Numpy, Pandas, matplotlib and Seaborn

Objectives

After completing this module, you should be able to:

Numpy Ndarrays

Descriptive Statistics with Numpy

Linear Algebra with Numpy

Pandas Series, Pandas DataFrame & Pandas Panel

Data Import and Export using Pandas

Selection, Filtering and Merging Data Frames

Removing Duplicates, Missing Values Handling, String Manipulation

Basic Visualization using Matplotlib

Boxplot, Heatmap, Distributionmap, violinplots, clustermap using seaborn

numpy

NumPy contains:

- A powerful N-dimensional array object.
- Sophisticated (broadcasting/universal) functions.
- Useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

3

Numpy arrays

In general, any numerical data that is stored in an array-like container can be converted to an ndarray through use of the array() function. The most obvious examples are sequence types like lists and tuples.

```
# Import modules
import numpy as np
# Create an array
civilian_birth = np.array([4352, 233, 3245, 256, 2394])
civilian_birth
#output
array([4352,  233, 3245,  256, 2394])
```


4

Create ndarrays from lists

```
data1 = [1, 2, 3, 4, 5]
        # list
arr1 = np.array(data1)
        # 1d array
data2 = [range(1, 5), range(5, 9)]                # list of lists
arr2 = np.array(data2)
        # 2d array
arr2.tolist()
        # convert array back to list
```


5

create special arrays

```
np.zeros(10)
np.zeros((3, 6))
np.ones(10)
np.linspace(0, 1, 5) # 0 to 1 (inclusive) with 5 points
np.logspace(0, 3, 4) # 10^0 to 10^3 (inclusive) with 4 points
```


6

Reshaping arrays

```
arr = np.arange(10, dtype=float).reshape((2, 5))
print(arr.shape)
print(arr.reshape(5, 2))
```


7

Stack arrays

```
a = np.array([0, 1])
b = np.array([2, 3])
ab = np.stack((a, b)).T
print(ab)
# or
np.hstack((a[:, None], b[:, None]))
```


8

Generating Random Numbers With NumPy

```
import numpy as np
#Generate A Random Number From The Normal Distribution
np.random.normal()

#Generate Four Random Numbers From The Normal Distribution
np.random.normal(size=4)

#Generate Four Random Numbers From The Uniform Distribution
np.random.uniform(size=4)

#Generate Four Random Integers Between 1 and 100
np.random.randint(low=1, high=100, size=4)
```


9

Linear Algebra with Numpy

Matrices

There is also a matrix class which inherits from the ndarray class.

There are some slight differences but matrices are very similar to general arrays.

In NumPy's own words, the question of whether to use arrays or matrices comes down to the short answer of "use arrays".

11

Transpose A Vector Or Matrix

```
import numpy as np
# Create vector
vector = np.array([1, 2, 3, 4, 5, 6])
# Create matrix
matrix = np.matrix([[1, 2, 3],
                    [4, 5, 6],
                    [7, 8, 9]])
vector.T                          # Tranpose vector
matrix.T                  # Transpose matrix
```


12

Flatten A Matrix

```
import numpy as np
# Create matrix
matrix = np.matrix([[1, 2, 3],
                    [4, 5, 6],
                    [7, 8, 9]])


# Flatten matrix
matrix.flatten()
```


13

Difference between numpy dot() and inner()

```
a=np.array([[1,2],[3,4]])
b=np.array([[11,12],[13,14]])
np.dot(a,b)
#output
array([[37, 40],
       [85, 92]])

np.inner(a,b)
#otuput
array([[35, 41],
       [81, 95]])
```

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}11 & 12\\13 & 14\end{bmatrix}$$

With dot():

$$\begin{bmatrix}1*11+2*13 & 1*12+2*14\\3*11+4*13 & 3*12+4*14\end{bmatrix}=\begin{bmatrix}37 & 40\\85 & 92\end{bmatrix}$$

With inner():

$$\begin{bmatrix}1*11+2*12 & 1*13+2*14\\3*11+4*12 & 3*13+4*14\end{bmatrix}=\begin{bmatrix}35 & 41\\81 & 95\end{bmatrix}$$

14

Linear Algebra

```
# Return determinant of matrix
np.linalg.det(matrix)


# Calculate the tracre of the matrix
matrix.diagonal().sum()


# Calculate inverse of matrix
np.linalg.inv(matrix)
```


15

Linear Algebra

```
# Add two matrices
np.add(matrix_a, matrix_b)


# Subtract two matrices
np.subtract(matrix_a, matrix_b)
```


16

Eigen Values & Eigen Vectors

```
import numpy as np
from numpy.linalg import eig
A = np.array([[1,2],[3,4]])
eig(A)


#output
(array([-0.37228132, 5.37228132]), array([[-0.82456484, -
0.41597356], [ 0.56576746, -0.90937671]]))
```


17

Linear Algebra - Solving Equations

```
import numpy as np
from numpy.linalg import solve
A = np.array([[4,5],[6,-3]])
b = np.array([23, 3])
x = solve(A,b)
#output - value of x
array([ 2, 3])
```

$$4x + 5y = 23$$
$$6x - 3y = 3$$

18

Getting started with pandas

Pandas Series

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.).

```
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print(s)
```


20

Pandas Series

```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve a single element
print(s['a’])
```


21

Pandas DataFrame

A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

```
# Create an dataframe
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'year': [2012, 2012, 2013, 2014, 2014],
        'reports': [4, 24, 31, 2, 3]}
df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa
Cruz', 'Maricopa', 'Yuma'])
```


22

Pandas DataFrame

```
raw_data = {'first_name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', 'Jacobson', ".", 'Milner', 'Cooze'],
        'age': [42, 52, 36, 24, 73],
        'preTestScore': [4, 24, 31, ".", "."],
        'postTestScore': ["25,000", "94,000", 57, 62, 70]}
```


23

Pandas DataFrame

```
import pandas as pd
import numpy as np


df = pd.DataFrame(raw_data, columns = ['first_name',
'last_name', 'age', 'preTestScore', 'postTestScore'])
```


24

Pandas Panel

A **panel** is a 3D container of data.

The names for the 3 axes are intended to give some semantic meaning to describing operations involving panel data. They are –

- **items** – axis 0, each item corresponds to a DataFrame contained inside.
- **major_axis** – axis 1, it is the index (rows) of each of the DataFrames.
- **minor_axis** – axis 2, it is the columns of each of the DataFrames.

25

Pandas Panel

```
import pandas as pd
import numpy as np
data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
        'Item2' : pd.DataFrame(np.random.randn(4, 2))}
p = pd.Panel(data)
print(p['Item1'])
print(p.major_xs(1))
print(p.minor_xs(1))
```


26

Loading A CSV Into pandas

```
# Load a csv
df =
pd.read_csv('https://raw.githubusercontent.com/anshupandey/datasets/ma
ster/datawh.csv’)

# Load a csv with no headers
df =
pd.read_csv(https://raw.githubusercontent.com/anshupandey/datasets/mas
ter/datanh.csv', header=None)
```


27

Export DataFrame

```
# Save dataframe as csv in the working director
df.to_csv('pandas_dataframe_importing_csv/example.csv’)
```


28

Load A JSON File Into Pandas

```
import pandas as pd
url =
'https://raw.githubusercontent.com/anshupandey/datasets/master/
server-metrics.json'
# Load the first sheet of the JSON file into a data frame
df = pd.read_json(url, orient='columns')
```


29

Descriptive Statistics with Pandas

Descriptive Statistics For pandas Dataframe

```
import pandas as pd
data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'age': [42, 52, 36, 24, 73],
        'preTestScore': [4, 24, 31, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70]}
df = pd.DataFrame(data, columns = ['name', 'age',
'preTestScore', 'postTestScore'])
```


31

Descriptive Statistics For pandas Dataframe

```
Summary statistics on preTestScore
df['preTestScore'].describe()


# The sum of all the ages
df['age'].sum()


# Mean preTestScore
df['preTestScore'].mean()
```


32

Descriptive Statistics For pandas Dataframe

```
# Cumulative sum of preTestScores, moving from the rows
df['preTestScore'].cumsum()

# Count the number of non-NA values
df['preTestScore'].count()

# Minimum value of preTestScore
df['preTestScore'].min()
```


33

Descriptive Statistics For pandas Dataframe

```
# Maximum value of preTestScore
df['preTestScore'].max()


# Median value of preTestScore
df['preTestScore'].median()


# Sample variance of preTestScore values
df['preTestScore'].var()
```


34

Descriptive Statistics For pandas Dataframe

```
# Sample standard deviation of preTestScore values
df['preTestScore'].std()


# Skewness of preTestScore values
df['preTestScore'].skew()


# Kurtosis of preTestScore values
df['preTestScore'].kurt()
```


35

Descriptive Statistics For pandas Dataframe

```
# Correlation Matrix Of Values
df.corr()


# Covariance Matrix Of Values
df.cov()
```


36

Data Manipulation with Pandas

Dropping Rows & Columns in Dataframe

```
Import pandas as pd


data = {'name': ['Jason', 'Molly', 'Tina', 'Jake', 'Amy'],
        'year': [2012, 2012, 2013, 2014, 2014],
        'reports': [4, 24, 31, 2, 3]}
df = pd.DataFrame(data, index = ['Cochice', 'Pima', 'Santa
Cruz', 'Maricopa', 'Yuma'])
```


38

Dropping Rows And Columns in Dataframe

```
# Drop an observation (row)
df.drop(['Cochice', 'Pima’])


# Drop a variable (column)
# axis=1 denotes that we are referring to a column, not a row
df.drop('reports', axis=1)
```


39

Dropping Rows And Columns in Dataframe

```
# Drop a row if it contains a certain value (in this case, “Tina”)
df[df.name != 'Tina’]


# Drop a row by row number (in this case, row 3)
df.drop(df.index[2])
# can be extended to dropping a range
df.drop(df.index[[2,3]])
```


40

Join And Merge Pandas Dataframe

```
import pandas as pd


#first Dataframe

raw_data = {
        'subject_id': ['1', '2', '3', '4', '5'],
        'first_name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
        'last_name': ['Anderson', 'Ackerman', 'Ali', 'Aoni',
'Atiches']}
df_a = pd.DataFrame(raw_data, columns = ['subject_id',
'first_name', 'last_name'])
```


41

Join And Merge Pandas Dataframe

```
# Create a second dataframe
raw_data = {
        'subject_id': ['4', '5', '6', '7', '8'],
        'first_name': ['Bill', 'Brian', 'Bran', 'Bryce', 'Betty'],
        'last_name': ['Bonder', 'Black', 'Balwner', 'Brice',
'Btisan']}
df_b = pd.DataFrame(raw_data, columns = ['subject_id',
'first_name', 'last_name'])
```


42

Join And Merge Pandas Dataframe

```
# Create a third dataframe
raw_data = {
        'subject_id': ['1', '2', '3', '4', '5', '7', '8', '9',
'10', '11'],
        'test_id': [51, 15, 15, 61, 16, 14, 15, 1, 61, 16]}
df_n = pd.DataFrame(raw_data, columns = ['subject_id','test_id'])
```


43

Join And Merge Pandas Dataframe

```
# Join the two dataframes along rows
df_new = pd.concat([df_a, df_b])

# Join the two dataframes along columns
pd.concat([df_a, df_b], axis=1)

# Merge two dataframes along the subject_id value
pd.merge(df_new, df_n, on='subject_id')
```


44

Find Unique Values In Pandas Dataframes

```
raw_data = {'regiment': ['51st', '29th', '2nd', '19th', '12th', '101st',
'90th', '30th', '193th', '1st', '94th', '91th'],
            'trucks': ['MAZ7310', np.nan, 'MAZ7310', 'MAZ7310', 'Tatra10',
'Tatra10', 'Tatra10', 'Tatra10', 'ZIS150', 'Tatra10', 'ZIS150', 'ZIS150'],
            tanks': ['Merkava Mark 4', 'Merkava Mark 4', 'Merkava Mark 4',
'Leopard 2A6M', 'Leopard 2A6M', 'Leopard 2A6M', 'Arjun MBT', 'Leopard
2A6M', 'Arjun MBT', 'Arjun MBT', 'Arjun MBT', 'Arjun MBT'],
            'aircraft': ['none', 'none', 'none', 'Harbin Z-9', 'Harbin Z-
9', 'none', 'Harbin Z-9', 'SH-60B Seahawk', 'SH-60B Seahawk', 'SH-60B
Seahawk', 'SH-60B Seahawk', 'SH-60B Seahawk']}
```


45

Find Unique Values In Pandas Dataframes

```
# Create a list of unique values by turning the
# pandas column into a set
list(set(df.trucks))


# Create a list of unique values in df.trucks
list(df['trucks'].unique())
```


46

Delete Duplicates In pandas

```
import pandas as pd
# Create dataframe with duplicates
raw_data = {'first_name': ['Jason', 'Jason', 'Jason','Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', 'Miller', 'Miller','Ali', 'Milner', 'Cooze'],
        'age': [42, 42, 1111111, 36, 24, 73],
        'preTestScore': [4, 4, 4, 31, 2, 3],
        'postTestScore': [25, 25, 25, 57, 62, 70]}
```


47

Delete Duplicates In pandas

```
df = pd.DataFrame(raw_data, columns = ['first_name',
'last_name', 'age', 'preTestScore', 'postTestScore’])


# Identify which observations are duplicates
df.duplicated()


df.drop_duplicates()
# Drop duplicates in the  name column, but take the last obs in
the duplicated set
df.drop_duplicates(['first_name'], keep='last')
```


48

groupby operations

```
raw_data = {'regiment': ['Nighthawks', 'Nighthawks', 'Nighthawks',

         'Nighthawks','Dragoons', 'Dragoons', 'Dragoons',
                                                  'Dragoons',
'Scouts','Scouts', 'Scouts', 'Scouts'],
        'company': ['1st', '1st', '2nd', '2nd', '1st', '1st', '2nd',
                                                      '2nd','1st',
'1st', '2nd', '2nd'],
        'name': ['Miller', 'Jacobson', 'Ali', 'Milner', 'Cooze', 'Jacon',
                                                  'Ryaner', 'Sone',
'Sloan', 'Piger', 'Riani', 'Ali'],
        'preTestScore': [4, 24, 31, 2, 3, 4, 24, 31, 2, 3, 2, 3],
        'postTestScore': [25, 94, 57, 62, 70, 25, 94, 57, 62, 70, 62, 70]}
```


49

groupby operations

```
Import pandas as pd
# Create dataframe
df = pd.DataFrame(raw_data, columns = ['regiment', 'company',
'name', 'preTestScore', 'postTestScore’])


# Create a groupby variable that groups preTestScores by regiment
groupby_regiment = df['preTestScore'].groupby(df['regiment’])


# Mean of each regiment’s preTestScore
groupby_regiment.mean()
```


50

groupby operations

```
# View a grouping - Use list() to show what a grouping looks like
list(df['preTestScore'].groupby(df['regiment’]))


# Descriptive statistics by group
df['preTestScore'].groupby(df['regiment']).describe()
```


51

groupby operations

```
# Mean preTestScores grouped by regiment and company
df['preTestScore'].groupby([df['regiment'], df['company']]).mean()


# Mean preTestScores grouped by regiment and company without
# heirarchical indexing
df['preTestScore'].groupby([df['regiment'],
df['company']]).mean().unstack()
```


52

groupby operations

```
# Group the entire dataframe by regiment and company
df.groupby(['regiment', 'company']).mean()

# Number of observations in each regiment and company
df.groupby(['regiment', 'company']).size()

# Group the dataframe by regiment, and for each regiment,
for name, group in df.groupby('regiment'):
    print(name) # print the name of the regiment
    print(group) # print the data of that regiment
```


53

Missing Data In pandas Dataframes

```
# Drop missing observations
df_no_missing = df.dropna()

# Drop rows where all cells in that row is NA
df_cleaned = df.dropna(how='all’)

# Create a new column full of missing values
df['location'] = np.nan
```


54

Missing Data In pandas Dataframes

```
# Drop column if they only contain missing values
df.dropna(axis=1, how='all’)


# Drop rows that contain less than five observations
# This is really mostly useful for time series
df.dropna(thresh=5)


# Fill in missing data with zeros
df.fillna(0)
```


55

Missing Data In pandas Dataframes

```
# inplace=True means that the changes are saved to the df right away
df["preTestScore"].fillna(df["preTestScore"].mean(), inplace=True)

# Fill in missing in postTestScore with each sex's mean value of
postTestScore
df["postTestScore"].fillna(df.groupby("sex")["postTestScore"].transf
orm("mean"), inplace=True)
```


56

Missing Data In pandas Dataframes

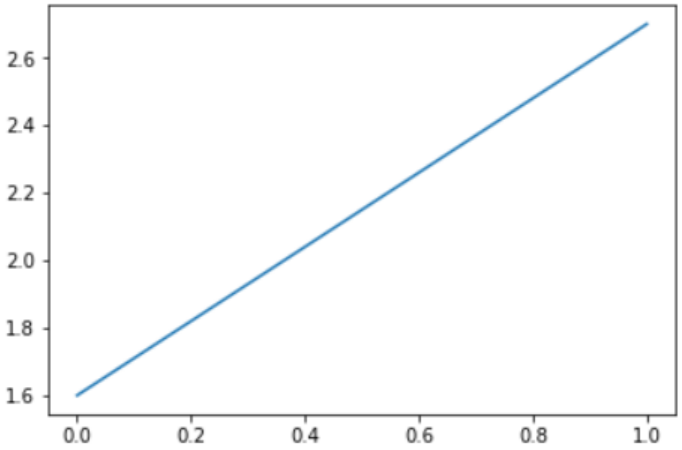
```
# Select some raws but ignore the missing data points
# Select the rows of df where age is not NaN and sex is not NaN
df[df['age'].notnull() & df['sex'].notnull()]
```


57

Basic Plots using matplotlib

Line Plot

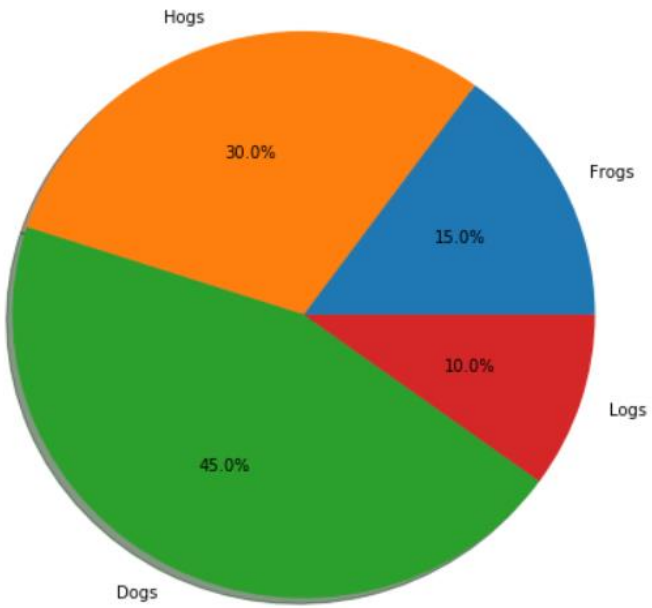
```
import matplotlib.pyplot as plt
plt.plot([1.6, 2.7])
```



59

Piechart

```
import matplotlib.pyplot as plt
from matplotlib.gridspec import GridSpec
plt.figure(figsize=(8,8))
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = [15, 30, 45, 10]
explode = (0, 0.05, 0, 0)
plt.pie(fracs, labels=labels,
autopct='%1.1f%%', shadow=True)
plt.show()
```



60

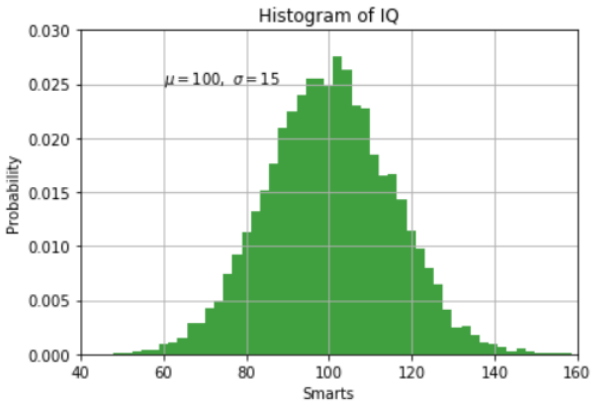
Working with text

The text() command can be used to add text in an arbitrary location, and the xlabel(), ylabel() and title() are used to add text in the indicated locations (see Text introduction for a more detailed example)

```
mu, sigma = 100, 15
x = mu + sigma * numpy.random.randn(10000)

n, bins, patches = plt.hist(x, 50, normed=1,
facecolor='g', alpha=0.75)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60,.025,r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



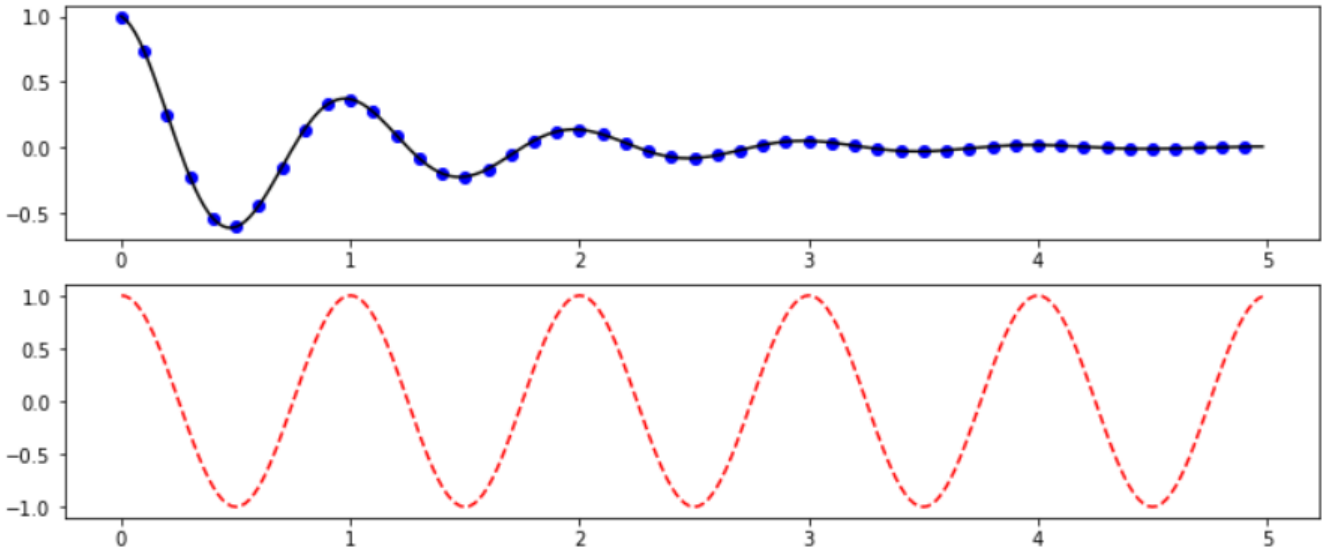
61

Working with multiple figures and axes

```
import numpy
t1 = numpy.arange(0.0, 5.0, 0.1)
t2 = numpy.arange(0.0, 5.0, 0.02)
f=numpy.exp(-t1)*numpy.cos(2*numpy.pi*t1)
f2=numpy.exp(-t2)*numpy.cos(2*numpy.pi*t2)
plt.figure(figsize=(12,5))
plt.subplot(211)
plt.plot(t1, f, 'bo', t2, f2, 'k')
plt.subplot(212)
plt.plot(t2, numpy.cos(2*numpy.pi*t2), 'r--')
plt.show()
```


62

Working with multiple figures and axes



63

Data Visualization using Seaborn

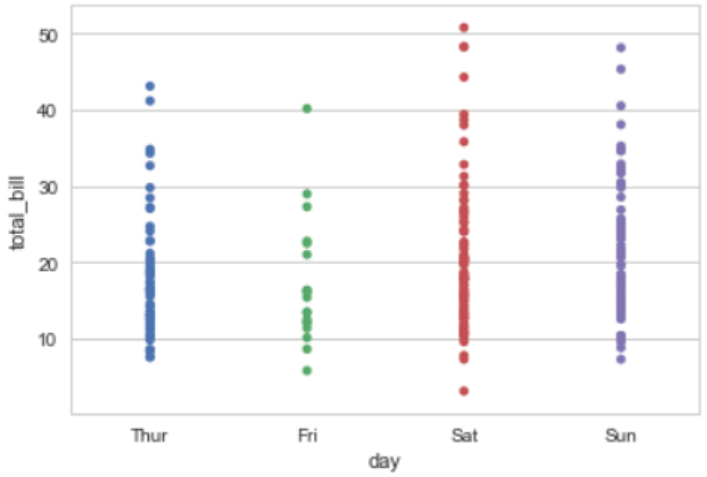
Plotting with categorical data

```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid", color_codes=True)
np.random.seed(sum(map(ord, "categorical")))
titanic = sns.load_dataset("titanic")
tips = sns.load_dataset("tips")
iris = sns.load_dataset("iris")
```


65

```
sns.stripplot(x="day", y="total_bill", data=tips);
```

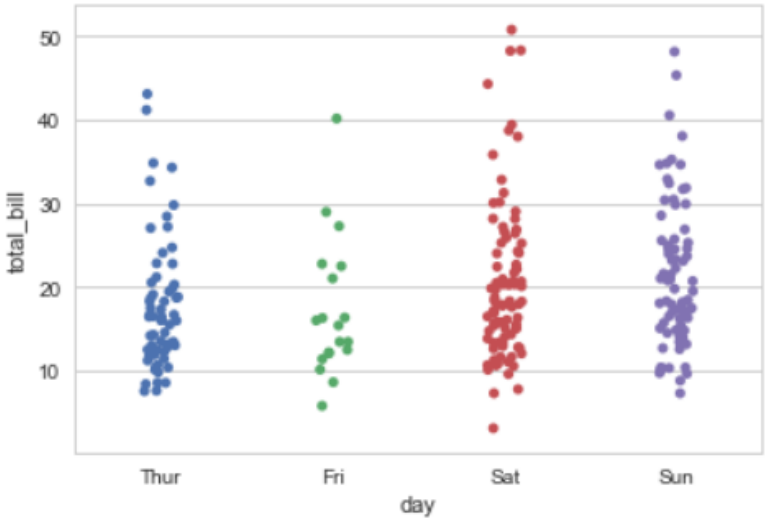
Categorical scatterplots
Stripplots



66

```
sns.stripplot(x="day", y="total_bill",
data=tips, jitter=True);
```

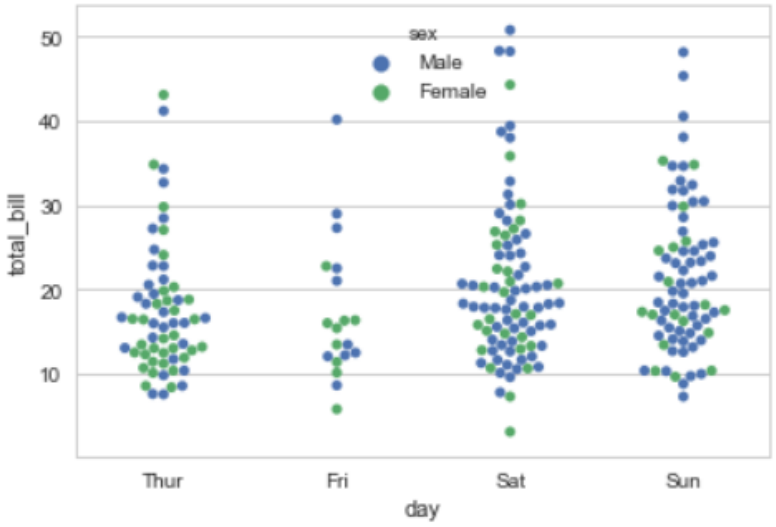
Categorical scatterplots
Stripplots



67

```
sns.swarmplot(x="day", y="total_bill",
hue="sex", data=tips);
```

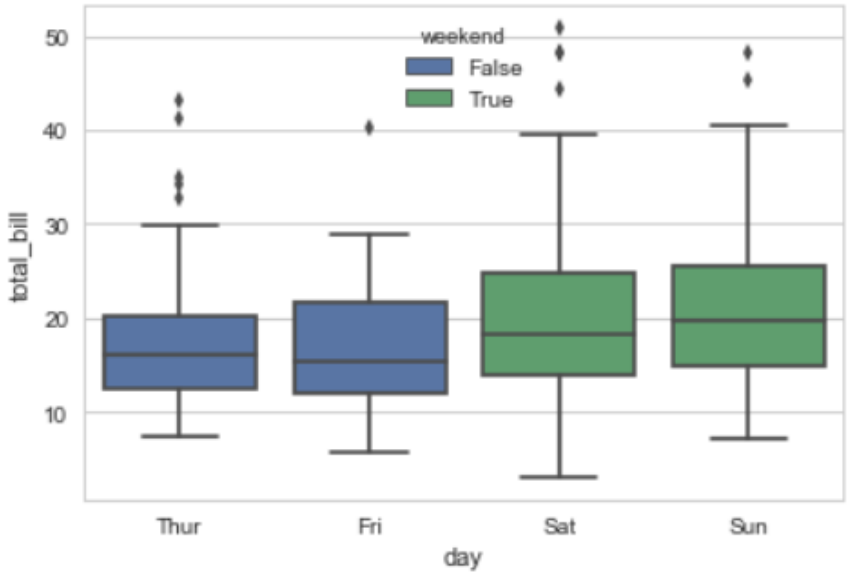
Categorical scatterplots swarmplots



68

```
tips["weekend"] = tips["day"].isin(["Sat", "Sun"])
sns.boxplot(x="day", y="total_bill", hue="weekend",
data=tips, dodge=False);
```

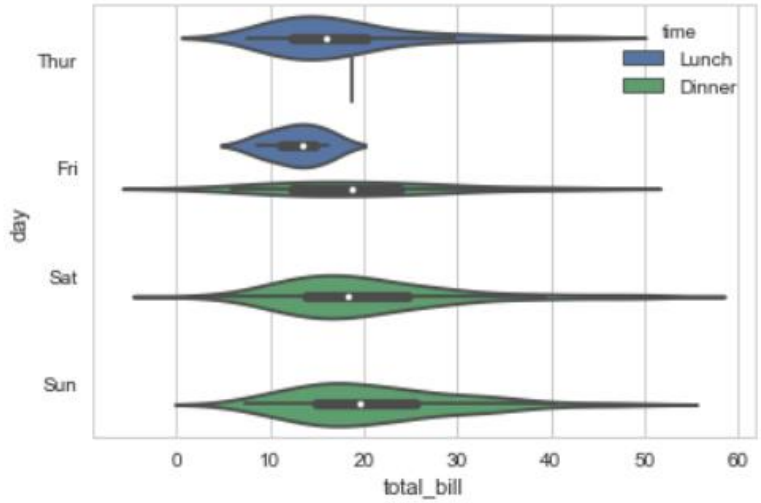
Boxplot



69

```
sns.violinplot(x="total_bill", y="day",
hue="time", data=tips);
```

Violin Plot



70

Using Seaborn To Visualize data

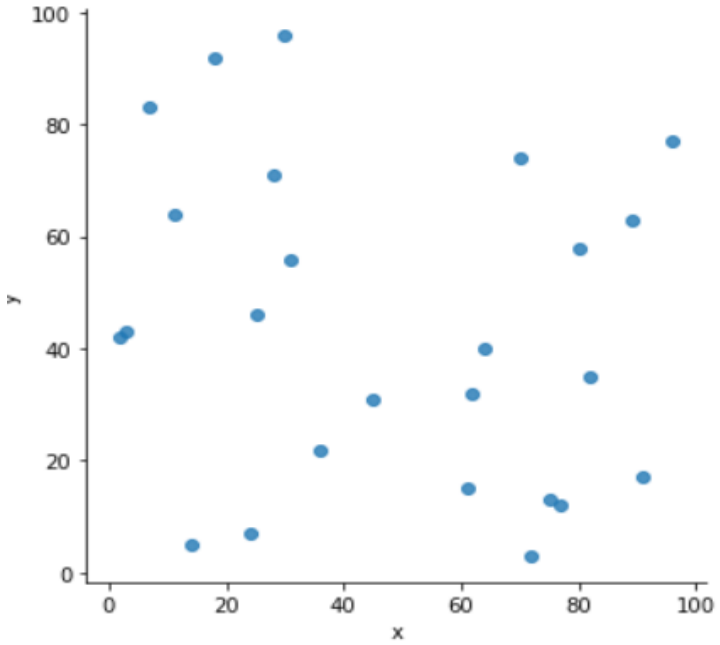
```
import pandas as pd
%matplotlib inline
import random
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.DataFrame()

df['x'] = random.sample(range(1, 100), 25)
df['y'] = random.sample(range(1, 100), 25)
```


71

Scatterplot

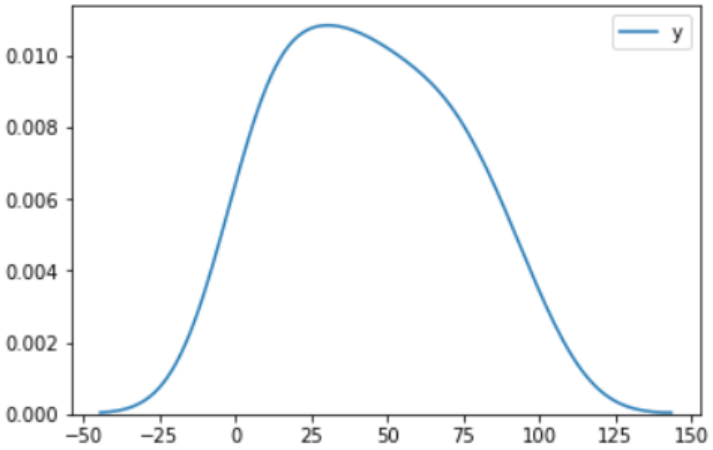
```
sns.lmplot('x', 'y', data=df, fit_reg=False)
```



72

Density Plot

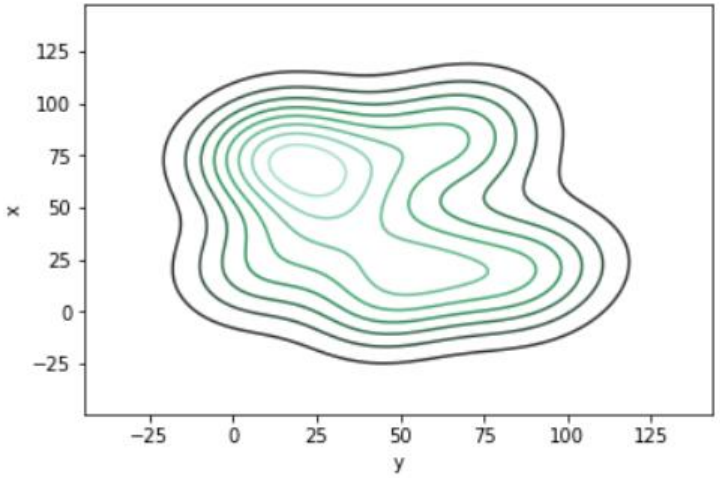
```
sns.kdeplot(df.y)
```



73

Density Plot

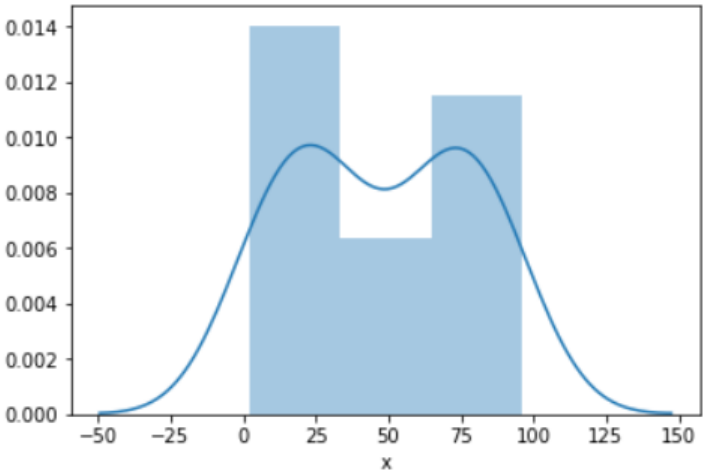
```
sns.kdeplot(df.y, df.x)
```



74

Density Plot

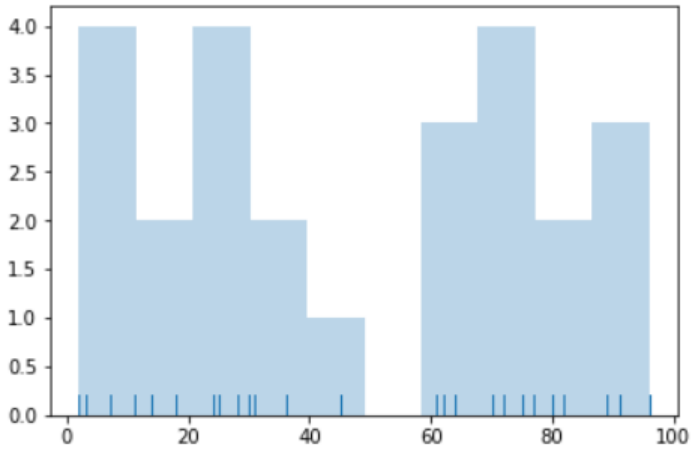
```
sns.distplot(df.x)
```



75

Histogram

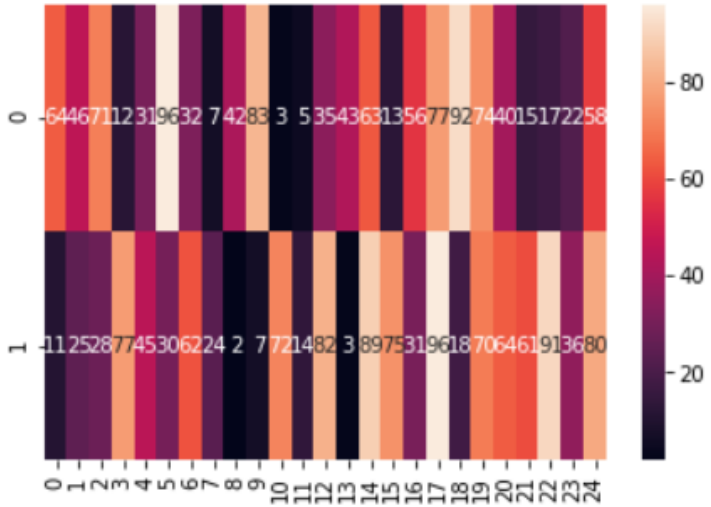
```
plt.hist(df.x, alpha=.3)
sns.rugplot(df.x);
```



76

Heatmap

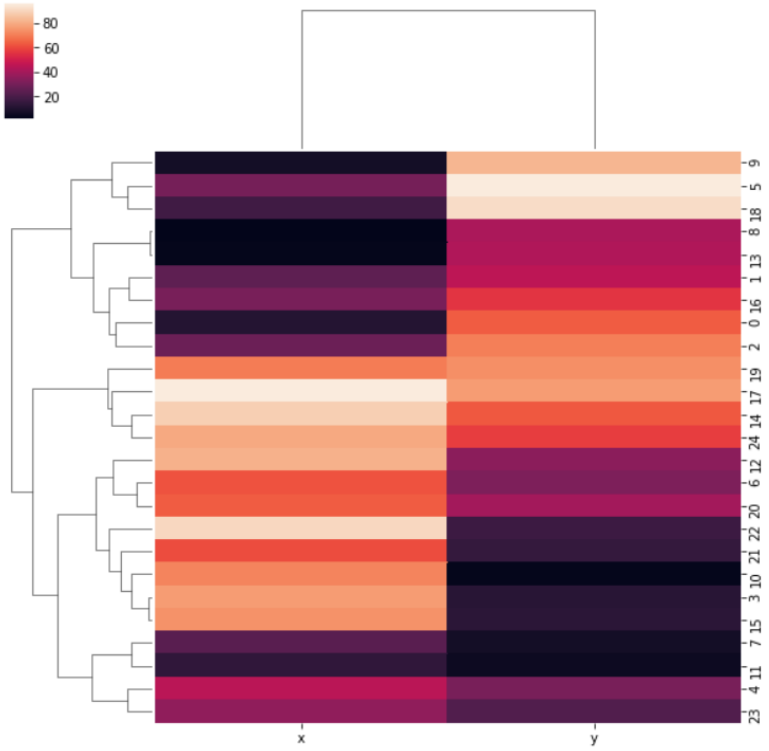
```
sns.heatmap([df.y, df.x], annot=True, fmt="d")
```



77

Clustermap

```
sns.clustermap(df)
```



78

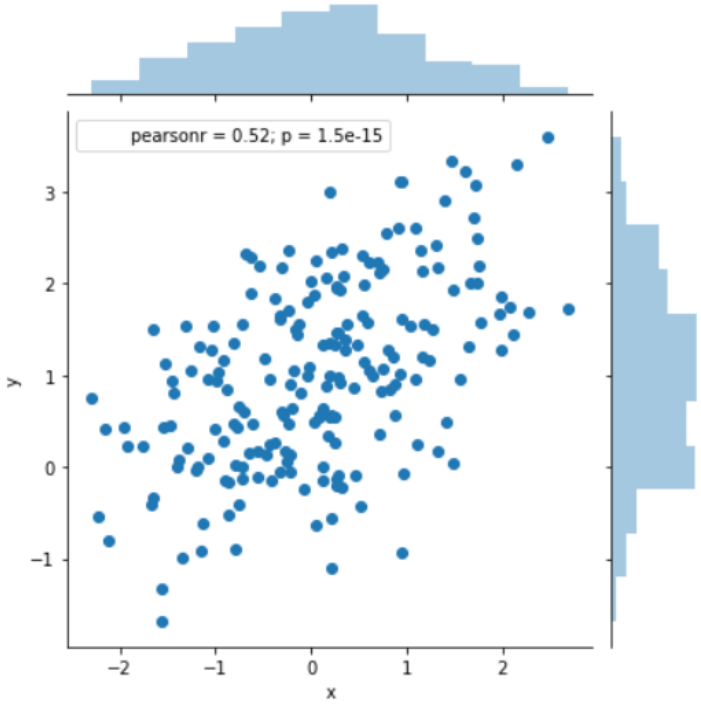
Plotting bivariate distributions

```
mean, cov = [0, 1], [(1, .5), (.5, 1)]
data = np.random.multivariate_normal(mean, cov, 200)
df = pd.DataFrame(data, columns=["x", "y"])
x, y = np.random.multivariate_normal(mean, cov, 1000).T
```


79

Jointplot

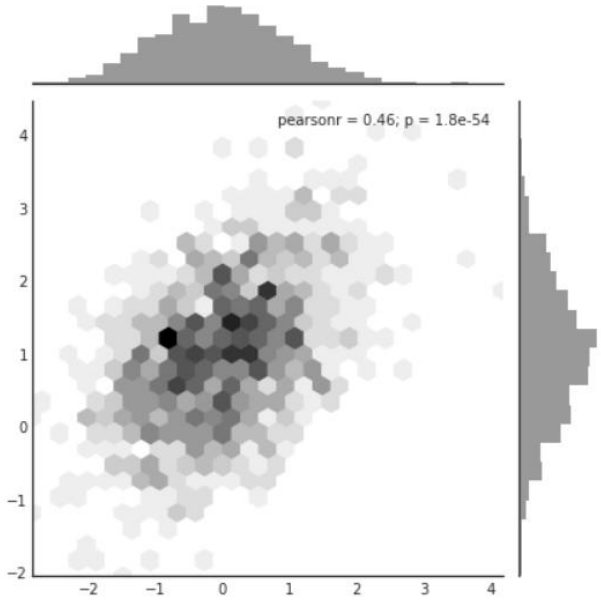
```
sns.jointplot(x="x", y="y", data=df)
```



80

Hexbin Plots

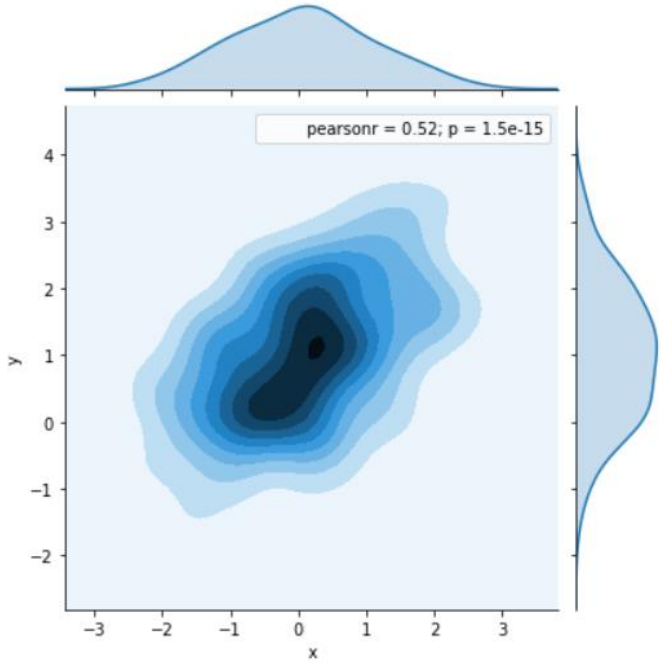
```
with sns.axes_style("white"):
    sns.jointplot(x=x, y=y, kind="hex", color="k")
```



81

Kernel density estimation

```
sns.jointplot(x="x", y="y", data=df, kind="kde");
```



82

Visualizing pairwise relationships in a dataset

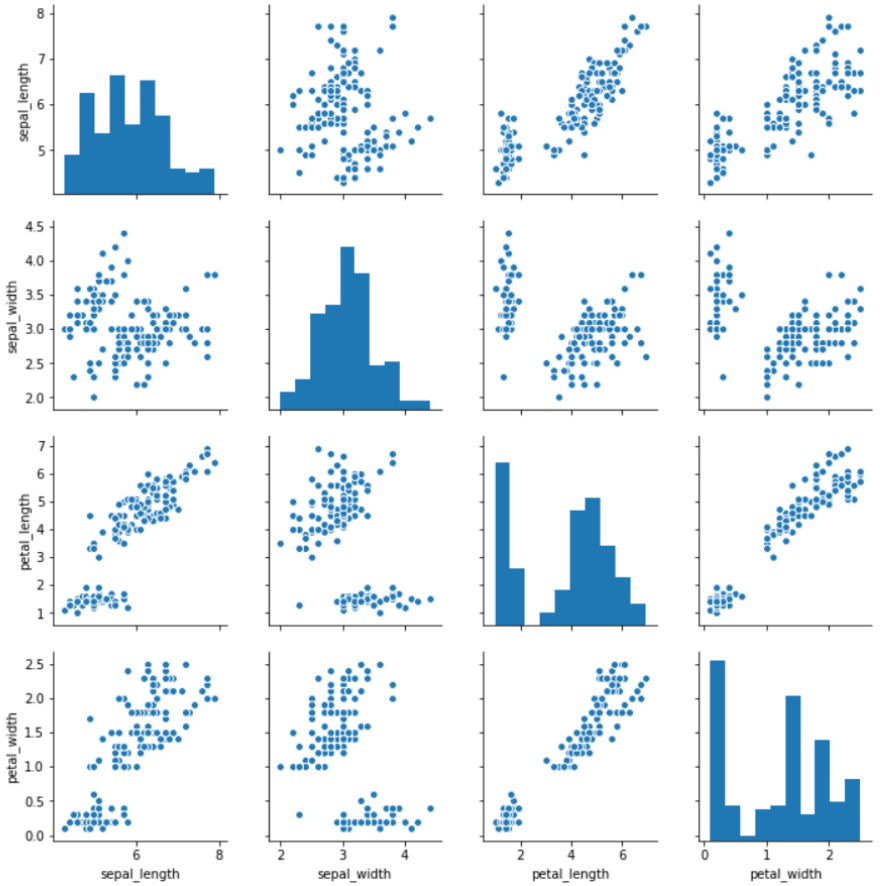
To plot multiple pairwise bivariate distributions in a dataset, you can use the pairplot() function. This creates a matrix of axes and shows the relationship for each pair of columns in a DataFrame. by default, it also draws the univariate distribution of each variable on the diagonal Axes.

```
iris = sns.load_dataset("iris")

sns.pairplot(iris);
```


83

Visualizing pairwise relationships in a dataset



84

Summary

This module covered the following topics:

- Mathematical Computing using numpy
- Linear Algebra using Numpy
- Getting started with Pandas
- Descriptive statistics using pandas
- Data Manipulation using Pandas
- Data Visualization using matplotlib and seaborn

Thank you