Workshop Python Análise de Dados

Numpy, Pandas e Matplotlib

Orador:
Ricardo Tonet, MIEB
Organizado por:

Conteúdo

- Apresentação do problema
- Numpy
- Scipy
- Pandas
- Matplotlib

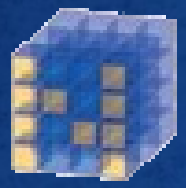
Problema

- Examinar quais as áreas da saúde que precisam de maior intervenção, analisando as taxas de mortalidade a nível europeu.

- Dados reais da Organização Mundial de Saúde – Divisão Europeia
(WHO Data Warehouse: https://dw.euro.who.int/)

- European mortality database

- Vamos usar os Jupyter Notebooks para o código

Numpy 

- Numpy = 'Numerical Python'
- É uma biblioteca de arrays multidimensionais e respectivas rotinas de processamento.
- É a base do Scipy, Pandas e outras bibliotecas científicas de Python
- Juntamente com Scipy, e Matplotlib formam a base de substituição do Matlab

Numpy

- Categorias de Operações:
  - Operações lógicas e matemáticas em arrays
  - Transformadas de Fourier e manipulação da forma das arrays
  - Algebra linear e geração de números aleatórios

Numpy: Tipos de Dados

- **Booleanos:**
  - bool_
- **Inteiros:**
  - int_, intc, intp, int8, int16, int32, int64
- **Inteiros positivos:**
  - uint8, uint16, uint32, uint64
- **Reais:**
  - float_, float16, float32, float64
- **Complexos:**
  - complex_, complex64, complex128

Numpy: Criação de arrays

```
import numpy as np

print(np.array([1, 2, 3]))
print(np.array([i for i in range(10)]))
print(np.array([1, 2, 3], dtype=complex))
print(np.empty([3,2], dtype = int))
print(np.zeros((5), dtype = np.int))
print(np.ones([2,2], dtype = int))
print(np.arange(10, 20, 2))
print(np.linspace(10, 20, 5))
print(np.logspace(1,10,num=10,base=2))
```

Numpy: Atributos dos arrays

```
arr = np.array([[1,2,3],[4,5,6]])

# shape - N.º de dados por dimensão do array
print(arr.shape)

# ndim - N.º de dimensões
print(arr.ndim)

# Existem outros...
```

Numpy: Indexing & Slicing

```
a = np.arange(10)
print(a[5]) # valor único
print(a[2:]) # intervalo 2 ao final
print(a[2:5]) # intervalo 2-4
print(a[2:7:2]) # intervalo 2-6, de 2 em 2

a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print(a)
print(a[1:]) # intervalo 1 ao final
print(a[...,1]) # Seleccionar coluna
```

Numpy: Indexing & Slicing

```
x = np.array([[1, 2], [3, 4], [5, 6]])
y = x[[0,1,2], [0,1,0]]
print(y)

x = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8],[9,
10, 11]])
rows = np.array([[0,0],[3,3]])
cols = np.array([[0,2],[0,2]])
y = x[rows,cols]
print(y)
```

Numpy: Array Manipulation

- **np.reshape:** alterar forma do array sem alterar os dados
- **ndarray.flat:** Iterador unidimensional
- **ndarray.flatten:** retorna array colapsada em 1D
- **np.ravel:** retorna array colapsada em 1D. Recebe array e cria cópia.
- **np.transpose:** transpõe uma matriz

Numpy: Array Manipulation

- **np.resize:** alterar forma da array
- **np.append:** adiciona valores ao final do array
- **np.insert:** insere valores ao longo de um eixo antes dos valores já existentes
- **np.delete:** eliminar valor(es) do array
- **np.unique:** retorna array com os valores únicos do array original. Elimina repetições.

Numpy: Array Manipulation

- **ndarray.T:** transpõe uma matriz
- **np.concatenate:** Concatena duas arrays ao longo de um eixo
- **np.stack:** junta uma sequência de arrays ao longo de um novo eixo
- **np.hstack:** concatena arrays em sequência horizontal (por coluna)
- **np.vstack:** Empinha arrays em sequência vertical (por coluna)

Numpy: String functions

- **np.char.lower:** Retorna array em minúsculas
- **np.char.upper:** Retorna array em maiúsculas
- **np.char.split:** Retorna uma lista de strings separadas pelo separador
- **np.char.join:** Retorna uma string concatenada com os caracteres unidos pelo separador
- **np.char.replace:** Substitui todas as ocorrências da substring por uma nova

Numpy: Mathematical functions

- **np.sin, np.cos, np.tan:** Funções seno e coseno e tangente
- **np.arcos, np.arcsin, np.arctan:** Funções inversas seno, coseno e tangente
- **np.pi:** Pi
- **np.degrees:** Converte radianos para graus
- **np.around, np.floor, np.ceil:** Funções de arredondamento

Numpy: Arithmetic functions

- **np.add, np.subtract, np.multiply, np.divide:** Operam elemento a elemento entre arrays
- **np.reciprocal:** Array dos inversos
- **np.power:** Array da potência dos elementos
- **np.mod, np.remainder:** Retorna array com resto da divisão inteira
- **np.real, np.imag, np.conj, np.angle:** Funções de números complexos

Numpy: Statistical functions

- **np.amin, np.amax:** Mínimo e máximo
- **np.ptp:** Distância pico a pico (|max-min|)
- **np.percentile:** Percentil
- **np.median:** Mediana
- **np.mean, np.average:** Média aritmética e ponderada
- **np.std, np.var:** Desvio padrão e variância

Numpy: Sort, Search & Counting

- **np.sort, np.argsort:** Ordenação de arrays; ordenação de array pelos indíces
- **np.argmax, np.argmin:** Indíces do valor máximo ou mínimo, respectivamente
- **np.nonzero:** Indíces de valores não nulos
- **np.where:** Indíces que verificam uma condição
- **np.extract:** Valores que verificam uma condição

Numpy: Linear Algebra

- **np.dot:** Produto interno entre arrays
- **np.vdot:** Produto interno entre vectores
- **np.inner:** Produto interno entre vectores
- **np.matmul:** Produto matricial entre arrays
- **np.linalg.det:** Determinante de matrizes
- **np.linalg.solve:** Resolve sistemas matrizes
- **np.linalg.inv:** Calcula inversa de matriz

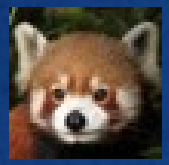
Scipy

- **Biblioteca para computação matemática, científica e de engenharia.**

cluster	Vector quantization / Kmeans
constants	Physical and mathematical constants
fftpack	Fourier transform
integrate	Integration routines
interpolate	Interpolation
io	Data input and output
linalg	Linear algebra routines
ndimage	n-dimensional image package
odr	Orthogonal distance regression
optimize	Optimization
signal	Signal processing
sparse	Sparse matrices

Scipy

spatial	Spatial data structures and algorithms
special	Any special mathematical functions
stats	Statistics

Pandas 

- **Biblioteca de elevada performance para manipulação e análise de dados. O nome Pandas deriva da expressão Panel Data.**

- **Características:**

Carregamento de dados através de vários formatos	Inserção e eliminação de colunas
Lida com dados inexistentes	Agrupamento de dados para agregação e transformações
Alteração da forma das estruturas de dados	Fusão e união de dados com elevada performance
Corte, indexação e subgrupos	Funcionalidade para lidar com séries temporais

Pandas

- Usa 3 estruturas base:
  - Series
  - DataFrame
  - Panel

- São construídas em cima das arrays de Numpy

Estrutura	Dimensões	Descrição
Series	1	Array homogénea de 1D com etiqueta, e tamanho imutável
Data Frames	2	Estrutura tabular 2D com etiqueta, e tamanho mutável e colunas heterógeneas
Panel	3	Estrutura 3D com etiqueta, e tamanho mutável

Pandas: Series

```
# pandas.Series(data, index, dtype, copy)

import pandas as pd
import numpy as np
s = pd.Series()
print(s) # Series vazia

data = np.array(['a','b','c','d'])
s = pd.Series(data)
print(s)

s = pd.Series(data,index=[100,101,102,103])
print(s)
```

Pandas: Series

```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data)
print(s)

data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data,index=['b','c','d','a'])
print(s)

s = pd.Series(5, index=[0, 1, 2, 3])
print(s)

s = pd.Series([1,2,3,4],index=['a','b','c','d'])
print(s[0])
```

Pandas: Series

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

print(s[:3]) # numeric indexing

print(s[-3:]) # numeric indexing

print(s['a']) # label indexing

print(s[['a','c','d']]) # multiple label indexing

print(s['f']) # Key Error
```

Pandas: DataFrame

```
# pandas.DataFrame(data,index,columns,dtype,copy)

df = pd.DataFrame()
print(df) # DataFrame vazio

data = [1,2,3,4,5]
df = pd.DataFrame(data)
print(df)

data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print(df)
```

Pandas: DataFrame

```
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],
dtype=float)
print(df)

data = {'Name':['Tom','Jack','Steve','Ricky'],
'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print(df)

df = pd.DataFrame(data,
index=['rank1','rank2','rank3','rank4'])
print(df)
```

Pandas: DataFrame

```
data = [{'a':1, 'b':2},{'a':5, 'b':10, 'c':20}]
df = pd.DataFrame(data)
print(df)

df = pd.DataFrame(data, index=['first','second'])
print(df)

df1 = pd.DataFrame(data,index=['first','second'],
columns=['a','b'])
df2 = pd.DataFrame(data,index=['first','second'],
columns=['a','b1'])
print(df1)
print(df2)
```

Pandas: DataFrame

```
d = {
'one':pd.Series([1,2,3],index=['a','b','c']),
'two':pd.Series([1,2,3,4],index=['a','b','c','d']
)}
df = pd.DataFrame(d)
print(df)

print(df['one']) # Selecção de coluna

# Adição de nova coluna
df['three']=pd.Series([10,20,30],index=['a','b','
c'])
print(df)
```

Pandas: DataFrame

```
d =
{'one':pd.Series([1,2,3],index=['a','b','c']),
'two':pd.Series([1,2,3,4],index=['a','b','c','d']
),
'three':pd.Series([10,20,30],index=['a','b','c'])
}
df = pd.DataFrame(d)
print(df)

del df['one'] # eliminar coluna

df.pop('two') # eliminar coluna retornando-a
```

Pandas: DataFrame

```
d =
{'one':pd.Series([1,2,3],index=['a','b','c']),
'two':pd.Series([1,2,3,4],index=['a','b','c','d']
)}
df = pd.DataFrame(d)
print(df.loc['b']) # selecção de linha por label

print(df.iloc[2]) # selecção de linha por número

print(df[2:4]) # slicing
```

Pandas: DataFrame

```
df = pd.DataFrame([[1, 2], [3, 4]], columns =
['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns =
['a','b'])
print(df)

df = df.append(df2) # adicionar linhas

df = df.drop(0) # eliminar linhas (elimina linhas
duplicadas com mesmo índice)

print(df)
```

Pandas: Panel

```
# pandas.Panel(data,items,major_axis,minor_axis,
dtype,copy)

p = pd.Panel()
print(p) # Panel vazio

data = np.random.rand(2,4,5)
print(pd.Panel(data)) # Criação a partir de 3D
ndarray

data={'Item1':pd.DataFrame(np.random.randn(4,3)),
'Item2':pd.DataFrame(np.random.randn(4,2))}
print(pd.Panel(data))
```

Pandas: IO Tools

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq

Pandas: IO Tools

```
# pandas.read_csv(filepath_or_buffer, sep='\t',
delimiter=None, header='infer',names=None,
index_col=None, usecols=None)

df = pd.read_csv('temp.csv')
print(df) # Criação de DataFrame

df=pd.read_csv("temp.csv",index_col=['Serial'])
print(df) # Definir coluna de índice

df = pd.read_csv("temp.csv", dtype={'Salary':
np.float64})
print(df.dtypes) # Tipo de dados das colunas
```

Pandas: IO Tools

```
df=pd.read_csv("temp.csv", names=['a', 'b',
'c','d','e'])
print(df) # adicionar cabeçalho extra

df=pd.read_csv("temp.csv",names=['a','b','c','d',
'e'],header=0)
print(df) # remover o cabeçalho

df=pd.read_csv("temp.csv", skiprows=2)
print(df) # saltar linhas
```

Pandas: Basic functionality

```
df=pd.DataFrame(np.random.randn(10,3),columns=['c
ol1','col2','col3'])
print(df)
print(df.T) # Transpor o DataFrame
print(df.axes) # Labels das linhas e colunas
print(df.dtypes) # Tipos de dados por coluna
print(df.empty) # True, se vazio
print(df.ndim) # N.º de dimensões: 2
print(df.shape) # tuple (linhas, colunas)
print(df.size) # N.º de elementos
print(df.values) # arrays de dados por linha
print(df.head(n)) # lista as primeiras n linhas
print(df.tail(n)) # lista as últimas n linhas
```

Pandas: Descriptive Statistics

```
df=pd.DataFrame(np.random.randn(10,3),columns=['c
ol1','col2','col3'])
print(df)
print(df.sum()) # Soma/coluna
print(df.count()) # Conta valores não nulos
print(df.mean()) # Média/coluna
print(df.median()) # Mediana/coluna
print(df.mode()) # Moda/coluna
print(df.std()) # Desvio padrão/coluna
print(df.min()) # Mínimo/coluna
print(df.max()) # Máximo/coluna
print(df.abs()) # Módulo/coluna
```

Pandas: Descriptive Statistics

```
print(df.prod()) # Produto/coluna
print(df.cumsum()) # Soma cumulativa/coluna
print(df.cumprod()) # Produto cumulativo/coluna
print(df.describe()) # Sumário/coluna
```

Pandas: Reindexing

```
N=20
df = pd.DataFrame({
'A': pd.date_range(start='2016-01-01', periods=N,
freq='D'),
'x': np.linspace(0,stop=N-1,num=N),
'y': np.random.rand(N),
'C':
np.random.choice(['Low','Medium','High'],N).tolis
t(),
'D': np.random.normal(100, 10, size=(N)).tolist()
})
dfr = df.reindex(index=[0,2,5],
columns=['A','C','B'])
```

Pandas: Reindexing

```
df1 =
pd.DataFrame(np.random.randn(10,3),columns=['col1
','col2','col3'])
df2 =
pd.DataFrame(np.random.randn(7,3),columns=['col1'
,'col2','col3'])

df3 = df1.reindex_like(df2)
df4 = df2.reindex_like(df1)
print(df3, df4)

df5 = df2.reindex_like(df1, method='ffill')
print(df5)
```

Pandas: Reindexing

```
df6 =
df2.reindex_like(df1,method='ffill',limit=1)
print(df6)

df1 =
pd.DataFrame(np.random.randn(6,3),columns=['col1'
,'col2','col3'])
print(df1)

print(df1.rename(columns={'col1':'c1','col2':'c2'
},index={0:'apple',1:'banana',2:'durian'}))
```

Pandas: Iteration

```
df = pd.DataFrame(np.random.randn(4,3),columns=
['col1','col2','col3'])

# Itera sobre as colunas
for key,value in df.iteritems():
   print(key,value)
# Itera sobre as linhas
for row_index,row in df.iterrows():
   print(row_index,row)
# Retorna um tuple com labels
for row in df.itertuples():
    print(row)
```

Pandas: Sorting

```
# Ordenar por label
df = pd.DataFrame(np.random.randn(10,2),index=
[1,4,6,2,3,5,9,8,0,7],columns=['col2','col1'])
print(df.sort_index())
print(df.sort_index(ascending=False))
print(df.sort_index(axis=1))

# Ordenar por valor
df = pd.DataFrame({'col1':[2,1,1,1],'col2':
[1,3,2,4]})
print(df.sort_values(by='col1'))
print(df.sort_values(by=['col1','col2']))
print(df.sort_values(by='col1',kind='mergesort'))
```

Pandas: Working with text

```
s = pd.Series(['Tom ', ' William Rick', 'John',
'Alber@t', np.nan, '1234','SteveSmith'])
print(s)

print(s.str.lower()) # minúsculas
print(s.str.upper()) # maiúsculas
print(s.str.len()) # tamanho das strings
print(s.str.strip()) # elimina espaços nas pontas
print(s.str.split(' ')) # separa strings
print(s.str.cat(sep='_')) # concatena c/ sep
print(s.str.contains(' ')) # procura padrão
print(s.str.replace('@','$')) # substitui padrão
print(s.str.count('m')) # n.º ocorrencias padrão
```

Pandas: Indexing and Selection

```
df = pd.DataFrame(np.random.randn(8, 4),
index = ['a','b','c','d','e','f','g','h'],
columns = ['A', 'B', 'C', 'D'])

print(df.loc[:,'A'])
print(df.loc[:,['A','C']])
print(df.loc[['a','b','f','h'],['A','C']])
print(df.loc['a':'h'])
print(df.loc['a']>0)
```

Pandas: Indexing and Selection

```
df = pd.DataFrame(np.random.randn(8, 4), columns
= ['A', 'B', 'C', 'D'])

print(df.iloc[:4])
print(df.iloc[1:5, 2:4])
print(df.iloc[[1, 3, 5], [1, 3]])
print(df.iloc[1:3, :])
print(df.iloc[:,1:3])

print(df.ix[:4])
print(df.ix[:,'A'])
```

Pandas: Statistical functions

```
# Calcula diferença percentual entre elementos
consecutivos
s = pd.Series([1,2,3,4,5,4])
print(s.pct_change())
df = pd.DataFrame(np.random.randn(5, 2))
print(df.pct_change())

# Covariância
s1 = pd.Series(np.random.randn(10))
s2 = pd.Series(np.random.randn(10))
print(s1.cov(s2))
```

Pandas: Statistical functions

```
# Correlação
f = pd.DataFrame(np.random.randn(10, 5),
columns=['a', 'b', 'c', 'd', 'e'])
print(f['a'].corr(f['b']))
print(f.corr())

# Ranking entre elementos
s = pd.Series(np.random.np.random.randn(5),
index=list('abcde'))
print(s.rank(method='average'))
# Métodos: average, min, max, first, dense
```

Pandas: Window functions

```
df = pd.DataFrame(np.ones([10,4]),
index = pd.date_range('1/1/2000', periods=10),
columns = ['A', 'B', 'C', 'D'])

print(df)
print(df.rolling(window=3).sum())
print(df.expanding(min_periods=3).sum())
print(df.ewm(com=0.5).mean())
```

Pandas: Aggregations

```
df = pd.DataFrame(np.ones([10,4]),
index = pd.date_range('1/1/2000', periods=10),
columns = ['A', 'B', 'C', 'D'])
r = df.rolling(window=3, min_periods=1).sum()
print(r)

print(r.aggregate(np.sum))
print(r['A'].aggregate(np.sum))
print(r[['A','B']].aggregate(np.sum))
print(r['A'].aggregate([np.sum,np.mean]))
print(r[['A','B']].aggregate([np.sum,np.mean]))
print(r.aggregate({'A' : np.sum,'B' : np.mean}))
```

Pandas: Missing data

```
df = pd.DataFrame(np.random.randn(5, 3),
index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f',
'g', 'h'])
print(df)

print(df['one'].isnull()) # valores nulos
print(df['one'].notnull()) # valores não nulos
print(df.fillna(0)) # Substituir NaN por 0
print(df.fillna(method='pad'))
print(df.fillna(method='bfill'))
```

Pandas: Missing data

```
print(df.dropna())
print(df.dropna(axis=1))

df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print(df.replace({1000:10,2000:60}))
```

Pandas: GroupBy

```
data = {'Team':['Riders','Riders','Devils',
'Devils','Kings','kings','Kings','Kings','Riders'
,'Royals','Royals','Riders'],
'Rank':[1,2,2,3,3, 4,1,1,2,4,1,2],
'Year':[2014,2015,2014,2015,2014,
2015,2016,2017,2016,2014,2015,2017],
'Points':[876,789,863,673,741,812,756,788,694,
701,804,690]}
df = pd.DataFrame(data)
print(df)

print(df.groupby('Team').groups)
print(df.groupby(['Team','Year']).groups)
```

Pandas: GroupBy

```
grouped = df.groupby('Year')
for name,group in grouped:
   print(name)
   print(group)
print(grouped.get_group(2014))
print(grouped['Points'].agg(np.mean))

grouped = df.groupby('Team')
print(grouped.agg(np.size))
print(grouped['Points'].agg([np.sum, np.mean,
np.std]))
print(grouped.filter(lambda x: len(x) >= 3))
```

Pandas: Merging/Joining

```
# pd.merge(left, right, how='inner', on=None,
left_on=None, right_on=None, left_index=False,
right_index=False, sort=True)

left = pd.DataFrame({'id':[1,2,3,4,5],'Name':
['Alex','Amy','Allen','Alice','Ayoung'],'subject_
id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame({'id':[1,2,3,4,5],'Name':
['Billy','Brian','Bran','Bryce','Betty'],'subject
_id':['sub2','sub4','sub3','sub6','sub5']})
print(left)
print(right)
```

Pandas: Merging/Joining

```
print(pd.merge(left,right,on='id'))
print(pd.merge(left,right,on=['id','subject_id'])
)
print(pd.merge(left,right,on='subject_id',
how='left'))
print(pd.merge(left,right,on='subject_id',
how='right'))
print(pd.merge(left,right,on='subject_id',
how='outer'))
print(pd.merge(left,right,on='subject_id',
how='inner'))
```

Pandas: Concatenation

```
# pd.concat(objs,axis=0,join='outer',join_axes=
None,ignore_index=False)

one = pd.DataFrame({'Name': ['Alex', 'Amy',
'Allen', 'Alice', 'Ayoung'],'subject_id':['sub1',
'sub2','sub4','sub6','sub5'],'Marks_scored':
[98,90,87,69,78]},index=[1,2,3,4,5])
two = pd.DataFrame({'Name': ['Billy', 'Brian',
'Bran', 'Bryce', 'Betty'],'subject_id':['sub2',
'sub4','sub3','sub6','sub5'],'Marks_scored':
[89,80,79,97,88]},index=[1,2,3,4,5])

print(pd.concat([one,two]))
```

Pandas: Concatenation

```
print(pd.concat([one,two],keys=['one','two']))

print(pd.concat([one,two],keys=['one','two'],igno
re_index=True))

print(pd.concat([one,two],axis=1))

print(one.append(two))

print(one.append([two,one,two]))
```

Pandas: Visualization

```
df =pd.DataFrame(np.random.randn(10,4),index=
pd.date_range('1/1/2000',periods=10), columns=
list('ABCD'))
df.plot()

df = pd.DataFrame(np.random.rand(10,4),columns=
['a','b','c','d'])
df.plot.bar()
df.plot.bar(stacked=True)
df.plot.barh()
df.plot.barh(stacked=True)
```

Pandas: Visualization

```
df=pd.DataFrame({'a':np.random.randn(1000)+1,'b':
np.random.randn(1000),'c':
np.random.randn(1000) - 1}, columns=['a', 'b',
'c'])
df.plot.hist(bins=20)

df = pd.DataFrame(np.random.rand(10, 5),
columns=['A', 'B', 'C', 'D', 'E'])
df.plot.box()

df = pd.DataFrame(np.random.rand(10, 4),
columns=['a', 'b', 'c', 'd'])
df.plot.area()
```

Pandas: Visualization

```
df = pd.DataFrame(np.random.rand(50, 4),
columns=['a', 'b', 'c', 'd'])
df.plot.scatter(x='a', y='b')

df = pd.DataFrame(3 * np.random.rand(4),
index=['a', 'b', 'c', 'd'], columns=['x'])
df.plot.pie(subplots=True)
```

Matplotlib 

- Biblioteca gráfica 2D que produz figures de elevada qualidade.

- Pode ser usada em scripts Python, consolas Python e Ipython, Jupyter notebooks, aplicações web e GUI toolkits.

- Gera gráficos, histogramas, gráficos de barras, scatterplots, espectros de potência, etc.

- O Pyplot é uma colecção de funções que possibilitam trabalhar como no MATLAB.

Pyplot

```
import matplotlib.pyplot as plt
# Gráfico linear
plt.plot([1, 2, 3, 4])
plt.show()

# Gráfico linear X,Y
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()

# definição do estilo da linha e dimensão eixos
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

Pyplot

```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3,
'g^')
plt.show()
```

Pyplot: Plot w/ keywords

```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

# c=color, s=marker_size
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a')
plt.ylabel('entry b')
plt.show()
```

Pyplot: Plot w/ categorical vars

```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(1, figsize=(9, 3))
plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```

Pyplot: line properties

```
x=[1, 2, 3, 4]
y=[1, 4, 9, 16]
# usar keywords
plt.plot(x, y, linewidth=2.0)

# usar setter methods
line, = plt.plot(x, y, '-')
line.set_antialiased(False)

lines = plt.plot(x1, y1, x2, y2)
plt.setp(lines, color='r', linewidth=2.0)
# or MATLAB style string value pairs
plt.setp(lines, 'color', 'r', 'linewidth', 2.0)
```

Análise de Dados

Property	Value Type
alpha	float
animated	[True | False]
antialiased or aa	[True | False]
clip_box	a matplotlib.transform.Bbox instance
clip_on	[True | False]
clip_path	a Path instance and a Transform instance, a F
color or c	any matplotlib color
contains	the hit testing function
dash_capstyle	['butt' | 'round' | 'projecting']
dash_joinstyle	['miter' | 'round' | 'bevel']
dashes	sequence of on/off ink in points
data	(np.array xdata, np.array ydata)
figure	a matplotlib.figure.Figure instance
label	any string
linestyle or ls	['-' | '--' | '-.' | ':' | 'steps' | ...]
linewidth or lw	float value in points
lod	[True | False]
marker	['+' | ',' | '.' | '1' | '2' | '3' | '4']
markeredgecolor or mec	any matplotlib color
markeredgewidth or mew	float value in points
markerfacecolor or mfc	any matplotlib color
markersize or ms	float
markevery	[None | integer | (startind, stride)]
picker	used in interactive line selection
pickradius	the line pick selection radius
solid_capstyle	['butt' | 'round' | 'projecting']
solid_joinstyle	['miter' | 'round' | 'bevel']
transform	a matplotlib.transforms.Transform instance
visible	[True | False]
xdata	np.array
ydata	np.array
zorder	any number

Pyplot: multiple figures & axes¶

```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```

Pyplot: Working with text

```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)
# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1,
facecolor='g', alpha=0.75)

plt.xlabel('Smarts', fontsize=14, color='red')
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()
```

Pyplot: Working with text

```
ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max',xy=(2,1),xytext=(3,1.5),
arrowprops=dict(facecolor='black',shrink=0.05),)

plt.ylim(-2, 2)
plt.show()
```

Pyplot: Logarithmic and other nonlinear axes

```
np.random.seed(19680801)
y=np.random.normal(loc=0.5, scale=0.4,size=1000)
y=y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))

plt.figure(1)
plt.subplot(221)
plt.plot(x, y)
plt.yscale('linear')
plt.title('linear')
plt.grid(True)
```

Pyplot: Logarithmic and other nonlinear axes

```
# log
plt.subplot(222)
plt.plot(x, y)
plt.yscale('log')
plt.title('log')
plt.grid(True)

# symmetric log
plt.subplot(223)
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthreshy=0.01)
plt.title('symlog')
plt.grid(True)
```

Questões?

Referências

- https://scipy.org/docs.html
- https://www.tutorialspoint.com/numpy/index.htm
- https://www.tutorialspoint.com/scipy/index.htm
- https://www.tutorialspoint.com/python_pandas/index.htm
- https://matplotlib.org/tutorials/introductory/pyplot.html #sphx-glr-tutorials-introductory-pyplot-py
- https://docs.scipy.org/doc/numpy/reference/index.html
- https://docs.scipy.org/doc/scipy/reference/
- http://pandas.pydata.org/pandas-docs/stable/
- https://matplotlib.org/api/api_overview.html