Project: Prosper Loan Data Analysis

> The Dataset contains 113,917 loans, each row include infomation on the borrow's APR, status, borrowed amount, debt, etc. This investigation will be analyzing factors that influence borrow's APR and how each loan were taken by what type of borrowers.

In [2]:

```python
# 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

%matplotlib inline
```

In [3]:

```python
# expand maximun number of columns

pd.set_option('display.max_column',None)

```

In [4]:

```python
# load the raw dataset
df_loan = pd.read_csv('ProsperLoanData.csv')
df_loan.head()
```

Out[4]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanSta
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Comple
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Curr
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	Comple
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	NaN	36	Curr
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	NaN	36	Curr

In [5]:

```
# Check dataframe information
df_loan.info()

```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
ListingKey                             113937 non-null object
ListingNumber                          113937 non-null int64
ListingCreationDate                    113937 non-null object
CreditGrade                            28953 non-null object
Term                                   113937 non-null int64
LoanStatus                             113937 non-null object
ClosedDate                             55089 non-null object
BorrowerAPR                            113912 non-null float64
BorrowerRate                           113937 non-null float64
LenderYield                            113937 non-null float64
EstimatedEffectiveYield                84853 non-null float64
EstimatedLoss                          84853 non-null float64
EstimatedReturn                        84853 non-null float64
ProsperRating (numeric)                84853 non-null float64
ProsperRating (Alpha)                  84853 non-null object
ProsperScore                           84853 non-null float64
ListingCategory (numeric)              113937 non-null int64
BorrowerState                          108422 non-null object
Occupation                             110349 non-null object
EmploymentStatus                       111682 non-null object
EmploymentStatusDuration               106312 non-null float64
IsBorrowerHomeowner                    113937 non-null bool
CurrentlyInGroup                       113937 non-null bool
GroupKey                               13341 non-null object
DateCreditPulled                       113937 non-null object
CreditScoreRangeLower                  113346 non-null float64
CreditScoreRangeUpper                  113346 non-null float64
FirstRecordedCreditLine                113240 non-null object
CurrentCreditLines                     106333 non-null float64
OpenCreditLines                        106333 non-null float64
TotalCreditLinespast7years             113240 non-null float64
OpenRevolvingAccounts                  113937 non-null int64
OpenRevolvingMonthlyPayment            113937 non-null float64
InquiriesLast6Months                   113240 non-null float64
TotalInquiries                         112778 non-null float64
CurrentDelinquencies                   113240 non-null float64
AmountDelinquent                       106315 non-null float64
DelinquenciesLast7Years                112947 non-null float64
PublicRecordsLast10Years               113240 non-null float64
PublicRecordsLast12Months              106333 non-null float64
RevolvingCreditBalance                 106333 non-null float64
BankcardUtilization                    106333 non-null float64
AvailableBankcardCredit                106393 non-null float64
TotalTrades                            106393 non-null float64
TradesNeverDelinquent (percentage)     106393 non-null float64
TradesOpenedLast6Months                106393 non-null float64
DebtToIncomeRatio                      105383 non-null float64
IncomeRange                            113937 non-null object
IncomeVerifiable                       113937 non-null bool
StatedMonthlyIncome                    113937 non-null float64
```

```
LoanKey                                113937 non-null object
TotalProsperLoans                      22085 non-null float64
TotalProsperPaymentsBilled             22085 non-null float64
OnTimeProsperPayments                  22085 non-null float64
ProsperPaymentsLessThanOneMonthLate    22085 non-null float64
ProsperPaymentsOneMonthPlusLate        22085 non-null float64
ProsperPrincipalBorrowed               22085 non-null float64
ProsperPrincipalOutstanding            22085 non-null float64
ScorexChangeAtTimeOfListing            18928 non-null float64
LoanCurrentDaysDelinquent              113937 non-null int64
LoanFirstDefaultedCycleNumber          16952 non-null float64
LoanMonthsSinceOrigination             113937 non-null int64
LoanNumber                             113937 non-null int64
LoanOriginalAmount                     113937 non-null int64
LoanOriginationDate                    113937 non-null object
LoanOriginationQuarter                 113937 non-null object
MemberKey                              113937 non-null object
MonthlyLoanPayment                     113937 non-null float64
LP_CustomerPayments                    113937 non-null float64
LP_CustomerPrincipalPayments           113937 non-null float64
LP_InterestandFees                     113937 non-null float64
LP_ServiceFees                         113937 non-null float64
LP_CollectionFees                      113937 non-null float64
LP_GrossPrincipalLoss                  113937 non-null float64
LP_NetPrincipalLoss                    113937 non-null float64
LP_NonPrincipalRecoverypayments        113937 non-null float64
PercentFunded                          113937 non-null float64
Recommendations                        113937 non-null int64
InvestmentFromFriendsCount             113937 non-null int64
InvestmentFromFriendsAmount            113937 non-null float64
Investors                              113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

In [6]:

```
# Check the statistical value
df_loan.describe()
```

Out[6]:

	ListingNumber	Term	BorrowerAPR	BorrowerRate	LenderYield	EstimatedE
count	1.139370e+05	113937.000000	113912.000000	113937.000000	113937.000000	ε
mean	6.278857e+05	40.830248	0.218828	0.192764	0.182701	
std	3.280762e+05	10.436212	0.080364	0.074818	0.074516	
min	4.000000e+00	12.000000	0.006530	0.000000	-0.010000	
25%	4.009190e+05	36.000000	0.156290	0.134000	0.124200	
50%	6.005540e+05	36.000000	0.209760	0.184000	0.173000	
75%	8.926340e+05	36.000000	0.283810	0.250000	0.240000	
max	1.255725e+06	60.000000	0.512290	0.497500	0.492500	

In [7]:

```python
# Check the duplicated rows in this dataframe
df_loan.duplicated().sum()
```

Out[7]: 0

What is the structure of your dataset?

The Prosper loan dataset contains 113,937 observations of 81 variables. The observations refer to loan listings on Prosper.com from late 2005 until 2014, and various characteristics of those loans. The data seems “tidy,” but there are still a lot work to do to wragle the data according our exploration.

What is/are the main feature(s) of interest in your dataset?

The Borrower's APR will be analyzied with many factors such as the borrower's rating, creditscore, occupation and Delinquencies that could influence change in borrower's APR. Another feature that looks interesting is Loan Status, in that some loans have performed while others have defaulted or been charged off (what I will call “non-performing”). It will be interesting to look at loan status/performance for different Occupations, Loan Origination Quarters, Prior Borrowers, and other variables. Also, with Credit Score being a proxy for risk, it will be interesting to see how loans with different Credit Scores have performed.

What other features in the dataset do you think will help support your investigation into your feature(s) of interest?

The Prosper Rating and score could show low Borrower's APR because higher rating reflect the borrower's personality to be more trustworthy. Creditscore could also have similar effect on Borrower's APR as Prosper Rating.

Data Wragling

In [8]:

```python
# Copy the orignal dataframe
df_loan_clean = df_loan.copy()
```

In [9]:

```python
#Drop columns we don't need
df_loan_clean = df_loan_clean.drop(columns = ['ListingKey', 'ListingNumb
                                   'CurrentlyInGroup', 'GroupKey', 'DateC
                                   'TotalProsperLoans', 'TotalProsperPaym
                                   'ProsperPaymentsLessThanOneMonthLate',
                                   'ProsperPrincipalBorrowed', 'ProsperPr
                                   'LoanCurrentDaysDelinquent', 'LoanFirs

# Drop the rows including missing data in ProsperScore and EmploymentSta
df_loan_clean = df_loan_clean.dropna(subset=['ProsperScore','EmploymentS

```

In [10]:
```
# Check the change
df_loan_clean.shape
```

Out[10]:
```
(84834, 61)
```

In [11]:
```
# Drop the rows including missing data in ProsperScore and EmploymentSta
df_loan_clean = df_loan_clean.dropna(subset=['ProsperScore','EmploymentS
```

In [12]:
```
# Convert from float to int for columns 'ProsperScore','CreditScoreRange

df_loan_clean['ProsperScore'] = df_loan_clean.ProsperScore.astype(int)
df_loan_clean['CreditScoreRangeLower'] = df_loan_clean.CreditScoreRangeL
df_loan_clean['CreditScoreRangeUpper'] = df_loan_clean.CreditScoreRangeU
df_loan_clean['EmploymentStatusDuration'] = df_loan_clean.EmploymentStat

# Convert from int to str for columns'CreditScoreRangeLower','CreditScor
df_loan_clean['CreditScoreRangeLower'] = df_loan_clean.CreditScoreRangeL
df_loan_clean['CreditScoreRangeUpper'] = df_loan_clean.CreditScoreRangeU
```

In [13]:
```
# LoanStatus
df_loan_clean["LoanStatus"].value_counts()

```

Out[13]:
```
Current                   56566
Completed                 19657
Chargedoff                 5334
Defaulted                  1005
Past Due (1-15 days)        806
Past Due (31-60 days)       363
Past Due (61-90 days)       313
Past Due (91-120 days)      304
Past Due (16-30 days)       265
FinalPaymentInProgress      205
Past Due (>120 days)         16
Name: LoanStatus, dtype: int64
```

In [14]:
```
# We will redefinite the 11 loan status as three status depending on the
#  Current: Current, Past Due
#  Defaulted: Defaulted, Chargedoff
#  Completed : Completed, FinalPaymentInProgress
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Charg
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Past
df_loan_clean["LoanStatus"] = df_loan_clean["LoanStatus"].replace('Final
```

In [15]:

```
# Check the change
df_loan_clean["LoanStatus"].value_counts()
```

Out[15]:

```
Current      58633
Completed    19862
Defaulted     6339
Name: LoanStatus, dtype: int64
```

In [16]:

```
# Add one column"CreditScoreRange"
df_loan_clean['CreditScoreRange'] = df_loan_clean['CreditScoreRangeUpper
df_loan_clean['CreditScoreRange'].value_counts()
```

Out[16]:

```
679-660    14130
699-680    14015
719-700    13607
739-720    11033
659-640     8846
759-740     7870
779-760     5252
799-780     3705
819-800     2107
639-620     1650
839-820     1042
619-600     1040
859-840      398
879-860      122
899-880       17
Name: CreditScoreRange, dtype: int64
```

In [17]:

```
# Save the wrangled dataframe as csv
df_loan_clean.to_csv("ProsperLoanDataclean.csv", index = False)
```

In [18]:

```
# Check the change
df_loan_clean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 84834 entries, 1 to 113936
Data columns (total 62 columns):
Term                                  84834 non-null int64
LoanStatus                            84834 non-null object
BorrowerAPR                           84834 non-null float64
BorrowerRate                          84834 non-null float64
LenderYield                           84834 non-null float64
EstimatedEffectiveYield               84834 non-null float64
EstimatedLoss                         84834 non-null float64
EstimatedReturn                       84834 non-null float64
ProsperRating (numeric)               84834 non-null float64
ProsperRating (Alpha)                 84834 non-null object
ProsperScore                          84834 non-null int32
ListingCategory (numeric)             84834 non-null int64
BorrowerState                         84834 non-null object
Occupation                            83507 non-null object
EmploymentStatus                      84834 non-null object
EmploymentStatusDuration              84834 non-null int32
IsBorrowerHomeowner                   84834 non-null bool
CreditScoreRangeLower                 84834 non-null object
CreditScoreRangeUpper                 84834 non-null object
CurrentCreditLines                    84834 non-null float64
OpenCreditLines                       84834 non-null float64
TotalCreditLinespast7years            84834 non-null float64
OpenRevolvingAccounts                 84834 non-null int64
OpenRevolvingMonthlyPayment           84834 non-null float64
InquiriesLast6Months                  84834 non-null float64
TotalInquiries                        84834 non-null float64
CurrentDelinquencies                  84834 non-null float64
AmountDelinquent                      84834 non-null float64
DelinquenciesLast7Years               84834 non-null float64
PublicRecordsLast10Years              84834 non-null float64
PublicRecordsLast12Months             84834 non-null float64
RevolvingCreditBalance                84834 non-null float64
BankcardUtilization                   84834 non-null float64
AvailableBankcardCredit               84834 non-null float64
TotalTrades                           84834 non-null float64
TradesNeverDelinquent (percentage)    84834 non-null float64
TradesOpenedLast6Months               84834 non-null float64
DebtToIncomeRatio                     77543 non-null float64
IncomeRange                           84834 non-null object
IncomeVerifiable                      84834 non-null bool
StatedMonthlyIncome                   84834 non-null float64
LoanMonthsSinceOrigination            84834 non-null int64
LoanNumber                            84834 non-null int64
LoanOriginalAmount                    84834 non-null int64
LoanOriginationDate                   84834 non-null object
LoanOriginationQuarter                84834 non-null object
MemberKey                             84834 non-null object
MonthlyLoanPayment                    84834 non-null float64
LP_CustomerPayments                   84834 non-null float64
LP_CustomerPrincipalPayments          84834 non-null float64
LP_InterestandFees                    84834 non-null float64
```

```
LP_ServiceFees                        84834 non-null float64
LP_CollectionFees                     84834 non-null float64
LP_GrossPrincipalLoss                 84834 non-null float64
LP_NetPrincipalLoss                   84834 non-null float64
LP_NonPrincipalRecoverypayments       84834 non-null float64
PercentFunded                         84834 non-null float64
Recommendations                       84834 non-null int64
InvestmentFromFriendsCount            84834 non-null int64
InvestmentFromFriendsAmount           84834 non-null float64
Investors                             84834 non-null int64
CreditScoreRange                      84834 non-null object
dtypes: bool(2), float64(37), int32(2), int64(9), object(12)
memory usage: 39.0+ MB
```

Observation: There are 84834 loans entries and 62 attributes saved after data wragling.Each loan contain information on the borrowered's background information and details regarding the loans.

Data Exploration

We will analyze the features which effect the the borrower's APR and the factors you think will reflect the borrowers' credits to be more trustworthy.

Univariate Exploration

BorrowAPR(Histogram)

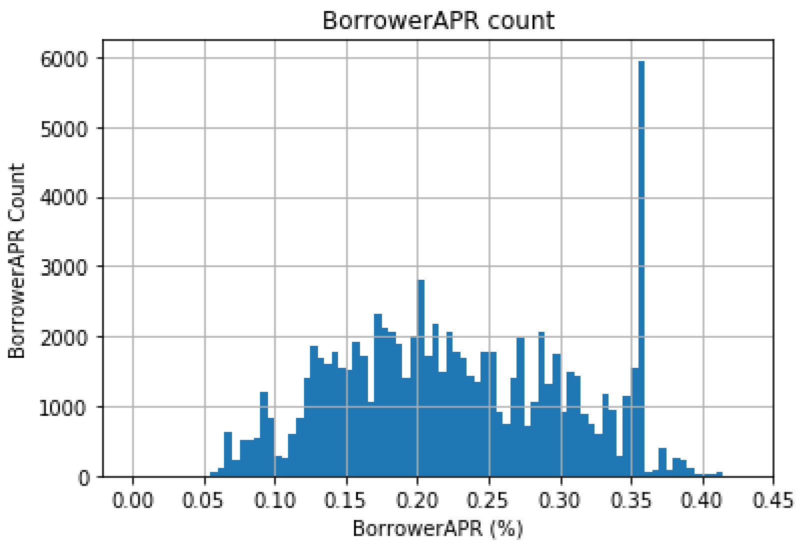
In [19]:

```
# check where APR has the most counts
df_BorrowerAPR = pd.DataFrame(df_loan_clean.BorrowerAPR.value_counts()).
df_BorrowerAPR.columns=['BorrowerAPR','BorrowerAPR Count']
df_BorrowerAPR.head(10)
```

Out[19]:

	BorrowerAPR	BorrowerAPR Count
0	0.35797	3671
1	0.35643	1644
2	0.30532	902
3	0.29510	747
4	0.35356	720
5	0.15833	650
6	0.24246	605
7	0.24758	601
8	0.12528	559
9	0.15324	547

In [20]:
```
# Plot BorrowerAPR histogram
bins = np.arange(0, df_BorrowerAPR['BorrowerAPR'].max(), 0.005)
x =  df_loan_clean['BorrowerAPR']
df_loan_clean['BorrowerAPR'].hist(bins=bins)
# Set title,ylabel,xlabel and xticks
plt.title('BorrowerAPR count')
plt.xlabel('BorrowerAPR (%)')
plt.ylabel('BorrowerAPR Count')
plt.xticks(np.arange(0, df_BorrowerAPR['BorrowerAPR'].max()+0.05, 0.05))
# Save plot
plt.savefig("image/BorrowerAPR count.png")
```



Observation: We can see borrowers' APR is between 0.045830% to 0.423950%, The highest density of borrowers is at 0.357(2329 people)

Loan Stutus(Horizontal Bar Chart)

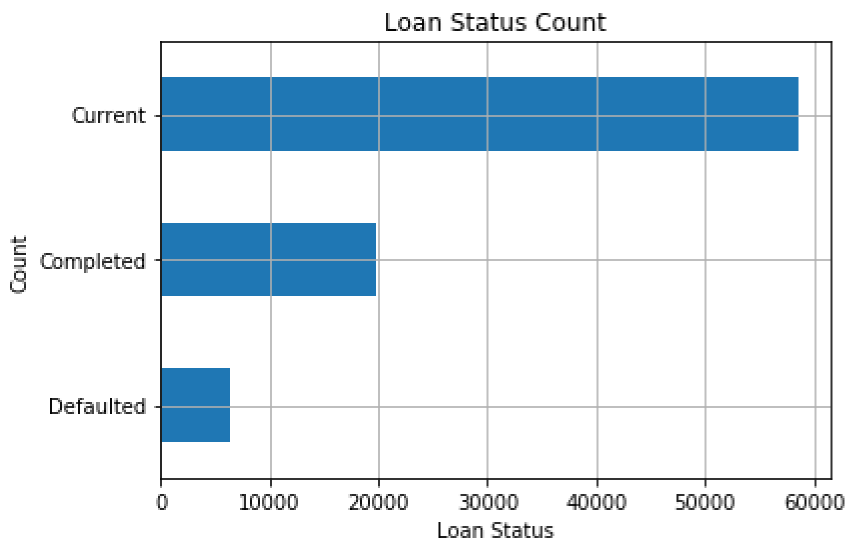
In [21]:
```
# Loan status
df_loan_clean["LoanStatus"].value_counts()
```

Out[21]:
```
Current      58633
Completed    19862
Defaulted     6339
Name: LoanStatus, dtype: int64
```

In [22]:
```
# Defaulted rating in completed loans
Defaulted_rating_pct = round(6339/(19862+ 6339)*100,2)
print(f"Observation: There are {Defaulted_rating_pct}% defaulted loans i
```

```
Observation: There are 24.19% defaulted loans in completed loans
```

In [23]:
```
# Plot the bar chart
df_loan_clean["LoanStatus"].value_counts()[3::-1].plot(kind="barh")
plt.title('Loan Status Count')
plt.xlabel('Loan Status')
plt.ylabel('Count')
plt.grid()

# Save plot
plt.savefig("image/LoanStatus Count.png")
```



Defaulted Loans Ration in Each ProsperRating Level

In [24]:
```
# Extract completed loans data from the whole dataset
df_completed = df_loan_clean[ df_loan_clean["LoanStatus"]!= 'Current']
defaulted_count = df_completed[df_completed["LoanStatus"]== 'Defaulted']
completed_count = df_completed.groupby('ProsperRating (Alpha)').LoanStat
Defaulted_pct = defaulted_count/completed_count *100

```

In [25]:

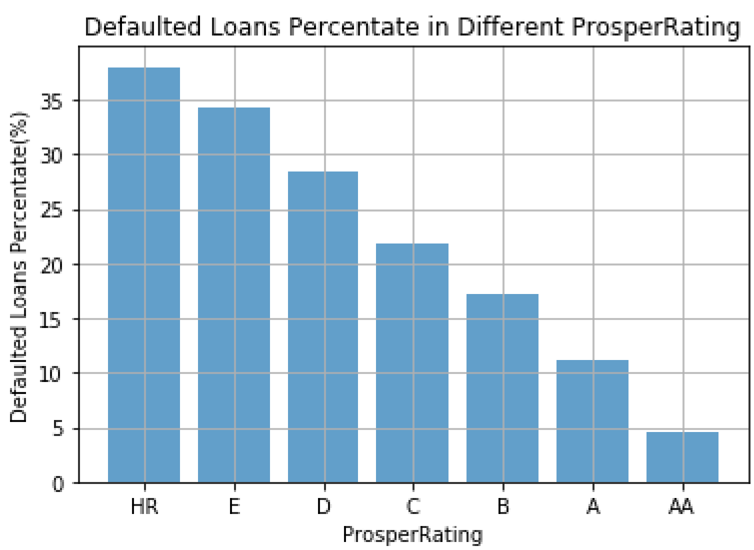
```
# Convert completed_count to dataframe and sort the values
Defaulted_pct_df = pd.DataFrame(Defaulted_pct).reset_index()
Defaulted_pct_df.columns=["ProsperRating (Alpha)", "Percentage"]
Defaulted_pct_df= Defaulted_pct_df.sort_values(by='Percentage',ascending
Defaulted_pct_df
```

Out[25]:

	ProsperRating (Alpha)	Percentage
6	HR	37.997330
5	E	34.282755
4	D	28.435697
3	C	21.806854
2	B	17.233294
0	A	11.123318
1	AA	4.623955

In [26]:

```
# plot bar chart
y_height= Defaulted_pct_df['Percentage']
x_pos = [i for i in range(len(Defaulted_pct_df))]
plt.bar(x_pos, y_height, align='center', alpha=0.7)
plt.xticks(x_pos, Defaulted_pct_df['ProsperRating (Alpha)'])
plt.grid()
plt.title("Defaulted Loans Percentate in Different ProsperRating")
plt.xlabel("ProsperRating")
plt.ylabel("Defaulted Loans Percentate(%)")
 # Save plot
plt.savefig("image/Defaulted Loans Percentate in Different ProsperRating
```



Observation The higher ProsperRating, the lower ration of defaulted loans

Defaulted Loans Ration in Each CreditScore Level

In [27]:

```
defaulted_count_2 = df_completed[df_completed["LoanStatus"]== 'Defaulted
# There are no defaulted loans in the CreditScorerange "899-880", we app
defaulted_count_2['899-880']=0
defaulted_count_2
```

Out[27]:

```
CreditScoreRange
619-600     220
639-620     327
659-640     863
679-660    1106
699-680     994
719-700     948
739-720     699
759-740     515
779-760     327
799-780     201
819-800      93
839-820      33
859-840       9
879-860       4
899-880       0
Name: CreditScoreRange, dtype: int64
```

In [28]:

```
completed_count_2 = df_completed.groupby('CreditScoreRange').CreditScore
completed_count_2
```

Out[28]:

```
CreditScoreRange
619-600     595
639-620     921
659-640    2926
679-660    3860
699-680    3706
719-700    3546
739-720    3099
759-740    2608
779-760    1879
799-780    1385
819-800     879
839-820     490
859-840     213
879-860      82
899-880      12
Name: CreditScoreRange, dtype: int64
```

In [29]:

```
# Convert completed_count to dataframe
Defaulted_pct_2 = round(defaulted_count_2/completed_count_2 *100, 2)

```

In [30]:

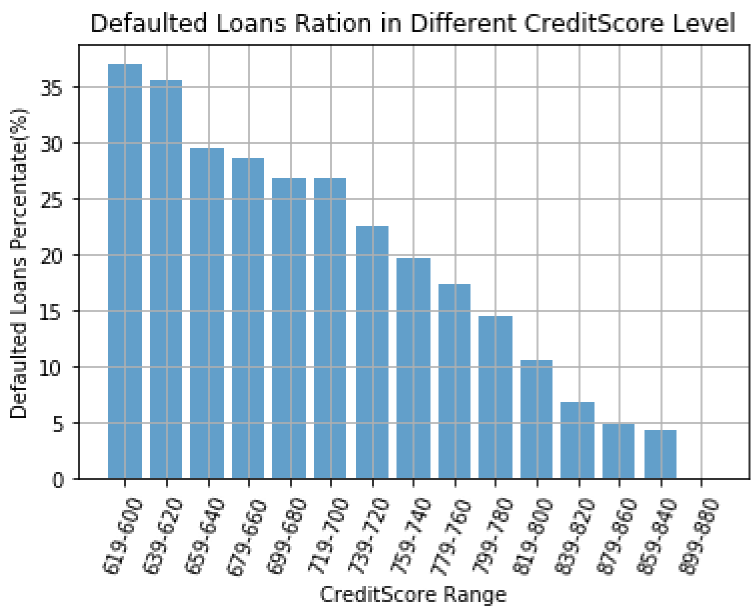
```
# Sort the values
Defaulted_pct_df_2 = pd.DataFrame(Defaulted_pct_2)
Defaulted_pct_df_2 = Defaulted_pct_df_2.rename(columns={'CreditScoreRang
Defaulted_pct_df_2 = Defaulted_pct_df_2.sort_values(by='Percentage',asce
Defaulted_pct_df_2

```

Out[30]:

	Percentage
CreditScoreRange	
619-600	36.97
639-620	35.50
659-640	29.49
679-660	28.65
699-680	26.82
719-700	26.73
739-720	22.56
759-740	19.75
779-760	17.40
799-780	14.51
819-800	10.58
839-820	6.73
879-860	4.88
859-840	4.23
899-880	0.00

In [31]:

```
# plot bar chart
y_height= Defaulted_pct_df_2['Percentage']
x_pos = [i for i in range(len(Defaulted_pct_df_2))]
plt.bar(x_pos, y_height, align='center', alpha=0.7)
plt.xticks(x_pos, Defaulted_pct_df_2.index,rotation = 70)
plt.grid()
plt.title("Defaulted Loans Ration in Different CreditScore Level")
plt.xlabel("CreditScore Range")
plt.ylabel("Defaulted Loans Percentate(%)")
 # Save plot
plt.savefig("image/Defaulted Loans Ration in Different CreditScore Level
```



Observation: The higher credit score the borrower have, the fewer defaulted loans they bring. There was even no defaulted loans for the borrower with 800-899

Occupation(Pie Chart)

In [32]:

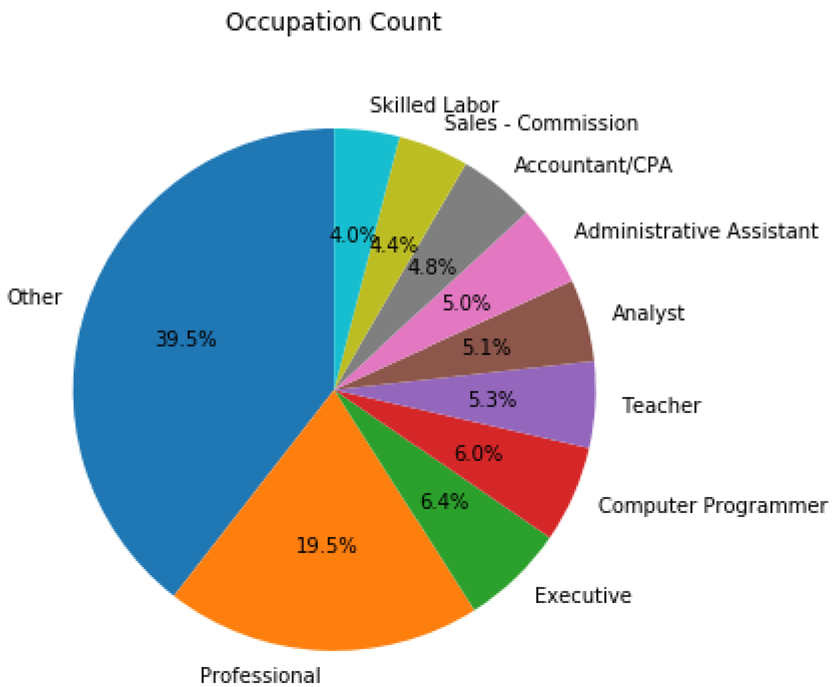
```
# The top 10 occupations of borrowers

# Filter the rows with null value in "occupation" column
df_loan_clean_1 = df_loan_clean.dropna(subset=['Occupation'])
df_Occupation = pd.DataFrame(df_loan_clean_1['Occupation'].value_counts(
df_Occupation.columns=['Occupation','Occupation Count']
df_Occupation.head(10)

```

Out[32]:

	Occupation	Occupation Count
0	Other	21317
1	Professional	10539
2	Executive	3468
3	Computer Programmer	3236
4	Teacher	2888
5	Analyst	2735
6	Administrative Assistant	2707
7	Accountant/CPA	2574
8	Sales - Commission	2350
9	Skilled Labor	2179

In [33]:

```
# Plot a pie chart for the top 10 occupation of borrowers
plt.pie(df_Occupation['Occupation Count'][:10], labels= df_Occupation['O
plt.title('Occupation Count',loc='center', y=1.3)
# Save plot
plt.savefig("image/Occupation Count.png")
```

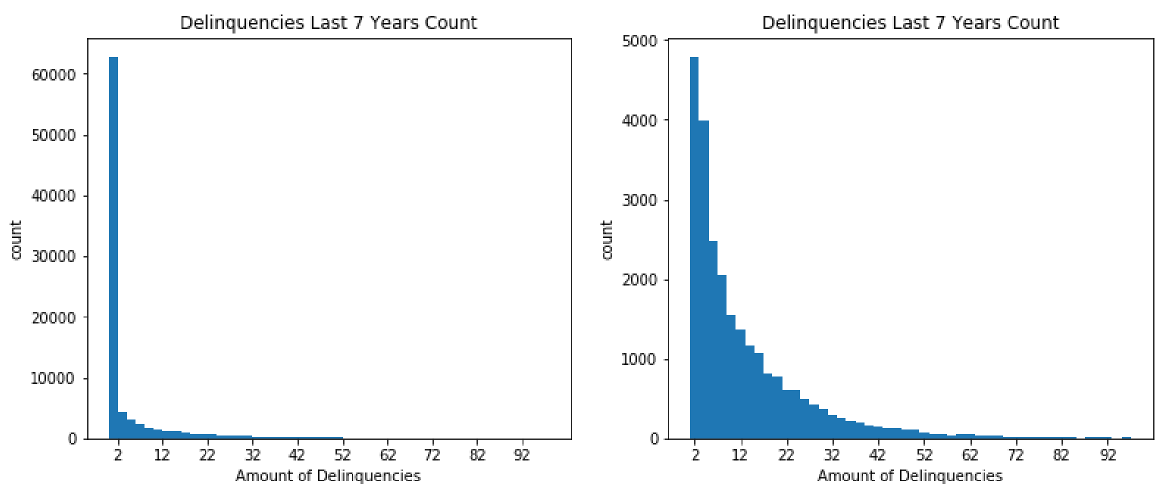


Observation: Most borrowers enter their occupations as "others" or "professional" because they might not want to share this information. The rest of the occupations do not show big increase compare to others.

Delinquencies records (Histogram)

In [37]:

```
# Histogram for Delinquencies records count from the Last 7Years.

plt.figure(figsize = [13, 5])

plt.subplot(1, 2, 1)
bins = np.arange(0, df_loan_clean['DelinquenciesLast7Years'].max(), 2)
plt.hist(data = df_loan_clean, x = 'DelinquenciesLast7Years', bins = bin
plt.xticks(np.arange(2, 100+1, 10))
plt.title('Delinquencies Last 7 Years Count')
plt.xlabel('Amount of Delinquencies')
plt.ylabel('count');

plt.subplot(1, 2, 2)
bins = np.arange(1, df_loan_clean['DelinquenciesLast7Years'].max(), 2)
plt.hist(data = df_loan_clean, x = 'DelinquenciesLast7Years', bins = bin
plt.xticks(np.arange(2, 100+1, 10))
plt.title('Delinquencies Last 7 Years Count')
plt.xlabel('Amount of Delinquencies')
plt.ylabel('count');
plt.savefig("image/DelinquenciesCount.png")
```



Observation: Most borrowers has no Delinquencies records. Another plot is ploted to exclude borrowers with 0 Delinquencies record. The counts seems to be decreased exponentially with higher number of Delinquencies.

ProsperRating vs BorrowerAPR mean

In [38]:

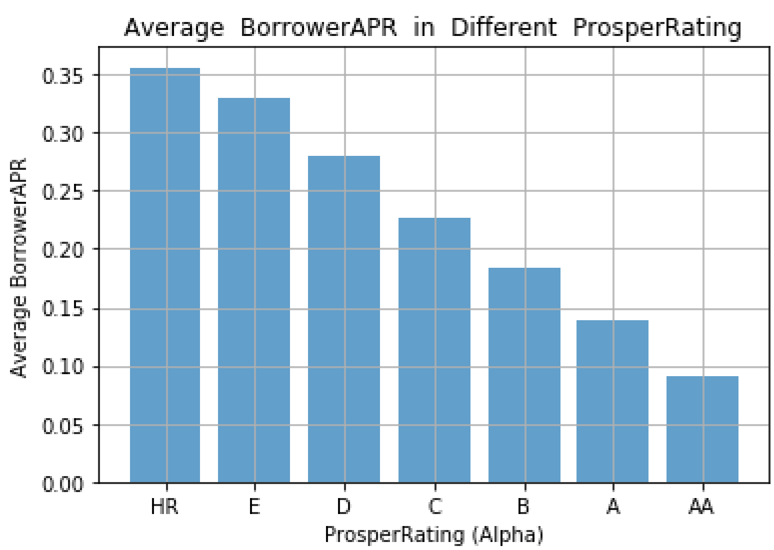
```
ProsperRating_mean = df_loan_clean.groupby('ProsperRating (Alpha)').Borr
ProsperRating_mean_df = pd.DataFrame(ProsperRating_mean)
ProsperRating_mean_df = ProsperRating_mean_df.sort_values(by="BorrowerAP

ProsperRating_mean_df

```

Out[38]:

	BorrowerAPR
ProsperRating (Alpha)	
HR	0.356059
E	0.330551
D	0.280584
C	0.226124
B	0.184031
A	0.138910
AA	0.090034

In [39]:

```
# Plot the bar chart
y_height= ProsperRating_mean_df['BorrowerAPR']
x_pos = [i for i in range(len(ProsperRating_mean_df))]
plt.bar(x_pos, y_height, align='center', alpha=0.7)
plt.xticks(x_pos, ProsperRating_mean_df.index)
plt.grid()
plt.title("Average  BorrowerAPR  in  Different  ProsperRating")
plt.xlabel("ProsperRating (Alpha)")
plt.ylabel("Average BorrowerAPR")
 # Save plot
plt.savefig("image/Average BorrowerAPR in Different ProsperRating.png")


```

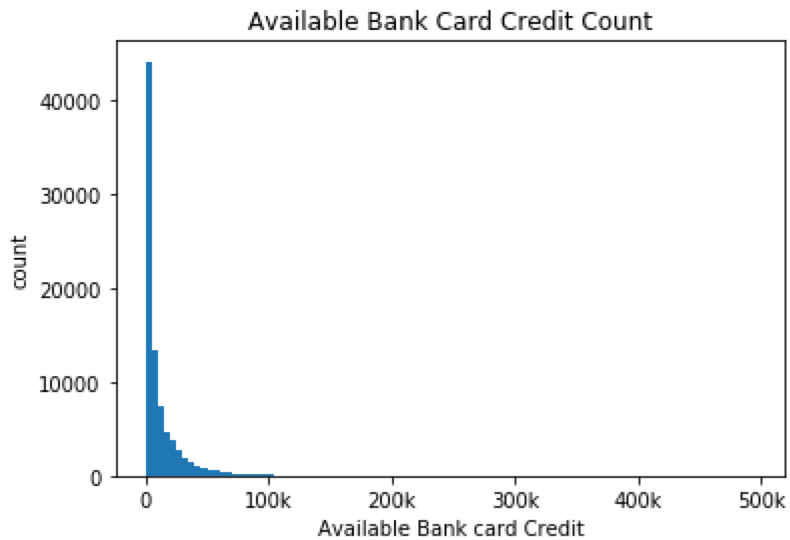


Observation: Borrowers Rating are displayed in order from highest rating to lowest rating (AA, A, B, C, D, E, HR). We can see a pattern that the highest rating of AA received lowest average APR

(0.09), whereas the lowerest rating received the highest average APR (0.356).

In [41]:

```
# see counts for AvailableBankcardCredit

bins = np.arange(0, df_loan_clean['AvailableBankcardCredit'].max(), 5000
plt.hist(data = df_loan_clean, x = 'AvailableBankcardCredit', bins = bin
plt.xticks([0, 1e5, 2e5, 3e5, 4e5, 5e5], [0, '100k', '200k', '300k', '40
plt.title('Available Bank Card Credit Count')
plt.xlabel('Available Bank card Credit')
plt.ylabel('count');
plt.savefig("image/AvailableBaknCreditCount.png")
```



Observation Most AvailableBankcardCredit counts fall in values from 0 to 100k. Clearly there are few percent of people who have higher bank total credits than majority of people.

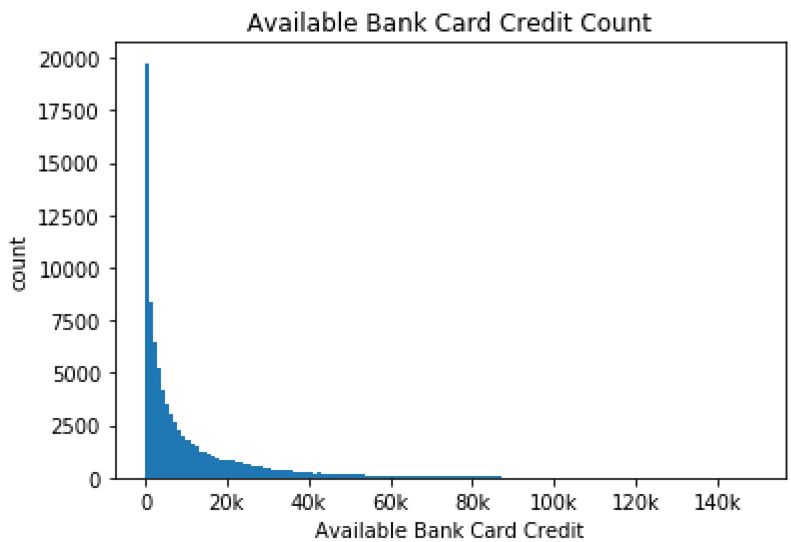
In [43]:

```
# Drop the loan case in which borrowers have AvailableBankcardCredit>150
High_credit_df = df_loan_clean[df_loan_clean['AvailableBankcardCredit']>
df_loan_clean_3 = df_loan_clean[df_loan_clean['AvailableBankcardCredit']
Low_credit_df = df_loan_clean[df_loan_clean['AvailableBankcardCredit']<=
print(f"There are {len(High_credit_df)} borrowers with very high Availab
print(f"There are {len(Low_credit_df)} borrowers with low AvailableBankc
df_loan_clean_3 = df_loan_clean[df_loan_clean['AvailableBankcardCredit']
```

```
There are 113 borrowers with very high AvailableBankcardCredit(greater than
150k)
There are 69645 borrowers with low AvailableBankcardCredit(less than 20k)
```

In [44]:

```
# plot again for AvailableBankcardCredit count with new filter data

bins = np.arange(0, df_loan_clean_3['AvailableBankcardCredit'].max(), 10
plt.hist(data =df_loan_clean_3, x = 'AvailableBankcardCredit', bins = bi
plt.xticks([0, 2e4, 4e4, 6e4, 8e4, 1e5, 1.2e5, 1.4e5],
           [0, '20k', '40k', '60k', '80k', '100k', '120k', '140k'])
plt.title('Available Bank Card Credit Count')
plt.xlabel('Available Bank Card Credit')
plt.ylabel('count');

```



Observation:" The variables are explored for more understanding of Borrower's APR. From AvailableBankcardCredit count plot shown above, most borrowers has AvailableBankcardCreditare within 1000k. 82% borrowers has AvailableBankcardCreditare less than 20k(69645).There are 113 borrowers with vailableBankcardCreditare higher than 150k were removed from the data because they are away from most of the data point shown in the plot.

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

```
The variables are explored for more understanding of Borrower's APR. Fro
m AvailableBankcardCredit count plot shown above, most borrowers has Ava
ilableBankcardCreditare within 1000k. There are 113 borrowers with Avail
ableBankcardCreditare higher than 150k were removed from the data becaus
e they are away from most of the data point shown in the plot. Also, loo
king at BorrowerAPR count, there are two BorrowerAPR counts that were hi
gher than rest of the values. Due to high number counts falling into tho
se two values, there might be resonable reasons these two values are use
d. Therefore, the two BorrowerAPR values are kept untouched.
```

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

```
The countplots for ProsperScore Delinquencies, Last7Years and AvailableB
ankcardCreditare are right skewed. CreditScoreRangeUpper & CreditScoreRa
ngeLower follow a normal distribution curve.
```

Bivariate Exploration

In [47]:

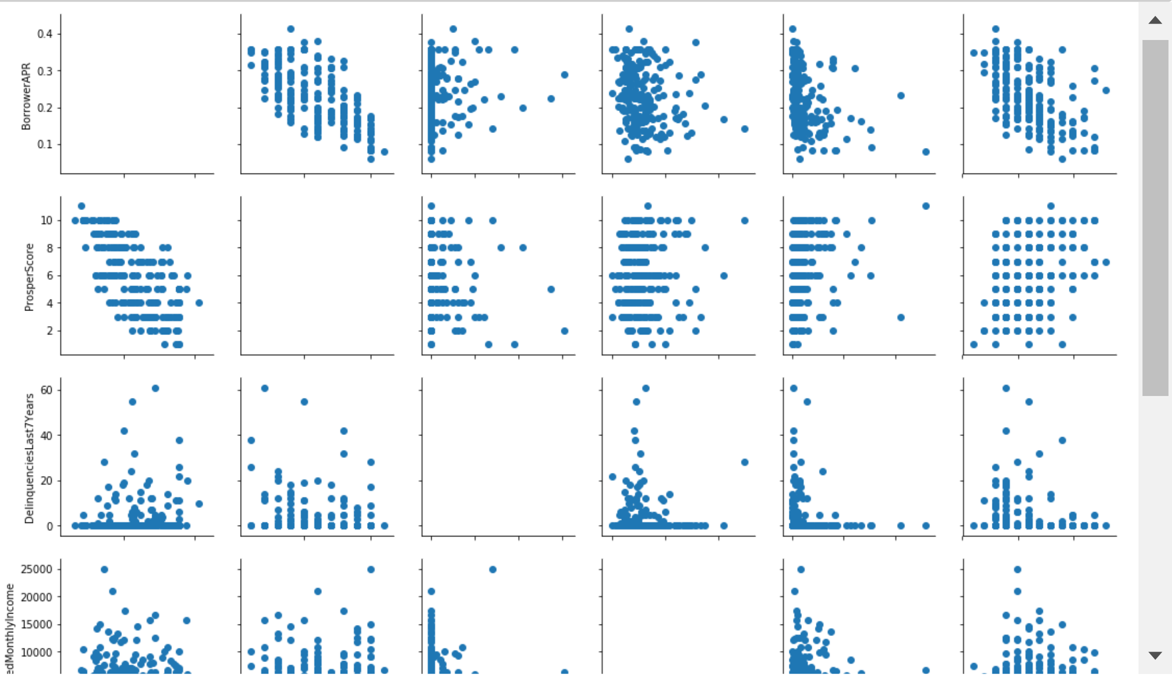
```python
# correlation plot
# Change CreditScoreRangeUpper from str to int
df_loan_clean_3['CreditScoreRangeUpper'] = df_loan_clean_3.CreditScoreRa
num_vars = ['BorrowerAPR', 'ProsperScore', 'DelinquenciesLast7Years',
            'StatedMonthlyIncome', 'AvailableBankcardCredit', 'CreditSco

plt.figure(figsize = [8, 5])
sb.heatmap(df_loan_clean_3[num_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'vlag_r', center = 0)
plt.title('Correlation Plot')
plt.show()

plt.savefig("image/Correlation.png")
```

Observation: There are no strong positive relationships between any pairs. It makes sense because higher AvailableBankcardCredit has better creditscore. BorrowerAPR and ProsperScore are negative because borrowers with lower score are more likely to pay higher APR. Similarly, higher CreditScore means the borrowers are more trustworthy, therefore it recevied lower APR.

Scatter plot to explore pair up all above variables.

In [48]:

```
# plot matrix: only 300 random loans are used to see the pattern more cl


num_vars = ['BorrowerAPR', 'ProsperScore', 'DelinquenciesLast7Years',
            'StatedMonthlyIncome', 'AvailableBankcardCredit', 'CreditSco

samples = np.random.choice(df_loan_clean_3.shape[0], 300, replace = Fals
loan_samp = df_loan_clean_3.loc[samples,:]

g = sb.PairGrid(data = loan_samp, vars = num_vars)
g.map_offdiag(plt.scatter)
plt.title('Matrix Plot');

plt.savefig("image/correlation2.png")
```

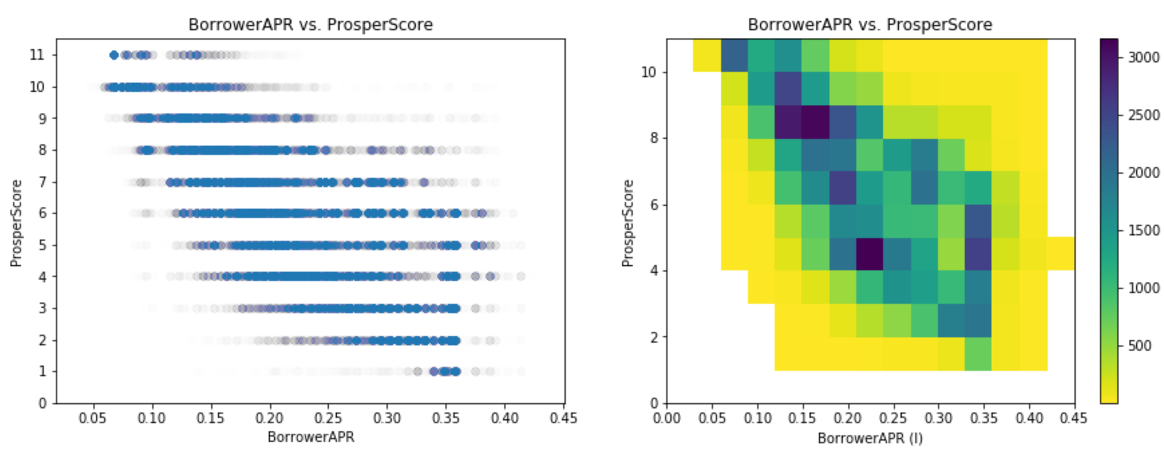


Observation: Similar to the correlation plot, we can determine which pair has negative or positive relationships from analyzing the pattern in each scatter plots. ProsperScore seems to be more related to BorrowerAPR compare to other variables. StatedMonthlyIncome does not give useful information on BorrowerAPR and will not be further analyzed.

More plots to look at ProsperScore vs BorrowerAPR more closely

In [49]:

```
# scatter and heat plot for comparing ProsperScore and BorrowerAPR.
plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 1)
plt.scatter(data = df_loan_clean_3, x = 'BorrowerAPR', y = 'ProsperScore
plt.yticks(np.arange(0, 12, 1))
plt.title('BorrowerAPR vs. ProsperScore')
plt.xlabel('BorrowerAPR')
plt.ylabel('ProsperScore')


plt.subplot(1, 2, 2)
bins_x = np.arange(0, df_loan_clean_3['BorrowerAPR'].max()+0.05, 0.03)
bins_y = np.arange(0, df_loan_clean_3['ProsperScore'].max()+1, 1)
plt.hist2d(data = df_loan_clean_3, x = 'BorrowerAPR', y = 'ProsperScore'
               cmap = 'viridis_r', cmin = 0.5)
plt.colorbar()
plt.title('BorrowerAPR vs. ProsperScore')
plt.xlabel('BorrowerAPR (l)')
plt.ylabel('ProsperScore');

plt.savefig("image/BorrowerAPR_ProsperScore.png")
```

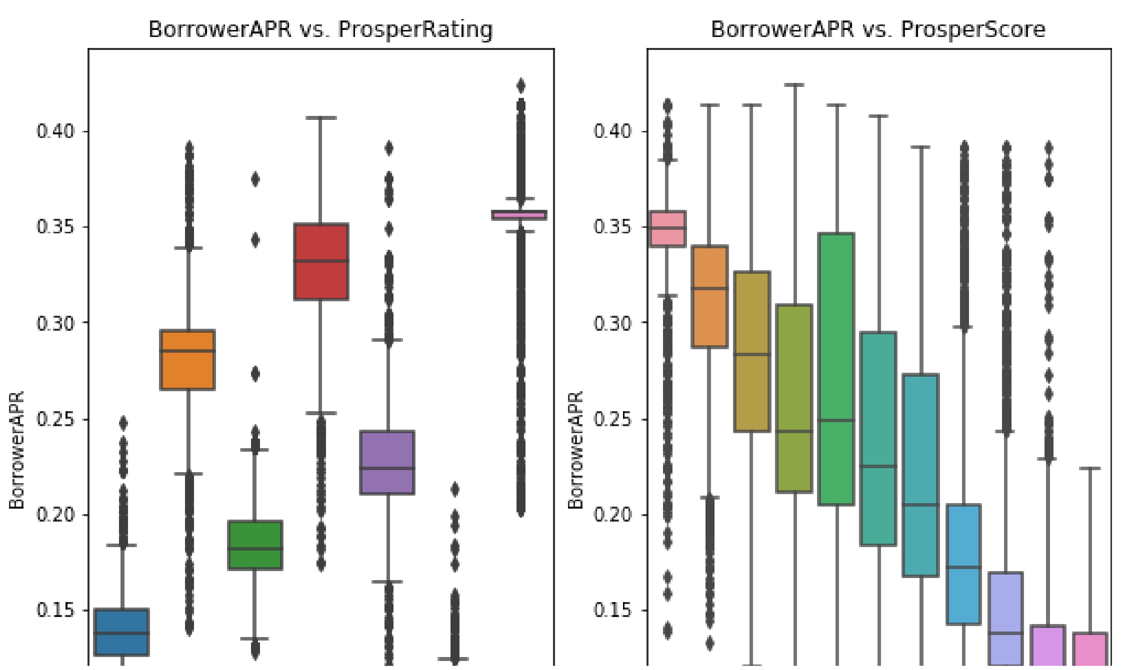


Observation: People with higher rating tend to be more reliable and therefore given lower BorrowerAPR

BorrowerAPR vs. ProsperRating & ProsperScore

In [50]:

```
# Violin plot for BorrowerAPR vs. ProsperRating & ProsperScore. Shows hi
plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 1)
sb.boxplot(data = df_loan_clean, x = 'ProsperRating (Alpha)', y = 'Borro
plt.gcf().set_size_inches(10, 8)
plt.title('BorrowerAPR vs. ProsperRating')
plt.xlabel('ProsperRating')
plt.ylabel('BorrowerAPR')

plt.subplot(1, 2, 2)
sb.boxplot(data = df_loan_clean, x = 'ProsperScore', y = 'BorrowerAPR')
plt.gcf().set_size_inches(10, 8)
plt.title('BorrowerAPR vs. ProsperScore')
plt.xlabel('ProsperScore')
plt.ylabel('BorrowerAPR');

plt.savefig("image/BorrowerAPR_ProsperRating.png")
```



Observation: For these two categorical variables, there is not much correlation on ProsperRating. Good or bad rating doesn't reflect the percentage of APR the borrower will get. For ProsperScore, there are clearly negative relationship with BorrowerAPR as discussed in Univariate Exploration.

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

```
The correlation and matrix plots are really helpful to preview all possi
bles variables related on BorrowerAPR we can trying to analyize. Out of
 all variables, ProsperScore has stronger relationship with BorrowerAPR
 (negative correlated). Univariate Exploration helps to examine data poi
nts and statistics about our variables. By looking into Bivariate Explor
ation, it is more clearly to gain more understanding and answer question
s about BorrowerAPR.
```

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

```
The CreditScoreRangeUpper, AvailableBankcardCredit and CreditScoreRangeU
pper are all positive correlated to ProsperScore and negative correlated
to BorrwerAPR.
```

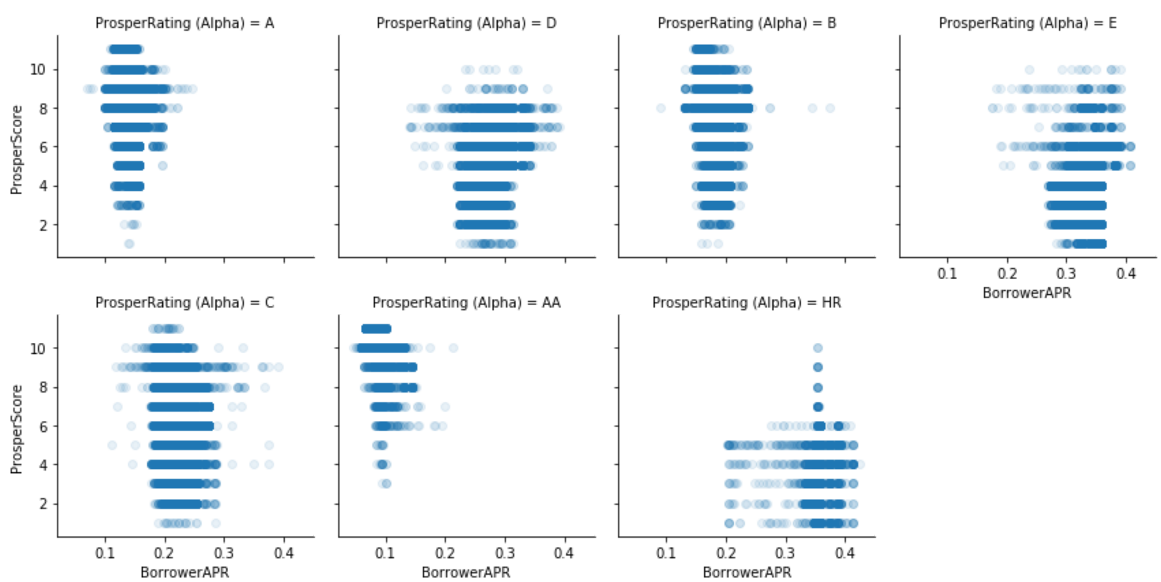
Multivariate Exploration

FacetGrid: BorrowerAPR vs ProsperScore

In [51]:

```
g = sb.FacetGrid(data = df_loan_clean_3, col = 'ProsperRating (Alpha)',
g.map(plt.scatter, 'BorrowerAPR', 'ProsperScore', alpha = 0.1)
g.set_xlabels('BorrowerAPR')
g.set_ylabels('ProsperScore')

plt.show()

plt.savefig("image/Correlation3.png")
```

```
C:\Users\fawnz\anaconda\Anaconda3\lib\site-packages\seaborn\axisgrid.py:23
0: UserWarning: The `size` paramter has been renamed to `height`; please up
date your code.
  warnings.warn(msg, UserWarning)
```



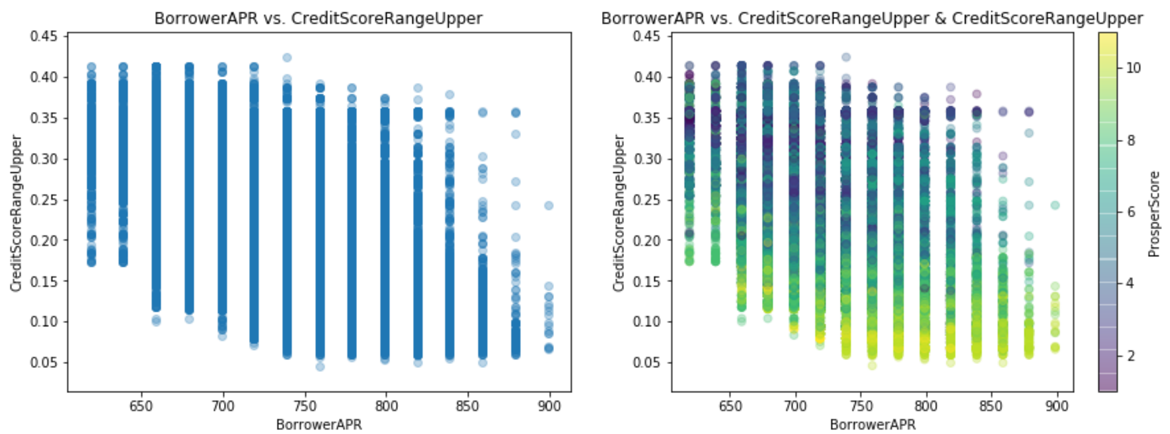

```
<Figure size 432x288 with 0 Axes>
```

Observation: This visualization helps to analyze BorrowerAPR vs ProsperScore on difference letter ratings. The patterns shows the lowerest rating(HR) of borrowers have the highest APR. For high rating A(A), the borrowers has the lowers APR. This visualization differenate groups of people in terms of APR received based on their rating and scores.

BorrowerAPR vs. CreditScoreRangeUpper & ProsperScore

In [52]:

```

plt.figure(figsize = [15, 5])

plt.subplot(1, 2, 1)
plt.scatter(data = df_loan_clean_3, x = 'CreditScoreRangeUpper', y = 'Bo
plt.title('BorrowerAPR vs. CreditScoreRangeUpper')
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');


plt.subplot(1, 2, 2)
plt.scatter(data = df_loan_clean_3, x = 'CreditScoreRangeUpper', y = 'Bo
plt.colorbar(label = 'ProsperScore')
plt.title('BorrowerAPR vs. CreditScoreRangeUpper & CreditScoreRangeUpper
plt.xlabel('BorrowerAPR')
plt.ylabel('CreditScoreRangeUpper');

plt.savefig("image/BorrowerAPR_CreditScoreRangeUpper.png")
```



Observation: We can see the CreditScoreRangeUpper increase as BorrowerAPR decrease in the plots. By adding ProsperScore to color encodings, BorrowerAPR decreases as ProsperScore increases. This proves the point that CreditScoreRangeUpper and ProsperScore negatively correlated to BorrowerAPR.

Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

> The correlation and matrix plots in previous plots can also be counted as part of Multivariate Exploration. To be more efficient, these two plots can be done ealier part of exploration to preview all variables and how they interact to each other. Adding to that, FacetGrid shows how each rating groups differ in terms of BorrowerAPR vs ProsperScore.

> From all above visualizations created from univariate exploration to multivariate exploration, many variable are found to be negatively correlated to BorrowerAPR, whereas ProspoerScore gives the strongest negative relationship.

Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

> The correlation and matrix plots are really helpful to preview all possibles variables related on BorrowerAPR we can trying to analyize. Out of all variables, ProsperScore has stronger relationship with BorrowerAPR (negative correlated). Univariate Exploration helps to examine data points and statistics about our variables. By looking into Bivariate Exploration, it is more clearly to gain more understanding and answer questions about BorrowerAPR.

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

> The CreditScoreRangeUpper, AvailableBankcardCredit and CreditScoreRangeUpper are all positive correlated to ProsperScore and negative correlated to BorrwerAPR.

In [ ]:

```
1
```