

Jueves de Big Data

Fecha:

22 de noviembre de 2018

De 16:30 a 18:00h

Jueves de Big Data

3/4 Analizando datos con Python

Jueves de Big Data en Sevilla

25/10/2018 Machine learning, una introducción

08/11/2018 Blockchain

22/11/2018 Analizando datos con Python

13/12/2018 Data Augmentation, haciendo grandes algoritmos con pocos datos



Jueves de Big Data

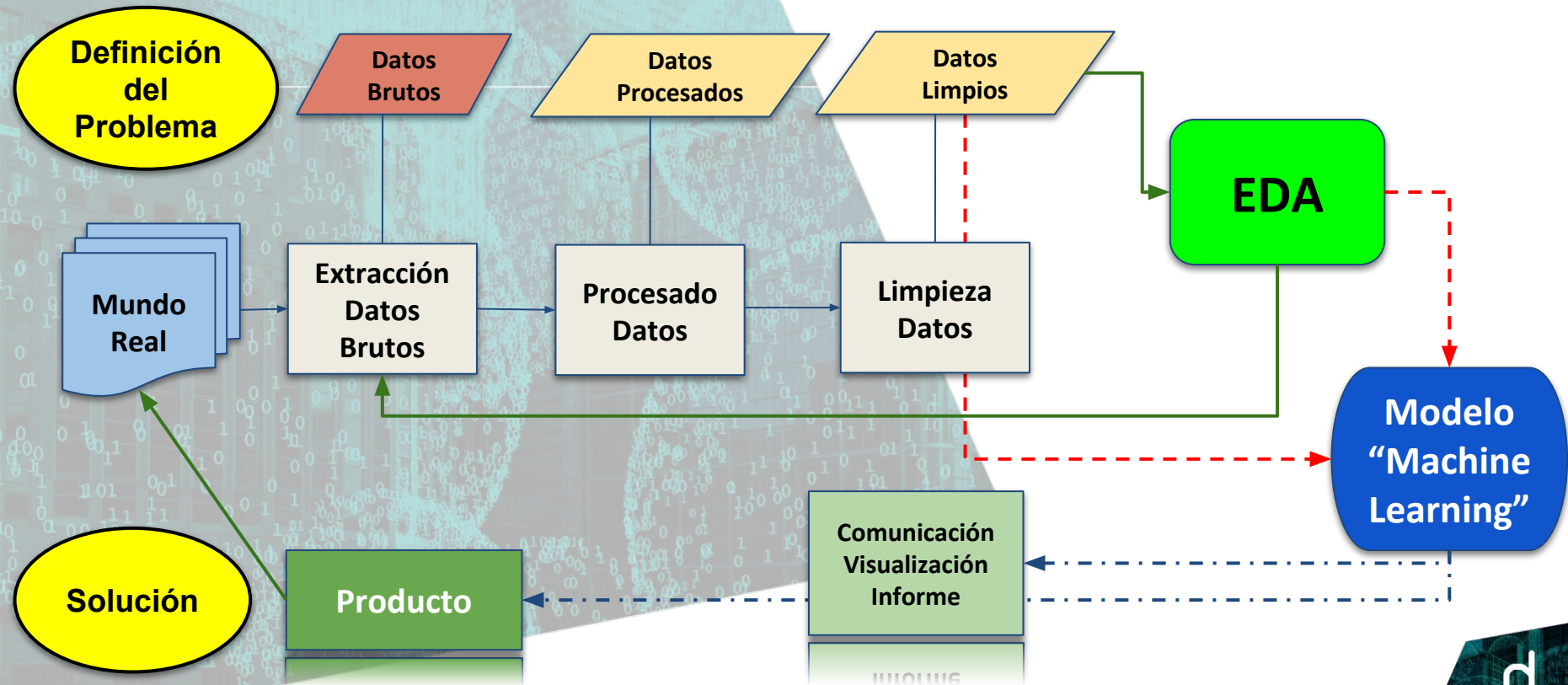
Analizando Datos con Python

Antonio Marín

DSC::Energy
Analytics

- 1. Proceso de Ciencia de Datos (“Data Science”)**
- 2. Etapas de un EDA (“Exploratory Data Analysis”)**
- 3. Python como herramienta**
- 4. Librería NumPy**
- 5. Librería Pandas**
- 6. Visualización: Matplotlib, Seaborn**
- 7. Entornos de trabajo: Anaconda, Jupyter, Spyder**
- 8. Ejemplo**

Proceso Ciencia de Datos



Etapas de un EDA

“Exploratory Data Analysis” es una actitud, un estado de flexibilidad, una voluntad de buscar aquellas cosas que creemos que no están allí, también aquellas que creemos que sí están.

— John Tukey (1977)

- **Tipos de datos**
- **Espacio de la solución (“target”)**
- **Espacio de las variables (“features”)**
- **Variables categóricas**
- **Variables numéricas**
- **Relación entre solución y variables**
- **Relación entre variables**

- ❑ **Valores únicos**
- ❑ **Valores perdidos (“Missing”)**
- ❑ **Valores cero**
- ❑ **Rangos válidos de valores**
- ❑ **Outliers**
- ❑ **Distribución de valores (media, mediana, desviación est., percentiles, máx-min, sesgo)**

Python como herramienta

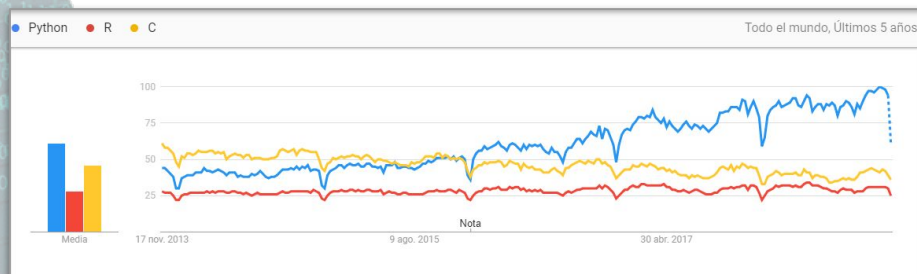


datahack

Python: lenguaje de programación interpretado (no compilado), creado por Guido Van Rossum en 1989 con un objetivo de que fuera fácil de usar y aprender

Motivos de su popularidad:

1. Comunidad que lo respalda
2. Esponsorizado por Google
3. Tiene Big Data
4. Tiene librerías para casi todo
5. Es eficiente y fiable
6. Es accesible



Librerías:

Python: 140000

(3700 específicas de DS)

R: 12000

(6000 específicas de DS)

NumPy: Arrays y vectorización (1/2)

Numpy (Numerical Python) es, probablemente, la librería más importante de todo el lenguaje Python (más allá de las incluidas en la librería interna estándar). Al mismo tiempo, es la librería de álgebra lineal más conocida y potente que hay a día de hoy.

Incluido en Numpy:

- ndarray, un eficiente array multidimensional con una rápida aritmética orientada a array
- Funciones matemáticas para operaciones rápidas sobre los arrays enteros, sin hacer bucles
- Herramientas para lectura/escritura a disco, trabajando con ficheros mapeados en memoria
- Álgebra lineal, generación de aleatorios, transformada de Fourier
- Una API a C para conectar NumPy con librerías escritas en C, C++ y FORTRAN



NumPy: Arrays y vectorización (2/2)

Principales aplicaciones en análisis de datos:

- Muy rápidas operaciones vectorizadas, para manipulación, limpieza, selección, filtrado y transformación
- Algoritmos sobre arrays comunes: ordenado, únicos y operaciones sobre conjuntos (“set”)
- Eficiente estadística descriptiva, agregación y resumen de datos
- Alineación y manipulación de datos relacionales, para “merging” y “joining”
- Lógica condicional usando expresiones de arrays en lugar de bucles “if-elif-else”

Razones de la importancia de NumPy:

- Está diseñado para arrays de datos muy grandes
- Internamente almacena los datos en bloques de memoria contigua, independiente de los otros objetos de Python
- Los algoritmos están escritos en C




NumPy: Cheat Sheet

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Numpy_Python_Cheat_Sheet.pdf

Python For Data Science Cheat Sheet

NumPy Basics

Learn Python for Data Science interactively at [www.DataCamp.com](https://www.datacamp.com)






NumPy

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

1D array: 
2D array: 
3D array: 

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([[1,5,2,3], [4,5,6]], dtype = float)
>>> c = np.array([[[1,5,2,3], [4,5,6]], [[3,2,1], [4,5,6]]], dtype = float)
```

Initial Placeholders

```
>>> np.zeros(13,4)
>>> np.ones((2,2,4), dtype=np.int64)
>>> d = np.arange(10,25,5)
>>> np.linspace(0,2,9)
>>> e = np.full((2,2),7)
>>> f = np.eye(2)
>>> np.random.random((2,2))
>>> np.empty(13,2)
```

Create an array of zeros
Create an array of ones
Create an array of evenly spaced values (step value)
Create an array of evenly spaced values (number of samples)
Create a constant array
Create a 2x2 identity matrix
Create an array with random values
Create an empty array

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npz')
```

Saving & Loading Text Files

```
>>> np.savetxt("myfile.csv")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64
>>> np.float32
>>> np.complex
>>> np.bool
>>> np.object
>>> np.string
>>> np.unicode
```

Signed 64-bit integer types
Standard double-precision floating point
Complex numbers represented by tall floats
Boolean type storing TRUE and FALSE values
Python object type
Fixed-length string type
Fixed-length unicode type

Inspecting Your Array

```
>>> a.shape
>>> len(a)
>>> a.ndim
>>> a.size
>>> b.dtype
>>> b.dtype.name
>>> b.astype(int)
```

Array dimensions
Length of array
Number of array dimensions
Number of array elements
Data type of array elements
Name of data type
Convert an array to a different type

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a + b
>>> array([[0, 5, 0, ..., 0, 1, 1],
        [-3, ..., -3, ..., 1]])
>>> np.subtract(a,b)
>>> b - a
>>> array([[ 2.5,  4, ..., 6, 1],
        [-5, ..., 7, ..., 8, 1]])
>>> np.add(b,a)
>>> a + b
>>> array([[0.565887, 1, ..., 1, 1],
        [ 4, ..., 18, ..., 18, 1]])
>>> np.multiply(a,b)
>>> np.exp(b)
>>> np.exp(b)
>>> np.sin(a)
>>> np.cos(b)
>>> np.log(a)
>>> a.dot(f)
>>> array([[ 7.7,  7.7],
        [ 7.7,  7.7)])
```

Subtraction
Subtraction
Addition
Addition
Division
Division
Multiplication
Exponentiation
Square root
Print lines of an array
Element-wise cosine
Element-wise natural logarithm
Dot product

Comparison

```
>>> a == b
>>> array([[True,  True,  True],
        [True,  True,  True]], dtype=bool)
>>> a < 2
>>> array([[True,  True,  True],
        [True,  True,  True]], dtype=bool)
>>> np.array_equal(a, b)
```

Element-wise comparison
Element-wise comparison
Array-wise comparison

Aggregate Functions

```
>>> a.mean()
>>> a.min()
>>> b.max(axis=0)
>>> b.cumsum(axis=1)
>>> a.mean()
>>> b.median()
>>> a.corrcoef()
>>> np.std(b)
```

Array-wise sum
Array-wise minimum value
Maximum value of an array row
Cumulative sum of the elements
Mean
Median
Correlation coefficient
Standard deviation

Copying Arrays

```
>>> b = a.view()
>>> np.copy(a)
>>> b = a.copy()
```

Create a view of the array with the same data
Create a copy of the array
Create a deep copy of the array

Sorting Arrays

```
>>> a.sort()
>>> c.sort(axis=0)
```

Sort an array
Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
>>> b[1,2]
>>> a[0]
>>> a[0:2]
>>> array([1, 2])
>>> b[0:2, 2:]
>>> b[1:]
>>> array([[0, 5, 2, ..., 3, 1]])
>>> a[1,...]
>>> array([[ 2,  2, ..., 1, 1],
        [ 4,  5, ..., 6, 1]])
```

Select the element at the 2nd index
Select the element at row 1 column 2 (equivalent to b[1][2])
Select items at index 0 and 1
Select items at rows 0 and 1 in column 1
Select all items at row 0 (equivalent to b[0][:, :])
Same as [1, ..., 1]
Reversed array a

Slicing

```
>>> a[0:2]
>>> array([1, 2])
>>> b[0:2, 2:]
>>> b[1:]
>>> array([[0, 5, 2, ..., 3, 1]])
>>> a[1,...]
>>> array([[ 2,  2, ..., 1, 1],
        [ 4,  5, ..., 6, 1]])
```

Select elements from a less than 2
Select a subset of the matrix's rows and columns

Boolean Indexing

```
>>> a[a<1]
>>> array([1, 2])
>>> b[b[1, 0, 1, 0], [0, 1, 2, 0]]
>>> array([[ 4,  2, ..., 5, ..., 6],
        [ 5,  6, ..., 6, ..., 6],
        [ 5,  6, ..., 6, ..., 6],
        [ 5,  6, ..., 6, ..., 6]])
```

Select elements from a less than 2
Select a subset of the matrix's rows and columns

Fancy Indexing

```
>>> b[[1, 0, 1, 0], [0, 1, 2, 0]]
>>> array([[ 4,  2, ..., 5, ..., 6],
        [ 5,  6, ..., 6, ..., 6],
        [ 5,  6, ..., 6, ..., 6],
        [ 5,  6, ..., 6, ..., 6]])
```

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```

Permute array dimensions
Permute array dimensions

Changing Array Shape

```
>>> b.reshape(3,-2)
>>> i.reshape((2,6))
```

Flatten the array
Reshape, but don't change data

Adding/Removing Elements

```
>>> np.append(b,0)
>>> np.insert(a, 1, 5)
>>> np.delete(a, [1])
```

Append items to an array
Insert items in an array
Delete items from an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
>>> array([ 1,  2, ..., 10, 15, 20])
>>> np.vstack([a,b])
>>> array([[ 1, ..., 1, ..., 1, ..., 0],
        [ 1, ..., 5, ..., 6, ..., 1]])
>>> np.r_[a,f]
>>> np.hstack([a,f])
>>> array([[ 1, ..., 1, ..., 1, ..., 0],
        [ 1, ..., 5, ..., 6, ..., 1],
        [ 7, ..., 7, ..., 0, ..., 1]])
>>> np.column_stack([a,d])
>>> array([[ 1, 15],
        [ 2, 20]])
>>> np.d_[a,d]
>>> array([[ 1, 15],
        [ 2, 20]])
```


Concatenate arrays
Stack arrays vertically (row-wise)
Stack arrays vertically (row-wise)
Stack arrays horizontally (column-wise)
Create stacked column-wise arrays
Create stacked column-wise arrays

Splitting Arrays

```
>>> np.hsplit(a,3)
>>> [array([1, 2, ..., 10, 15]), array([15]), array([15])]
>>> np.vsplit(a,3)
>>> [array([1, 2, ..., 1, ..., 1, ..., 0]), array([1, 5, ..., 6, ..., 1]), array([1, 5, ..., 6, ..., 1])]
```

Split the array horizontally at the 3rd index
Split the array vertically at the 2nd index

DataCamp
Learn Python for Data Science Interactively





Pandas: Series y DataFrames

Pandas es una librería construida sobre Numpy que nos permite tratar datos, principalmente estructurados (“tablas”), de forma sencilla, cómoda y rápida

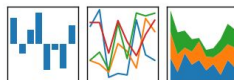
Los principales elementos son: los “DataFrame”, que son objetos tabulares orientados en columnas, y las “Series”, objetos array unidimensionales

Características principales de Pandas:

- Mezcla la base de alto rendimiento en computación de NumPy, con la manipulación flexible de hojas de cálculo y bases de datos relacionales (como SQL)
- Permite indexación sofisticada, para facilitar la transformación, selección, filtrado y todo tipo de operaciones sobre los DATOS
- Funcionalidad integrada de series temporales
- Manipulación flexible de datos “missing” (drop, ffill, bfill, interpolate, ...)
- “Merging” y otras operaciones relacionales que se usan habitualmente en bases de datos (SQL)

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$




Pandas: Cheat Sheet

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PandasPythonForDataScience.pdf

Python For Data Science Cheat Sheet

Pandas Basics

Learn Python for Data Science Interactively at www.datacamp.com



Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.


Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

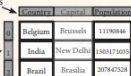
A one-dimensional labeled array capable of holding any data type



```
>>> s = pd.Series([3, -5, 7, 41], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns



A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
           'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
           'Population': [11180846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Getting

```
>>> a['b']
-5
```

Get one element

```
>>> df[1:]
   Country  Capital  Population
1  India    New Delhi  1303171035
2  Brazil   Brasilia  207847528
```

Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[0,0]
'Belgium'
>>> df.loc[0,0]
'Belgium'
>>> df.at[0,0]
'Belgium'
```

Select single value by row & column

By Label

```
>>> df.loc[0, ['Country']]
'Belgium'
>>> df.at[0, ['Country']]
'Belgium'
```

Select single value by row & column labels

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital      Brasilia
Population    207847528
```

Select single row of subset of rows

```
>>> df.ix[:, 'Capital']
0    Brussels
1    New Delhi
2    Brasilia
```

Select a single column of subset of columns

```
>>> df.ix[:, 'Capital']
New Delhi
New Delhi
Brasilia
```

Select rows and columns

Boolean Indexing

```
>>> a[(a > 1)]
a
1    7
2   41
```

Series a where value is not > 1 where value is < 1 or > 2

```
>>> df[df['Population'] > 1200000000]
   Country  Capital  Population
1  India    New Delhi  1303171035
```

Use filter to adjust DataFrame

Setting

```
>>> a['a'] = 6
```

Set index a of Series a to 6

Dropping

```
>>> a.drop(['a', 'c'])
a
1    7
2   41
```

Drop values from rows (axis=0)

```
>>> df.drop('Country', axis=1)
   Capital  Population
1  New Delhi  1303171035
2  Brasilia  207847528
```

Drop values from columns (axis=1)

Sort & Rank

```
>>> df.sort_index()
a    3
b   -5
c    7
d   41
```

Sort by labels along an axis

```
>>> df.sort_values(by='Country')
   Country  Capital  Population
1  India    New Delhi  1303171035
2  Brazil   Brasilia  207847528
```

Sort by the values along an axis

```
>>> df.rank()
a    1
b    2
c    3
d    4
```

Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
(4, 3)
>>> df.index
a
b
c
d
>>> df.columns
Country
Capital
Population
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4 entries, a to d
Data columns (total 3 columns):
Country      object
Capital      object
Population    int64
dtypes: object(2), int64(1)
memory usage: 112 bytes
```

Summary

```
>>> df.sum()
Country      1303171035
Capital      207847528
Population    11180846
dtypes: int64(3)
memory usage: 112 bytes
```

Sum of values
Cumulative sum of values
Minimum/maximum values
Minimum/maximum index value
Summary statistics
Mean of values
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
Country      2236342070
Capital      415695056
Population    2236342070
dtypes: int64(3)
memory usage: 112 bytes
```

Apply function element-wise

```
>>> df.applymap(f)
Country      2236342070
Capital      415695056
Population    2236342070
dtypes: int64(3)
memory usage: 112 bytes
```

Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> a3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> a
a    3
b   -5
c    7
d   41
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> a.add(a3, fill_value=0)
a
a    6
b   -5
c    7
d   41
```

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
>>> pd.read_excel('file.xlsx', sheet_name='Sheet1')
>>> df = pd.read_excel('file.xlsx')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql('SELECT * FROM my_table', engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query('SELECT * FROM my_table', engine)
>>> read_sql('SELECT * FROM my_table', engine)
>>> pd.to_sql('myOf', engine)
```



Matplotlib: visualización de datos 1

La Visualización es una de las tareas más importantes en el Análisis de Datos.

Matplotlib, es la librería por excelencia de creación de gráficos 2D en Python. El proyecto lo empezó John Hunter en 2002, para tener un interfaz gráfico parecido a Matlab en Python.

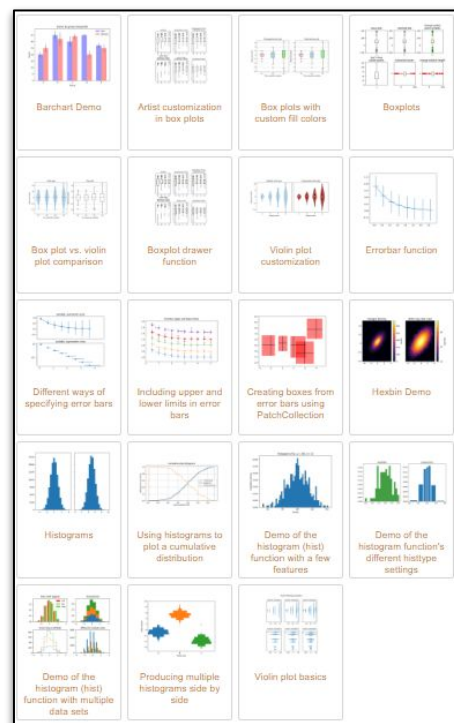
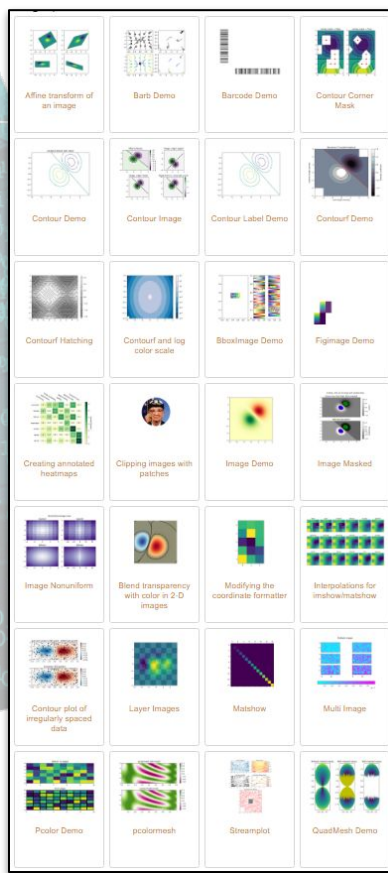
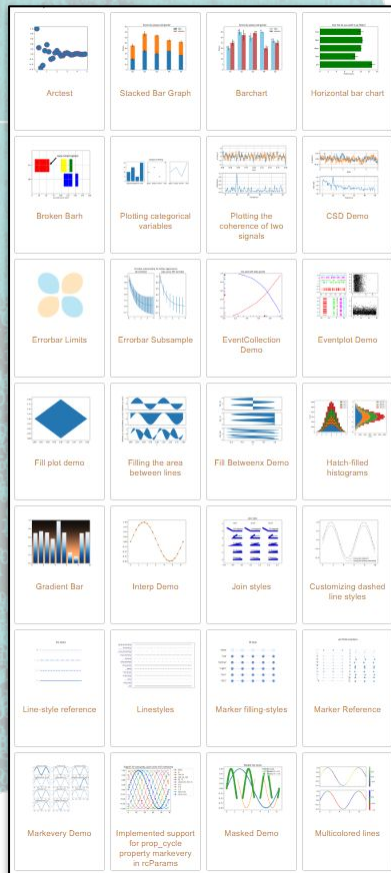
Objetivos que se marcó John Hunter cuando creó Matplotlib:

- Gráficos debería ser de alta calidad para publicaciones
- Debe tener salida Postscript para inclusión en documentos TeX
- Debe permitir embeberse en una interfaz gráfica de usuario (GUI) para desarrollo de aplicaciones
- Debe ser suficientemente fácil para entenderlo y extenderlo
- Generar gráficos debe ser fácil

Matplotlib: visualización de datos 1

Tipos de gráficos:

- Líneas
- Histograma
- Barras
- Contornos
- Polares
- Scatter
- Pastel
- Multipanel y grids
- Boxplots
- Violinplots
- Barras de error
- Hexbin
- Densidad
- Animación
- ...



Matplotlib: Cheat Sheet

datahack

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf

Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.datacamp.com

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

1 Prepare The Data Also see Lists & NumPy

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> y, X = np.mgrid[0:10, 0:10]
>>> U = 1 - x**2 + y
```

from matplotlib.cheek import get_sample_data

```
>>> img = np.load(get_sample_data('axes_grid/figure1.mat'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig, ax = plt.subplots()
>>> ax1 = fig.add_subplot(221) # row=col=um
>>> ax2 = fig.add_subplot(222)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)
>>> ax.scatter(x,y)
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].dash([0.5,2],[0.5,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.45)
>>> ax.dill(x,y,color='blue')
>>> ax.dill(x,y,color='yellow')
```

2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(data,
>>>                  cmap='gist_march',
>>>                  interpolation='nearest',
>>>                  vmin=-2,
>>>                  vmax=3)
```

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, y, x**2, y, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='r')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(data,
>>>                  cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker='*')
>>> ax.plot(x,y,marker='o')
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,linestyle='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
>>>         'Example Graph',
>>>         style='italic')
>>> ax.annotate('size',
>>>             xytext=(0, 1),
>>>             xycoords='data',
>>>             textcoords='figure fraction',
>>>             arrowprops=dict(arrowstyle='->',
>>>                             connectionstyle='arc3,2pts'))
```

Vector Fields

```
>>> axes[2,1].arrow(0.5,0.5,0.5,0.5)
>>> axes[1,1].quiver(y,x)
>>> axes[0,1].streamplot(x,y,0.5,0.5)
```

Data Distribution

```
>>> ax1.hist(y)
>>> ax1.boxplot(y)
>>> ax1.violinplot(x)
```

Plot a Histogram

```
>>> ax1.hist(x)
>>> ax1.boxplot(y)
>>> ax1.violinplot(x)
```

Pseudocolor plot of 2D array

```
>>> axes[0].pcolormesh(data2)
>>> axes[0].pcolormesh(data)
>>> CS = plt.contour(x,y,X)
>>> axes[2].contourf(data1)
>>> axes[2] = ax.contour(CS)
```

Plot Anatomy & Workflow

Plot Anatomy

Figure

Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]
>>> y = [10,20,25,30]
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111)
>>> ax.plot(x, y, color='lightblue', linewidth=3)
>>> ax.scatter(5,15,25),
>>>          color='darkgreen',
>>>          marker='*')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, y, x**2, y, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='r')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(data,
>>>                  cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker='*')
>>> ax.plot(x,y,marker='o')
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,linestyle='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
>>>         'Example Graph',
>>>         style='italic')
>>> ax.annotate('size',
>>>             xytext=(0, 1),
>>>             xycoords='data',
>>>             textcoords='figure fraction',
>>>             arrowprops=dict(arrowstyle='->',
>>>                             connectionstyle='arc3,2pts'))
```

Vector Fields

```
>>> axes[2,1].arrow(0.5,0.5,0.5,0.5)
>>> axes[1,1].quiver(y,x)
>>> axes[0,1].streamplot(x,y,0.5,0.5)
```

Data Distribution

```
>>> ax1.hist(y)
>>> ax1.boxplot(y)
>>> ax1.violinplot(x)
```

Plot a Histogram

```
>>> ax1.hist(x)
>>> ax1.boxplot(y)
>>> ax1.violinplot(x)
```

Pseudocolor plot of 2D array

```
>>> axes[0].pcolormesh(data2)
>>> axes[0].pcolormesh(data)
>>> CS = plt.contour(x,y,X)
>>> axes[2].contourf(data1)
>>> axes[2] = ax.contour(CS)
```

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

Clear an axis

```
>>> plt.cla()
```

Clear the entire figure

```
>>> plt.clf()
```

Close a window

```
>>> plt.close()
```

DataCamp

Learn Python for Data Science Interactively

datahack

Seaborn: visualización de datos 2

Seaborn es una librería basada en Matplotlib para hacer gráficos estadísticos, e integrada con las estructuras de Pandas, con un interfaz de alto nivel y visualizaciones muy cuidadas

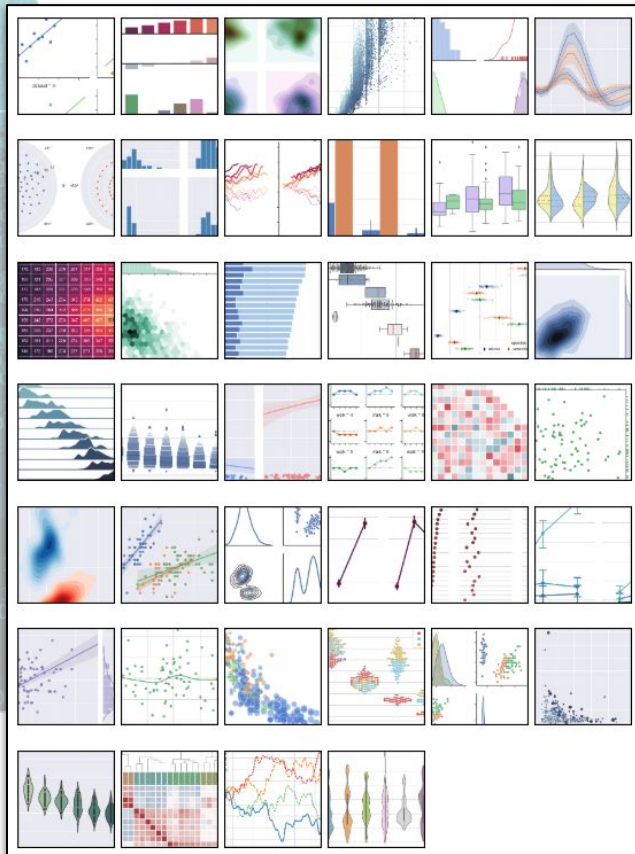
Características principales de Seaborn:

- API orientada a datasets para examinar relaciones entre múltiples variables
- Especial soporte para variables categóricas para mostrar observaciones y agregar estadística
- Opciones para visualizar univariable y multivariable
- Estimación automática de regresiones lineales
- Visualización general de datasets complejos
- Multi-plot de alto nivel para construir visualizaciones complejas
- Control conciso de los estilos de Matplotlib con temas incluidos
- Herramientas para elegir paletas para revelar patrones

Seaborn: visualización de datos 2

Tipos de gráficos:

- Relacionales (relación, scatter y líneas)
- Categoricos (stripplot, swarmplot, boxplot, catplot, violinplot, boxen, barras, contorno)
- Distribución (joint, pair, densidad, kde, rug)
- Regresión (lmlplot, regplot, residplot)
- Matrices (heatmap, clustermap)
- Multi-plots (facetgrid, pair grids, joint grids)
-



https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python Seaborn Cheat Sheet.pdf

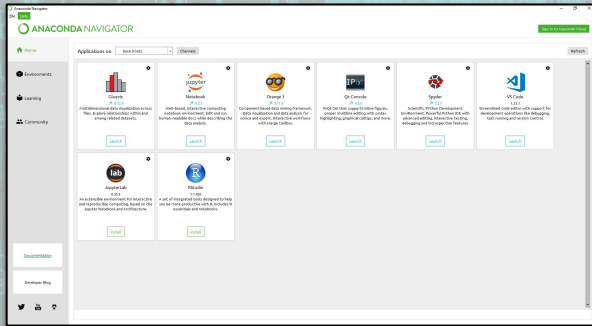
[illegible]

Entornos de trabajo

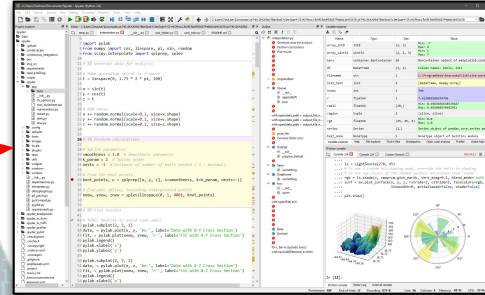


datahack

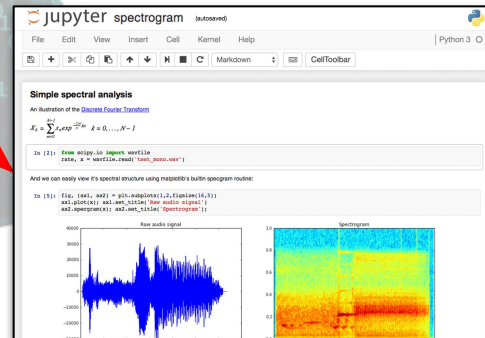
Anaconda



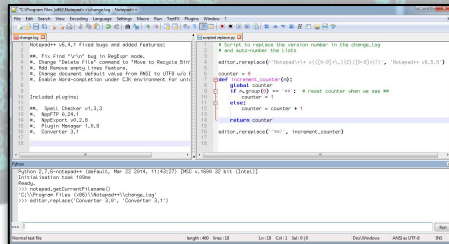
Spyder



Jupyter notebooks



Notepad++



¿Tienes alguna pregunta?



A word cloud featuring the word 'Gracias' in large, bold, dark blue letters at the bottom. Above it, numerous other words in various languages and scripts are arranged in a circular pattern, all meaning 'Thank you'. These include: 'Merci' (French), 'Danke' (German), 'Tack' (Swedish), 'Kaigai' (Japanese), 'Tsin'aen' (Mandarin), 'Matóndo' (Lingala), 'Merçi' (Romanian), 'Takk' (Norwegian), 'Yuspagara' (Yupik), 'Ngeyabonga' (Ndebele), 'Mahalo' (Hawaiian), 'maluhlap' (Xhosa), 'Kili' (Kikuyu), 'Dyakooyu' (Dinka), 'Gyalailaa' (Tibetan), 'Ashoge' (Igbo), 'so' (Swahili), 'Trugarez' (Breton), 'Tänan' (Lingala), 'Mantiox' (Mossi), 'Murakoze' (Kinyarwanda), 'Tack' (Swedish), 'leibh' (Yiddish), 'qúi' (Quechua), 'pai' (Tibetan), 'Xie' (Chinese), 'Barka' (Hausa), 'ankon' (Igbo), 'Maraba' (Hausa), 'laketai' (Igbo), 'Bedankt' (Dutch), 'Thanks' (English), 'nnaba' (Igbo), 'Mwebare' (Igbo), 'Emitekati' (Igbo), 'Tesekkür' (Turkish), 'jai' (Igbo), 'Dakujem' (Slovak), 'Syaabaas' (Arabic), 'chawe' (Chewa), 'magah' (Mossi), 'Hvala' (Croatian/Slovene), 'Alla' (Arabic), 'Uzbekco' (Uzbek), 'Ha'evete' (Xhosa), 'Rahmet' (Arabic), 'Danke' (German), 'Dios' (Spanish), 'raibh' (Irish), 'Netjer' (Egyptian), 'Gratias' (Latin), 'Ashi' (Akan), and 'hvala' (Slovene/Croatian). The words are in various colors and sizes, creating a vibrant and multicultural display.

Gracias
datahack
Equipo = dh

