

# Part\_2\_slide\_deck

October 1, 2022

## 1 Part II - Exploration and Analysis of Loan Data from Prosper

1.0.1 by F Njakai

[Github](#) | [LinkedIn](#)

### 1.1 Investigation Overview

#### 1.1.1 Questions that guide the investigation

- what are the characteristics of Prosper's loans over time?
  - how many loans have been written?
  - what is the average rate offered to borrowers?
  - what is the average amount of a loan?
- what does the average customer look like?
  - another way of saying it is: to whom does Prosper lend?
- what affects the rate offered to a customer (borrower, if you like)?
- what affects the number of investors of a loan?

### 1.2 Summary of Findings

#### 1.2.1 Loans

- average amount borrowed is c. USD 8,337.00; the range is USD 1,000.00 to 35,000.00
  - loans of amount USD 5k, 10k, 15k, 20k and 25k have higher-than-average frequency
- average rate is c. 0.19%; the range is zero to 0.5%
- the number of loans disbursed changes with time
  - the number fluctuates
    - \* there are seemingly regular but wide fluctuations
      - fluctuations in the periods Q4 2013 to Q2 2014 inclusive are remarkably wide
    - \* there are little to no fluctuations starting Q4 2008 to Q3 2009, inclusive (this is exactly 12 months)
  - the number of loans issued appears to increase exponentially despite the fluctuations

- the average number of loans issued, per annum, between 2005 and 2014 inclusive is *c.* 4.3 million
- an overwhelming majority of loans of all terms are in the range 0.05% to 0.35% borrowing rate
- majority of the tenure of the loans is 3 years or more
- the least funded loans are at 70% funding for all levels of borrowing rate

### 1.2.2 Occupations and employment status

- an overwhelming majority of the top ten occupations are [white collar](#)
- a majority of the bottom ten occupations, that are not students, are [blue collar](#); half of the occupations in the bottom ten are students
- Prosper lends to unemployed applicants

### 1.2.3 Income level

- Prosper appears to prefer to lend to high income earners at less-than-average rates
  - the region of high income and low rates is quite dense
  - an overwhelming majority of loans are issued to those whose income level is above USD 25,000.00 *p.a.*
    - \* the top 2 groups account for 55.8% of loans issued
    - \* the top 4 groups account for 85.4% of loans issued

### 1.2.4 Home ownership status

- there is an almost even split between those that own a home and those that do not; 50.4% of borrowers own a home

### 1.2.5 Location of borrower

- a majority of the states with the least borrowers are from the so-called "[fly-over country](#)"
- all of the states with the most borrowers are on either [seaboard](#)

### 1.2.6 Factors that affect the nominal borrowing rate

- there is weak, positive correlation between BorrowerRate and Term -> *c.* 0.02
- there is weak, negative correlation between
  - BorrowerRate and StatedMonthlyIncome -> *c.* -0.09
  - BorrowerRate and EmploymentStatusDuration -> *c.* -0.02
- there is strong, positive correlation between
  - BorrowerRate and BorrowerAPR -> *c.* 0.99
- none of the variables of interest have a strong, negative correlation

### 1.2.7 Factors that affect the number of investors

- none of the variables of interest have a weak, positive correlation
- there is weak, negative correlation between
  - Investors and BorrowerRate -> c. -0.27
  - Investors and PercentFunded -> c. -0.05
- none of the variables of interest have a strong, positive correlation
- none of the variables of interest have a strong, negative correlation

## 1.3 Overview of dataset

### 1.3.1 Overall

- 113,937 observations
- 81 variables
  - 3 of type bool
  - 50 of type float
  - 11 of type int
  - 17 of type str

### 1.3.2 Missing and null values

- some observations in the data set have missing or null values
  - case(s) in point
    - \* 100,596 under variable GroupKey
    - \* 96,985 under variable LoanFirstDefaultedCycleNumber

### 1.3.3 Duplicated observations

- the data set has no duplicated observations

### 1.3.4 Multiple values for a variable

- observations in the data set have 1 value per variable

```
In [2]: # 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

%matplotlib inline

# suppress warnings from final output
import warnings
warnings.simplefilter("ignore")
from contextlib import suppress
```

```
In [3]: # load in the dataset into a pandas dataframe
df = pd.read_csv('prosper_loans_data.csv', sep=',')
df.head()
```

```
Out[3]:
```

	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	

	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	

	BorrowerRate	LenderYield	...	LoanOriginationDate	\
0	0.1580	0.1380	...	2007-09-12 00:00:00	
1	0.0920	0.0820	...	2014-03-03 00:00:00	
2	0.2750	0.2400	...	2007-01-17 00:00:00	
3	0.0974	0.0874	...	2012-11-01 00:00:00	
4	0.2085	0.1985	...	2013-09-20 00:00:00	

	LoanOriginationQuarter	MemberKey	MonthlyLoanPayment	\
0	Q3 2007	1F3E3376408759268057EDA	330.43	
1	Q1 2014	1D13370546739025387B2F4	318.93	
2	Q1 2007	5F7033715035555618FA612	123.32	
3	Q4 2012	9ADE356069835475068C6D2	321.45	
4	Q3 2013	36CE356043264555721F06C	563.97	

	LP_CustomerPayments	LP_CustomerPrincipalPayments	LP_InterestandFees	\
0	11396.14	9425.00	1971.14	
1	0.00	0.00	0.00	
2	4186.63	3001.00	1185.63	
3	5143.20	4091.09	1052.11	
4	2819.85	1563.22	1256.63	

	LP_ServiceFees	PercentFunded	Investors
0	-133.18	1.0	258
1	0.00	1.0	1
2	-24.20	1.0	41
3	-108.01	1.0	158
4	-60.27	1.0	20

[5 rows x 72 columns]

Note that the above cells have been set as "Skip"-type slides. That means that when the

notebook is rendered as http slides, those cells won't show up.

## 1.4 Visualisation 1

**Q: how does the number of loans written behave over the years?**

- does it change at all?
  - if yes, what is the direction of change?

**A: the number of loans written changes with time**

- how does it change?
  - the number fluctuates
    - \* there are seemingly regular but wide fluctuations according to the time series
      - fluctuations in the periods Q4 2013 to Q2 2014 inclusive are remarkably wide
    - \* there are little to no fluctuations starting Q4 2008 to Q3 2009, inclusive (this is exactly 12 months)
    - \* the highest number of loans issued, in a year, before 2011 were issued in 2008 according to the bar chart
- there appears to be an exponential trend
  - trend shows that number of loans issued appears to increase exponentially despite the fluctuations

```
In [4]: #default blue
default_blue = sns.color_palette('tab10')[0]

#default orange, if required
default_orange = sns.color_palette('tab10')[1]

#default palette
default_palette = sns.color_palette('tab10')

#template no. 2

'''
simple function to create `Figure` object
using matplotlib that contains an x-lab,
y-lab and title.

"Father Figure", if you like :)

3 params, all type `str`:
x_lab, y_lab and title

Please make sure all args passed to the
```

```

function are type `str`

return: None
'''

def create_fig(x_lab: str, y_lab: str, title: str):
    """create_fig function"""
    try:
        #fig size
        plt.figure(figsize=(10, 6.18), dpi=216, frameon=False, clear=True)
        #x-axis name
        plt.xlabel(x_lab)
        #y-axis name
        plt.ylabel(y_lab)
        #title
        plt.title(title)
    except ModuleNotFoundError:
        print(f'Please `import matplotlib.pyplot as plt` and try again')
    except:
        print(f'Failed to create template')
        raise

#template no. 3

'''
simple function to create `Figure` object
using matplotlib for sub-plots.

"Father Figure" for subplots :)

2 params, type int; the number of sub-plots.
@n_row -> #rows
@n_col -> #cols

return: fig and ax objects
'''

def create_sub(n_row: int=1, n_col: int=1):
    """create_sub function"""
    try:
        #fig, ax and figsize
        fig, ax = plt.subplots(n_row, n_col, figsize=(10, 6.18), dpi=216)
        fig.tight_layout(pad=10.0)
        return fig, ax
    except ModuleNotFoundError:
        print(f'Please `import matplotlib.pyplot as plt` and try again')
    except:
        print(f'Failed to create template')
        raise

```

```

#group data and find sum

'''
Function to calculate sum of
grouped data

Takes in 2 args: var_1 and var_2

Both args must be columns of a pandas DF

return: sum of grouped data
'''

def group_sum(var_1, var_2):
    """ function group_sum"""
    if var_1 and var_2:
        sum_gp_data = df.groupby(var_1)[var_2].sum()
        return sum_gp_data
    else:
        print(f'Check that both arguments are columns of a pandas DF and try again')

df.LoanOriginationDate = pd.to_datetime(df.LoanOriginationDate)

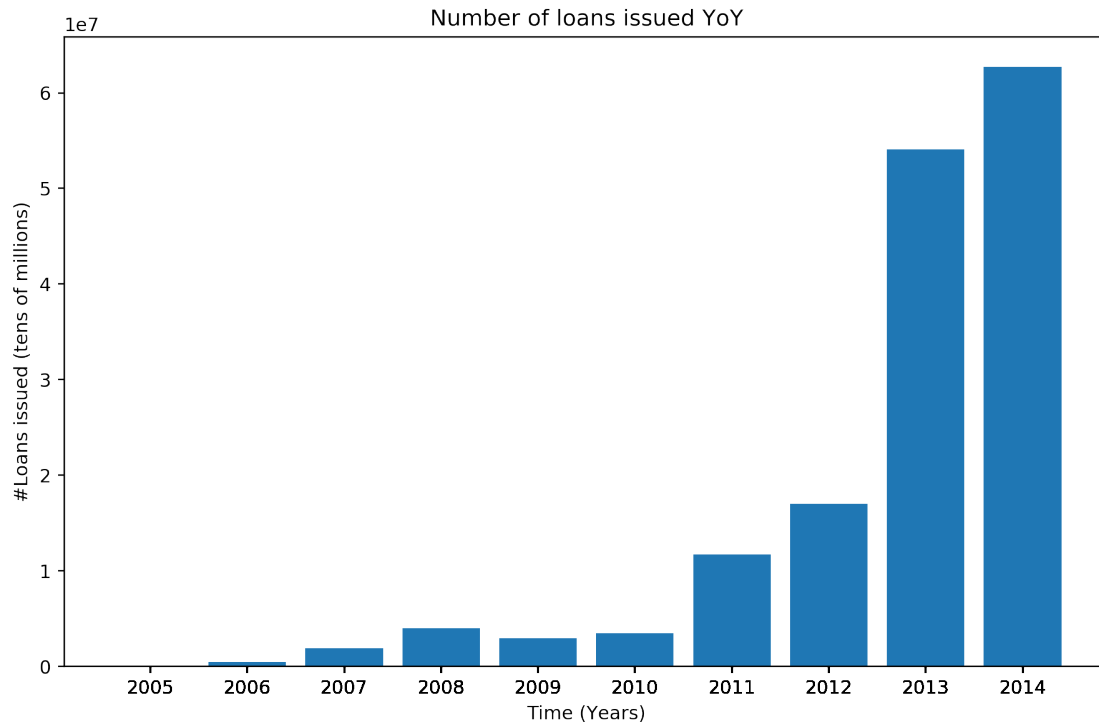
#sum of loans by year
sum_loan_time = group_sum('LoanOriginationDate', 'LoanNumber')

#number of loans issued
y = [sum_loan_time.values[i] for i in range(len(sum_loan_time))]

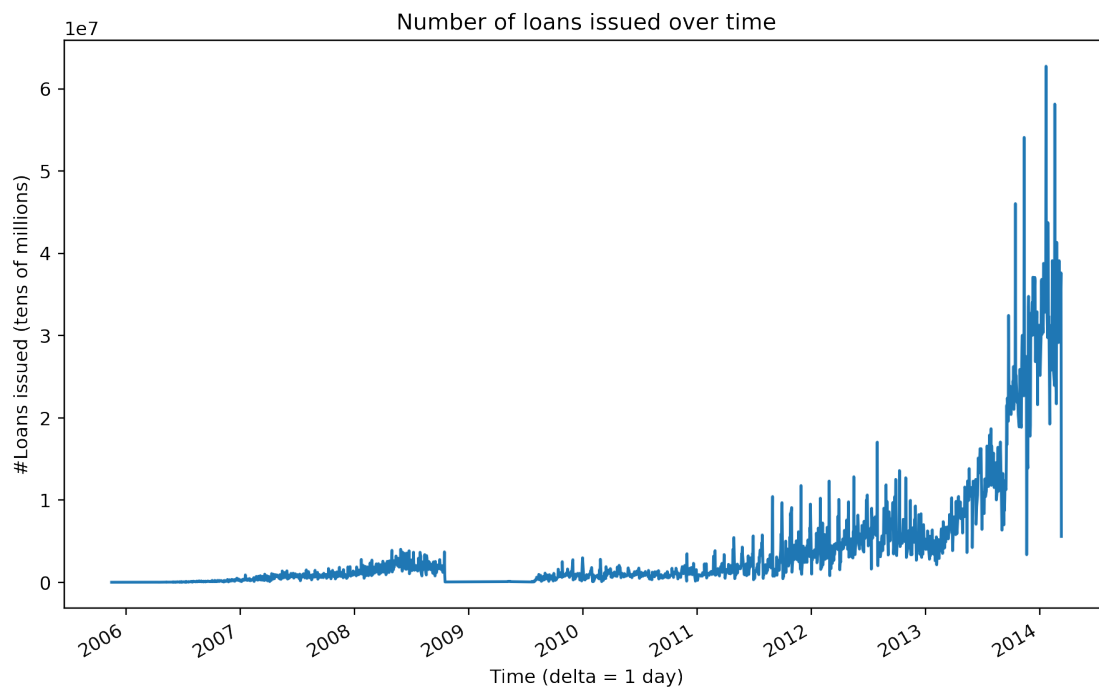
#time (years)
x = [sum_loan_time.index[i].year for i in range(len(sum_loan_time))]

#visualise said timelines
#create plot
create_fig('Time (Years)', '#Loans issued (tens of millions)', 'Number of loans issued Y
plt.bar(x, y, tick_label=x, color=default_blue)
plt.yticks(np.arange(0, 65000000, step=10000000));

```



```
In [5]: #time series plot for `sum_loan_plot` by the day
create_fig('Time (Days)', '#Loans issued (tens of millions)', 'Number of loans issued over time')
sum_loan_time.plot(kind='line',color=default_blue)
plt.xlabel('Time (delta = 1 day)');
```





**Q: what, really, happens:**

- before 2007
- during Q4 2008 to Q3 2009, inclusive

**A: nothing, for the most part**

- what happens before 2007...
- activity begins at the beginning of Q2 2007
- little activity; less than a million loans, in total, are issued
- what happens during Q4 2008 to Q3 2009, inclusive...
- activity ceases during the middle of Q4 2008
- said activity stays at zero until late Q2 2009
- little activity; less than 15 million loans, in total, are issued

**conclusion**

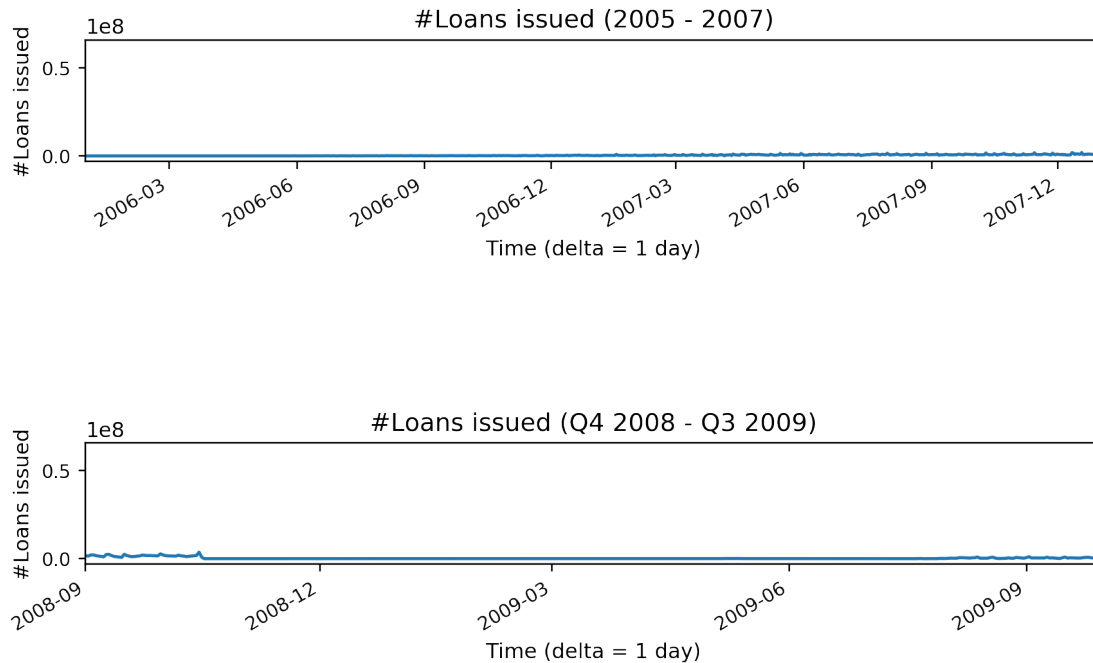
- the first few years had little to no activity
- Prosper is growing its loan book in an exponential-like manner
- Prosper's ability to grow its loan book are affected directly by events in the financial system

```
In [6]: create_sub(2, 1)
        plt.suptitle('Loans written by Prosper: outliers')

        #sub-plot #3: loans issued pre-2007
        plt.subplot(2, 1, 1)
        sum_loan_time.plot(kind='line',color=default_blue, xlim=['2005-12-31','2008-01-01'], tit
        plt.xlabel('Time (delta = 1 day)')
        plt.ylabel('#Loans issued');

        #sub-plot #4: loans issued during Q4 2008 to Q3 2009
        plt.subplot(2, 1, 2)
        sum_loan_time.plot(kind='line',color=default_blue, xlim=['2008-09-01','2009-09-30'], tit
        plt.xlabel('Time (delta = 1 day)')
        plt.xticks(['2008-09', '2008-12', '2009-03', '2009-06', '2009-09'])
        plt.ylabel('#Loans issued');
```

## Loans written by Prosper: outliers



### 1.5 Visualisation 2

**Q:** what is the correlation between the rate, term, stated monthly income and employment status duration of a loan?

**A:** a mix of weak positive and negative correlations, mostly

- there is weak, positive correlation between
  - Term and BorrowerRate -> *c.* 0.02
  - Term and StatedMonthlyIncome -> *c.* 0.03
  - Term and EmploymentStatusDuration -> *c.* 0.09
  - StatedMonthlyIncome and EmploymentStatusDuration -> *c.* 0.07
- there is weak, negative correlation between
  - BorrowerRate and StatedMonthlyIncome -> *c.* -0.09
  - BorrowerRate and EmploymentStatusDuration -> *c.* -0.02
  - Term and BorrowerAPR -> *c.* -0.02
  - StatedMonthlyIncome and BorrowerAPR -> *c.* -0.08
  - EmploymentStatusDuration and BorrowerAPR -> *c.* -0.01
- none of the variables of interest have a strong, negative correlation

and then there's this...

- strong, positive correlation between
  - BorrowerAPR and BorrowerRate -> c. 0.99

### conclusion

- the factors investigated do not affect the borrowing rate significantly; a more extensive and rigorous investigation must be done

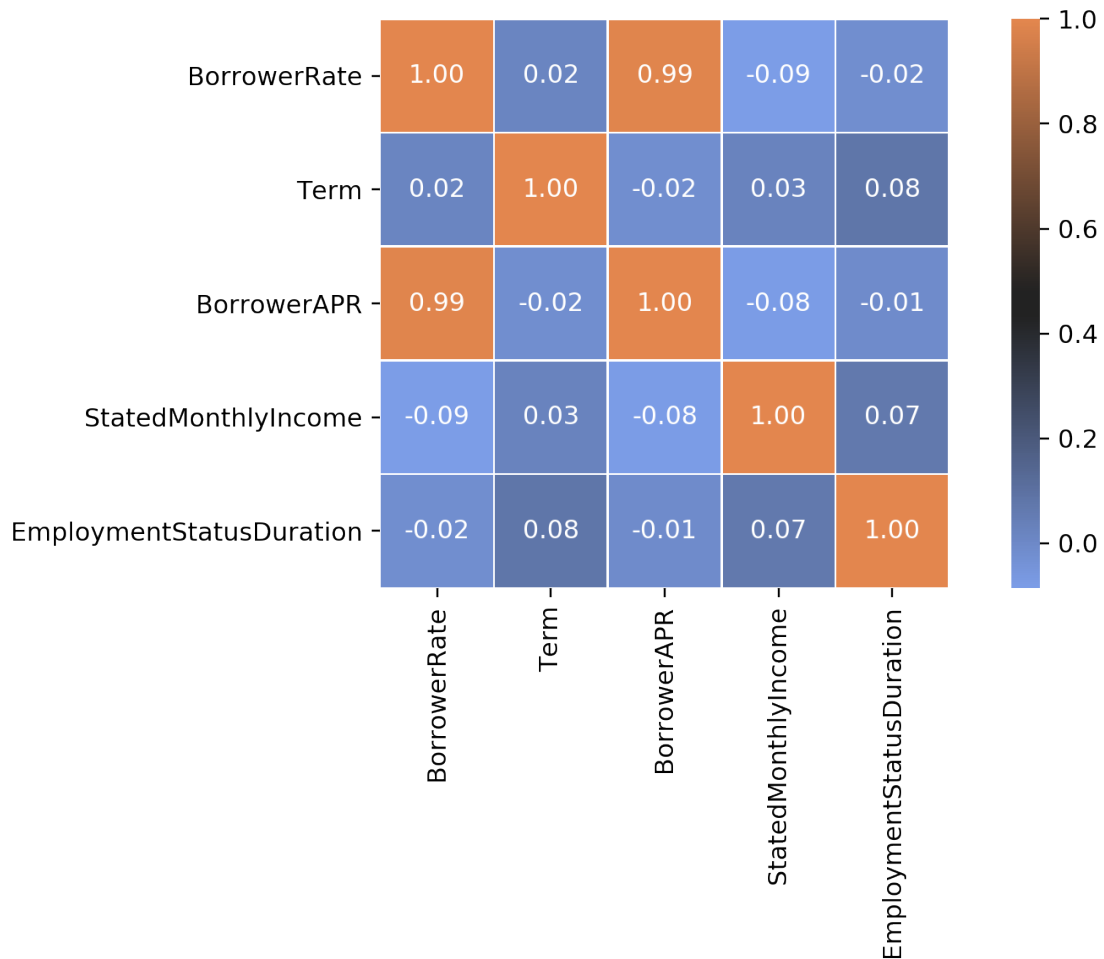
```
In [7]: corr_df = pd.DataFrame([df.BorrowerRate, df.Term, df.BorrowerAPR, df.StatedMonthlyIncome])
corr_df = corr_df[pd.notnull(corr_df['EmploymentStatusDuration'])]

#sub-plots
fig, ax = create_sub()
plt.suptitle('Correlation between select factors that affect borrower rate')

#correlation matrix
corr = corr_df.corr()

#plot
with suppress(DeprecationWarning, ImportWarning):
    sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(
```

## Correlation between select factors that affect borrower rate



```
In [ ]: corr
```

### 1.6 Visualisation 3

**Q:** what is the correlation between the rate, proportion funded and number of investors of a loan?

**A:** weak positive correlation

- none of the variables of interest have a weak, positive correlation
- there is weak, negative correlation between

- BorrowerRate and PercentFunded -> c. -0.03
- BorrowerRate and Investors -> c. -0.27
- PercentFunded and Investors -> c. -0.05

- none of the variables of interest have a strong, positive correlation
- none of the variables of interest have a strong, negative correlation

## conclusion

- the factors investigated do not affect the proportion of loans invested in significantly; a more extensive and rigorous investigation must be done

```
In [8]: corr_df = pd.DataFrame([df.BorrowerRate, df.PercentFunded, df.Investors]).transpose()
        #sub-plots
        fig, ax = create_sub()
        plt.suptitle('Correlation between select factors that affect the number of investors')

        #correlation matrix
        corr = corr_df.corr()

        #plot
        with suppress(DeprecationWarning, ImportWarning):
            sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(
```

Correlation between select factors that affect the number of investors



```
In [ ]: corr
```

### 1.6.1 Generate Slideshow

Once you're ready to generate your slideshow, use the `jupyter nbconvert` command to generate the HTML slide show.

```
In [ ]: # Use this command if you are running this file in local
!jupyter nbconvert Part_2_slide_deck.ipynb --to slides --post serve --no-input --no-prom

[NbConvertApp] Converting notebook Part_2_slide_deck.ipynb to slides
[NbConvertApp] Writing 837969 bytes to Part_2_slide_deck.slides.html
[NbConvertApp] Redirecting reveal.js requests to https://cdnjs.cloudflare.com/ajax/libs/reveal.js/3.7.1/
Serving your slides at http://127.0.0.1:8000/Part_2_slide_deck.slides.html
Use Control-C to stop this server
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: x-www-browser: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: firefox: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: iceweasel: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: seamonkey: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: mozilla: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: epiphany: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: konqueror: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: chromium-browser: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: google-chrome: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: www-browser: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: links2: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: elinks: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: links: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: lynx: not found
/usr/bin/xdg-open: 778: /usr/bin/xdg-open: w3m: not found
xdg-open: no method available for opening 'http://127.0.0.1:8000/Part_2_slide_deck.slides.html'
```

In the classroom workspace, the generated HTML slideshow will be placed in the home folder.

In local machines, the command above should open a tab in your web browser where you can scroll through your presentation. Sub-slides can be accessed by pressing 'down' when viewing its parent slide. Make sure you remove all of the quote-formatted guide notes like this one before you finish your presentation! At last, you can stop the Kernel.

### 1.6.2 Submission

If you are using classroom workspace, you can choose from the following two ways of submission:

1. **Submit from the workspace.** Make sure you have removed the example project from the /home/workspace directory. You must submit the following files:

- Part\_I\_notebook.ipynb
- Part\_I\_notebook.html or pdf
- Part\_II\_notebook.ipynb
- Part\_I\_slides.html
- README.md
- dataset (optional)

2. **Submit a zip file on the last page of this project lesson.** In this case, open the Jupyter terminal and run the command below to generate a ZIP file.

```
zip -r my_project.zip .
```

The command above will ZIP every file present in your /home/workspace directory. Next, you can download the zip to your local, and follow the instructions on the last page of this project lesson.

```
In [ ]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Part_2_slide_deck.html'])
```

```
In [ ]:
```