Python Program

Chapter 5: Plotting and Visualization

Chapter Objectives

In this chapter, we will introduce:

- Matplotlib
- Plotting functions in pandas
- Python visualization tool ecosystem

Chapter Concepts

Introducing Matplotlib

- `matplotlib` is a plotting package designed for creating publication quality plots
  - Has a number of add-on toolkits
    - 3D plots
    - Mapping and projections
- `pyplot` is a module built on `matplotlib` usually imported as `plt`
- Run in `pylab` mode in IPython
- In this chapter, we provide enough detail to begin working with `matplotlib`
  - Full documentation including extensive examples can be found at:
    - http://matplotlib.org/2.0.0/index.html

Figures and Subplots

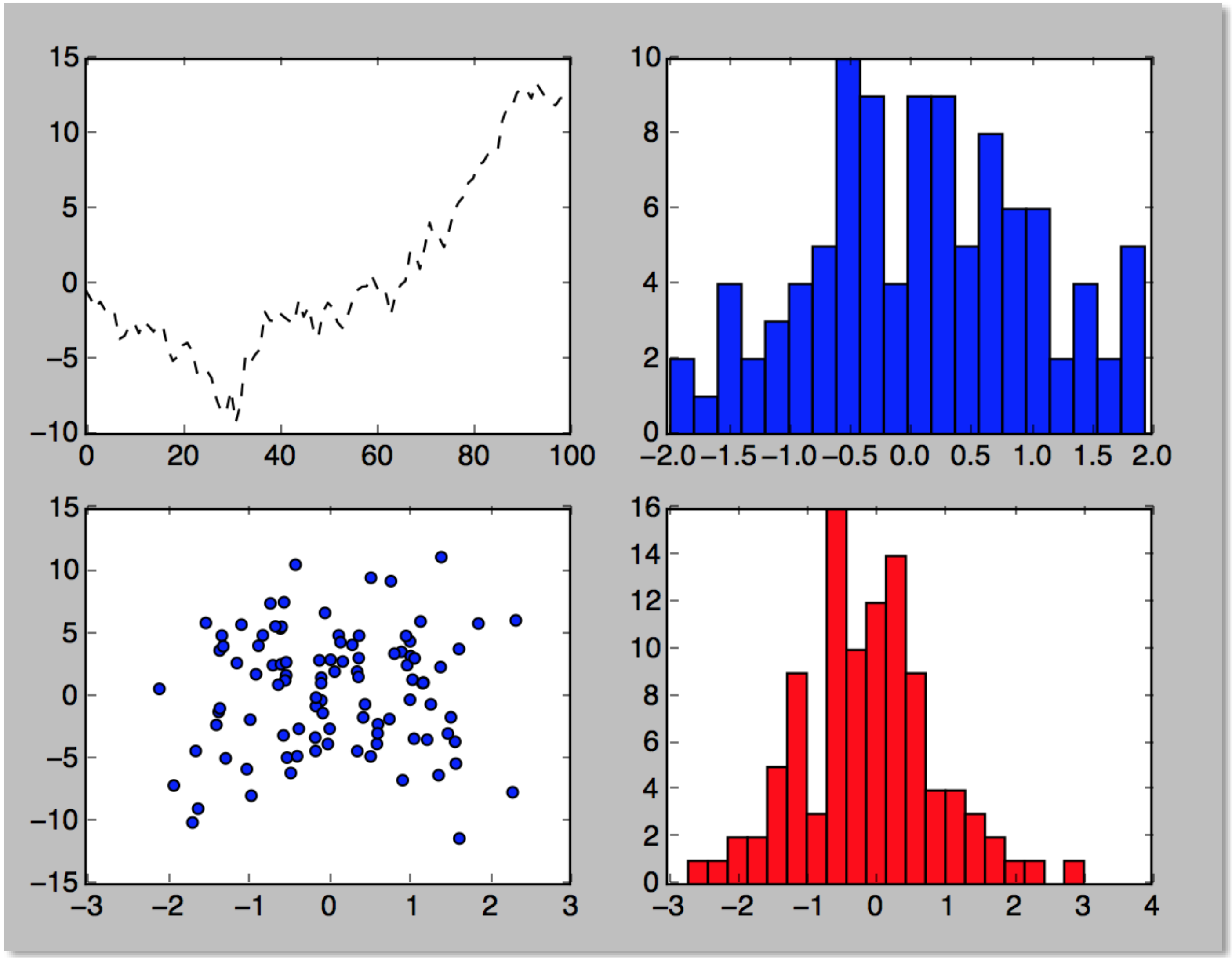
- Plots reside within a `Figure` object
- Subplots are added to a `Figure` object
  - Using `add_subplot(rows, columns, plot number)`
  - Returns `AxesSubplot` objects

```
import matplotlib
from matplotlib import pyplot as plt
from numpy.random import randn

figure = plt.figure()

sp1 = figure.add_subplot(2,2,1)
sp2 = figure.add_subplot(2,2,2)
sp3 = figure.add_subplot(2,2,3)
sp4 = figure.add_subplot(2,2,4)

sp1.plot(randn(100).cumsum(), 'k--')
sp2.hist(randn(100), bins=20)
sp3.scatter(randn(100), randn(100)-5*randn(100))
sp4.hist(randn(100), bins=20, color='r')
plt.show()
```

2x2 subplots, subplot 1

2x2 subplots, subplot 2

Figures and Subplots Example



Saving Plot

- Plots can be saved using the `savefig` method
- Various file formats are supported and can be listed with the following command:

```
plt.gcf().canvas.get_supported_filetypes_grouped()

{'Postscript': ['ps'],
 'Encapsulated Postscript': ['eps'],
 'Portable Document Format': ['pdf'],
 'PGF code for LaTeX': ['pgf'],
 'Portable Network Graphics': ['png'],
 'Raw RGBA bitmap': ['raw', 'rgba'],
 'Scalable Vector Graphics': ['svg', 'svgz'],
 'Joint Photographic Experts Group': ['jpeg', 'jpg'],
 'Tagged Image File Format': ['tif', 'tiff']}
```

- Using the extension indicates which format to save as

```
plt.savefig('chart1.jpg')
plt.savefig('chart1.pdf')
```

Colors and Styles

- The plot function accepts arrays of x and y coordinates and also an optional string
  - Optional string is for color and style
    - E.g., `sp1.plot(x, y, 'r--')`
      - `r` indicates red color and `--` is the dashed style
- More explicit requests for color and style can be made
  - E.g., `sp1.plot(x, y, linestyle='--', color='r')`
- Plots will have continuous line plots and, therefore, will have data interpolated
  - Can request data points to be shown
    - E.g., `sp1.plot(x, y, 'ro--')`
    - Or `sp1.plot(x, y, linestyle='--', color='r', marker='o')`

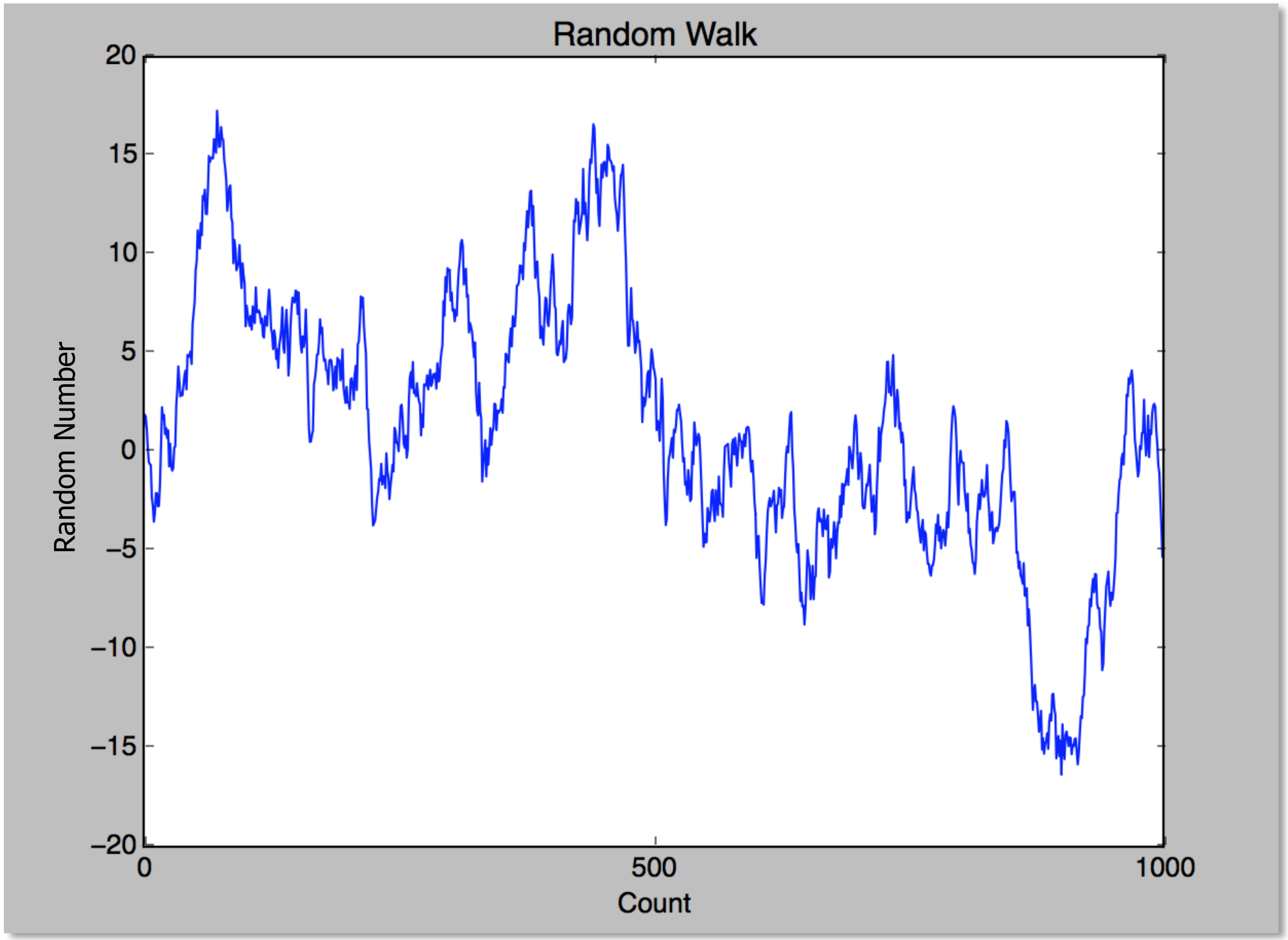
Labels and Legends

- Following example shows how to change axis ticks, labels, and add a title

```
figure = plt.figure()
p1 = figure.add_subplot(1,1,1)

p1.plot(randn(1000).cumsum())
p1.set_title('Random Walk')

p1.set_xticks([0,500,1000])

p1.set_xlabel('Count')

p1.set_ylabel('Random Number')
```

`set_yticks` for Y axis

Labels and Legends Example



Chapter Concepts

Plotting Functions in Pandas

- Pandas objects have built-in plotting functions
  - Simplify working with Matplotlib
    - In particular, for DataFrame objects
- Provide support for a number of different chart types such as:
  - Line plots
  - Bar plots
  - Histograms
  - Density plots
  - Scatter plots
  - Etc.

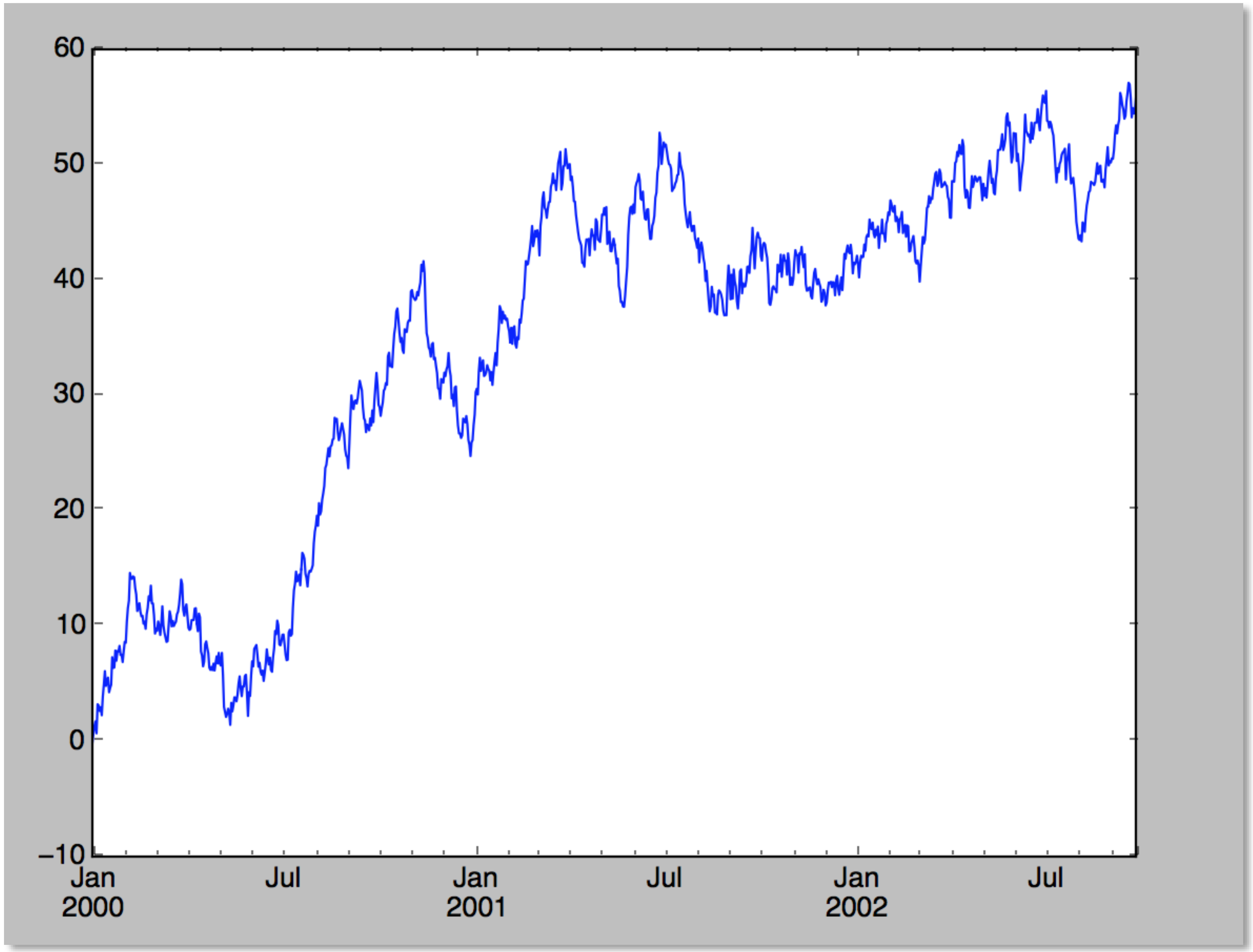
A Simple Example

- Consider plotting a series of data

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame

ts = Series(np.random.randn(1000), \
      index=pd.date_range('1/1/2000', periods=1000))

ts = ts.cumsum()

ts.plot()
```

Built-in function

A Simple Example (continued)



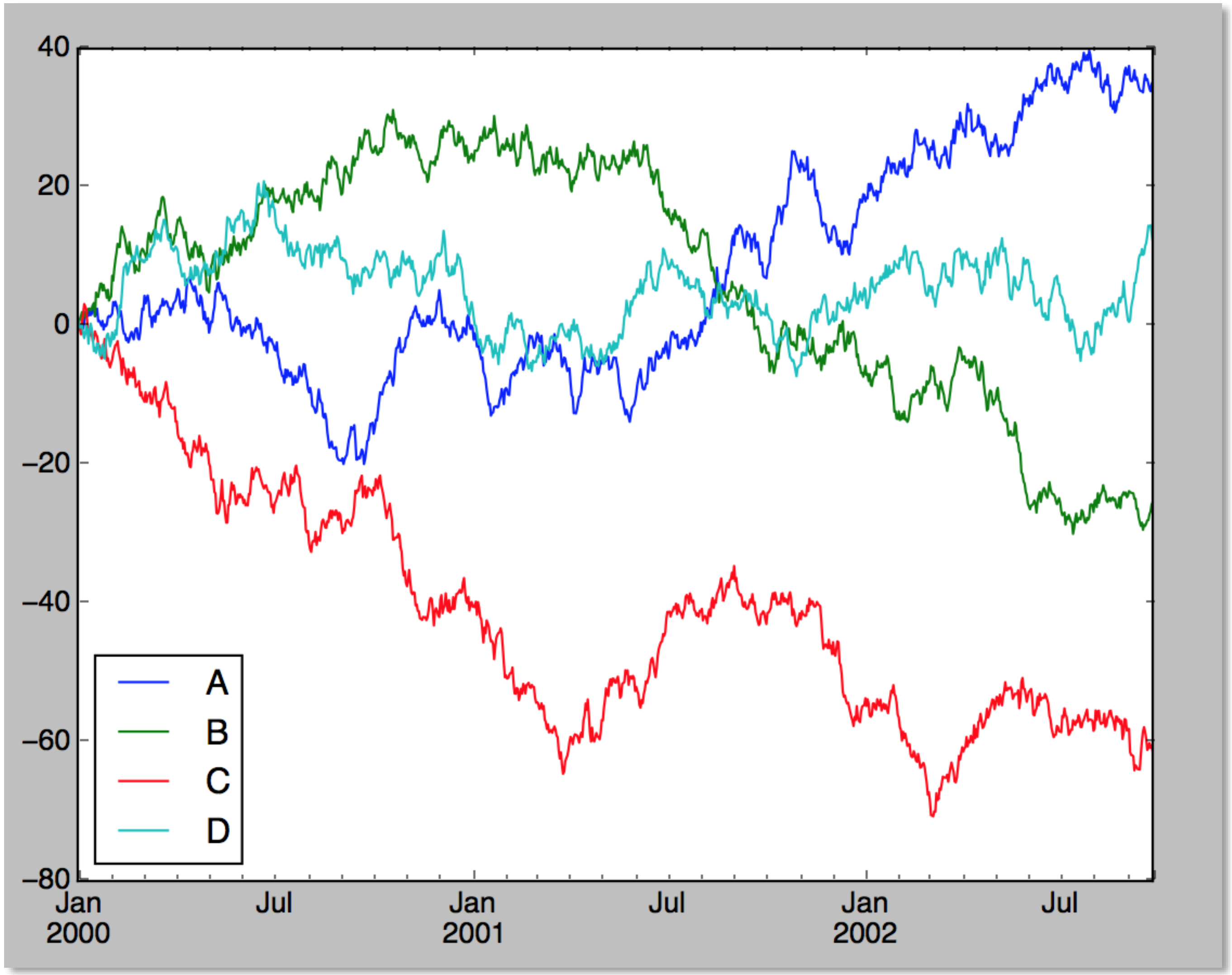
Line Plot with DataFrame

- `DataFrame’s` plot method plots each of its columns as a different line
  - On the same plot
  - A legend is created automatically

```
ts = Series(np.random.randn(1000), \
            index=pd.date_range('1/1/2000', periods=1000))

df = DataFrame(np.random.randn(1000,4), \
            index= ts.index, columns=list('ABCD'))

df = df.cumsum()

df.plot()
```

Used in legend

Line Plot with `DataFrame` (continued)



`Series plot()` Arguments

- Series plots can be customized using arguments to `plot()`
  - `label`
    - Label for plot legend
  - `style`
    - String such as `'g--'` for Matplotlib
  - `alpha`
    - Fill opacity from 0 to 1
  - `kind`
    - Line, bar, barh, kde
  - `grid`
    - Display axis grid
  - `logy`
    - Use logarithmic scaling on the Y axis
- For full list, see:
  - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.plot.html

`DataFrame plot()` Arguments

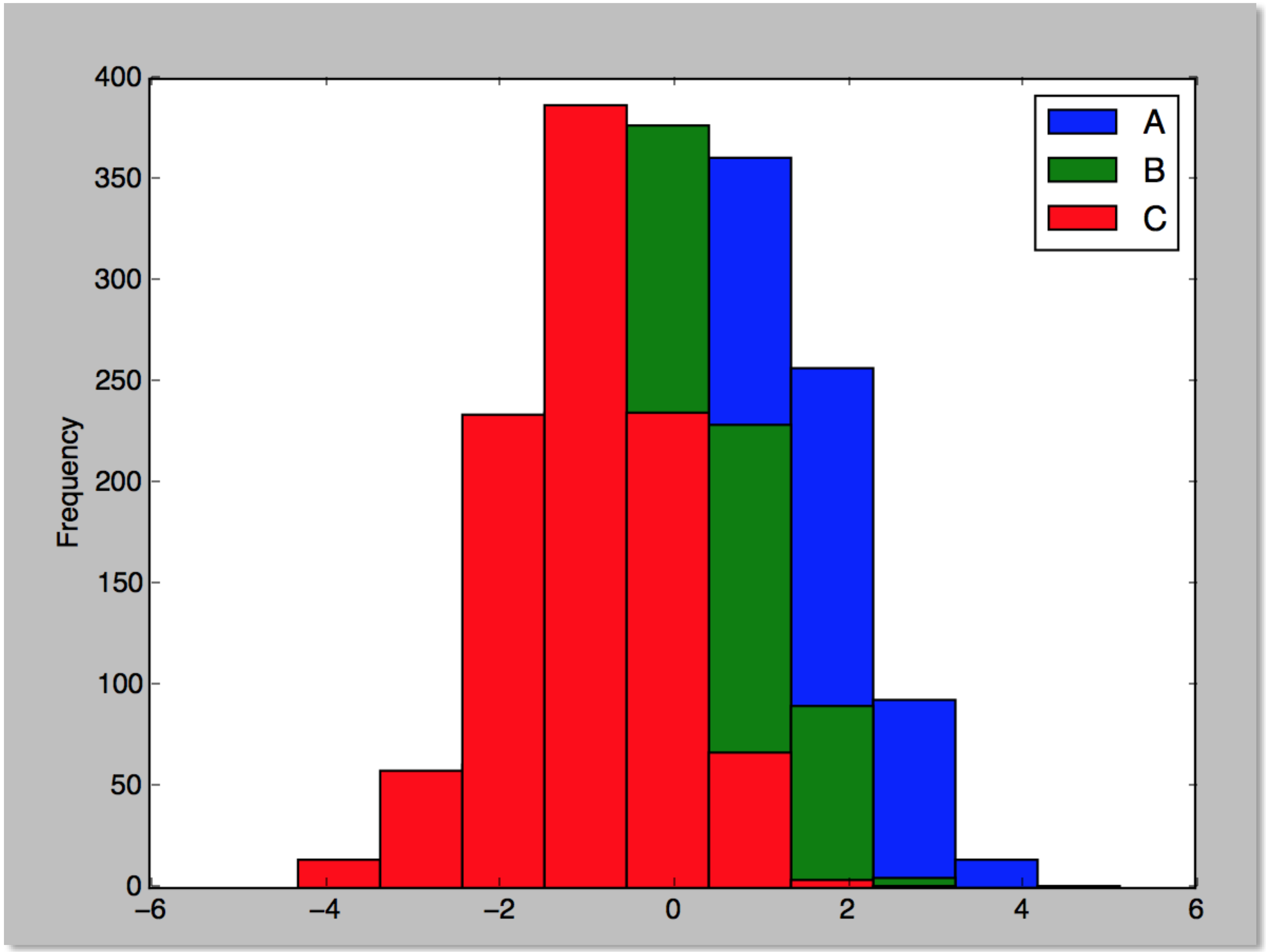
- Series plots can be customized using arguments to `plot()`
  - `subplots`
    - Plot each DataFrame in separate subplot
  - `sharex`
    - Share same x axis for subplots
  - `sharey`
    - Share same y axis for subplots
  - `figsize`
    - Size of figure to create
  - `title`
    - Plot title as a string
  - `legend`
    - Add a subplot legend
  - `sort_columns`
    - Plot columns in alphabetical order using existing column order
- For full list, see:
  - http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html

Histogram Example

```
df = pd.DataFrame({'A': np.random.randn(1000) + 1, \
          'B': np.random.randn(1000), \
          'C': np.random.randn(1000)-1})

print (df.head())

          A         B         C
0  0.627152  1.984009  0.785683
1  1.316856  0.318605  0.143795
2 -0.763011 -0.261403 -1.346760
3  1.174517  1.044114  0.556043
4  1.052025 -0.021766 -1.868798

df.plot(kind='hist')
```

Select histogram

Histogram Example (continued)



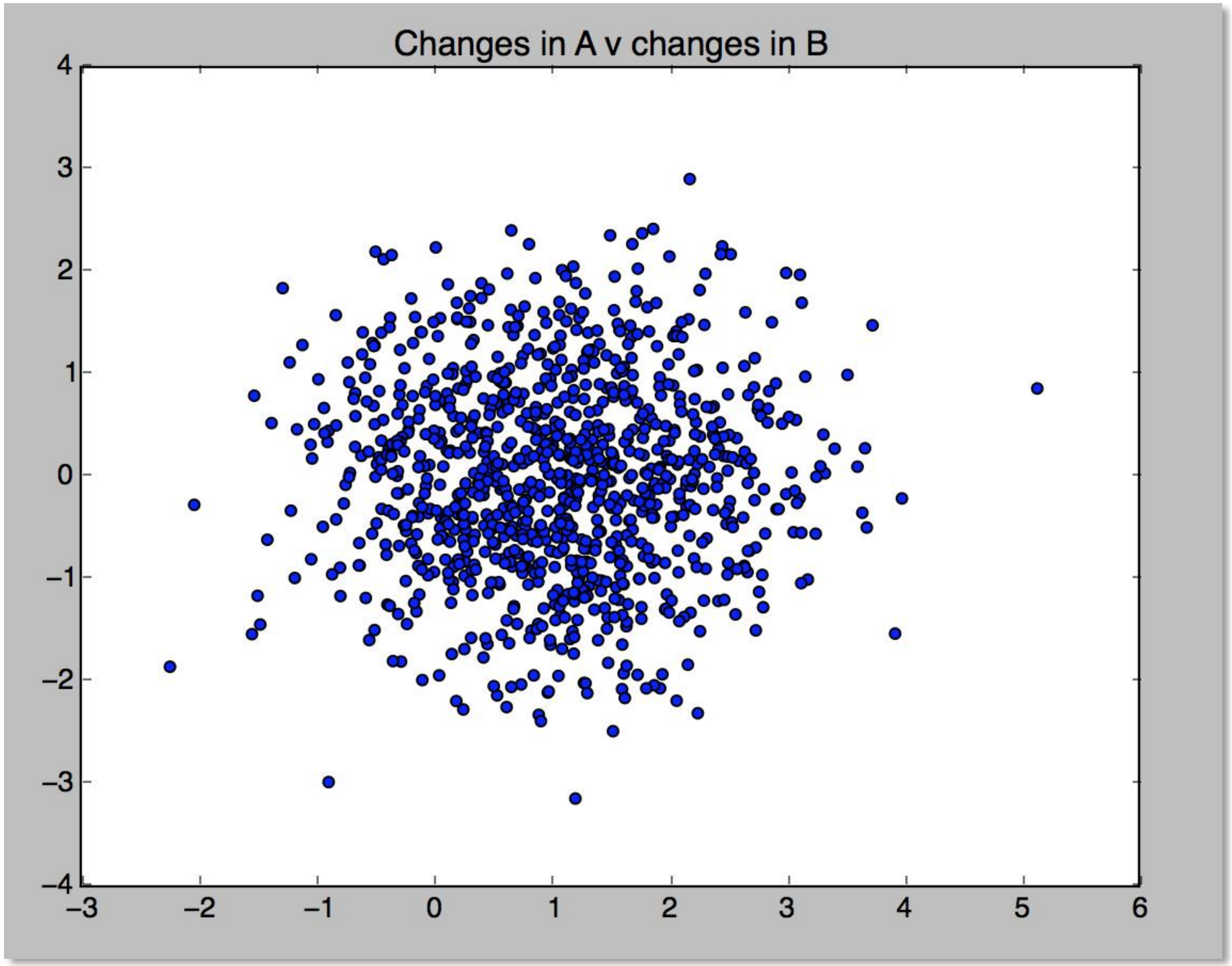
Scatter Plots

- Useful way of visualizing relationship between two one-dimensional data series
- `matplotlib` and `pyplot` have a scatter method for plotting charts

```
df.head()

          A         B         C
0  0.627152  1.984009  0.785683
1  1.316856  0.318605  0.143795
2 -0.763011 -0.261403 -1.346760
3  1.174517  1.044114  0.556043
4  1.052025 -0.021766 -1.868798

plt.scatter(df['A'], df['B'])

plt.title('Changes in A v changes in B')
```

Scatter plot

Scatter Plots (continued)



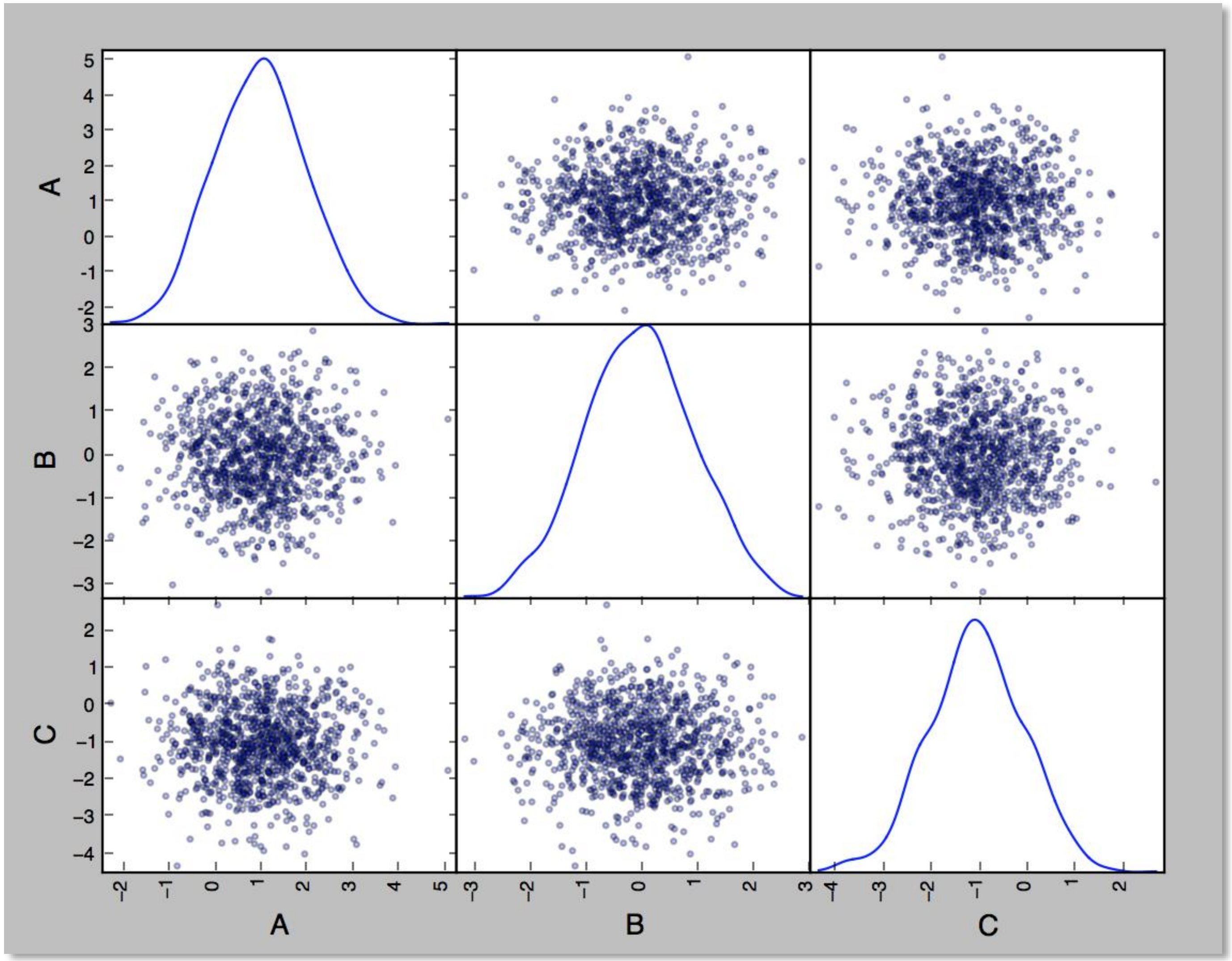
Scatter Plot Matrix

- For exploratory data analysis, it may be helpful to look at all scatter plots amongst a group of variables
  - Known as a pair plot or scatter plot matrix
- Pandas has `scatter_matrix` function
  - Works with a `DataFrame`
  - Also supports placing histograms or density plots of each variable along the diagonal

```
df.head()

          A         B         C
0  0.627152  1.984009  0.785683
1  1.316856  0.318605  0.143795
2 -0.763011 -0.261403 -1.346760
3  1.174517  1.044114  0.556043
4  1.052025 -0.021766 -1.868798

pd.plotting.scatter_matrix(df, diagonal='kde', alpha=0.3)
```

Density plot on diagonal

Scatter Plot Matrix Example



Chapter Concepts

Seaborn

- A library to simplify making presentation quality graphs from matplotlib
  - Short-cut methods to create advanced graphics
    - Especially combining more than one graph
  - Themes and palettes to simplify applying consistent styles
    - Applies themes immediately to all matplotlib graphs
    - Even those created without Seaborn
- To use
  - Usually imported as `sns`

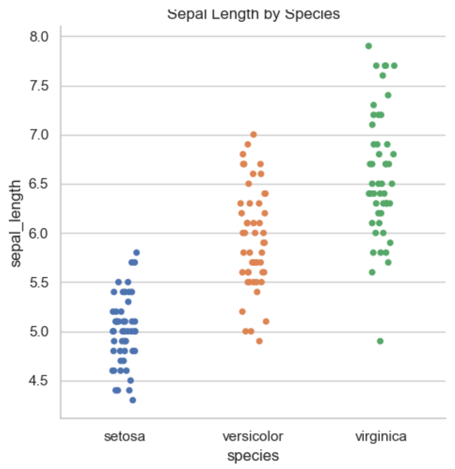
```
import seaborn as sns
sns.set()
```

Apply basic pre-set style

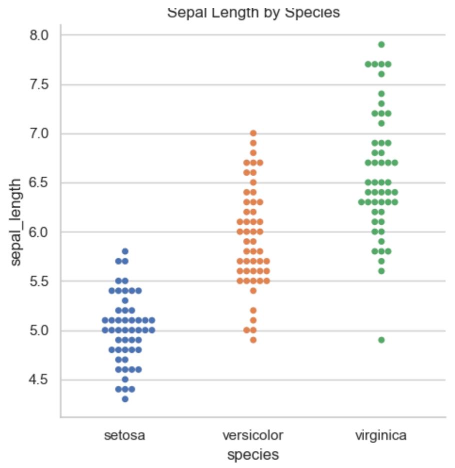
- Many Seaborn methods accept additional parameters
  - Passed to the underlying implementation (e.g., matplotlib, pandas)
  - Sometimes makes it hard to understand all the options
- Seaborn has many options
  - Too many for this course to cover all of them
  - The following slides pick out some key areas of interest

Categorical Plots

- Compare a numerical value with one, or more, categories
  – Using scatter, boxes, violins, error bars, histograms

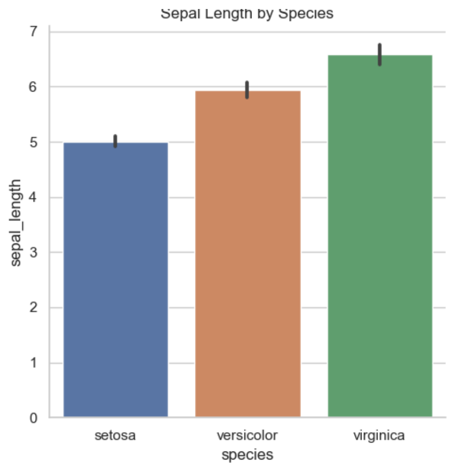


```
iris = sns.load_dataset('iris');
ax = sns.catplot(
  x = 'species',
  y = 'sepal_length',
  data = iris,
  kind = 'strip'
);
```

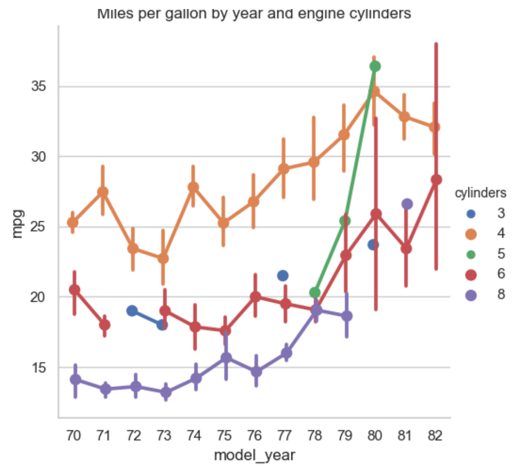


```
iris = sns.load_dataset('iris');
ax = sns.catplot(
  x = 'species',
  y = 'sepal_length',
  data = iris,
  kind = 'swarm'
);
```

Categorical Plots (continued)



```
iris = sns.load_dataset('iris');
ax = sns.catplot(
  x = 'species’,
  y = 'sepal_length’,
  data = iris,
  kind = bar’
);
```



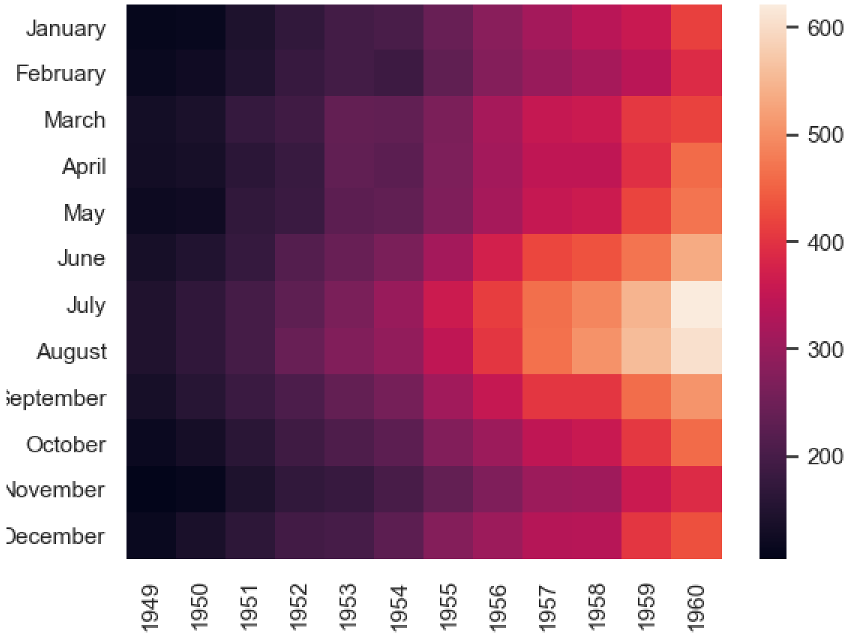
```
mpg = sns.load_dataset('mpg')
ax = sns.catplot(
  x = 'model_year',
  y = 'mpg',
  data = mpg,
  kind = 'point',
  hue = 'cylinders',
  dodge = True
);
```

Automatically calculates mean and shows error bars

Heatmaps

- Use color to show the scale of data at the intersection of two categories
  - A colored matrix
  - Good for highlighting correlation

```
flights = sns.load_dataset('flights')
flights = flights.pivot('month', 'year', 'passengers')
ax = sns.heatmap(flights)
```



Other Data Visualization Tools

- Plotly
  - Based on the popular plotly.js library
  - Creates interactive plots
- Folium
  - Visualize geospatial data on maps
- Ggplot
  - Graphing package based on ggplot2 from R
  - Uses *The Grammar of Graphics* to create plots at a high level without thinking about implementation details
- Bokeh
  - Also based on *The Grammar of Graphics*
  - Create interactive plots
- Altair
  - A declarative library based on the Vega-lite visualization grammar

Chapter Concepts

Chapter Summary

In this chapter, we have introduced:

- Matplotlib
- Plotting functions in pandas
- Python visualization tool ecosystem