Review of Python for Data Science

CADS Workshop Fall 2022

What will we cover in this workshop?

- **Yes**
  - Basics of strings and lists, because these are used all the time in data work
  - List comprehensions, because of their connection to map/reduce
  - Computing in notebooks
  - Fundamental data types: Series, DataFrame
  - Subsetting and sorting DataFrames
  - Other commonly useful pandas tools
- **No**
  - Control structures (if, for, etc.)
  - Software development
  - How to install Python or Jupyter
  - Anything really advanced

Data Science is not software development

Loops and conditionals are standard coding tools:

```
for i in range( len( my_data ) ):
    if my_data.loc[i,"age"] > 21:
        my_data.loc[i,"party_eligible"] = True
```

Data science code can almost always omit them!

```
my_data["party_eligible"] = my_data["age"] > 21
```

Data work uses strings and lists frequently

Common string tools

```
S = "my string"
S[3]            # get one character
S[3:]           # get several
S.split()       # split into words
"my" in S       # check for text
S.index("my")   # find position
S + " is long"
```

Common list tools

```
L = [ "my","little","list" ]
L[1]              # get one item
L[1:]             # get several

"my" in L         # check for item
L.index("my")     # find position
L.append(":)")    # extend
```

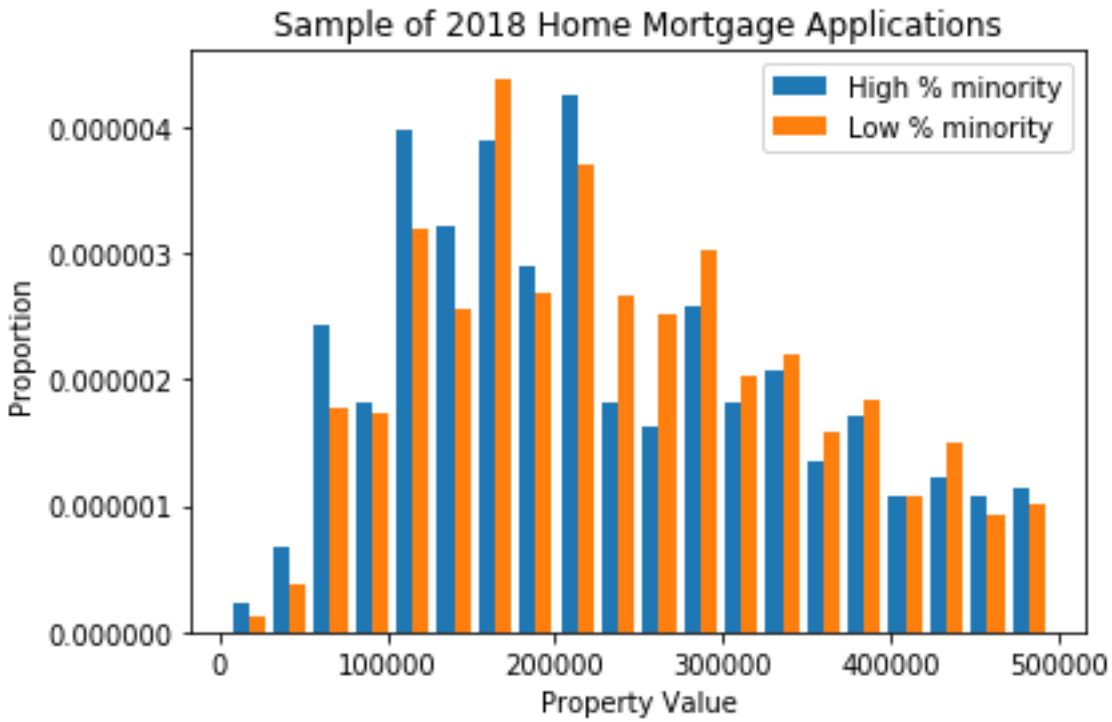
List Comprehensions

(How to play with lists without writing loops.)

```
[1]: L = ["my","little","list"]
     [ len(word) for word in L ]

[1]: [2, 6, 4]

[2]: L = ["my","little","list"]
     [ len(word) for word in L if "i" in word ]

[2]: [6, 4]

[3]: L = ["my","little","list"]
     [ word for word in L if "i" in word ]

[3]: ['little', 'list']
```

Notebooks (Jupyter, etc.)

[11]:
```
import matplotlib.pyplot as plt
plt.hist( [ high_minority['property_value'], low_minority['property_value'] ],
          bins=20, density=True )
plt.legend( [ 'High % minority', 'Low % minority' ] )
plt.title( 'Sample of 2018 Home Mortgage Applications' )
plt.xlabel( 'Property Value' )
plt.ylabel( 'Proportion' )
plt.show()
```

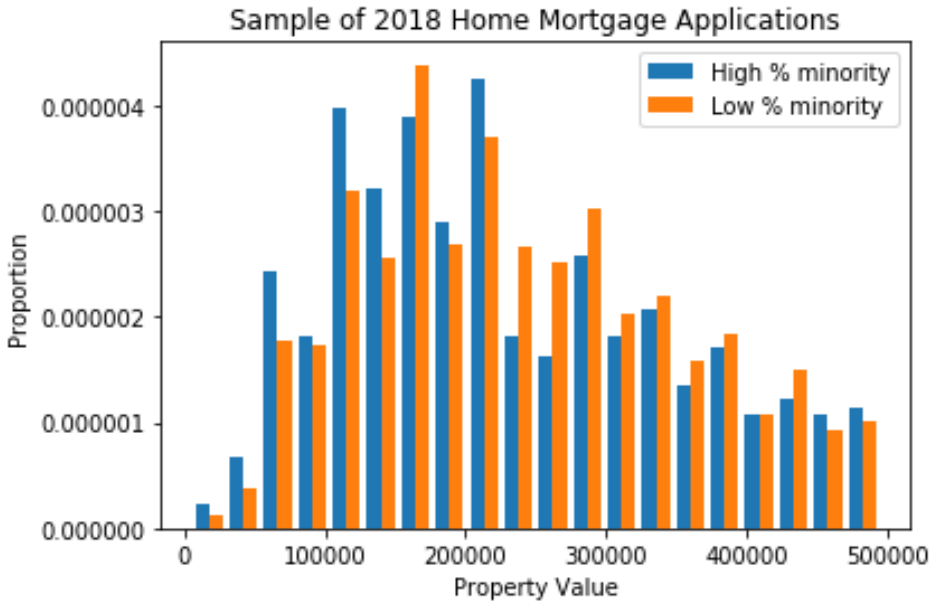


[14]:
```
high_minority['property_value'].mean(), low_minority['property_value'].mean()
```

[14]:
```
(229579.64601769912, 240573.24840764332)
```

Now let's plot the distribution of home prices for each of those two subsamples, the high minority areas and low minority areas. Perhaps the graph will show us whether there's any difference in home prices in these areas. We use two overlapping histograms and normalize them to proportions rather than actual frequencies, to make them more comparable, since the sizes of the two subsamples differ.

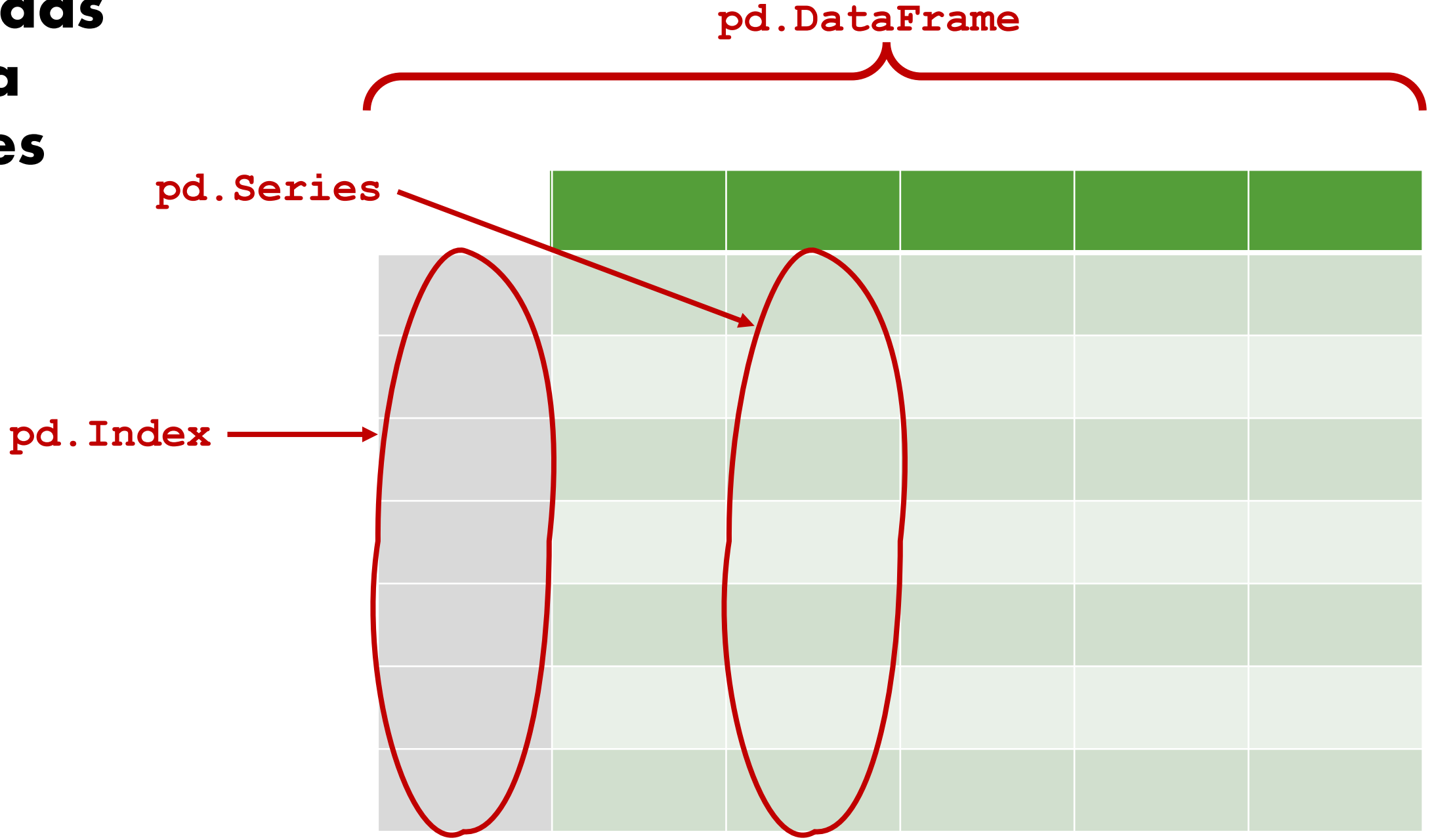
[11]:
```
import matplotlib.pyplot as plt
plt.hist( [ high_minority['property_value'], low_minority['property_value'] ],
          bins=20, density=True )
plt.legend( [ 'High % minority', 'Low % minority' ] )
plt.title( 'Sample of 2018 Home Mortgage Applications' )
plt.xlabel( 'Property Value' )
plt.ylabel( 'Proportion' )
plt.show()
```



One might hypothesize, even before looking at the graph, that for a variety of societal reasons (some of which are bad), areas with a high minority population have lower property values. The graph seems to reinforce this: On the right half of the histogram (higher home values) the orange bars (low minority %) tend to be larger, but on the left side, the opposite is true.

pandas data types

pandas data types



Common actions to take on Series (individual columns)

Inspecting the contents of a column

[2]:
```
df["state_code"].unique()
```

[2]:
```
array(['CA', 'WA', 'GA', 'SC', 'KY', 'MI', 'NC', 'OR', 'MD', 'PA', 'TX',
       'NJ', 'NY', 'IL', 'LA', 'FL', 'OH', 'AL', 'CO', 'OK', 'VA', 'MA',
       'ID', 'MO', 'AR', 'SD', 'ME', 'NH', 'AZ', 'MS', 'DC', 'WI', 'NE',
       'TN', 'NV', 'CT', 'MN', nan, 'WV', 'IN', 'IA', 'UT', 'VT', 'HI',
       'AK', 'WY', 'KS', 'NM', 'DE', 'RI', 'ND', 'MT', 'PR'], dtype=object)
```

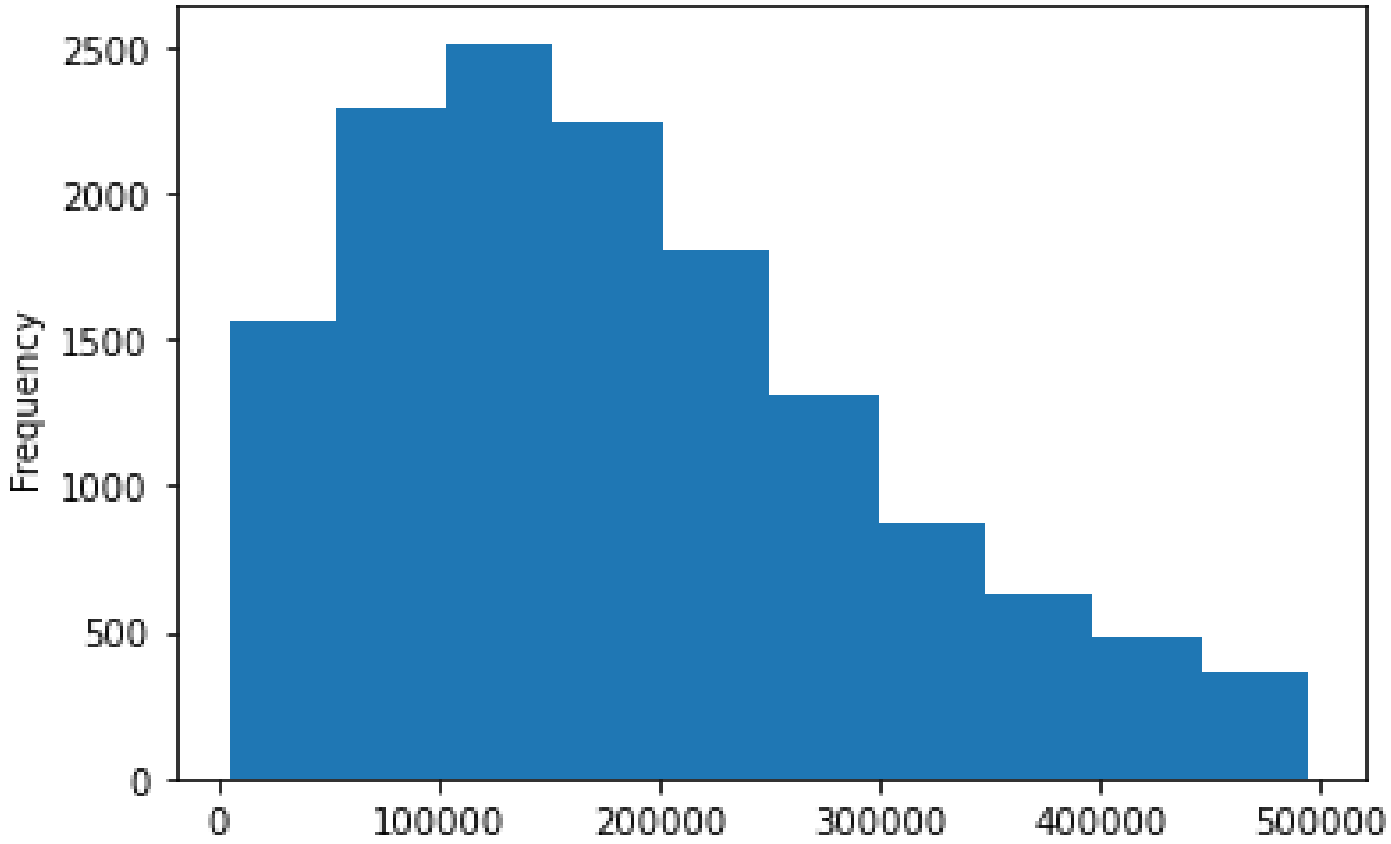
[3]:
```
df["state_code"].value_counts()
```

[3]:
```
CA    1684
FL    1136
TX    1119
PA     564
GA     558
OH     542
NY     535
NC     524
IL     508
MI     469
WA     454
AZ     449
```

Simple visualization with .plot.type()

```
[8]: df["loan_amount"].plot.hist()
```

```
[8]: <AxesSubplot:ylabel='Frequency'>
```

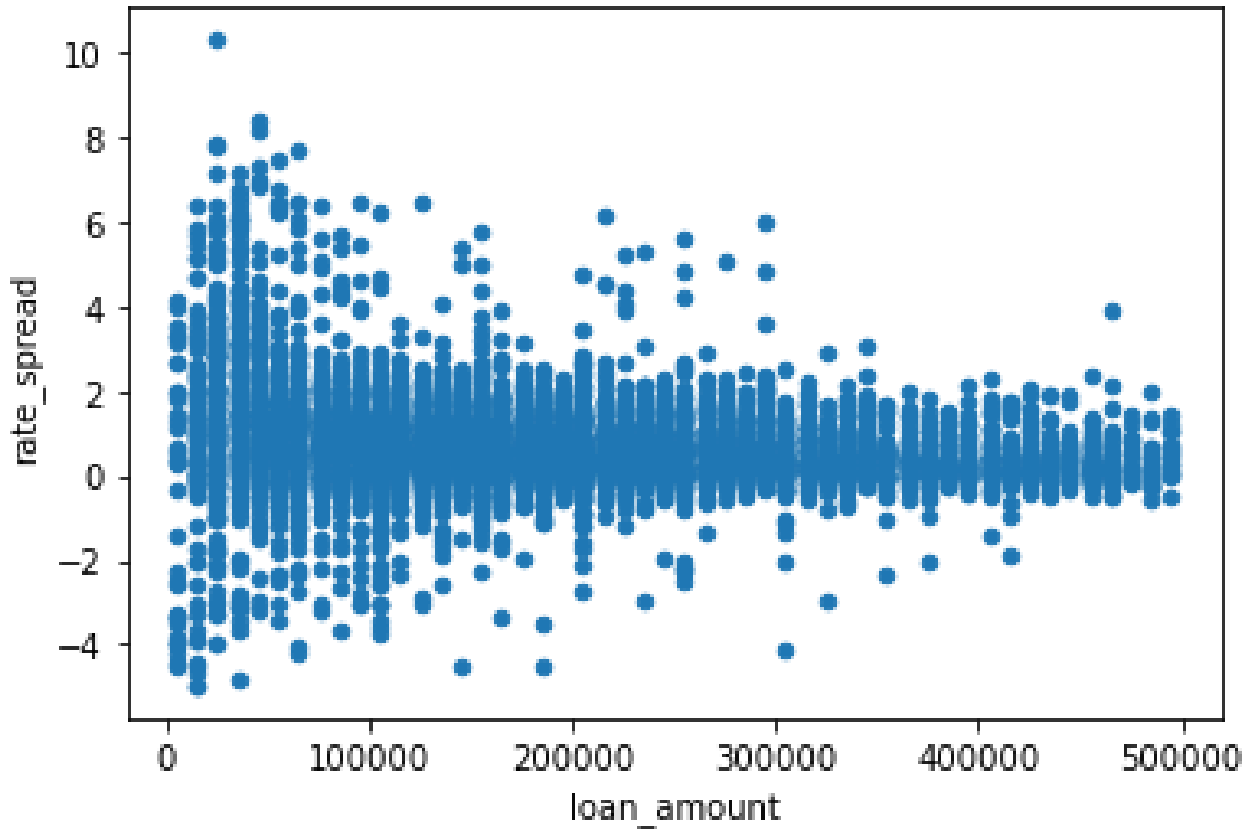


Common actions to take on DataFrames (entire tables)

Simple visualization with .plot.type()

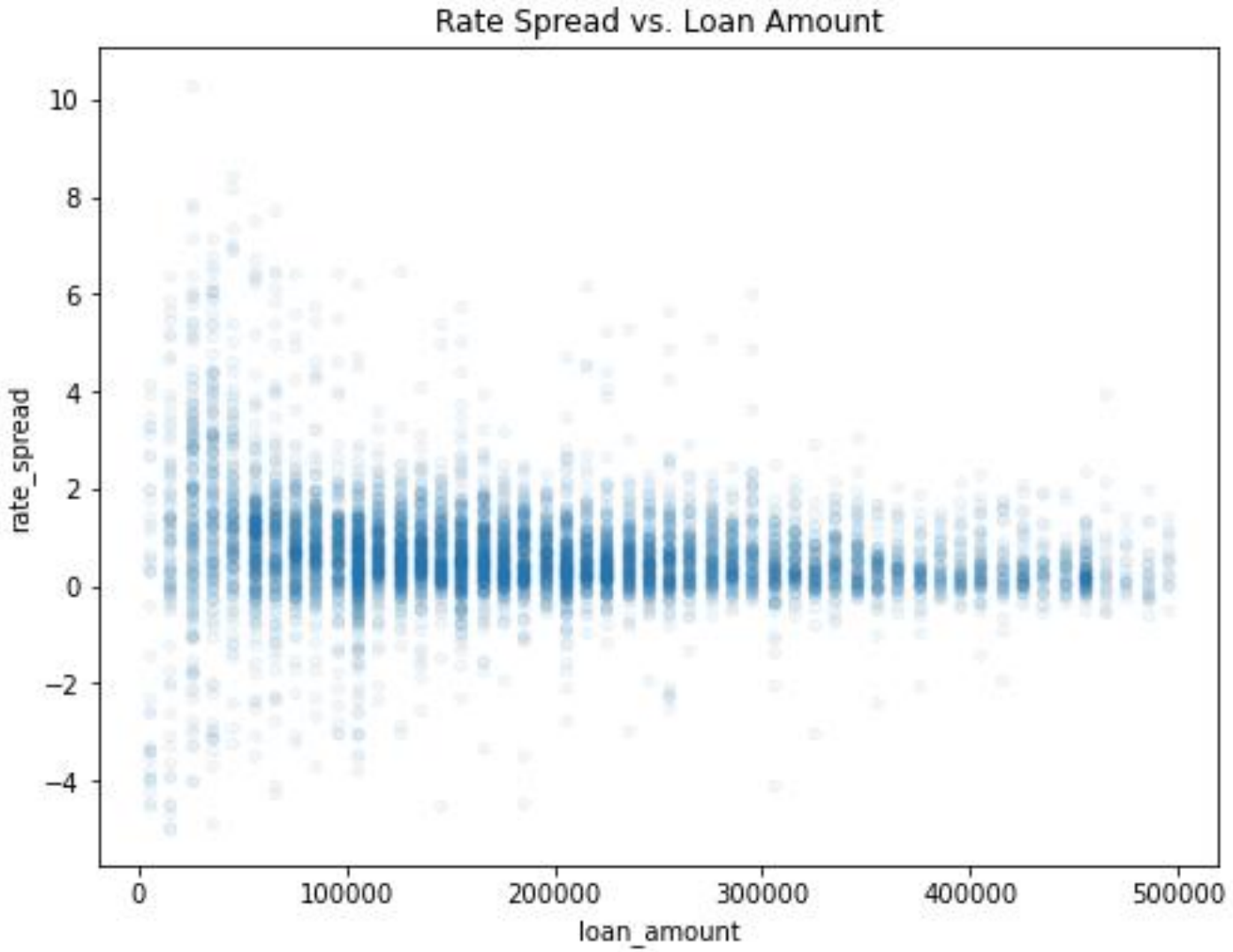
```
df.plot.scatter( x="loan_amount", y="rate_spread" )
```

```
<AxesSubplot:xlabel='loan_amount', ylabel='rate_spread'>
```



To do better, use Matplotlib explicitly

[116]:
```
import matplotlib.pyplot as plt
df.plot.scatter( x="loan_amount", y="rate_spread", alpha=0.05 )
plt.title( "Rate Spread vs. Loan Amount" )
plt.gcf().set_size_inches( 8, 6 )
plt.show()
```



Choose columns

[9]:
```
df[["state_code","loan_amount"]]
```

[9]:

	state_code	loan_amount
1	WA	115000
2	GA	105000
3	SC	185000
4	KY	235000
5	MI	35000
...	...	...

[27]:
```
df = df[["state_code","loan_amount"]]
```

Choose rows

[28]:
```
df[df["loan_amount"] > 20000000]
```

[28]:

	state_code	loan_amount
915	GA	20845000
7287	GA	25835000

Create a new column

[42]:
```
df["applicant_age"].unique()
```

[42]:
```
array(['65-74', '35-44', '25-34', '55-64', '8888', '>74', '45-54', '<25'],
      dtype=object)
```

[43]:
```
df["applicant_age_known"] = df["applicant_age"] != "8888"
```

[44]:
```
df["applicant_age_known"].head()
```

[44]:
```
0    True
1    True
2    True
3    True
4    True
Name: applicant_age_known, dtype: bool
```

Sort a table

[58]:
```
df.sort_values( by="loan_amount" ).head()
```

[58]:

	state_code	loan_amount
1054	ID	5000
3459	NY	5000
3273	MN	5000
3766	IL	5000
13751	TX	5000

[59]:
```
df.sort_values( by="loan_amount", ascending=False ).head()
```

[59]:

	state_code	loan_amount
7287	GA	25835000
915	GA	20845000
11871	VA	19055000
7394	NY	19005000
2307	AZ	16915000