

# Part\_I\_exploration\_template

February 27, 2023

## 1 Part I - (Analysis of Loan Data from Prosper)

### 1.1 by (Abdulraqib Omotosho)

### 1.2 Introduction

The dataset I will be exploring and visualizing for this project is the [Loan Data from Prosper](#) gotten from Udacity's curated datasets. This dataset 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.

### 1.3 Preliminary Wrangling

This notebook explores the dataset for the purpose of Exploratory Data Analysis and Explanatory Data Analysis.

```
[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 sns
sns.set_theme(style='white')

%matplotlib inline
```

Loading the dataset and describing its properties.

```
[2]: df = pd.read_csv('prosperLoanData.csv')
df.head(10)
```

```
[2]:
```

	ListingKey	ListingNumber	ListingCreationDate	\
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	
5	0F05359734824199381F61D	1074836	2013-12-14 08:26:37.093000000	
6	0F0A3576754255009D63151	750899	2013-04-12 09:52:56.147000000	
7	0F1035772717087366F9EA7	768193	2013-05-05 06:49:27.493000000	
8	0F043596202561788EA13D5	1023355	2013-12-02 10:43:39.117000000	

9 0F043596202561788EA13D5

1023355 2013-12-02 10:43:39.117000000

	CreditGrade	Term	LoanStatus	ClosedDate	BorrowerAPR	\
0	C	36	Completed	2009-08-14 00:00:00	0.16516	
1	NaN	36	Current	NaN	0.12016	
2	HR	36	Completed	2009-12-17 00:00:00	0.28269	
3	NaN	36	Current	NaN	0.12528	
4	NaN	36	Current	NaN	0.24614	
5	NaN	60	Current	NaN	0.15425	
6	NaN	36	Current	NaN	0.31032	
7	NaN	36	Current	NaN	0.23939	
8	NaN	36	Current	NaN	0.07620	
9	NaN	36	Current	NaN	0.07620	

	BorrowerRate	LenderYield	...	LP_ServiceFees	LP_CollectionFees	\
0	0.1580	0.1380	...	-133.18	0.0	
1	0.0920	0.0820	...	0.00	0.0	
2	0.2750	0.2400	...	-24.20	0.0	
3	0.0974	0.0874	...	-108.01	0.0	
4	0.2085	0.1985	...	-60.27	0.0	
5	0.1314	0.1214	...	-25.33	0.0	
6	0.2712	0.2612	...	-22.95	0.0	
7	0.2019	0.1919	...	-69.21	0.0	
8	0.0629	0.0529	...	-16.77	0.0	
9	0.0629	0.0529	...	-16.77	0.0	

	LP_GrossPrincipalLoss	LP_NetPrincipalLoss	LP_NonPrincipalRecoverypayments	\
0	0.0	0.0	0.0	
1	0.0	0.0	0.0	
2	0.0	0.0	0.0	
3	0.0	0.0	0.0	
4	0.0	0.0	0.0	
5	0.0	0.0	0.0	
6	0.0	0.0	0.0	
7	0.0	0.0	0.0	
8	0.0	0.0	0.0	
9	0.0	0.0	0.0	

	PercentFunded	Recommendations	InvestmentFromFriendsCount	\
0	1.0	0	0	
1	1.0	0	0	
2	1.0	0	0	
3	1.0	0	0	
4	1.0	0	0	
5	1.0	0	0	
6	1.0	0	0	
7	1.0	0	0	

```

8          1.0          0          0
9          1.0          0          0

```

```

InvestmentFromFriendsAmount Investors
0          0.0          258
1          0.0           1
2          0.0          41
3          0.0         158
4          0.0          20
5          0.0           1
6          0.0           1
7          0.0           1
8          0.0           1
9          0.0           1

```

```
[10 rows x 81 columns]
```

```
[3]: df.shape
```

```
[3]: (113937, 81)
```

```
[5]: df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ListingKey                           113937 non-null object
1   ListingNumber                         113937 non-null int64
2   ListingCreationDate                  113937 non-null object
3   CreditGrade                          28953 non-null  object
4   Term                                113937 non-null int64
5   LoanStatus                           113937 non-null object
6   ClosedDate                           55089 non-null  object
7   BorrowerAPR                          113912 non-null float64
8   BorrowerRate                         113937 non-null float64
9   LenderYield                          113937 non-null float64
10  EstimatedEffectiveYield               84853 non-null  float64
11  EstimatedLoss                         84853 non-null  float64
12  EstimatedReturn                       84853 non-null  float64
13  ProsperRating (numeric)               84853 non-null  float64
14  ProsperRating (Alpha)                 84853 non-null  object
15  ProsperScore                          84853 non-null  float64
16  ListingCategory (numeric)             113937 non-null int64
17  BorrowerState                         108422 non-null object
18  Occupation                            110349 non-null object
19  EmploymentStatus                     111682 non-null object

```

20	EmploymentStatusDuration	106312	non-null	float64
21	IsBorrowerHomeowner	113937	non-null	bool
22	CurrentlyInGroup	113937	non-null	bool
23	GroupKey	13341	non-null	object
24	DateCreditPulled	113937	non-null	object
25	CreditScoreRangeLower	113346	non-null	float64
26	CreditScoreRangeUpper	113346	non-null	float64
27	FirstRecordedCreditLine	113240	non-null	object
28	CurrentCreditLines	106333	non-null	float64
29	OpenCreditLines	106333	non-null	float64
30	TotalCreditLinespast7years	113240	non-null	float64
31	OpenRevolvingAccounts	113937	non-null	int64
32	OpenRevolvingMonthlyPayment	113937	non-null	float64
33	InquiriesLast6Months	113240	non-null	float64
34	TotalInquiries	112778	non-null	float64
35	CurrentDelinquencies	113240	non-null	float64
36	AmountDelinquent	106315	non-null	float64
37	DelinquenciesLast7Years	112947	non-null	float64
38	PublicRecordsLast10Years	113240	non-null	float64
39	PublicRecordsLast12Months	106333	non-null	float64
40	RevolvingCreditBalance	106333	non-null	float64
41	BankcardUtilization	106333	non-null	float64
42	AvailableBankcardCredit	106393	non-null	float64
43	TotalTrades	106393	non-null	float64
44	TradesNeverDelinquent (percentage)	106393	non-null	float64
45	TradesOpenedLast6Months	106393	non-null	float64
46	DebtToIncomeRatio	105383	non-null	float64
47	IncomeRange	113937	non-null	object
48	IncomeVerifiable	113937	non-null	bool
49	StatedMonthlyIncome	113937	non-null	float64
50	LoanKey	113937	non-null	object
51	TotalProsperLoans	22085	non-null	float64
52	TotalProsperPaymentsBilled	22085	non-null	float64
53	OnTimeProsperPayments	22085	non-null	float64
54	ProsperPaymentsLessThanOneMonthLate	22085	non-null	float64
55	ProsperPaymentsOneMonthPlusLate	22085	non-null	float64
56	ProsperPrincipalBorrowed	22085	non-null	float64
57	ProsperPrincipalOutstanding	22085	non-null	float64
58	ScorexChangeAtTimeOfListing	18928	non-null	float64
59	LoanCurrentDaysDelinquent	113937	non-null	int64
60	LoanFirstDefaultedCycleNumber	16952	non-null	float64
61	LoanMonthsSinceOrigination	113937	non-null	int64
62	LoanNumber	113937	non-null	int64
63	LoanOriginalAmount	113937	non-null	int64
64	LoanOriginationDate	113937	non-null	object
65	LoanOriginationQuarter	113937	non-null	object
66	MemberKey	113937	non-null	object
67	MonthlyLoanPayment	113937	non-null	float64

```

68 LP_CustomerPayments          113937 non-null float64
69 LP_CustomerPrincipalPayments 113937 non-null float64
70 LP_InterestandFees           113937 non-null float64
71 LP_ServiceFees                113937 non-null float64
72 LP_CollectionFees            113937 non-null float64
73 LP_GrossPrincipalLoss        113937 non-null float64
74 LP_NetPrincipalLoss          113937 non-null float64
75 LP_NonPrincipalRecoverypayments 113937 non-null float64
76 PercentFunded                113937 non-null float64
77 Recommendations              113937 non-null int64
78 InvestmentFromFriendsCount    113937 non-null int64
79 InvestmentFromFriendsAmount   113937 non-null float64
80 Investors                    113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB

```

```
[6]: df.describe()
```

```

[6]:      ListingNumber      Term  BorrowerAPR  BorrowerRate  \
count    1.139370e+05  113937.000000  113912.000000  113937.000000
mean      6.278857e+05    40.830248      0.218828      0.192764
std       3.280762e+05    10.436212      0.080364      0.074818
min       4.000000e+00    12.000000      0.006530      0.000000
25%       4.009190e+05    36.000000      0.156290      0.134000
50%       6.005540e+05    36.000000      0.209760      0.184000
75%       8.926340e+05    36.000000      0.283810      0.250000
max       1.255725e+06    60.000000      0.512290      0.497500

      LenderYield  EstimatedEffectiveYield  EstimatedLoss  EstimatedReturn  \
count    113937.000000      84853.000000      84853.000000      84853.000000
mean         0.182701         0.168661         0.080306         0.096068
std         0.074516         0.068467         0.046764         0.030403
min        -0.010000        -0.182700         0.004900        -0.182700
25%         0.124200         0.115670         0.042400         0.074080
50%         0.173000         0.161500         0.072400         0.091700
75%         0.240000         0.224300         0.112000         0.116600
max         0.492500         0.319900         0.366000         0.283700

      ProsperRating (numeric)  ProsperScore  ...  LP_ServiceFees  \
count          84853.000000  84853.000000  ...    113937.000000
mean             4.072243      5.950067  ...     -54.725641
std             1.673227      2.376501  ...      60.675425
min             1.000000      1.000000  ...    -664.870000
25%             3.000000      4.000000  ...    -73.180000
50%             4.000000      6.000000  ...    -34.440000
75%             5.000000      8.000000  ...    -13.920000
max             7.000000     11.000000  ...     32.060000

```

	LP_CollectionFees	LP_GrossPrincipalLoss	LP_NetPrincipalLoss	\
count	113937.000000	113937.000000	113937.000000	
mean	-14.242698	700.446342	681.420499	
std	109.232758	2388.513831	2357.167068	
min	-9274.750000	-94.200000	-954.550000	
25%	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	
max	0.000000	25000.000000	25000.000000	

	LP_NonPrincipalRecoverypayments	PercentFunded	Recommendations	\
count	113937.000000	113937.000000	113937.000000	
mean	25.142686	0.998584	0.048027	
std	275.657937	0.017919	0.332353	
min	0.000000	0.700000	0.000000	
25%	0.000000	1.000000	0.000000	
50%	0.000000	1.000000	0.000000	
75%	0.000000	1.000000	0.000000	
max	21117.900000	1.012500	39.000000	

	InvestmentFromFriendsCount	InvestmentFromFriendsAmount	Investors
count	113937.000000	113937.000000	113937.000000
mean	0.023460	16.550751	80.475228
std	0.232412	294.545422	103.239020
min	0.000000	0.000000	1.000000
25%	0.000000	0.000000	2.000000
50%	0.000000	0.000000	44.000000
75%	0.000000	0.000000	115.000000
max	33.000000	25000.000000	1189.000000

[8 rows x 61 columns]

### 1.3.1 What is the structure of your dataset?

The dataset contains 113,937 rows with 81 features. Most of the variables are numeric in nature while others are ordered factor variables such as `EmploymentStatus`, `IncomeRange`, `ListingCategory`, `Loan Status`, etc.

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

I'm most interested in finding out the features related to the borrowers and the loans they took.

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

Of the many features available in the dataset, I expect the following to be of great help in my exploration; - **Term**: The length of the loan expressed in months - **LoanStatus**: The current status of the loan: Cancelled, Chargedoff, Completed, Current, Defaulted, FinalPaymentInProgress, PastDue. The PastDue status will be accompanied by a delinquency bucket. - **StatedMonthlyIncome**: The monthly income the borrower stated at the time the listing was created. - **BorrowerAPR**: The Borrower's Annual Percentage Rate (APR) for the loan. - **BorrowerRate**: The Borrower's interest rate for this loan. - **BorrowerState**: The two letter abbreviation of the state of the address of the borrower at the time the Listing was created. - **ProsperScore**: A custom risk score built using historical Prosper data. The score ranges from 1-10, with 10 being the best, or lowest risk score. Applicable for loans originated after July 2009. - **ListingCategory (numeric)**: The category of the listing that the borrower selected when posting their listing: 0 - Not Available, 1 - Debt Consolidation, 2 - Home Improvement, 3 - Business, 4 - Personal Loan, 5 - Student Use, 6 - Auto, 7- Other, 8 - Baby&Adoption, 9 - Boat, 10 - Cosmetic Procedure, 11 - Engagement Ring, 12 - Green Loans, 13 - Household Expenses, 14 - Large Purchases, 15 - Medical/Dental, 16 - Motorcycle, 17 - RV, 18 - Taxes, 19 - Vacation, 20 - Wedding Loans. - **EmploymentStatus**: The employment status of the borrower at the time they posted the listing. - **IsBorrowerHomeowner**: A Borrower will be classified as a homeowner if they have a mortgage on their credit profile or provide documentation confirming they are a homeowner. - **IncomeRange**: The income range of the borrower at the time the listing was created. - **Occupation**: The Occupation selected by the Borrower at the time they created the listing. - **DebtToIncomeRatio**: The debt to income ratio of the borrower at the time the credit profile was pulled. This value is Null if the debt to income ratio is not available. This value is capped at 10.01 (any debt to income ratio larger than 1000% will be returned as 1001%). - **TotalProsperLoans**: Number of Prosper loans the borrower at the time they created this listing. This value will be null if the borrower had no prior loans. - **LoanOriginalAmount**: The origination amount of the loan. - **Investors**: The number of investors that funded the loan. - **LoanOriginationDate**: The date the loan was originated.

### 1.3.4 Selecting the features of interest.

```
[11]: to_use = ['Term', 'LoanStatus', 'StatedMonthlyIncome', 'BorrowerAPR',  
             ↪ 'BorrowerRate', 'BorrowerState', 'ProsperScore',  
             'ListingCategory (numeric)', 'EmploymentStatus',  
             ↪ 'IsBorrowerHomeowner', 'IncomeRange', 'Occupation', 'DebtToIncomeRatio',  
             'TotalProsperLoans', 'LoanOriginalAmount', 'Investors',  
             ↪ 'LoanOriginationDate']
```

```
[12]: loans = df[to_use]  
loans.head()
```

```
[12]:   Term LoanStatus  StatedMonthlyIncome  BorrowerAPR  BorrowerRate  \  
0    36   Completed           3083.333333         0.16516         0.1580  
1    36    Current           6125.000000         0.12016         0.0920
```

2	36	Completed	2083.333333	0.28269	0.2750
3	36	Current	2875.000000	0.12528	0.0974
4	36	Current	9583.333333	0.24614	0.2085

	BorrowerState	ProsperScore	ListingCategory (numeric)	EmploymentStatus	\
0	CO	NaN		0	Self-employed
1	CO	7.0		2	Employed
2	GA	NaN		0	Not available
3	GA	9.0		16	Employed
4	MN	4.0		2	Employed

	IsBorrowerHomeowner	IncomeRange	Occupation	DebtToIncomeRatio	\
0	True	\$25,000-49,999	Other	0.17	
1	False	\$50,000-74,999	Professional	0.18	
2	False	Not displayed	Other	0.06	
3	True	\$25,000-49,999	Skilled Labor	0.15	
4	True	\$100,000+	Executive	0.26	

	TotalProsperLoans	LoanOriginalAmount	Investors	LoanOriginationDate
0	NaN	9425	258	2007-09-12 00:00:00
1	NaN	10000	1	2014-03-03 00:00:00
2	NaN	3001	41	2007-01-17 00:00:00
3	NaN	10000	158	2012-11-01 00:00:00
4	1.0	15000	20	2013-09-20 00:00:00

```
[13]: import warnings;warnings.filterwarnings('ignore')
# convert 'LoanOriginationDate' data type to datetime and extract year, month,
# and week from it
loans.LoanOriginationDate = pd.to_datetime(loans.LoanOriginationDate)
loans['Year'] = loans.LoanOriginationDate.dt.year
loans['Month'] = loans.LoanOriginationDate.dt.strftime('%B')
loans['Week'] = loans.LoanOriginationDate.dt.week

#convert the numeric values in the Listing Category to object and change the
#column data type.
mapping = {0: 'Not available', 1: 'Debt Consolidation', 2: 'Home Improvement',
3: 'Business', 4: 'Personal Loan', 5: 'Student Use',
6: 'Auto', 7: 'Other', 8: 'Baby&Adoption', 9: 'Boat', 10: 'Cosmetic
Procedure', 11: 'Engagement Ring', 12: 'Green Loans',
13: 'Household Expenses', 14: 'Large Purchases', 15: 'Medical/
Dental', 16: 'Motorcycle', 17: 'RV', 18: 'Taxes', 19: 'Vacation',
20: 'Wedding Loans'}

loans['ListingCategory (numeric)'] = loans['ListingCategory (numeric)'].
replace(mapping)
loans['ListingCategory (numeric)'] = loans['ListingCategory (numeric)'].
astype(object)
```



```
loans = loans.rename(columns={'ListingCategory (numeric)': 'Listing Category'})
```

```
[15]: loans.head(1)
```

```
[15]:   Term LoanStatus   StatedMonthlyIncome   BorrowerAPR   BorrowerRate \
0    36   Completed          3083.333333          0.16516          0.158

   BorrowerState   ProsperScore Listing Category EmploymentStatus \
0              CO             NaN   Not available   Self-employed

   IsBorrowerHomeowner   IncomeRange Occupation   DebtToIncomeRatio \
0                   True   $25,000-49,999      Other              0.17

   TotalProsperLoans   LoanOriginalAmount   Investors LoanOriginationDate   Year \
0                 NaN                9425          258      2007-09-12   2007

   Month   Week
0  September    37
```

```
[16]: print(loans.shape)
      print(loans.dtypes)
```

```
(113937, 20)
Term                int64
LoanStatus          object
StatedMonthlyIncome float64
BorrowerAPR         float64
BorrowerRate        float64
BorrowerState       object
ProsperScore         float64
Listing Category    object
EmploymentStatus    object
IsBorrowerHomeowner bool
IncomeRange         object
Occupation          object
DebtToIncomeRatio   float64
TotalProsperLoans   float64
LoanOriginalAmount  int64
Investors           int64
LoanOriginationDate datetime64[ns]
Year               int64
Month              object
Week              int64
dtype: object
```

```
[17]: loans.describe()
```

```
[17]:
```

	Term	StatedMonthlyIncome	BorrowerAPR	BorrowerRate \
count	113937.000000	1.139370e+05	113912.000000	113937.000000
mean	40.830248	5.608026e+03	0.218828	0.192764
std	10.436212	7.478497e+03	0.080364	0.074818
min	12.000000	0.000000e+00	0.006530	0.000000
25%	36.000000	3.200333e+03	0.156290	0.134000
50%	36.000000	4.666667e+03	0.209760	0.184000
75%	36.000000	6.825000e+03	0.283810	0.250000
max	60.000000	1.750003e+06	0.512290	0.497500

	ProsperScore	DebtToIncomeRatio	TotalProsperLoans	LoanOriginalAmount \
count	84853.000000	105383.000000	22085.000000	113937.000000
mean	5.950067	0.275947	1.421100	8337.01385
std	2.376501	0.551759	0.764042	6245.80058
min	1.000000	0.000000	0.000000	1000.00000
25%	4.000000	0.140000	1.000000	4000.00000
50%	6.000000	0.220000	1.000000	6500.00000
75%	8.000000	0.320000	2.000000	12000.00000
max	11.000000	10.010000	8.000000	35000.00000

	Investors	Year	Week
count	113937.000000	113937.000000	113937.000000
mean	80.475228	2011.042611	26.953860
std	103.239020	2.506634	15.520827
min	1.000000	2005.000000	1.000000
25%	2.000000	2008.000000	13.000000
50%	44.000000	2012.000000	28.000000
75%	115.000000	2013.000000	41.000000
max	1189.000000	2014.000000	53.000000

```
[18]: for i, col in enumerate(loans.columns):
        print('Value counts for {}: '.format(col))
        print(loans[col].value_counts())
        if i < len(loans.columns) - 1:
            print('-' * 100)
```

Value counts for Term:

```
36    87778
60    24545
12     1614
```

Name: Term, dtype: int64

Value counts for LoanStatus:

```
Current          56576
Completed        38074
Chargedoff       11992
Defaulted         5018
```

```
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
Cancelled                  5
```

```
Name: LoanStatus, dtype: int64
```

---

```
Value counts for StatedMonthlyIncome:
```

```
4166.666667      3526
5000.000000      3389
3333.333333      2917
3750.000000      2428
5416.666667      2374
```

```
...
```

```
7069.916667       1
4266.333333       1
2211.750000       1
7032.916667       1
18756.000000       1
```

```
Name: StatedMonthlyIncome, Length: 13502, dtype: int64
```

---

```
Value counts for BorrowerAPR:
```

```
0.35797      3672
0.35643      1644
0.37453      1260
0.30532       902
0.29510       747
```

```
...
```

```
0.37266       1
0.27518       1
0.18477       1
0.29961       1
0.19543       1
```

```
Name: BorrowerAPR, Length: 6677, dtype: int64
```

---

```
Value counts for BorrowerRate:
```

```
0.3177      3672
0.3500      1905
0.3199      1651
0.2900      1508
0.2699      1319
```

```
...
```

```
0.2201      1
0.0752      1
0.1416      1
0.2812      1
0.0739      1
Name: BorrowerRate, Length: 2294, dtype: int64
```

```
-----
Value counts for BorrowerState:
```

```
CA      14717
TX       6842
NY       6729
FL       6720
IL       5921
GA       5008
OH       4197
MI       3593
VA       3278
NJ       3097
NC       3084
WA       3048
PA       2972
MD       2821
MO       2615
MN       2318
MA       2242
CO       2210
IN       2078
AZ       1901
WI       1842
OR       1817
TN       1737
AL       1679
CT       1627
SC       1122
NV       1090
KS       1062
KY        983
OK        971
LA        954
UT        877
AR        855
MS        787
NE        674
ID        599
NH        551
NM        472
RI        435
```

HI	409
WV	391
DC	382
MT	330
DE	300
VT	207
AK	200
SD	189
IA	186
WY	150
ME	101
ND	52

Name: BorrowerState, dtype: int64

Value counts for ProsperScore:

4.0	12595
6.0	12278
8.0	12053
7.0	10597
5.0	9813
3.0	7642
9.0	6911
2.0	5766
10.0	4750
11.0	1456
1.0	992

Name: ProsperScore, dtype: int64

Value counts for Listing Category:

Debt Consolidation	58308
Not available	16965
Other	10494
Home Improvement	7433
Business	7189
Auto	2572
Personal Loan	2395
Household Expenses	1996
Medical/Dental	1522
Taxes	885
Large Purchases	876
Wedding Loans	771
Vacation	768
Student Use	756
Motorcycle	304
Engagement Ring	217
Baby&Adoption	199

Cosmetic Procedure	91
Boat	85
Green Loans	59
RV	52

Name: Listing Category, dtype: int64

-----

Value counts for EmploymentStatus:

Employed	67322
Full-time	26355
Self-employed	6134
Not available	5347
Other	3806
Part-time	1088
Not employed	835
Retired	795

Name: EmploymentStatus, dtype: int64

-----

Value counts for IsBorrowerHomeowner:

True	57478
False	56459

Name: IsBorrowerHomeowner, dtype: int64

-----

Value counts for IncomeRange:

\$25,000-49,999	32192
\$50,000-74,999	31050
\$100,000+	17337
\$75,000-99,999	16916
Not displayed	7741
\$1-24,999	7274
Not employed	806
\$0	621

Name: IncomeRange, dtype: int64

-----

Value counts for Occupation:

Other	28617
Professional	13628
Computer Programmer	4478
Executive	4311
Teacher	3759
...	
Dentist	68
Student - College Freshman	41
Student - Community College	28
Judge	22

Student - Technical School 16  
Name: Occupation, Length: 67, dtype: int64

---

Value counts for DebtToIncomeRatio:

0.18000	4132
0.22000	3687
0.17000	3616
0.14000	3553
0.20000	3481

...	
0.06375	1
0.06281	1
0.19960	1
0.17775	1
0.23284	1

Name: DebtToIncomeRatio, Length: 1207, dtype: int64

---

Value counts for TotalProsperLoans:

1.0	15538
2.0	4540
3.0	1447
4.0	417
5.0	104
6.0	29
7.0	8
8.0	1
0.0	1

Name: TotalProsperLoans, dtype: int64

---

Value counts for LoanOriginalAmount:

4000	14333
15000	12407
10000	11106
5000	6990
2000	6067

...	
5284	1
7936	1
1201	1
10593	1
4292	1

Name: LoanOriginalAmount, Length: 2468, dtype: int64

---

Value counts for Investors:

```
1      27814
2      1386
3       991
4       827
5       753
```

```
...
665      1
634      1
555      1
752      1
754      1
```

Name: Investors, Length: 751, dtype: int64

---

Value counts for LoanOriginationDate:

```
2014-01-22    491
2013-11-13    490
2014-02-19    439
2013-10-16    434
2014-01-28    339
```

```
...
2006-02-03      1
2006-01-24      1
2005-11-18      1
2009-07-20      1
2005-11-15      1
```

Name: LoanOriginationDate, Length: 1873, dtype: int64

---

Value counts for Year:

```
2013    34345
2012    19553
2014    12172
2008    11552
2007    11460
2011    11228
2006     5906
2010     5652
2009     2047
2005         22
```

Name: Year, dtype: int64

---

Value counts for Month:

```
January    11395
October    11043
December   10708
February    9728
```



November	9635
August	9592
September	9221
July	9154
June	8847
March	8555
May	8500
April	7559

Name: Month, dtype: int64

-----

Value counts for Week:

39	3145
4	2813
51	2801
9	2777
50	2664
3	2599
10	2598
43	2592
35	2591
2	2578
5	2532
42	2468
30	2454
26	2452
8	2441
48	2409
41	2404
7	2399
46	2379
44	2372
38	2294
49	2269
45	2247
34	2240
40	2232
25	2210
29	2201
31	2180
21	2178
52	2156
6	2151
17	2096
11	2088
33	2033
47	2032
20	2013

```

23    1968
24    1959
32    1911
28    1864
19    1847
37    1842
13    1812
16    1804
1     1750
18    1701
12    1694
22    1645
15    1615
36    1467
14    1428
27    1373
53     169
Name: Week, dtype: int64

```

```

[19]: import warnings;warnings.simplefilter(action='ignore')

# convert LoanStatus, EmploymentStatus, and IncomeRange into ordered
# categorical types
ordinal_var_dict = {
    'LoanStatus': ['Cancelled', 'Defaulted', 'Past Due (>120
    days)', 'Past Due (91-120 days)', 'Past Due (61-90 days)',
    'Past Due (31-60 days)', 'Past Due (16-30
    days)', 'Past Due (1-15 days)', 'Chargedoff',
    'Current', 'FinalPaymentInProgress',
    'Completed'],
    'EmploymentStatus': ['Not available', 'Not employed',
    'Self-employed', 'Part-time',
    'Employed', 'Full-time', 'Retired',
    'Other'],
    'IncomeRange': ['Not employed', 'Not displayed', '$0',
    '$1-24,999', '$25,000-49,999', '$50,000-74,999',
    '$75,000-99,999', '$100,000+']
}

for var in ordinal_var_dict:
    ordered_var = pd.api.types.CategoricalDtype(ordered = True, categories =
    ordinal_var_dict[var])
    loans[var] = loans[var].astype(ordered_var)

[20]: cats = loans[ordinal_var_dict]
for i, col in enumerate(cats.columns):
    print('Unique values for {}: '.format(col))

```

```
print(cats[col].unique())
if i < len(cats.columns) - 1:
    print('-' * 328)
```

Unique values for LoanStatus:

```
['Completed', 'Current', 'Past Due (1-15 days)', 'Defaulted', 'Chargedoff', ...,
'Past Due (61-90 days)', 'Past Due (31-60 days)', 'Past Due (91-120 days)',
'FinalPaymentInProgress', 'Past Due (>120 days)']
```

Length: 12

Categories (12, object): ['Cancelled' < 'Defaulted' < 'Past Due (>120 days)' < 'Past Due (91-120 days)' ... 'Chargedoff' < 'Current' < 'FinalPaymentInProgress' < 'Completed']

Unique values for EmploymentStatus:

```
['Self-employed', 'Employed', 'Not available', 'Full-time', 'Other', NaN, 'Not
employed', 'Part-time', 'Retired']
```

Categories (8, object): ['Not available' < 'Not employed' < 'Self-employed' < 'Part-time' < 'Employed' < 'Full-time' < 'Retired' < 'Other']

Unique values for IncomeRange:

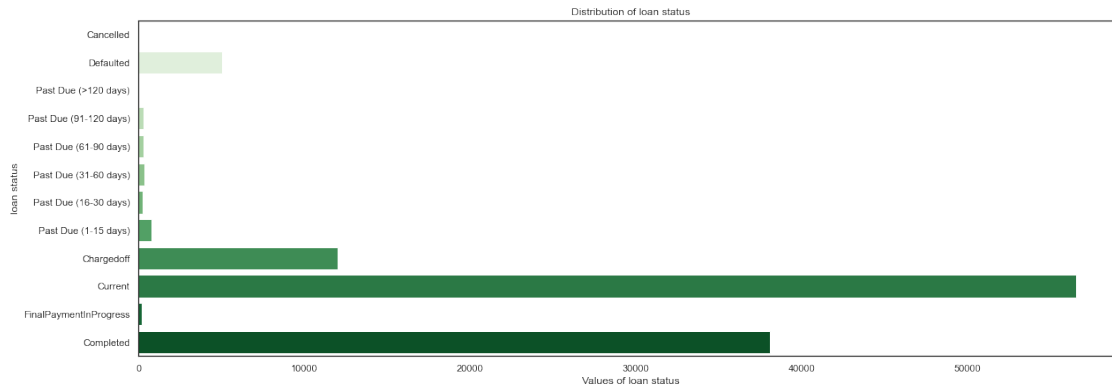
```
['$25,000-49,999', '$50,000-74,999', 'Not displayed', '$100,000+',
'$75,000-99,999', '$1-24,999', 'Not employed', '$0']
```

Categories (8, object): ['Not employed' < 'Not displayed' < '\$0' < '\$1-24,999' < '\$25,000-49,999' < '\$50,000-74,999' < '\$75,000-99,999' < '\$100,000+']

## 1.4 Univariate Exploration

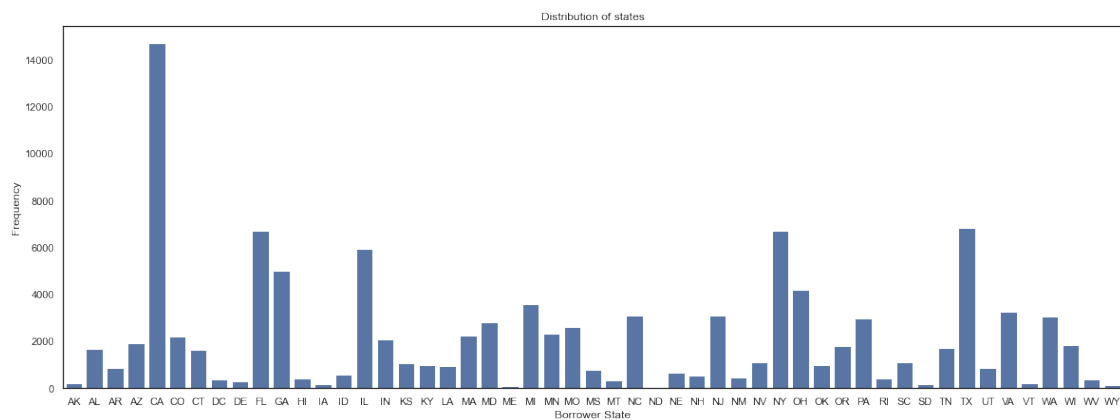
Exploring the distributions of individual features.

```
[21]: # exploring the statuses of loans that were took.
plot = loans.LoanStatus.value_counts()
x = plot.index
y = plot.values
plt.figure(figsize=[20,7])
sns.barplot(y=plot.index, x=plot.values, palette='Greens').set(xlabel='Values_
↳of loan status',
                                                                    ylabel='loan_
↳status', title='Distribution of loan status')
plt.show();
```



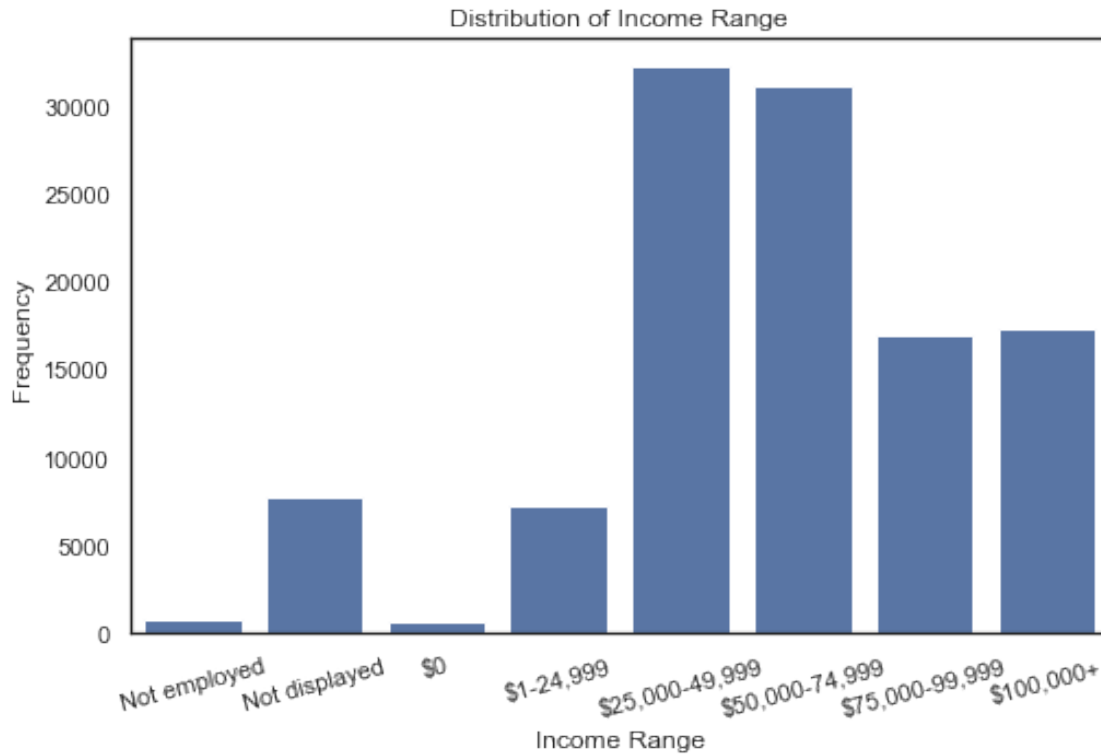
The most common loan status amongst the borrowers is **Current**.

```
[22]: # exploring the distribution of states.
plot = loans.BorrowerState
plot = plot.sort_values(ascending=True)
color = sns.color_palette()[0]
plt.figure(figsize=[20,7])
sns.countplot(plot, color=color).set(xlabel='Borrower State',
    ↳ylabel='Frequency', title='Distribution of states');
```



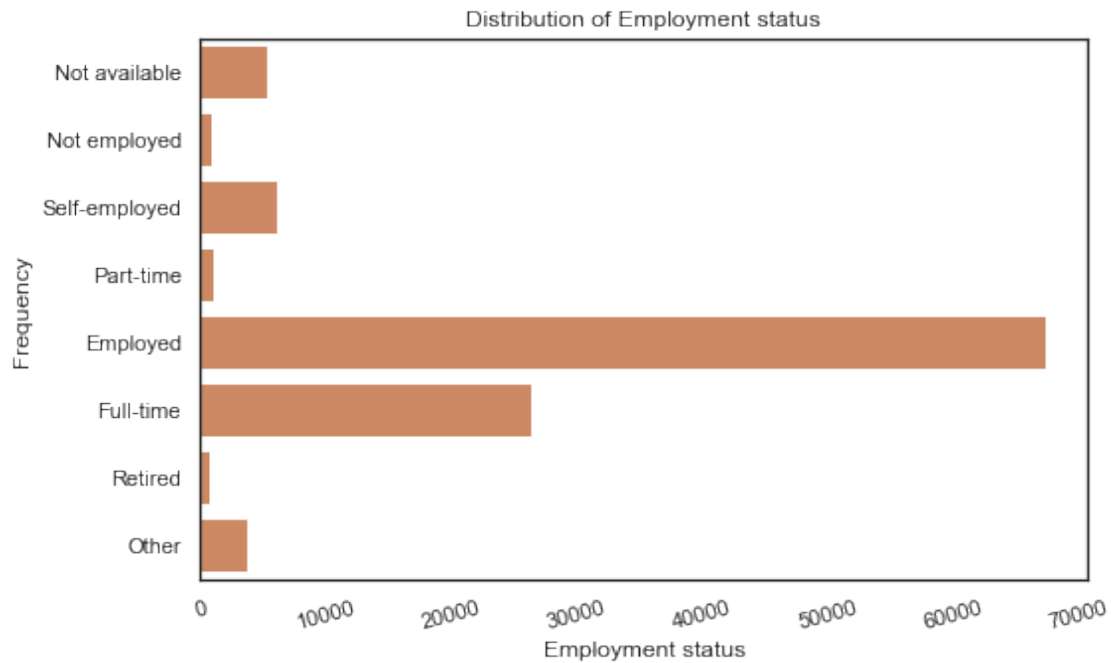
Majority of the borrowers originated from **California (CA)**

```
[24]: # exploring the Income range of the borrowers
plot = loans.IncomeRange
color = sns.color_palette()[0]
plt.figure(figsize=[8,5])
sns.countplot(plot, color=color).set(xlabel='Income Range', ylabel='Frequency',
    ↳title='Distribution of Income Range')
plt.xticks(rotation=15);
```



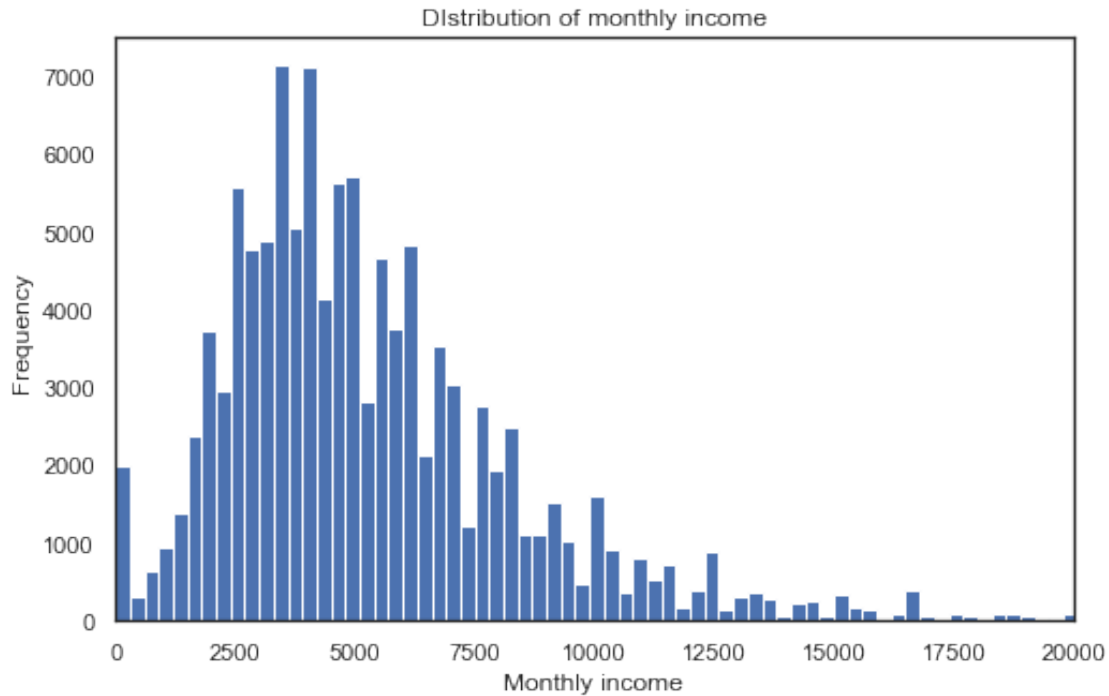
Majority of the borrowers had an income range between \$25,000-\$50,000.

```
[27]: # employment status of the borrowers
plot = loans.EmploymentStatus.value_counts()
x = plot.values
y = plot.index
plt.figure(figsize=[8,5])
color = sns.color_palette()[1]
sns.barplot(x=x, y=y, color=color).set(xlabel='Employment status',
                                       ylabel='Frequency', title='Distribution_
of Employment status')
plt.xticks(rotation=15);
```



The majority of the borrowers are **Employed and Full-time**

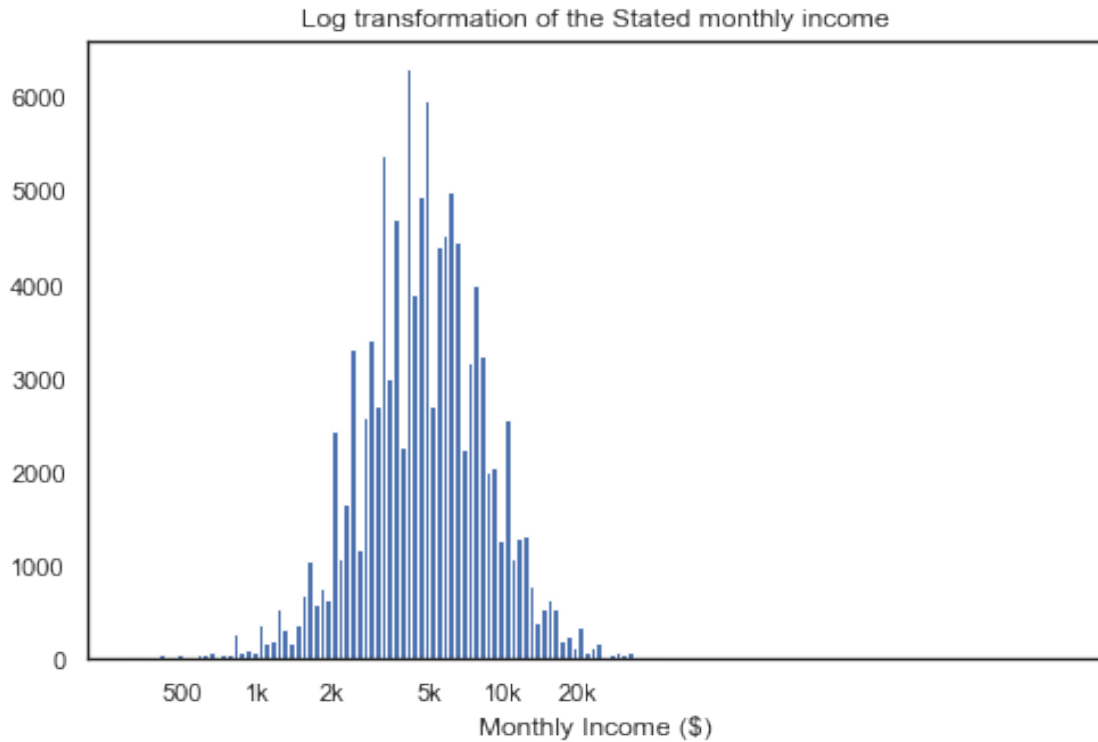
```
[28]: # distribution of monthly income plotted with more bins
bins = np.arange(0, loans.StatedMonthlyIncome.max()+300, 300)
plt.figure(figsize=[8,5])
plt.hist(x=loans.StatedMonthlyIncome, bins=bins)
plt.title('DIstribution of monthly income')
plt.xlabel('Monthly income')
plt.ylabel('Frequency')
plt.xlim(0, 20000);
```



The monthly income of the borrowers is positively skewed with majority of the borrowers having an income less than \$5,000

```
[34]: # using a log transformation for the stated monthly income viz
log_binsize = 0.025
bins = 10 ** np.arange(2.5, np.log10(loans['StatedMonthlyIncome'].
    ↪max())+log_binsize, log_binsize)

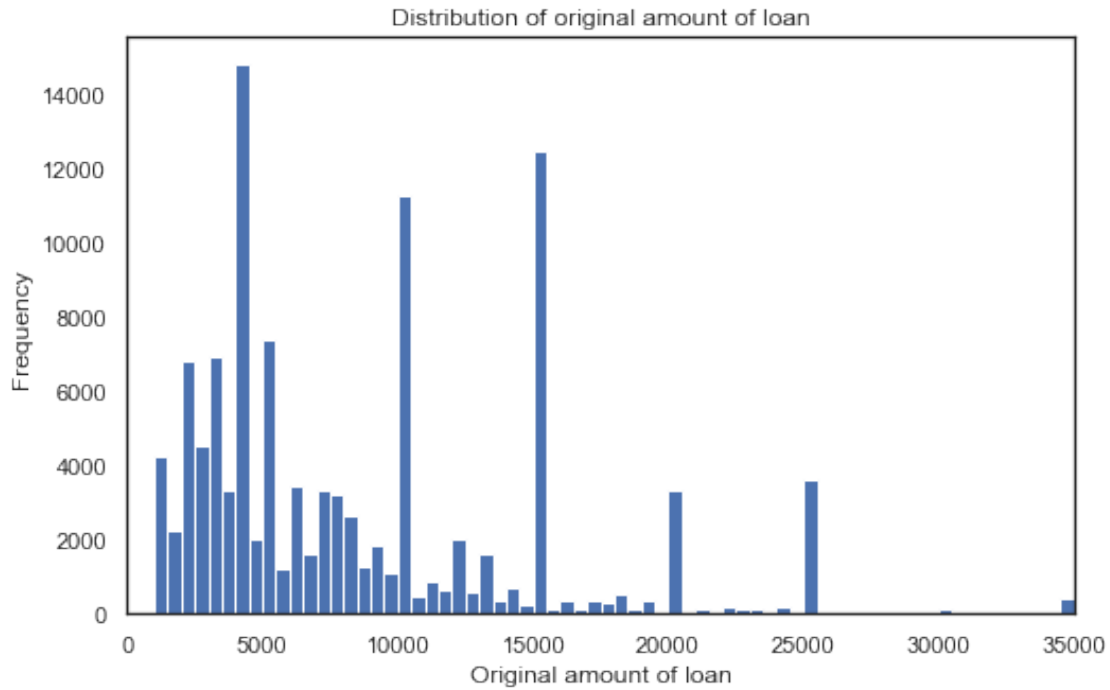
plt.figure(figsize=[8,5])
plt.hist(data = loans, x = 'StatedMonthlyIncome', bins = bins)
plt.xscale('log')
plt.xticks([500, 1e3, 2e3, 5e3, 1e4, 2e4], [500, '1k', '2k', '5k', '10k', '20k'])
plt.title('Log transformation of the Stated monthly income')
plt.xlabel('Monthly Income ($)')
plt.show();
```



Became much more skewed to the right

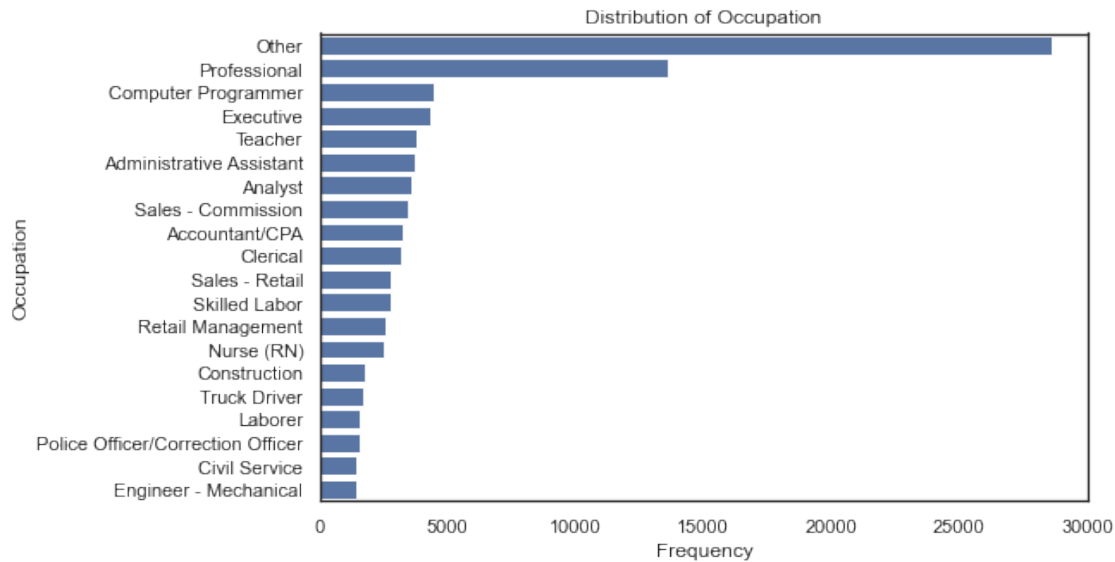
```
[35]: # Distribution of the original amount of loan taken
plot = loans.LoanOriginalAmount
bins = np.arange(0, loans.LoanOriginalAmount.max()+500, 500)
plt.figure(figsize=[8,5])
plt.hist(plot, bins=bins)
plt.title('Distribution of original amount of loan')
plt.xlabel('Original amount of loan')
plt.ylabel('Frequency')
plt.xlim([0, 35000]);
```





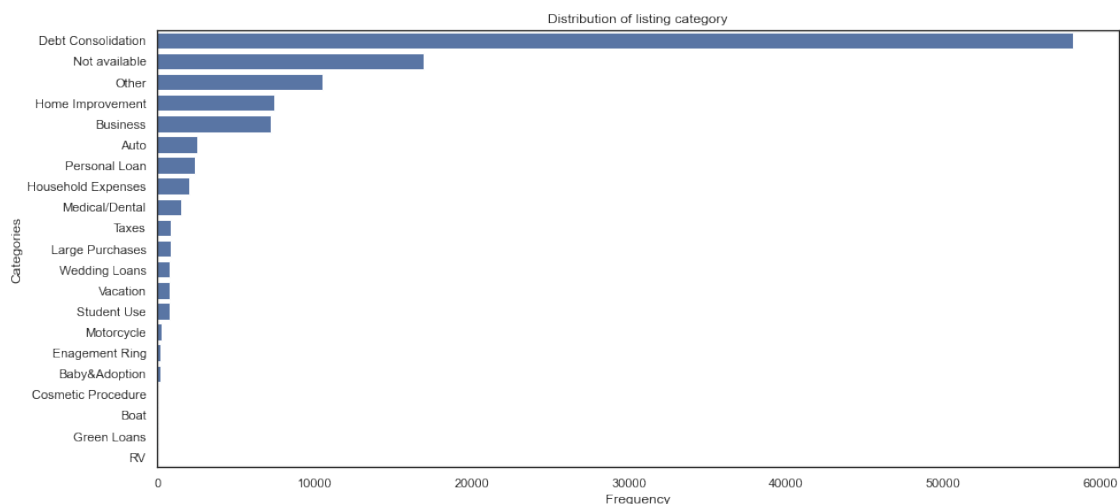
Most borrowers took about \$5,000 as loan.

```
[39]: # occupation of borrowers
plot = loans.Occupation.value_counts()[:20]
plt.figure(figsize=[8,5])
x = plot.index
y = plot.values
color = sns.color_palette()[0]
sns.barplot(x=y, y=x, order=x, color=color).set(title='Distribution of
↳Occupation', xlabel='Frequency', ylabel='Occupation')
plt.show();
```



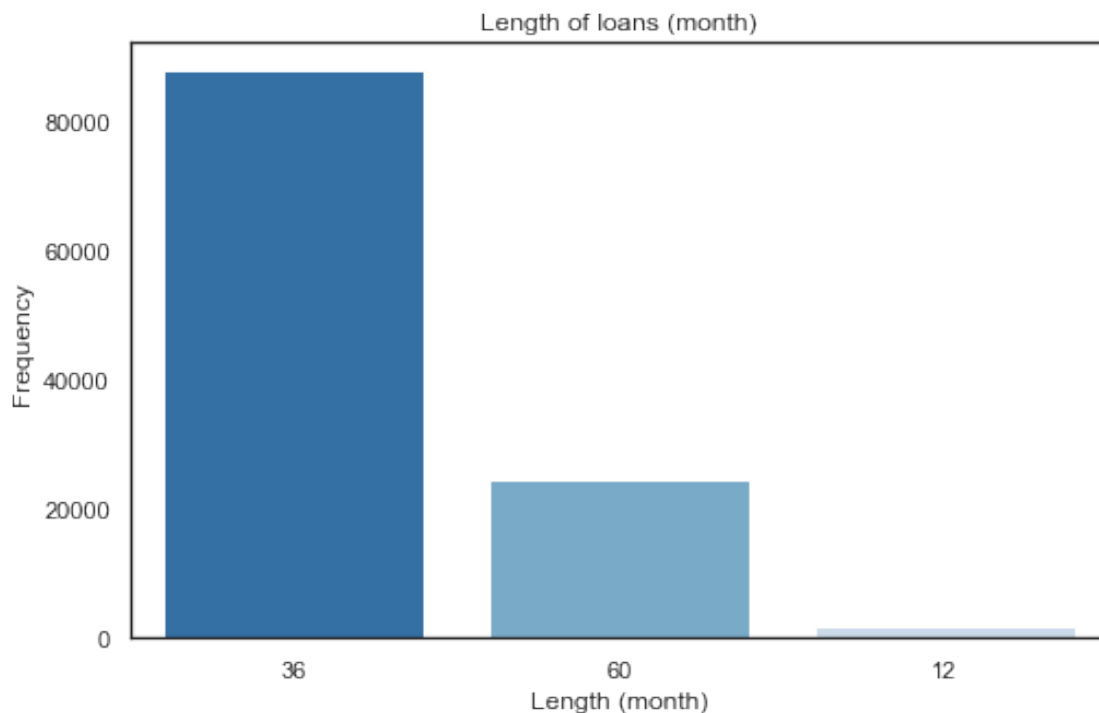
Most borrowers had other occupations apart from the ones provided in the listing. Professionals are also common among the borrowers

```
[40]: # listing category
plot = loans['Listing Category'].value_counts()
x = plot.index
y = plot.values
color = sns.color_palette()[0]
plt.figure(figsize=[15,7])
sns.barplot(x=y, y=x, color=color, order=x).set(
    title='Distribution of listing category', xlabel='Frequency',
    ylabel='Categories')
plt.show();
```



A vast majority applied for debt consolidation loans while 16965 borrowers didn't have their listing available.

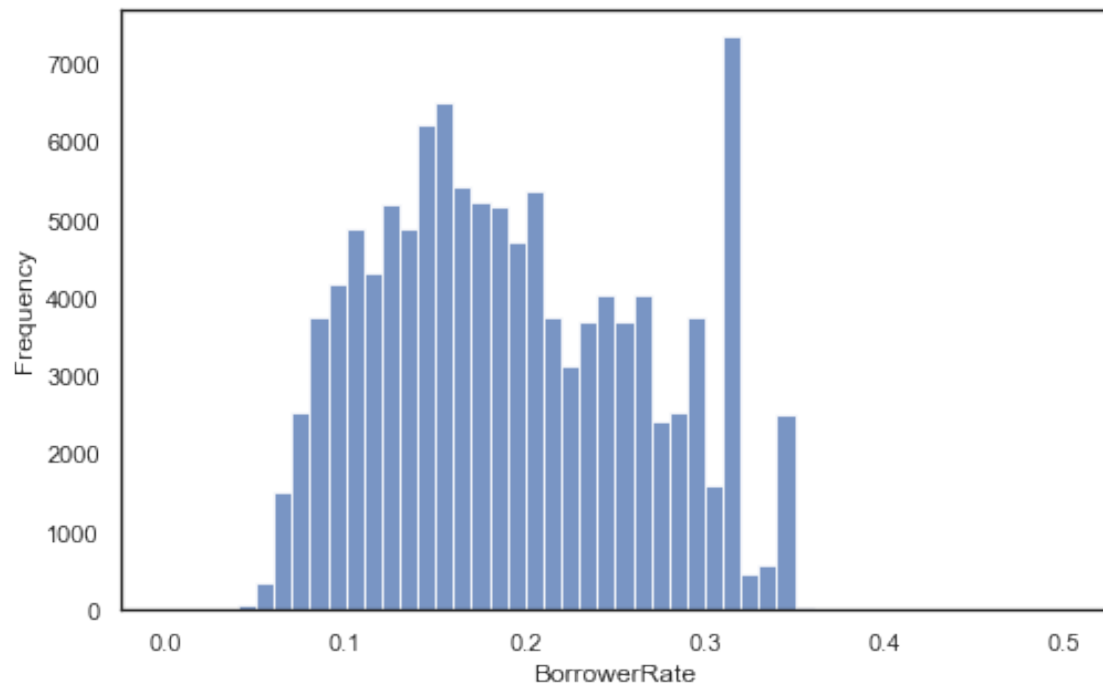
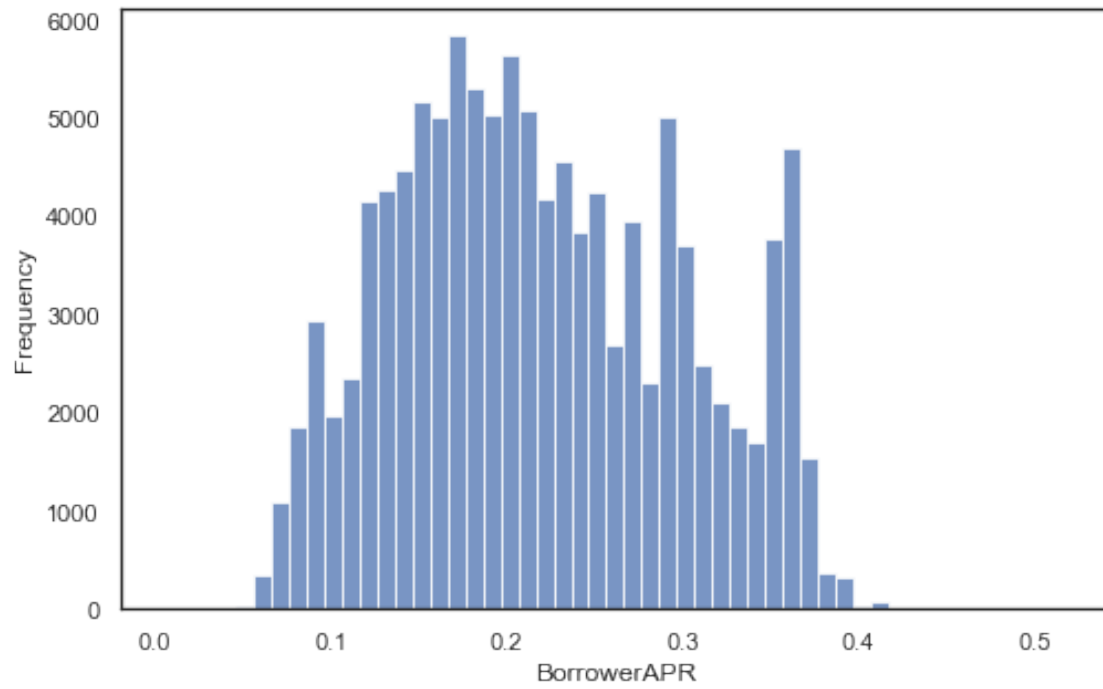
```
[42]: # term distribution
plot = loans.Term
plt.figure(figsize=[8,5])
sns.countplot(plot, order=loans.Term.value_counts().index, palette='Blues_r').
    ↪set(title='Length of loans (month)',
                                              xlabel='Length_
    ↪(month)', ylabel='Frequency')
plt.show();
```



A large bloc of borrowers applied for a loan for 36 months while a very small number applied for the shortest loan term -12 months.

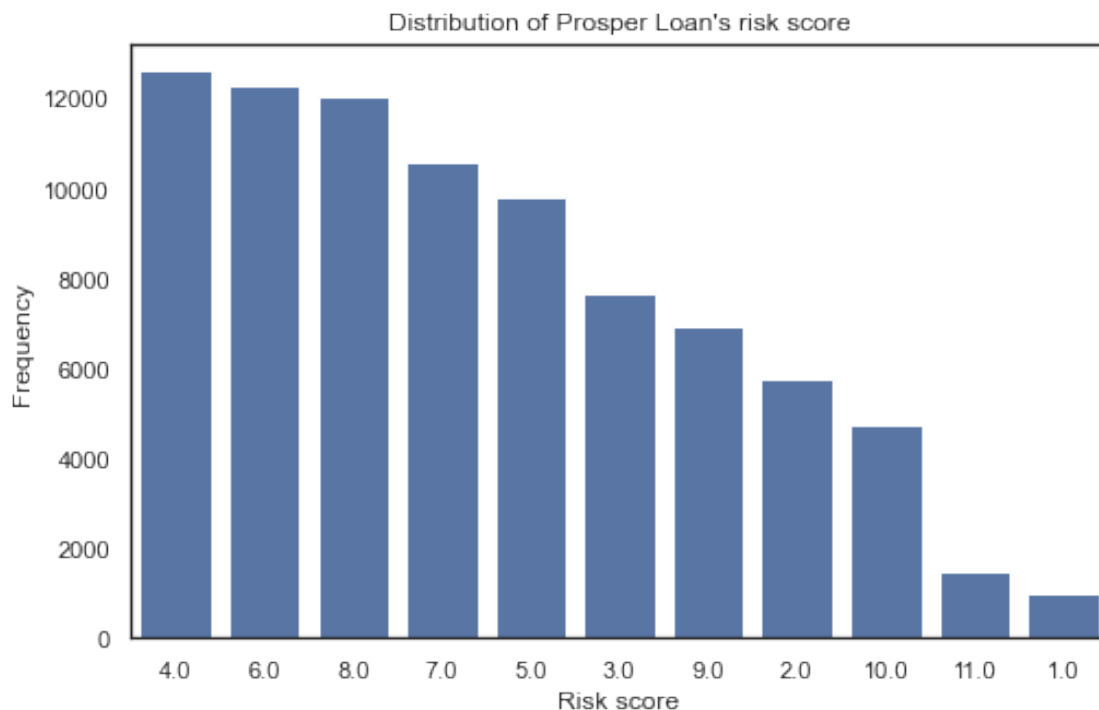
```
[45]: # borrower APR and Interest Rate
review = loans[['BorrowerAPR', 'BorrowerRate']]
for i in review:
    plt.figure(figsize=[8,5])
    bins = np.arange(min(loans[i].values), max(loans[i].values)+0.01, 0.01)
    sns.histplot(review[i], bins=bins).set(xlabel='{}'.format(i),
    ↪ylabel='Frequency')
```

```
plt.show();
```



The Borrowers's APR and Interest rates distributions are very similar which may indicate that the two features are highly correlated.

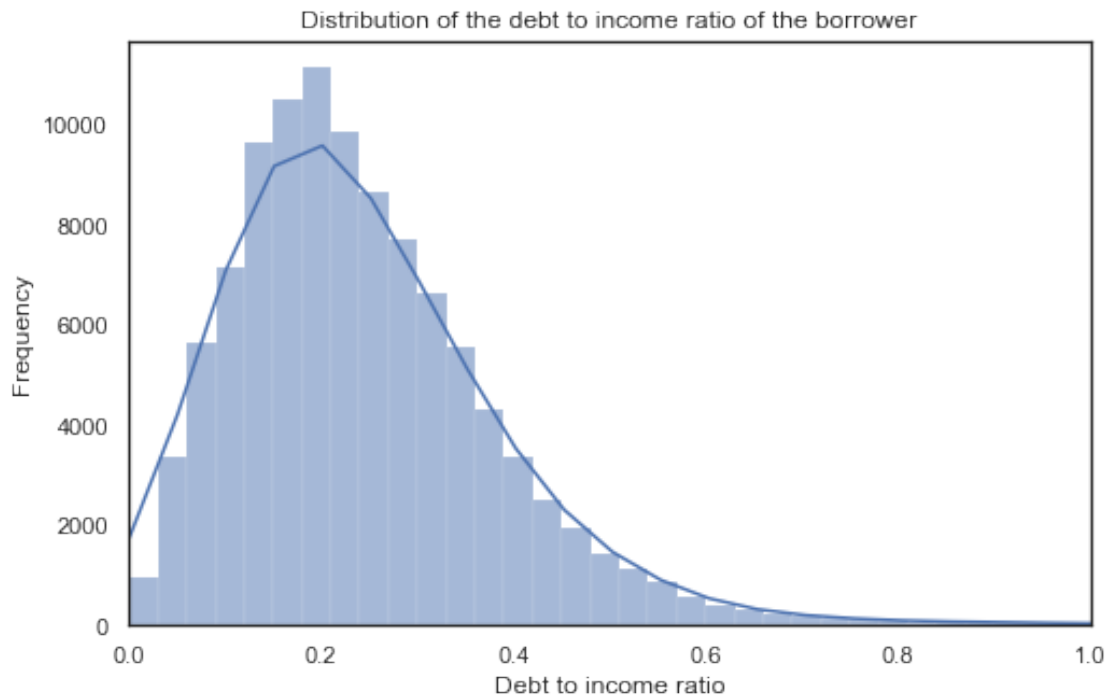
```
[46]: # prosper loan risk score
plot = loans.ProspersScore
color = sns.color_palette()[0]
plt.figure(figsize=[8,5])
sns.countplot(plot, order=plot.value_counts().index, color=color).set(
    title='Distribution of Prosper Loan\'s risk score', xlabel='Risk score',
    ylabel='Frequency')
plt.show();
```



Most borrowers have a risk score of 4.0 which is very close to bad while about 5000 borrowers had a perfect risk score.

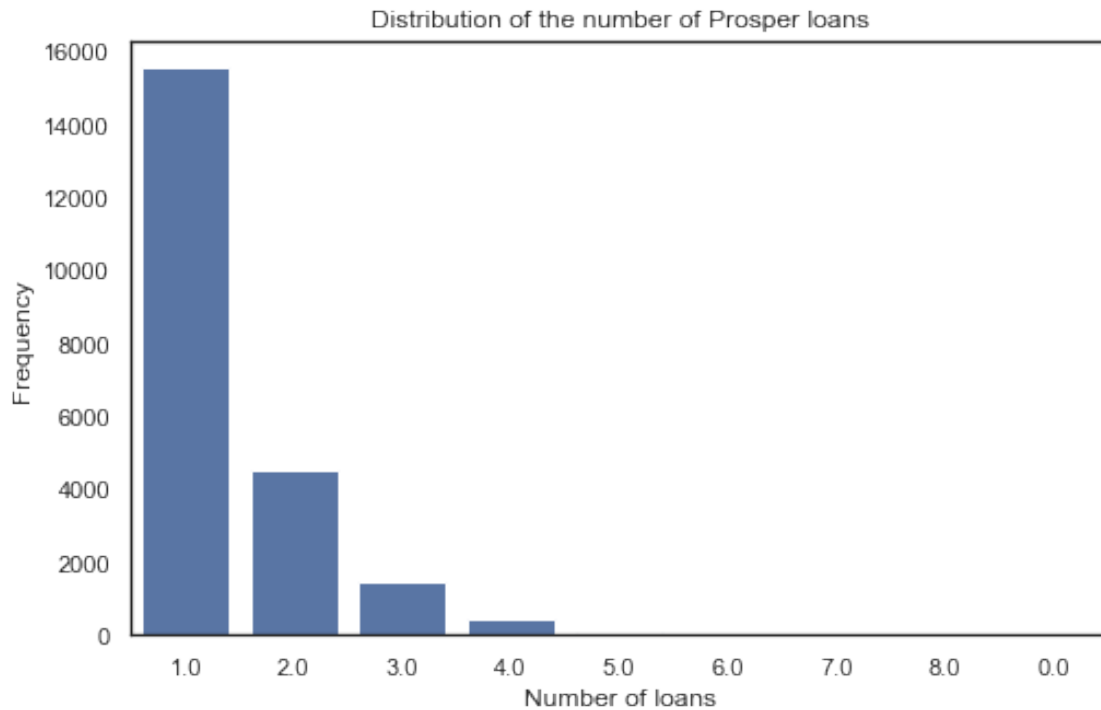
```
[48]: # debt to income ratio
bins = (np.arange(0, loans.DebtToIncomeRatio.max()+0.03, 0.03))
plt.figure(figsize=[8,5])
color = sns.color_palette()[0]
sns.histplot(data= loans, x='DebtToIncomeRatio', bins=bins, kde=True,
    color=color).set(
    title='Distribution of the debt to income ratio of the borrower', xlabel='Debt_
    to income ratio', ylabel='Frequency')
plt.xlim([0, 1])
```

```
plt.show();
```



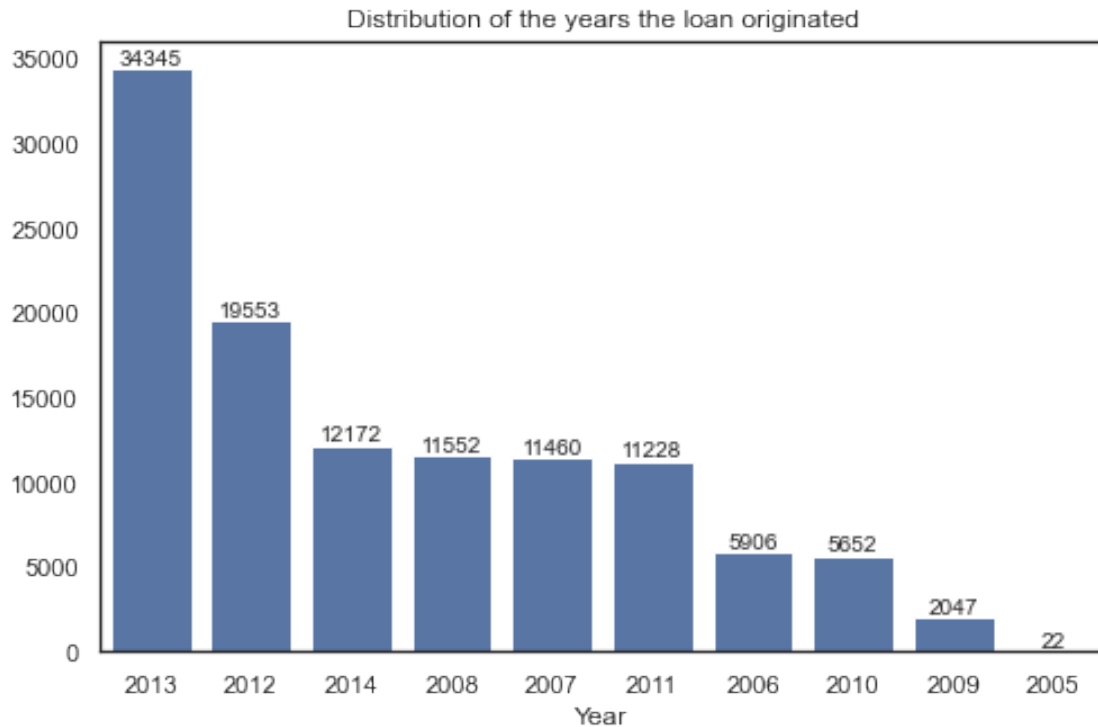
The debt to income ratio distribution is highly skewed to the right and hence highly positively skewed

```
[49]: # number of prosper loans
plot = loans.TotalProsperLoans
color = sns.color_palette()[0]
plt.figure(figsize=[8,5])
sns.countplot(plot, order=plot.value_counts().index, color=color).set(
    title='Distribution of the number of Prosper loans', xlabel='Number of ↵
    ↵loans', ylabel='Frequency')
plt.show();
```



A large bloc of the borrowers had only one prosper loan and a single borrower took the highest loan -8.

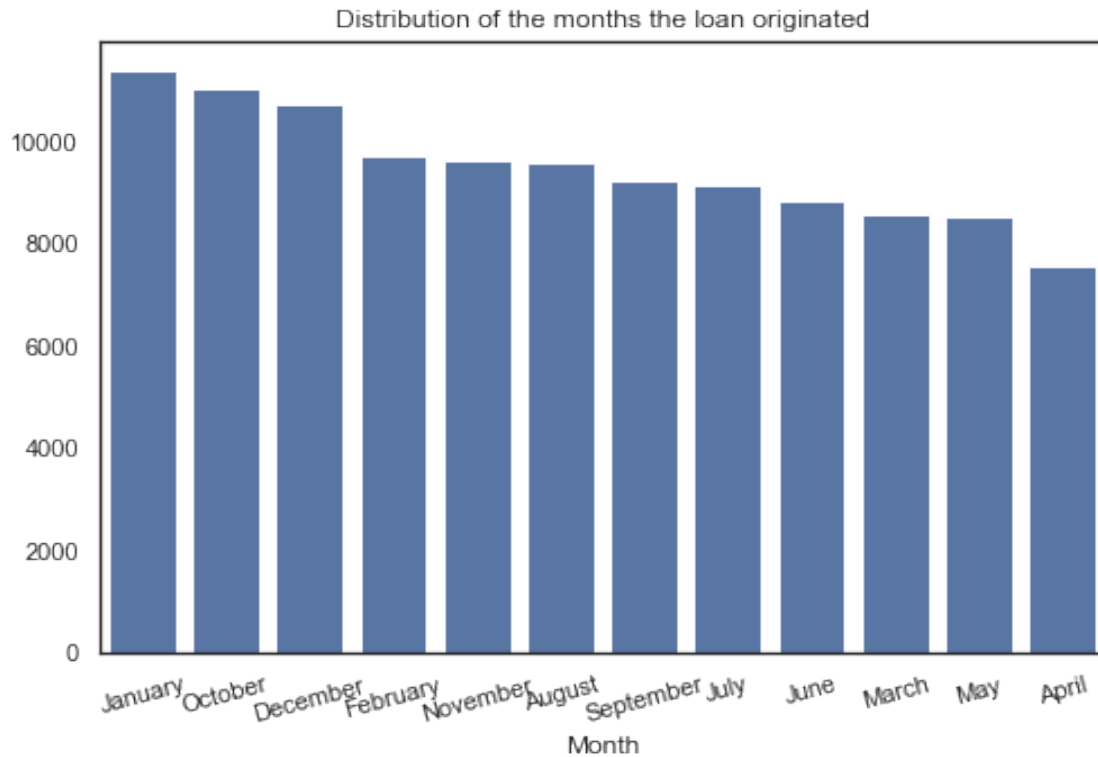
```
[51]: # year loans originated
plot = loans.Year.value_counts()
x = plot.index
y = plot.values
color = sns.color_palette()[0]
plt.figure(figsize=[8,5])
sns.barplot(x=x, y=y, color=color, order=x).set(
    title='Distribution of the years the loan originated', xlabel='Year')
for i, v in enumerate(plot):
    plt.annotate(str(v), xy=(i, v), ha='center', va='bottom')
plt.show();
```



About 34,500 loans originated in the year 2013 which is the highest of any year while the fewest occurred in 2005.

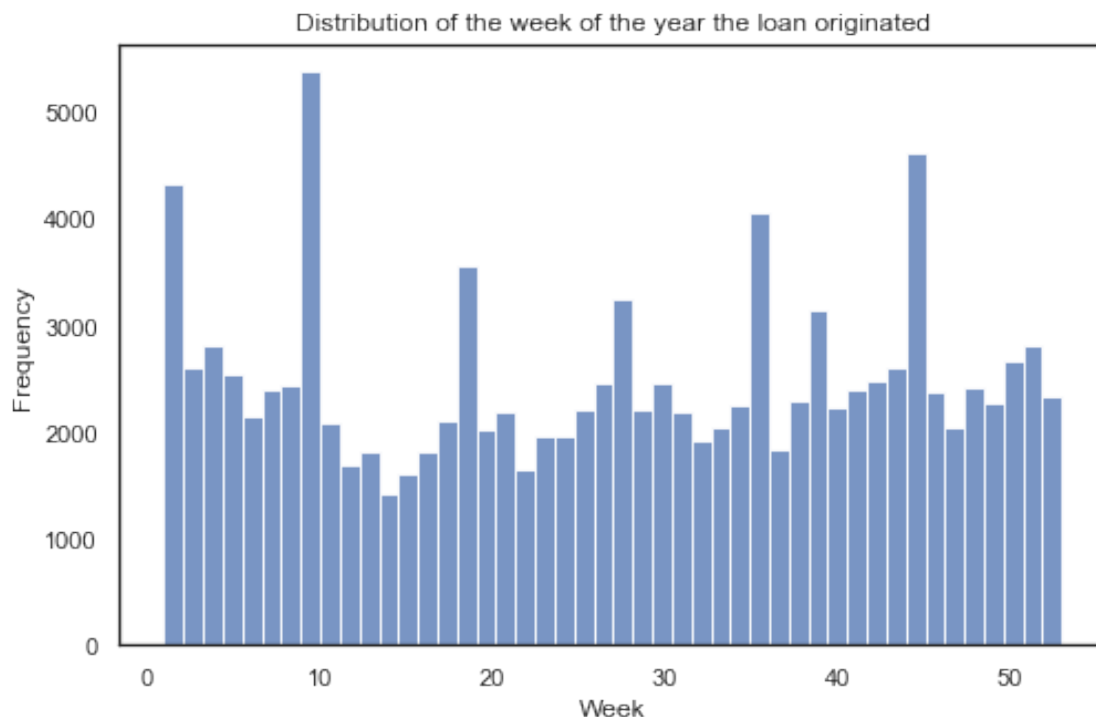
```
[52]: # months loans originated
plot = loans.Month.value_counts()
x = plot.index
y = plot.values
color = sns.color_palette()[0]
plt.figure(figsize=[8,5])
sns.barplot(x=x, y=y, color=color, order=x).set(
    title='Distribution of the months the loan originated', xlabel='Month')
plt.xticks(rotation=15)
plt.show();
```





Most borrowers took a loan at the beginning of the year while fewer borrowers took loans around the spring.

```
[55]: # week of loan origin
plot = loans.Week
color = sns.color_palette()[0]
bins = np.arange(0, loans.Week.max()+2, 1)
plt.figure(figsize=[8,5])
sns.histplot(plot).set(
    title='Distribution of the week of the year the loan originated',
    xlabel='Week', ylabel='Frequency')
plt.show();
```



Most borrowers took loans in the 10th week of the year

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

Yeah. I had to make some transformations. The `LoanOriginationDate` variable had to be transformed into years, months and weeks of the year to enable me gain more insights into the time people started taking Prosper loans. Apart from others that didn't have their occupation listed, professionals were the most common borrowers at Prosper loans. Majority of the borrowers have their loan status at current and are employed or either Full-time. The term of most loans was pegged at 36 months while a small minority took a loan of a year. A large faction of borrowers originated from California and took their loans at the beginning of the year with the most loans also originating from 2013. The listing category of a majority of the loans was for Debt consolidation purposes. Again, most borrowers have a not so good risk score of 4.0.

#### 1.4.2 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?

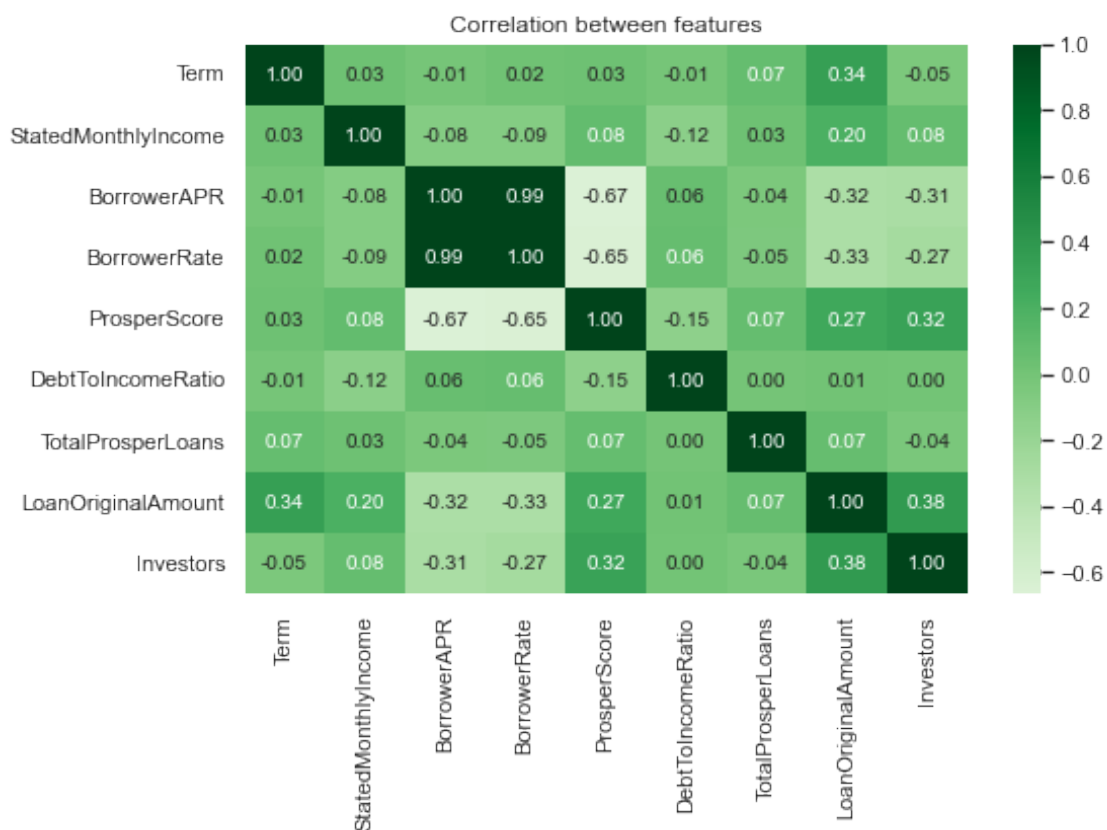
The `LoanOriginationDate` variable was originally recorded as an object type variable instead of a datetime. Again, the `Listing` category variable was an integer which I also had to map onto the alpha equivalent which I sourced from the internet. I also converted the data type to an object data type.

## 1.5 Bivariate Exploration

In this section, I investigated relationships between pairs of variables in the data.

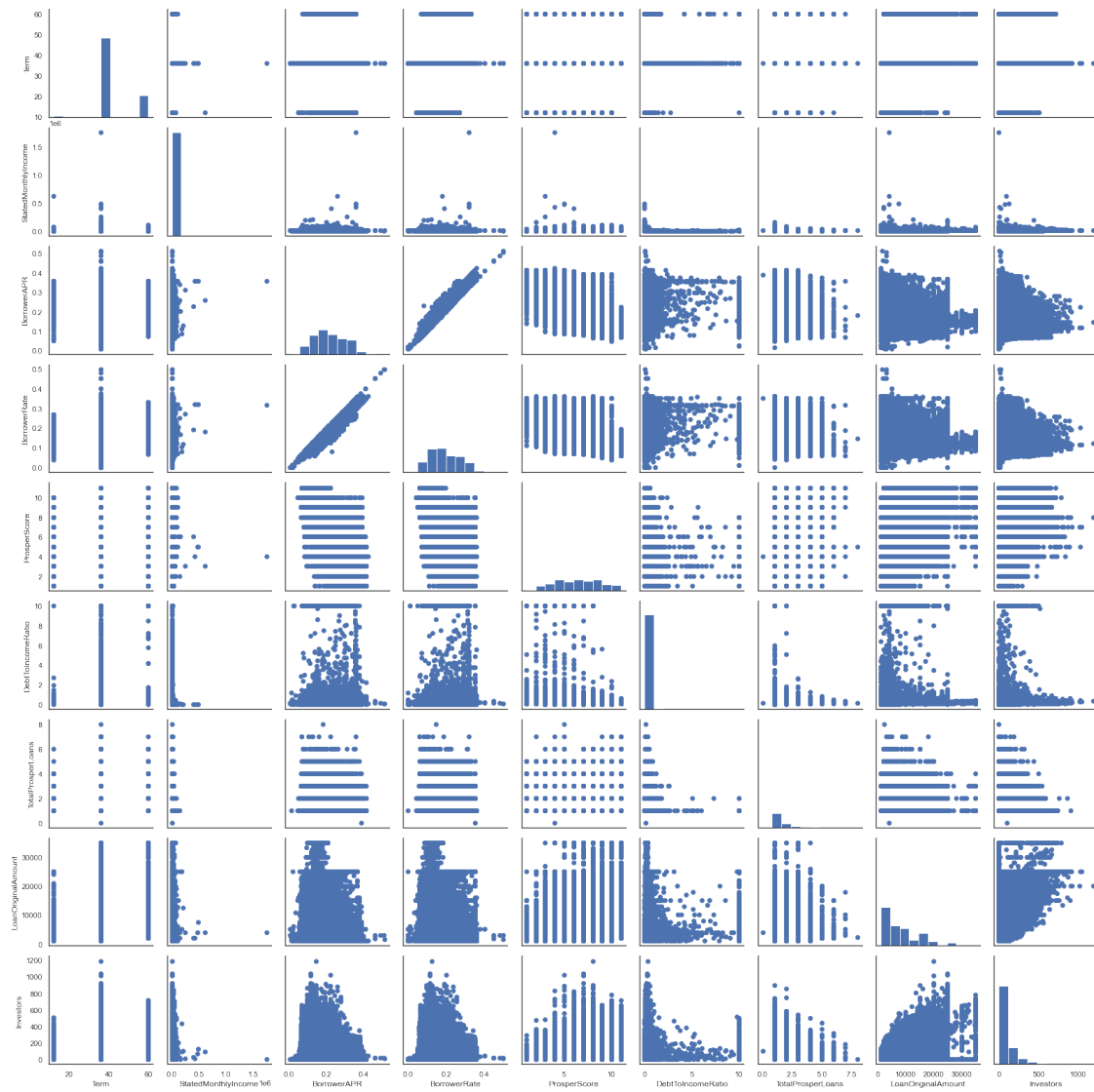
```
[58]: # correlation matrix of numeric variables
numeric_vars = ['Term', 'StatedMonthlyIncome', 'BorrowerAPR', 'BorrowerRate',
                'ProsperScore',
                'DebtToIncomeRatio', 'TotalProsperLoans', 'LoanOriginalAmount',
                'Investors']

plt.figure(figsize=[8,5])
sns.heatmap(loans[numeric_vars].corr(), fmt='.2f', annot=True, cmap='Greens',
            center=0).set(title='Correlation between features');
```

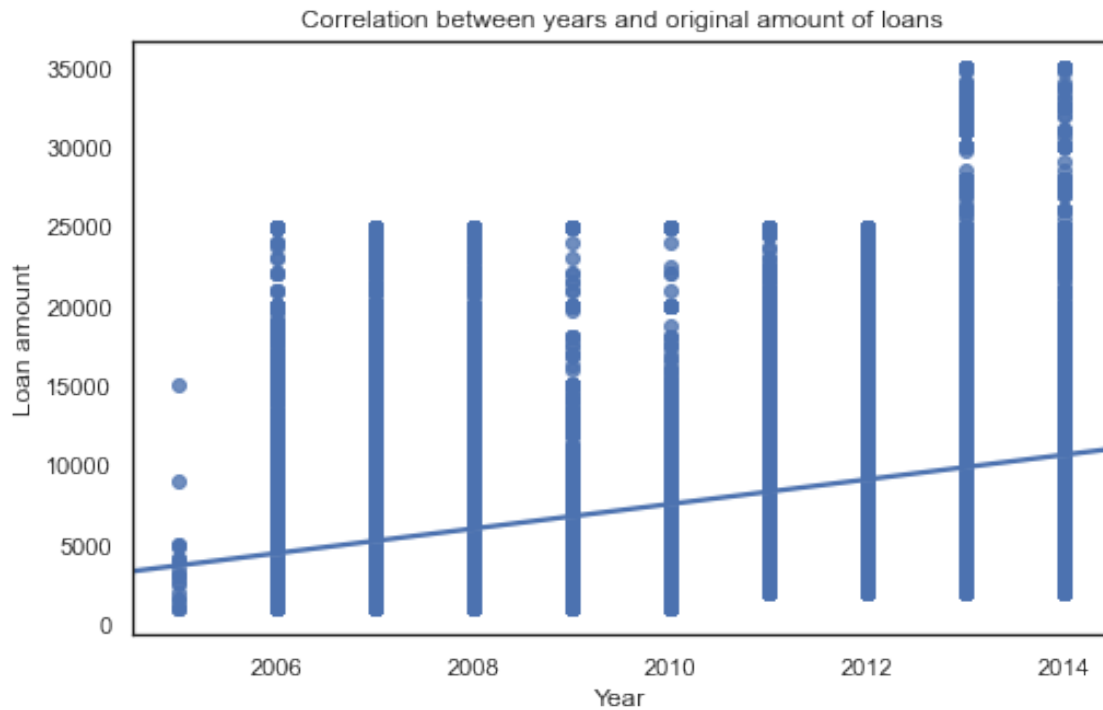


As can be seen above, BorrowerAPR and BorrowerRate have a very high correlation. Investors also have a correlation of 0.38 with LoanOriginalAmount.

```
[57]: plot_sample = np.random.choice(loans[numeric_vars].shape[0], 500, replace=False)
loans_plot_sample = loans[numeric_vars].loc[plot_sample]
g = sns.PairGrid(data=loans, vars=loans_plot_sample)
g = g.map_diag(plt.hist)
g.map_offdiag(plt.scatter);
```

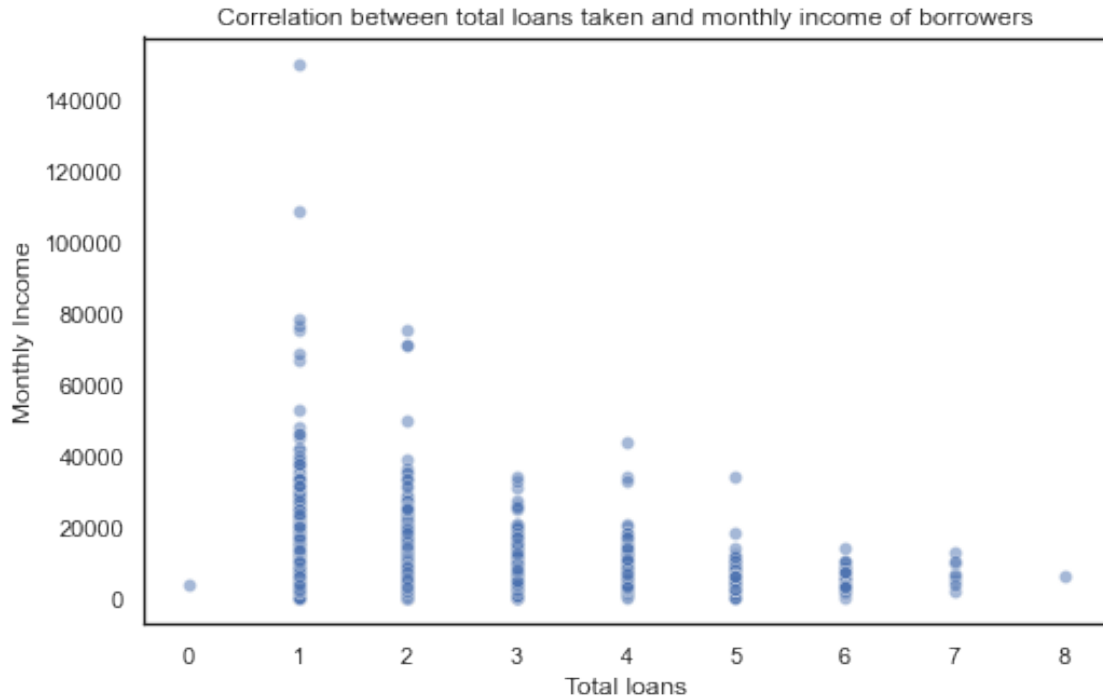


```
[59]: # Correlation between years and original amount of loans
plt.figure(figsize=[8,5])
sns.regplot(data=loans, x='Year', y='LoanOriginalAmount', truncate=False).set(
    title='Correlation between years and original amount of loans',
    xlabel='Year', ylabel='Loan amount');
```



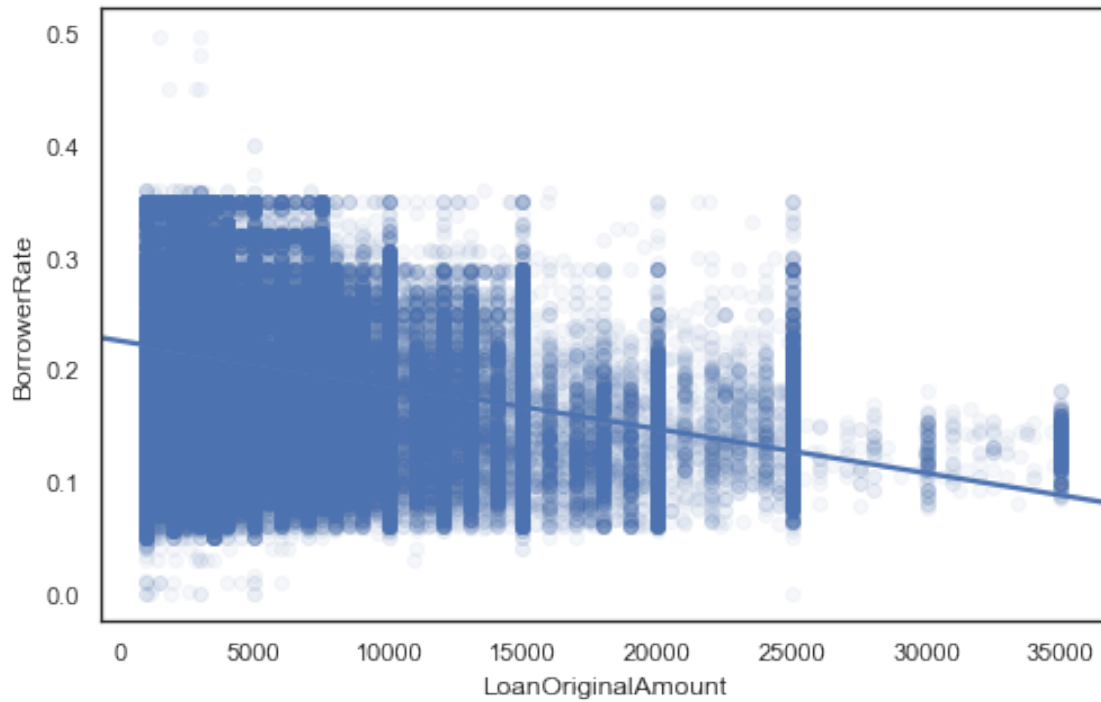
There is a positive correlation between Year and LoanOriginalAmount.

```
[60]: # Correlation between total loans taken and monthly income of borrowers
plt.figure(figsize = [8,5])
sns.scatterplot(data=loans, x='TotalProsperLoans', y='StatedMonthlyIncome',
               ↪alpha=5/10, palette='Blues_r').set(
title='Correlation between total loans taken and monthly income of borrowers',
               ↪xlabel='Total loans', ylabel='Monthly Income')
plt.show();
```



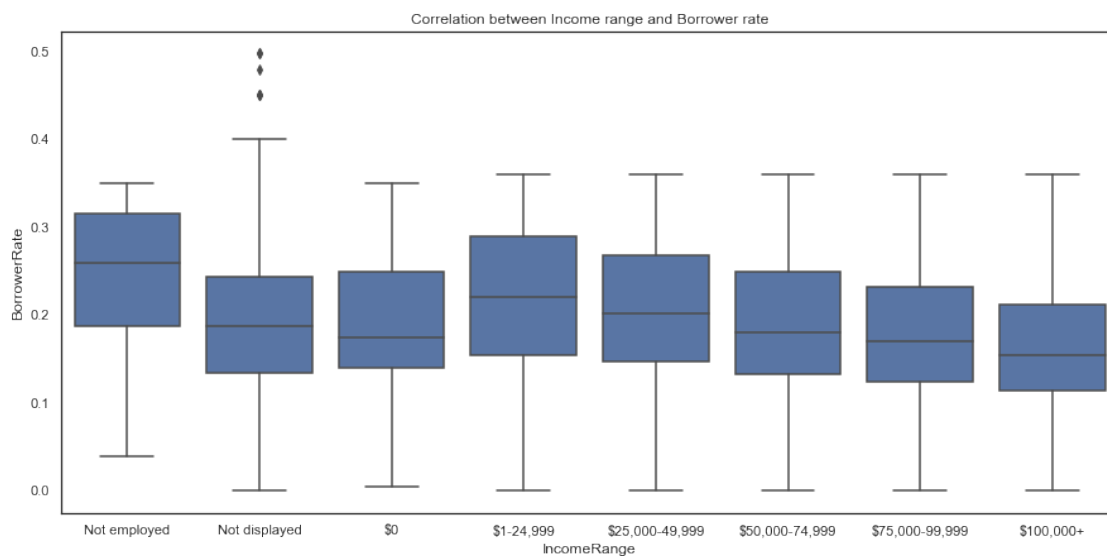
The borrowers with the highest monthly income took the least number of loans while those with the most loans taken are the low income earners. It can be noted that `TotalProsperLoans` is negatively correlated with `StatedMonthlyIncome`

```
[61]: # regplot of BorrowerRate and LoanOriginalAmount
plt.figure(figsize=[8,5])
sns.regplot(data=loans, x='LoanOriginalAmount', y='BorrowerRate',
            truncate=False, x_jitter=0.3, scatter_kws={'alpha':1/20});
```



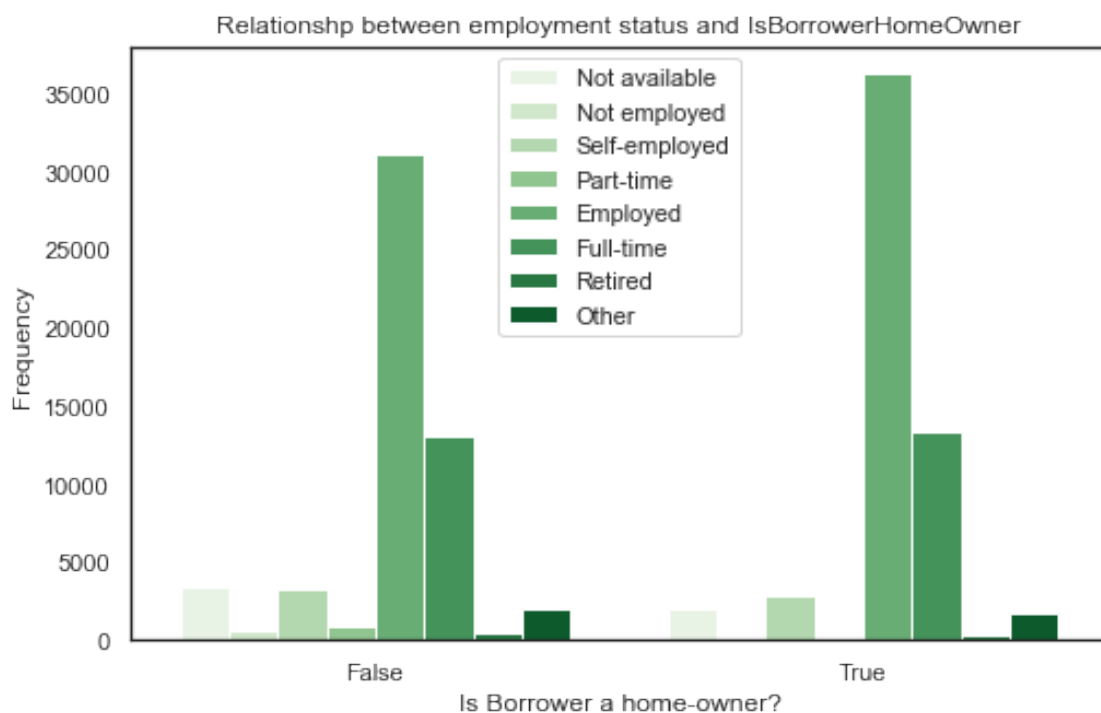
Range of BorrowerRate decreases with an increase of loan amount.

```
[65]: # boxplot of IncomeRange vs BorrowerRate
plt.figure(figsize=[15,7])
color = sns.color_palette()[0]
sns.boxplot(data=loans, x='IncomeRange', y='BorrowerRate', color=color).
    set(title='Correlation between Income range and Borrower rate');
```



The **not displayed** IncomeRange has a sizeable number of outliers. It also has the highest value of interest rate.

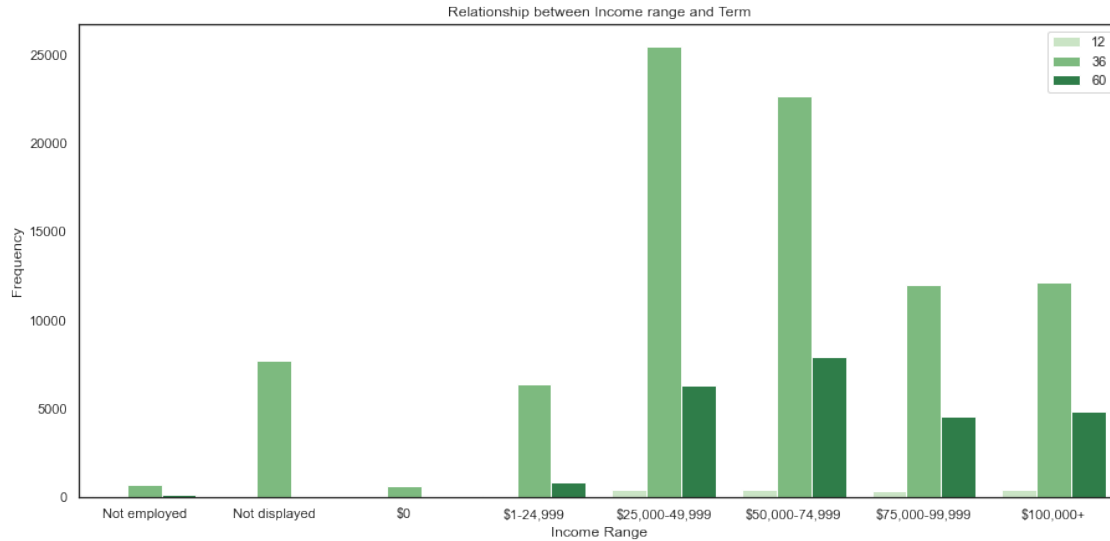
```
[68]: # relationship between employment status and IsBorrowerHomeOwner
plt.figure(figsize=[8,5])
sns.countplot(data=loans, x='IsBorrowerHomeowner', hue='EmploymentStatus',
              palette='Greens')
plt.legend(ncol=1)
plt.title('Relationship between employment status and IsBorrowerHomeOwner')
plt.xlabel('Is Borrower a home-owner?')
plt.ylabel('Frequency');
```



Employed folks make up the majority of borrowers that own a home. Interestingly, it's also the same for borrowers that don't own any.

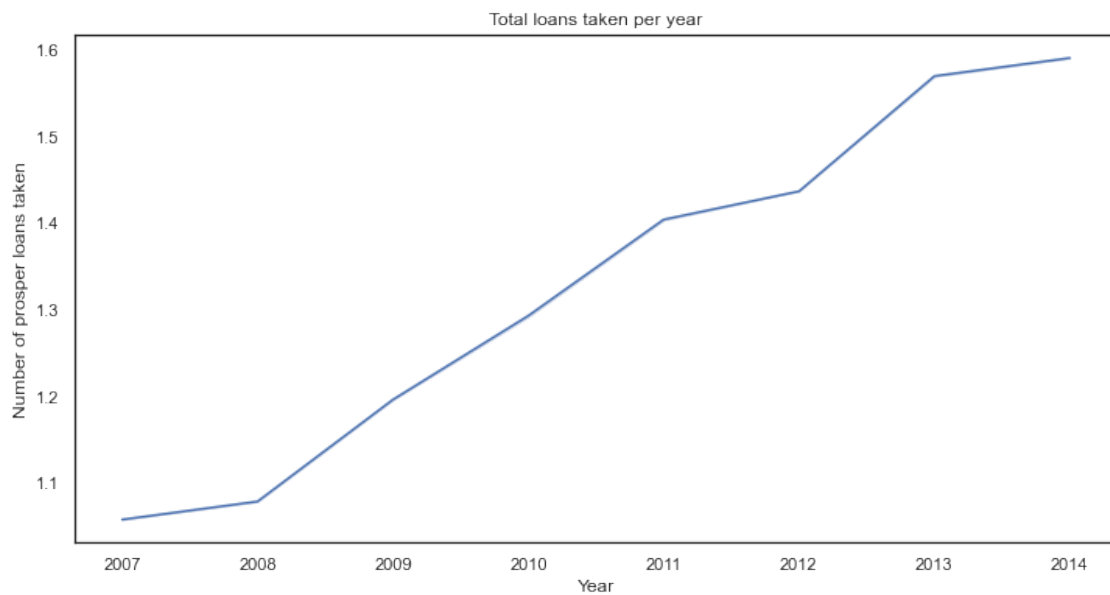
```
[74]: # term vs IncomeRange relationship
plt.figure(figsize=[15,7])
sns.countplot(data=loans, x='IncomeRange', hue='Term', palette='Greens')
plt.legend(ncol=1)
plt.title('Relationship between Income range and Term')
plt.xlabel('Income Range')
plt.ylabel('Frequency');
```





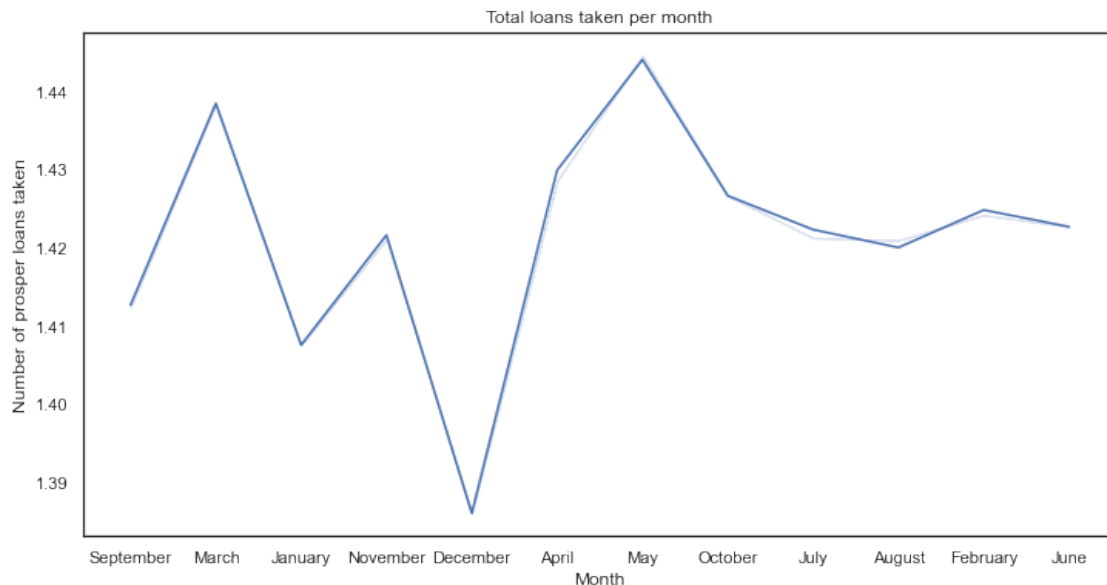
Majority of borrowers who took the shortest loan term have an income range between 25k and 100k dollars. It shows that the higher earning borrowers took the shortest loan term available.

```
[75]: # Relationship between TotalProsperLoans Vs Year
plt.figure(figsize=[12,6])
sns.lineplot(data=loans, x='Year', y='TotalProsperLoans', palette='Blues', ci=0)
plt.title('Total loans taken per year')
plt.xlabel('Year')
plt.ylabel('Number of prosper loans taken');
```



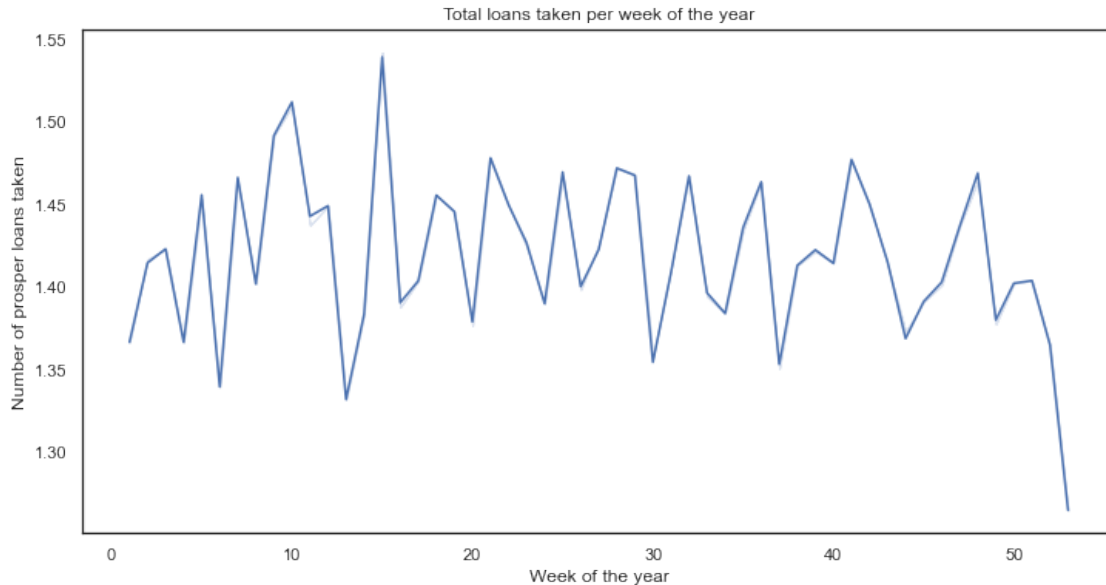
Generally, a positive correlation exists between `TotalProsperLoans` and `Year`. The number of loans taken decreased from 2008 to 2009 before going up again in 2012 thereafter with 2014 becoming the year with the highest number of loans taken.

```
[78]: # Relationship between TotalProsperLoans Vs Month
plt.figure(figsize=[12,6])
sns.lineplot(data=loans, x='Month', y='TotalProsperLoans', palette='Blues', ci=0)
plt.title('Total loans taken per month')
plt.xlabel('Month')
plt.ylabel('Number of prosper loans taken');
```



May is the month with the highest number of loans that were taken while borrowers seldom take loans at the end of the year.

```
[79]: # Relationship between TotalProsperLoans Vs Week
plt.figure(figsize=[12,6])
sns.lineplot(data=loans, x='Week', y='TotalProsperLoans', palette='Blues', ci=0)
plt.title('Total loans taken per week of the year')
plt.xlabel('Week of the year')
plt.ylabel('Number of prosper loans taken');
```



The least number of loans was taken during the last weeks of the year.

**1.5.1 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?**

- BorrowerAPR and BorrowerRate have a very high correlation.
- Investors have a correlation of 0.38 with LoanOriginalAMount.
- There is a positive correlation between Year and LoanOriginalAmount.
- The borrowers with the highest monthly income took the least number of loans while those with the most loans taken are the low income earners.
- It can also be noted that TotalProsperLoans is negatively correlated with StatedMonthlyIncome.
- The not displayed IncomeRange has a sizeable number of outliers
- Majority of borrowers who took the shortest loan term have an income range between 25k and 100k dollars. It shows that the higher earning borrowers took the shortest loan term available.
- Generally, a positive correlation exists between TotalProsperLoans and Year.
- May is the month with the highest number of loans that were taken while borrowers seldom take loans at the end of the year.
- The least number of loans was taken during the last weeks of the year.

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

- Employed folks make up the majority of borrowers that own a home.
- The higher earning borrowers took the shortest loan term available.

## 1.6 Multivariate Exploration

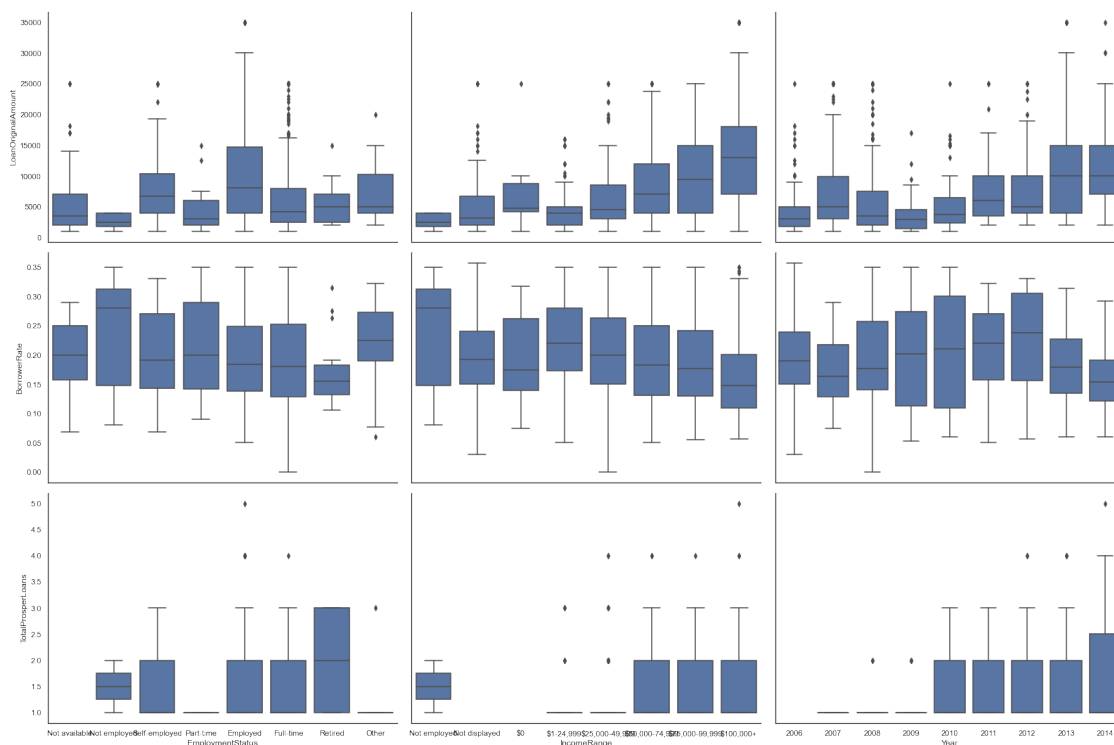
Creating plots of three or more variables to investigate the data even further.

```
[82]: # plot matrix of numeric features against categorical features with only 2000
      ↪ random samples taken
samples = np.random.choice(loans.shape[0], 2000, replace = False)
loans_samp = loans.loc[samples,:]
cat_vars = ['EmploymentStatus', 'IncomeRange', 'Year']

def boxgrid(x, y, **kwargs):
    """ Quick hack for creating box plots with seaborn's PairGrid. """
    default_color = sns.color_palette()[0]
    sns.boxplot(x, y, color = default_color)

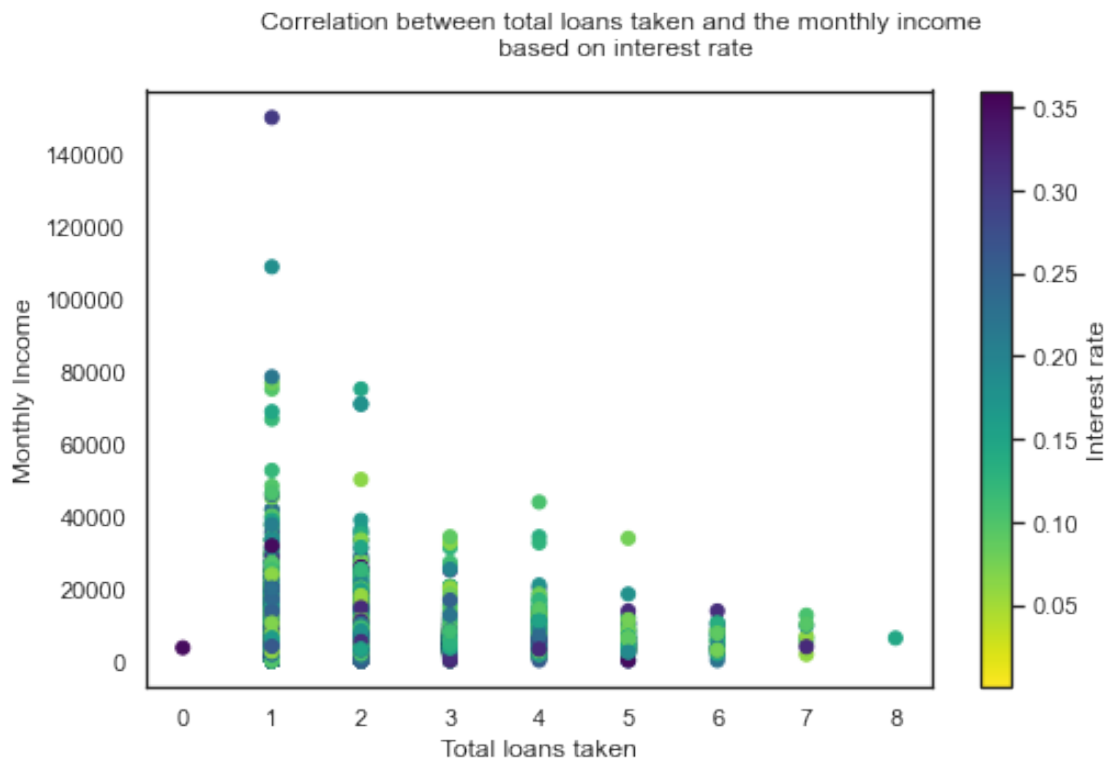
plt.figure(figsize = [20, 20])
g = sns.PairGrid(data = loans_samp, y_vars = ['LoanOriginalAmount',
      ↪ 'BorrowerRate', 'TotalProsperLoans'], x_vars = cat_vars, size=5, aspect=1.5,
      ↪ height=5)
g.map(boxgrid)
plt.show();
```

<Figure size 1440x1440 with 0 Axes>



It can be noted that IncomeRange and LoanOriginalAMount are positively correlated.

```
[83]: # relationship between TotalProsperLoans and StatedMonthlyIncome based on
      ↪BorrowerRate
plt.figure(figsize = [8,5])
plt.scatter(data=loans, x='TotalProsperLoans', y='StatedMonthlyIncome',
           ↪c='BorrowerRate', cmap='viridis_r')
plt.suptitle('Correlation between total loans taken and the monthly income\n
           ↪based on interest rate')
plt.xlabel('Total loans taken')
plt.ylabel('Monthly Income')
plt.colorbar(label='Interest rate')
plt.show();
```

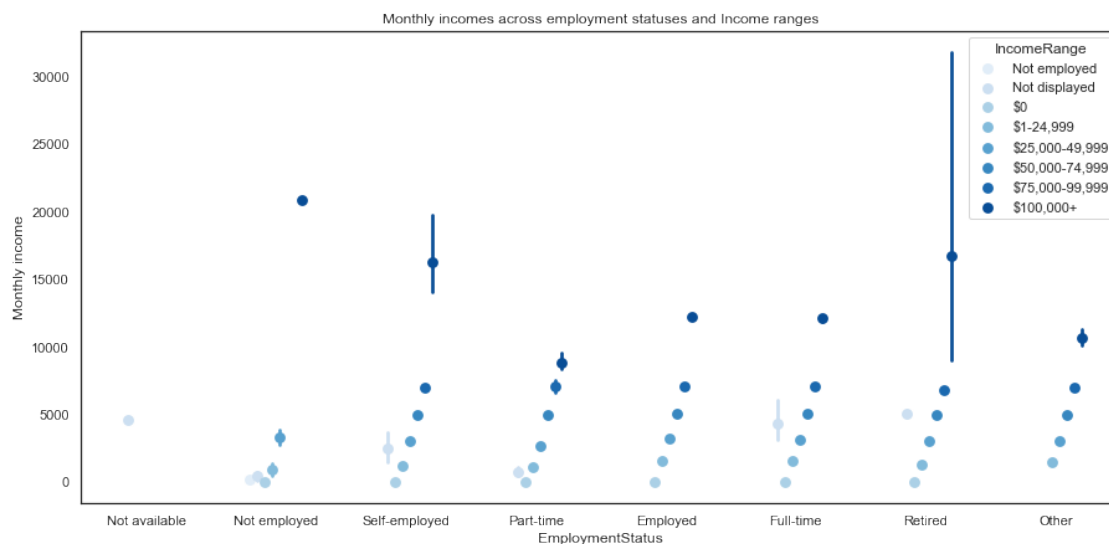


Borrowers with the least number of loans taken are the highest earners and they also have the highest interest rate (BorrowerRate). Generally, it can be observed that the more you take loans, the higher your interest rate (safe for the borrowers who took a single loan). This category of borrowers also have the lowest monthly income.

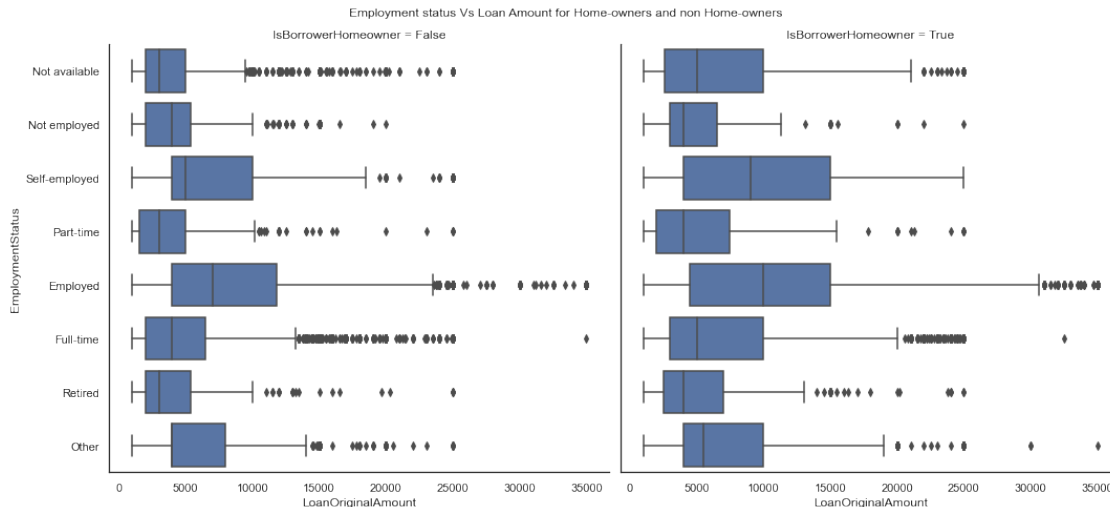
```
[87]: # fig = plt.figure(figsize = [15,7])
      # ax = sns.pointplot(data = loans, y = 'Listing Category', x =
      ↪'StatedMonthlyIncome', hue = 'EmploymentStatus',
      # palette = 'Blues', linestyle = '', dodge = 0.4)
      # plt.title('Monthly incomes across employment statuses and Income ranges')
```

```
# plt.ylabel('Monthly income')
# plt.show();
```

```
[88]: # Monthly incomes across employment statuses and Income ranges
fig = plt.figure(figsize = [15,7])
ax = sns.pointplot(data = loans, x = 'EmploymentStatus', y = 'StatedMonthlyIncome', hue = 'IncomeRange',
                  palette = 'Blues', linestyle = '', dodge = 0.4)
plt.title('Monthly incomes across employment statuses and Income ranges')
plt.ylabel('Monthly income')
plt.show();
```



```
[89]: # Employment status Vs Loan Amount for Home-owners and non Home-owners
g = sns.FacetGrid(data=loans, col='IsBorrowerHomeowner', height=5, aspect=1.5,
                  size=5)
g.map(sns.boxplot, 'LoanOriginalAmount', 'EmploymentStatus')
g.fig.set_size_inches(15, 7)
plt.suptitle('Employment status Vs Loan Amount for Home-owners and non Home-owners', y=1)
plt.show();
```



Borrowers who are either employed or Full-time and are home-owners tend to have more loan original amount. Also, self-employed borrowers who are home owners tend to have much loan original amount.

#### 1.6.1 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?

- It can be noted that IncomeRange and LoanOriginalAMount are positively correlated.
- Borrowers who are either employed or Full-time and are home-owners tend to have more loan original amount. Also, self-employed borrowers who are home owners tend to have much loan original amount.
- Generally, it can be observed that the more you take loans, the larger your interest rate.

#### 1.6.2 Were there any interesting or surprising interactions between features?

I noticed that folks who have a larger Income range tend to have the larger amount of loan original amount.

### 1.7 Conclusions

For this project, my main task was to explore the features related to the borrowers and how it relates to the loans they took. I started off with the univariate exploration where I explored the distribution of individual features. I got some notable insights from this exploration such as the majority of borrowers originating from California. I also observed that most borrowers are either employed or Full-time. A little minority took their loans for a year while most took their for 36 months. Exploring the Listing Category further, majority of the loans was taken for Debt consolidation. Taking a look at the ProsperLoans score, it can be seen that most borrowers have a not so good risk score of 4.0. Next I moved on to the Bivariate exploration where I investigated relationships between pairs of variables in the dataset. Using a heatmap, I explored the correlation of all numeric variables in the data and found out that BorrowerAPR and BorrowerRate are highly correlated. Looking down the

chart, I observed Investors have a correlation of 0.38 with LoanOriginalAmount. I also observed that borrowers with the highest monthly income took the least number of loans while those with the most loans taken are the low income earners. Exploring further, TotalProsperLoans is negatively correlated with StatedMonthlyIncome. Most borrowers also took loans in the month of May while very few loans were taken at the end of the year. Next, I moved on to the Multivariate exploration where I created plots of three or more variables to investigate the data even further. I used plot matrices to investigate the relationship between numeric features and categorical features in the data. I noted that IncomeRange and LoanOriginalAMount are positively correlated. I then checked the relationship between TotalProsperLoans and StatedMonthlyIncome based on BorrowerRate. Here, I observed that borrowers with the least number of loans taken are the highest earners and they also have the highest interest rate (BorrowerRate). I also noted that the more you take loans, the higher your interest rate(safe for the borrowers who took a single loan). Exploring the data further, I reviewed the relationship between Employment status and Loan Amount for Home-owners and non Home-owners. I found out that Borrowers who are either employed or Full-time and are home-owners tend to have more loan original amount. Also, self-employed borrowers who are home owners tend to have much loan original amount. An interesting interaction I noticed was that folks who have a larger Income range tend to have the larger amount of loan original amount. Finally, it was fun making this analysis and applying what I have learnt from the Data Visualization course of the Data Analyst Nanodegree to this exciting project. One issue I faced though was picking the best features from the many variables in the dataset for this analysis.