Data Science with Python

Part 8: Graphics

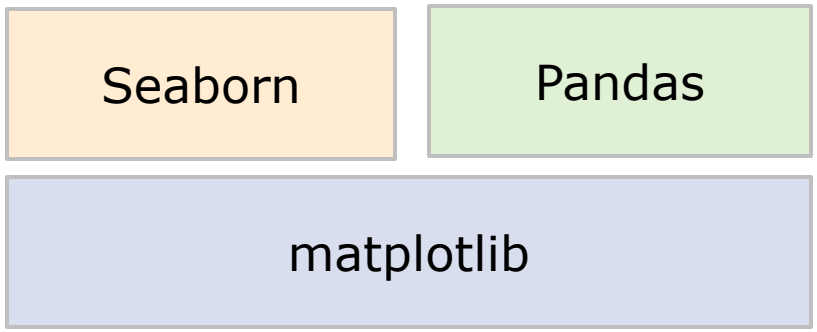
Based on
Dr. Glenn Bruns
CSUMB

Main source materials:

- Python Data Science Handbook, Jake VanderPlas, chapter 4.
- matplotlib.org/users/index.html
- matplotlib.org/3.1.1/tutorials/introductory/usage.html
- python-graph-gallery.com/
- seaborn.pydata.org/tutorial.html
- pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

Python plotting packages

We'll use three packages:



There are lots of others, including plotly.

I highly recommend the matplotlib tutorials:
https://matplotlib.org/3.1.1/tutorials/index.html

Matplotlib: basic line plot

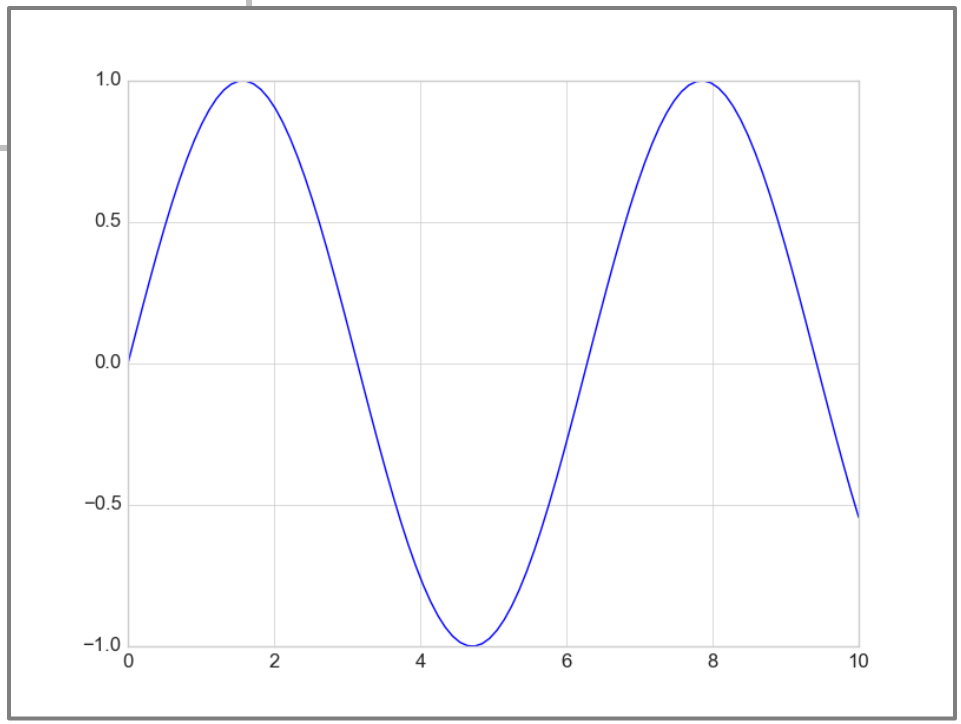
```
import matplotlib.pyplot as plot
plt.style.use('seaborn-whitegrid')

# some data
x = np.linspace(0,10,100)

plt.plot(x, np.sin(x))
plt.show()
```

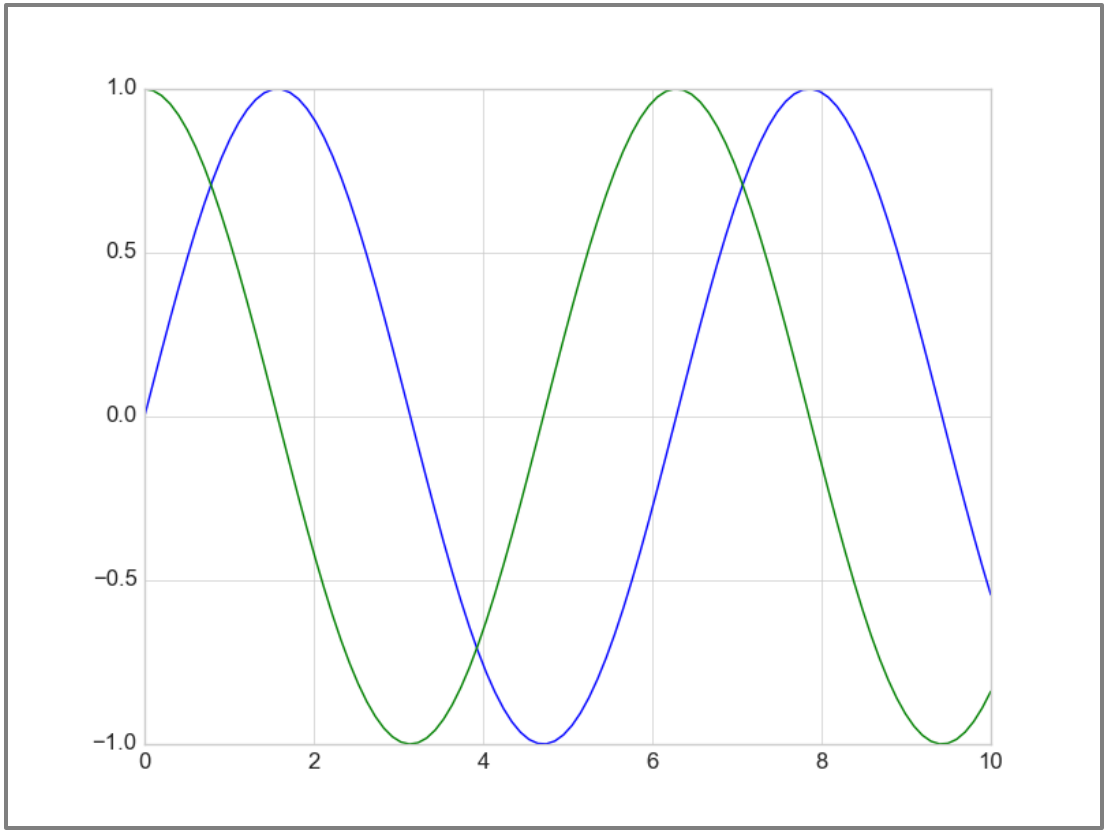
We use matplotlib's pyplot module

plt.show() not needed in Spyder or Jupyter notebooks



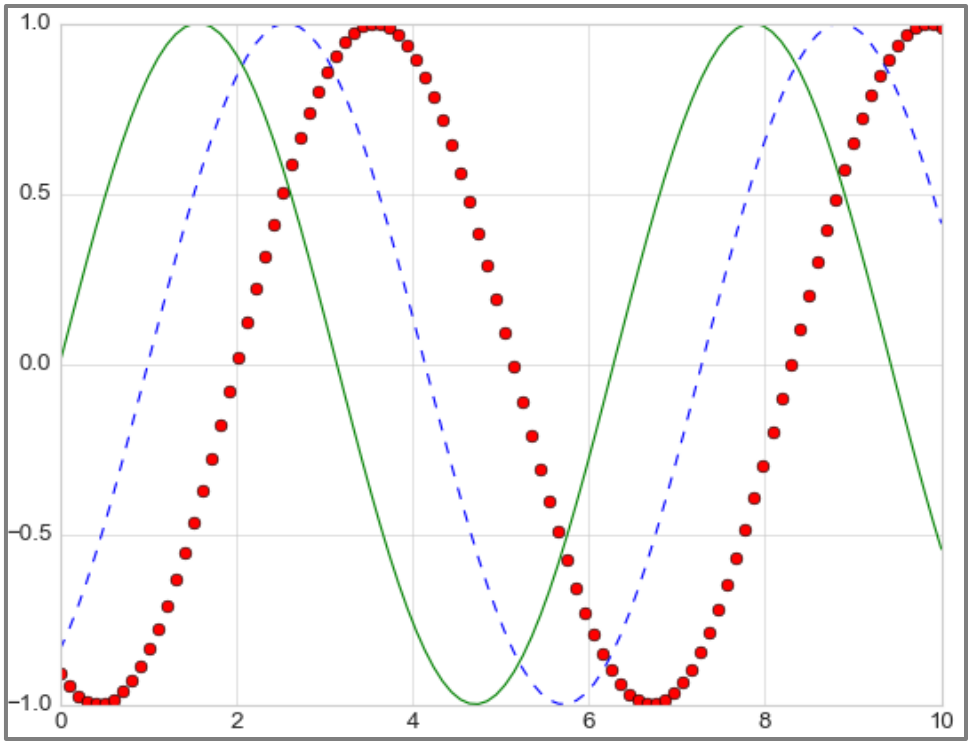
Multi-line plot

```
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
```



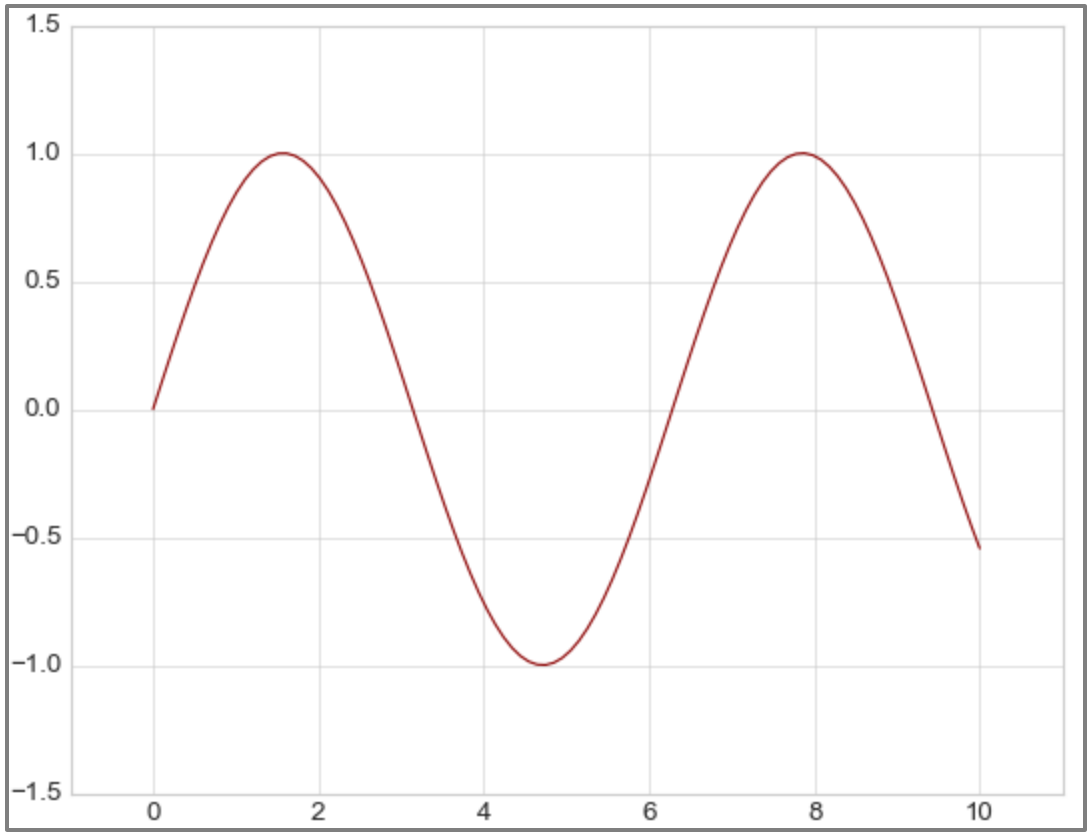
Colors, line styles

```
plt.plot(x, np.sin(x-0), color='green')
# line styles include 'dotted', 'dotdash'
plt.plot(x, np.sin(x-1), linestyle='dashed')
# abbreviated style; 'ro' for 'red, circles'
plt.plot(x, np.sin(x-2), 'ro')
```



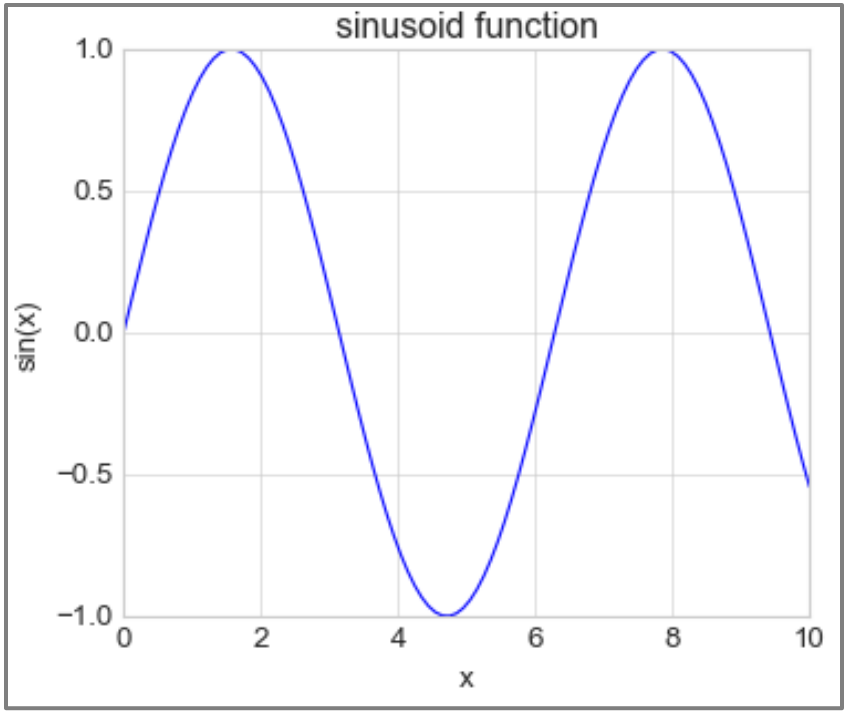
Adjusting axis limits

```
plt.plot(x, np.sin(x), color='darkred')
plt.xlim(-1,11)
plt.ylim(-1.5, 1.5)
```



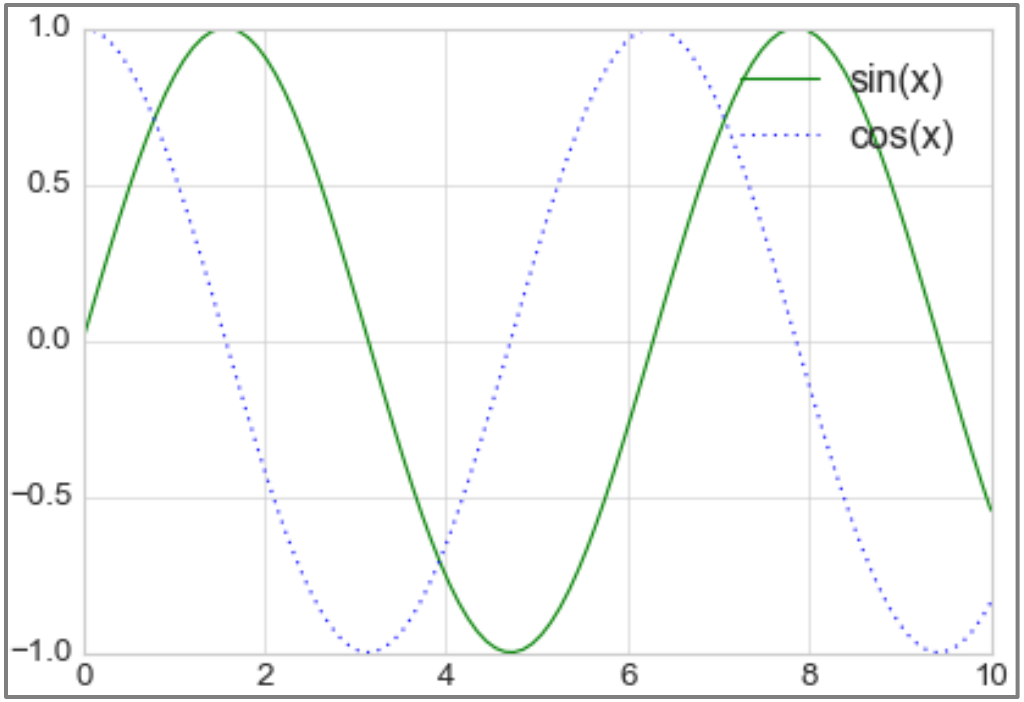
Titles and axis labels

```
plt.plot(x, np.sin(x))
plt.title('sinusoid function')
plt.xlabel('x')
plt.ylabel('sin(x)')
```



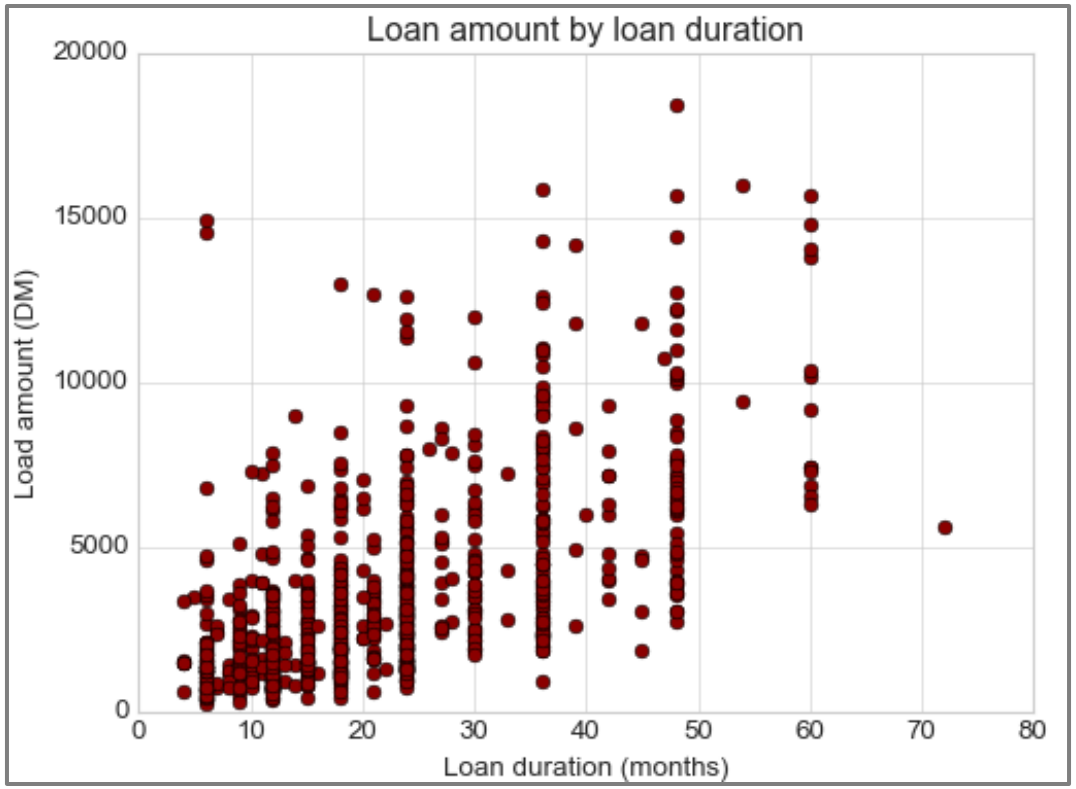
Legends

```
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')
plt.legend()
```



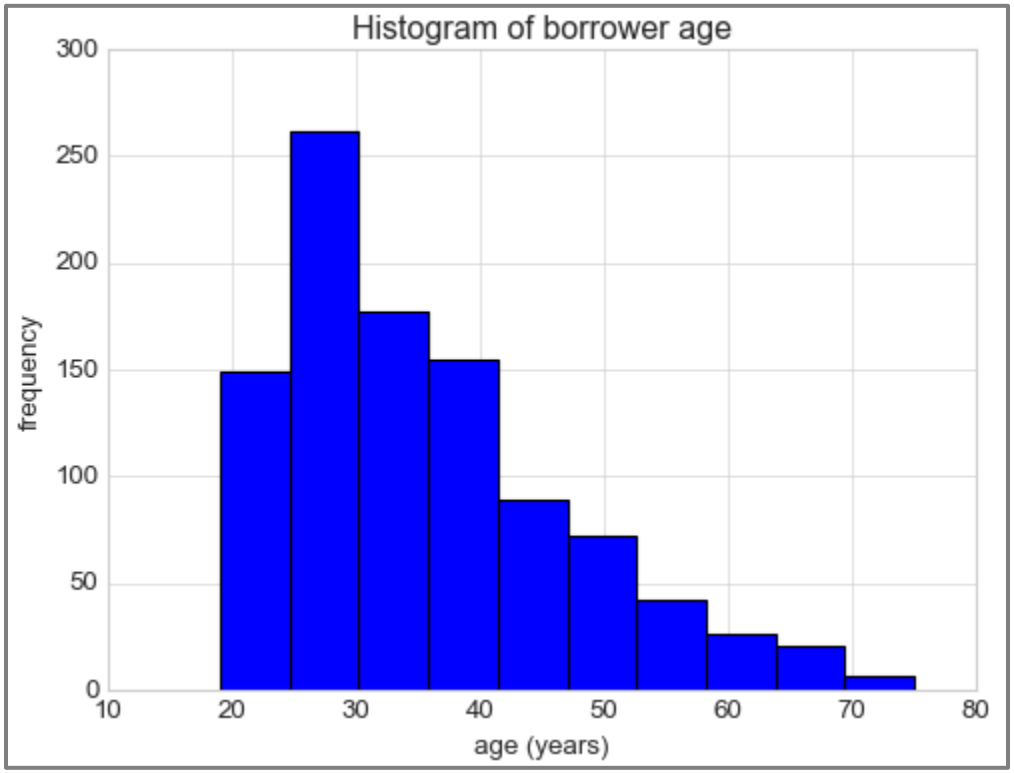
Basic scatter plot

```
plt.scatter(df['duration.in.months'], df['amount'], 'ro')
plt.title('Loan amount by loan duration')
plt.xlabel('Loan duration (months)')
plt.ylabel('Load amount (DM)')
```



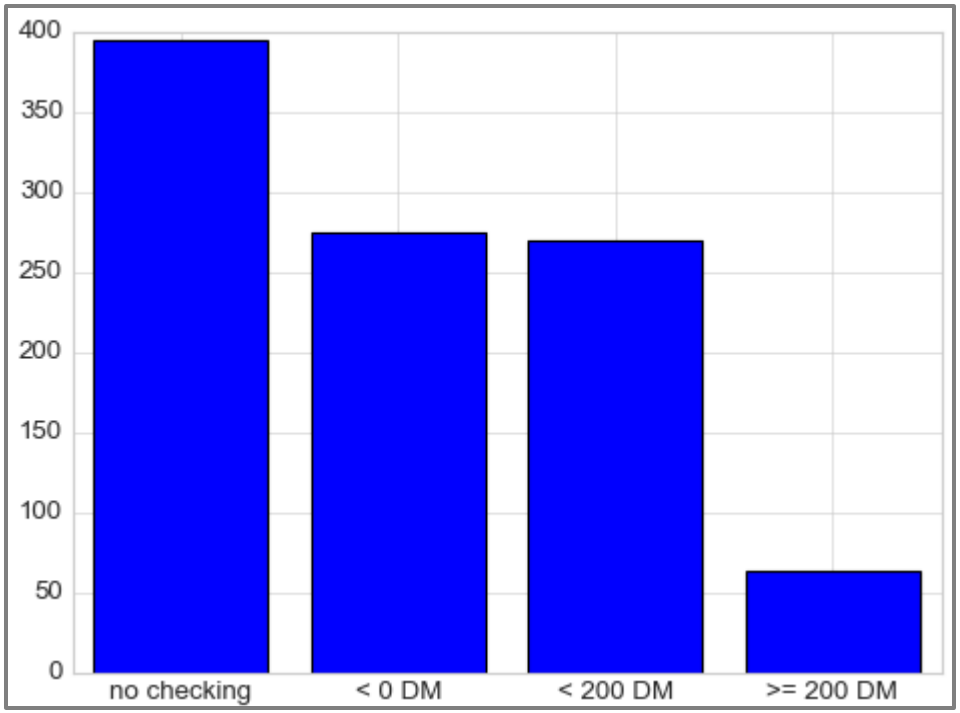
Basic histogram

```
plt.hist(df['age.in.years'])
plt.title('Histogram of borrower age')
plt.xlabel('age (years)')
plt.ylabel('frequency')
```



Bar plot

```
status_counts = df['checking.status'].value_counts()
labels = status_counts.index   # label bars with series index
x_pos = np.arange(len(labels)) # bar positions
plt.bar(x_pos, status_counts)
plt.xticks(x_pos, labels)
```



Matplotlib has 2 different interfaces

state-machine interface, like MATLAB
use functions like plt.plot() and plt.title()

Easier

```
plt.hist(df['age.in.years'])
plt.title('Histogram of borrower age')
plt.xlabel('age (years)')
plt.ylabel('frequency')
```

object-oriented interface
use methods on figure, axes objects
like ax.plot() and ax.set_title()

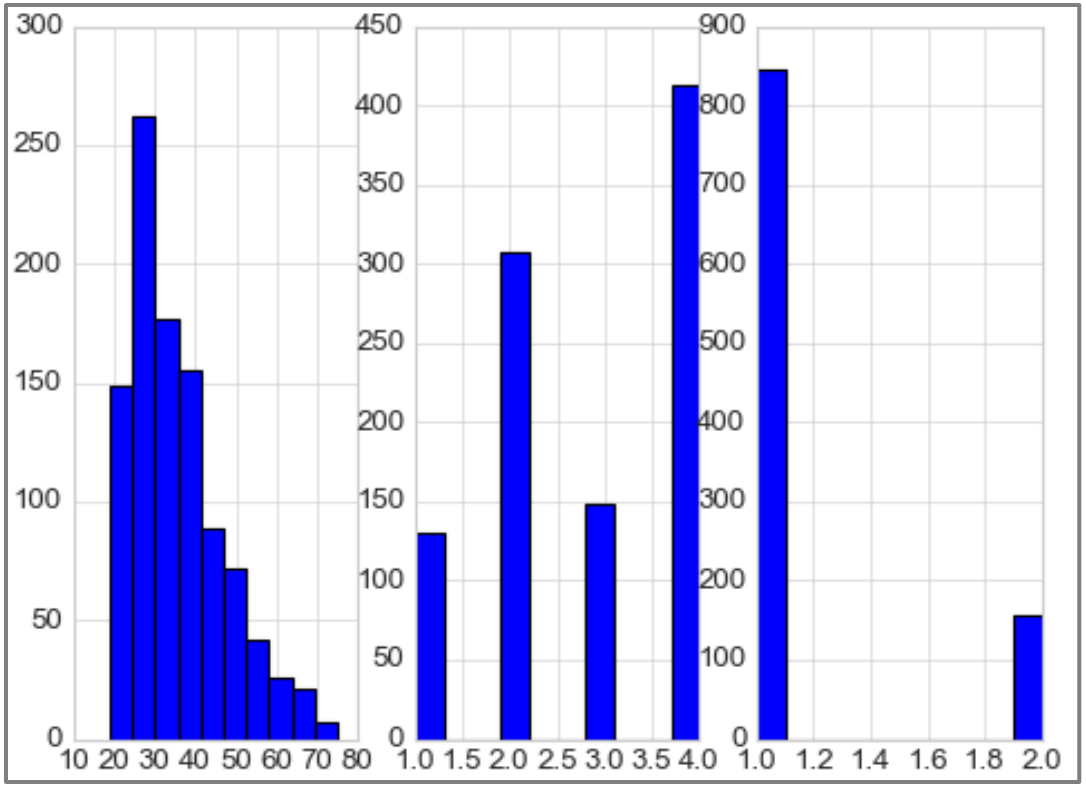
More control

```
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x))
ax.set_title('sinusoid')
```

To understand figure, axis, etc. see the matplotlib Usage Guide

Multiple subplots

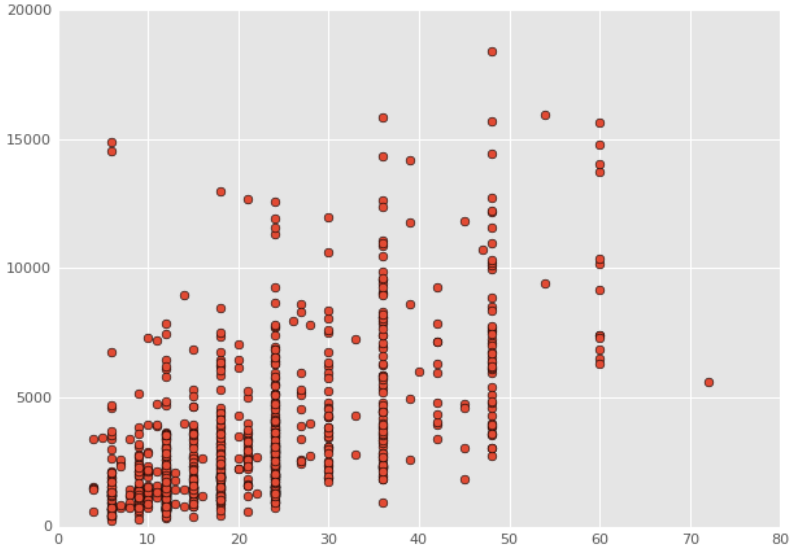
```
fig, ax = plt.subplots(1,3)
ax[0].hist(df['age.in.years'])
ax[1].hist(df['at.residence.since'])
ax[2].hist(df['num.dependents'])
```



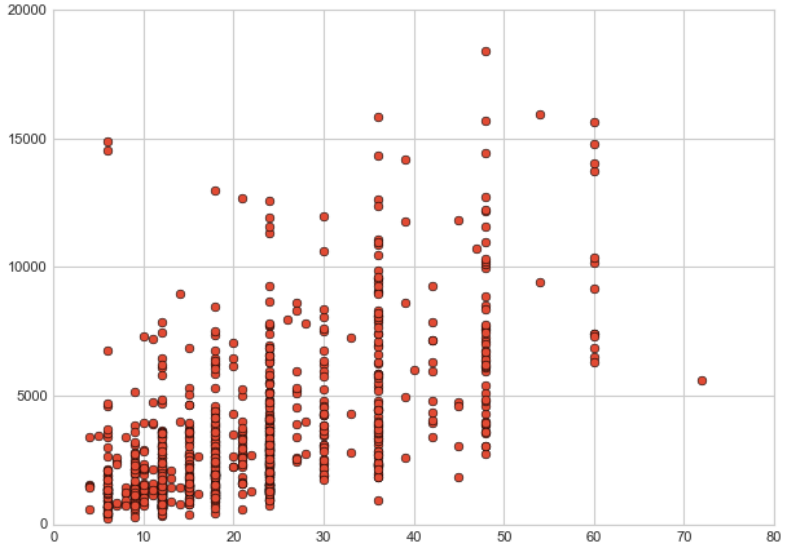
See p. 222 of text for the MATLAB-style approach.

Plotting styles

```
# list some of the available styles
plt.style.available[0:10]
# set the style
plt.style.use('ggplot')
# plot
plt.plot(df['duration.in.months'], df['amount'], 'o')
```



'ggplot'



'seaborn-whitegrid'

Saving a plot to a file

```
fig = plt.figure()
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
fig.savefig('sincos.png')
```

Pandas

```
# lineplot
df = pd.Series(np.sin(x), index=x)
df.plot()

# scatterplot
df.plot.scatter(x='duration.in.months', y='amount',
                color='darkred')

# barplot
status_counts = df['checking.status'].value_counts()
status_counts.plot.bar()
# alternatively
status_counts.plot(kind='bar')

# histogram
df['age.in.years'].plot.hist()
```

Seaborn

```
import seaborn as sns

# line plot
sns.lineplot(x=x, y=np.sin(x))

# scatterplot
sns.scatterplot(x='duration.in.months', y='amount', data=df)

# barplot
sns.countplot(df['checking.status'], color='darkred')

# histogram
sns.distplot(df['age.in.years'])
```

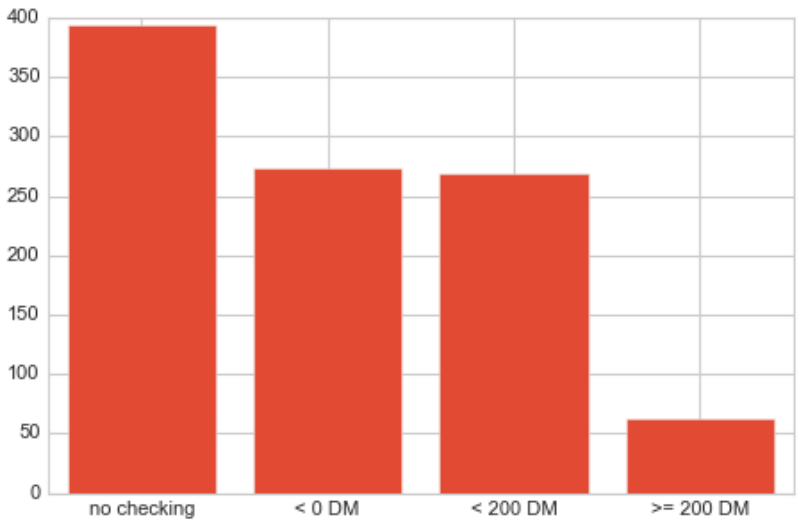
Saving a seaborn plot to a file:

```
sns_plot = sns.lineplot(x=x, y=np.sin(x))
sns_plot.savefig('output.png')
```

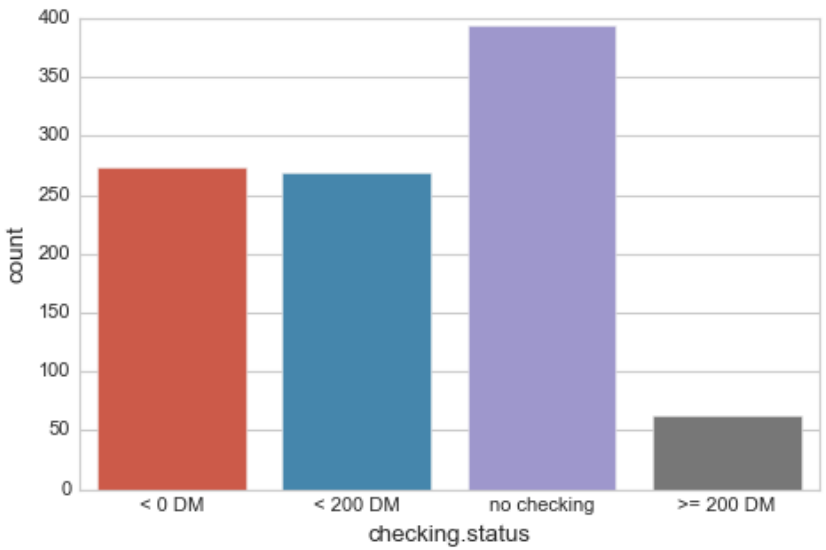
Seaborn vs matplotlib

```
status_counts = df['checking.status'].value_counts()
labels = status_counts.index
y_pos = np.arange(len(labels))
plt.bar(y_pos, status_counts)
plt.xticks(y_pos, labels)
```

```
sns.countplot(df['checking.status'])
```



matplotlib

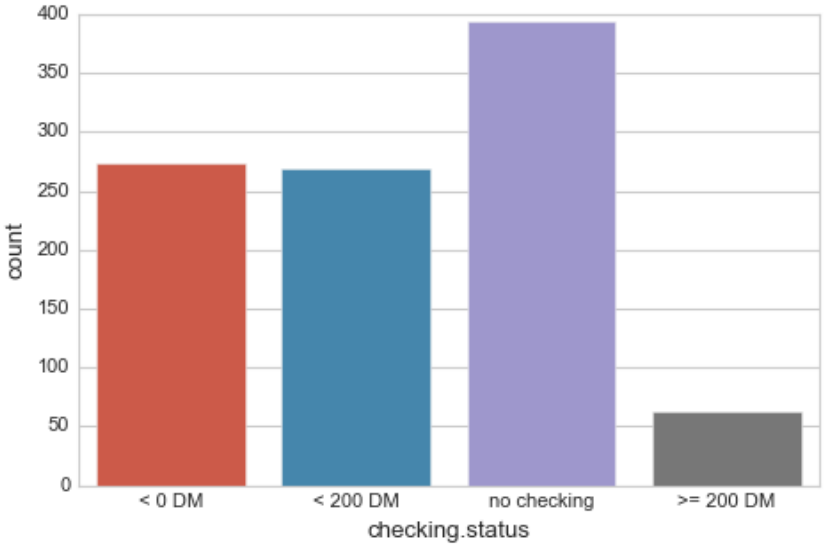


seaborn

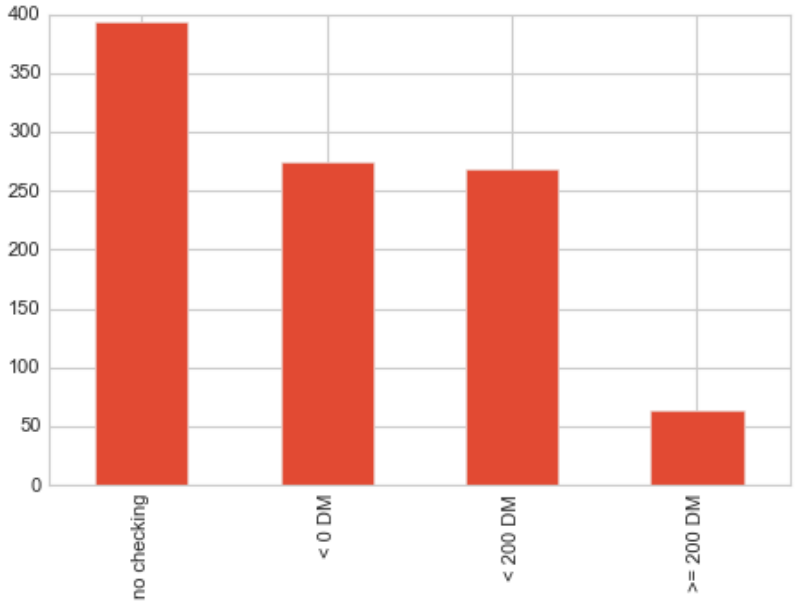
Seaborn vs pandas

```
sns.countplot(df['checking.status'])
```

```
status_counts = df['checking.status'].value_counts()
status_counts.plot.bar()
```



seaborn



pandas

Plotting in a notebook

When plotting in a notebook, use ';' after last plotting command in the cell.

```
sns.countplot(df['checking.status'])
plt.title('Checking account status');
```

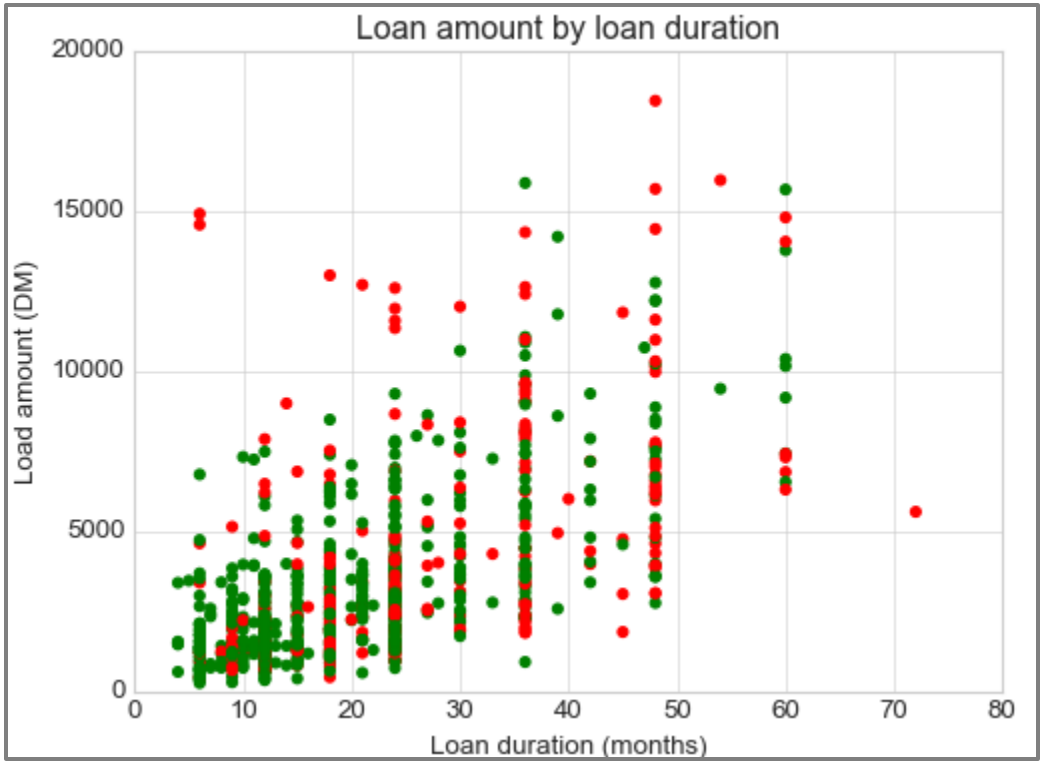
This hides text output from plotting command.

Advanced: Fancier scatter plot

```
colors = ['red' if x == 'BadLoan' else 'green' \
            for x in df['Good.Loan']]
plt.scatter(df['duration.in.months'], df['amount'],
            c=colors, edgecolors='face')
```

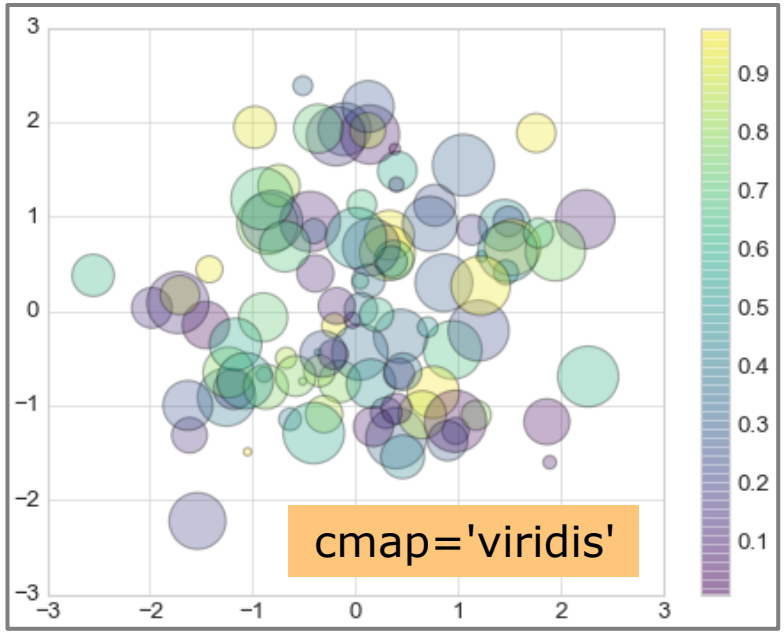
Another way to do this is to split the data into the bad loan and good loan data, and then use two plt.scatter() calles.

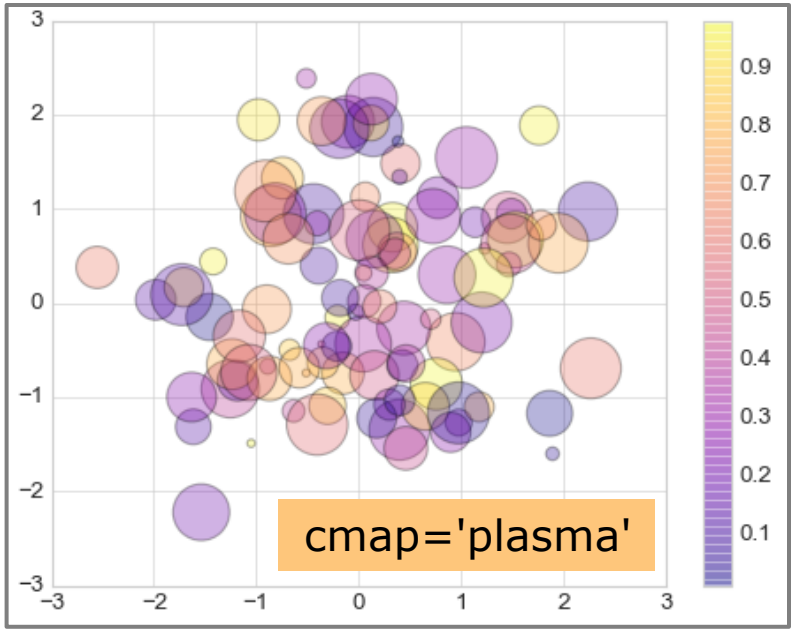
Every point can be given its own color, shape, etc.



Advanced: even fancier scatter plot

```
rng = np.random.RandomState(0)  # set random seed
x = rng.randn(100)              # 100 random samples
y = rng.randn(100)
colors = rng.rand(100)          # here, colors are numbers
sizes = 1000 * rng.rand(1000)
plt.scatter(x, y, c=colors, s=sizes, alpha=0.3, cmap='viridis')
plt.colorbar(), edgecolors='face')
```





Summary

We learned how to plot with matplotlib:

- create line plots, scatter plots, bar plots, and histograms
- modify the plot style
- add titles, axis labels, legends
- create multi-plots
- store plots as files

We also briefly saw how to plot with Seaborn and Pandas.

Learning outcomes

After this lecture you should be able to:

1. Perform basic plotting with matplotlib, Seaborn, and Pandas
2. Save your plots to files