



Afi

Escuela
de Finanzas

Programación en Python

Rocío Parrilla

Marzo 2020

Contenido

- 1. Introducción a Python**
- 2. Python básico**
- 3. Introducción al análisis de datos con Python: Numpy y Pandas**
- 4. Regresión lineal**
- 5. Visualización estática y dinámica**
- 6. Referencias**

1 | Introducción a Python

Historia de Python

- Creado en 1990 por Guido van Rossum (actualmente empleado de Dropbox).
- El nombre está basado en los humoristas británicos Monty Python.
- A partir del año 2001 pasa a ser administrado por Python Software Foundation, una compañía sin ánimo de lucro con un funcionamiento similar al de Apache Software Foundation.



¿Qué es Python?

¿Qué es Python?

- Es un lenguaje de programación **open source**.
- Es un lenguaje de programación **conciso**.
- Es un lenguaje de programación de **alto nivel**.

Se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas.

Ventajas: Genera un código más sencillo y comprensible.

- Es un lenguaje de programación de **propósito general**.

Son lenguajes que pueden ser usados para varios propósitos, acceso a bases de datos, comunicación entre computadoras, comunicación entre dispositivos, captura de datos, cálculos matemáticos, diseño de imágenes o páginas.

¿Qué es Python?

¿Qué es Python?

- o Es un lenguaje de programación **orientado a objetos**.

Los lenguajes de programación orientados a objetos tratan a los programas como conjuntos de objetos que se ayudan entre ellos para realizar acciones (entendiendo como objeto a entidades que contienen datos), permitiendo que los programas sean más fáciles de escribir, mantener y reutilizar.

- o Es un lenguaje de programación **con una comunidad muy activa**.

- o Es un lenguaje de programación **con infinidad de módulos y/o add-ins orientados a muy diferentes dominios** (tratamiento de imagen, videojuegos, bases de datos, análisis de datos...).

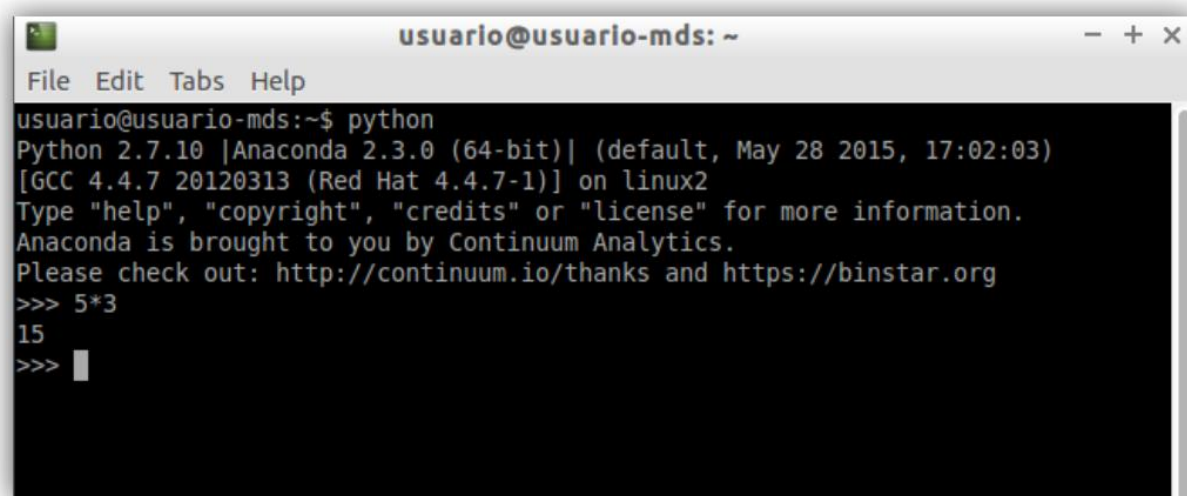
IDEs disponibles

- Existen muchos IDEs (Integrated Development Environment) para Python.
- Cada uno de ellos está diseñado para dar soporte a una forma de trabajo en función del dominio (análisis de datos, desarrollo general, programación reproducible...) al que se orienten.
- Se pueden encontrar desde consolas básicas (tipo R o Matlab) hasta entornos completos de desarrollo y despliegue de aplicaciones y servicios (tipo Visual Studio, Eclipse, NetBeans, etc.).

IDEs disponibles

Intérprete / Consola Python

- Se trata de la consola interactiva de Python.
- Se accede al intérprete y se trabaja con el mismo desde la línea de comandos del sistema operativo.
- Interpreta los inputs introducidos, evalúa los mismos y ofrece un output por cada uno de ellos.

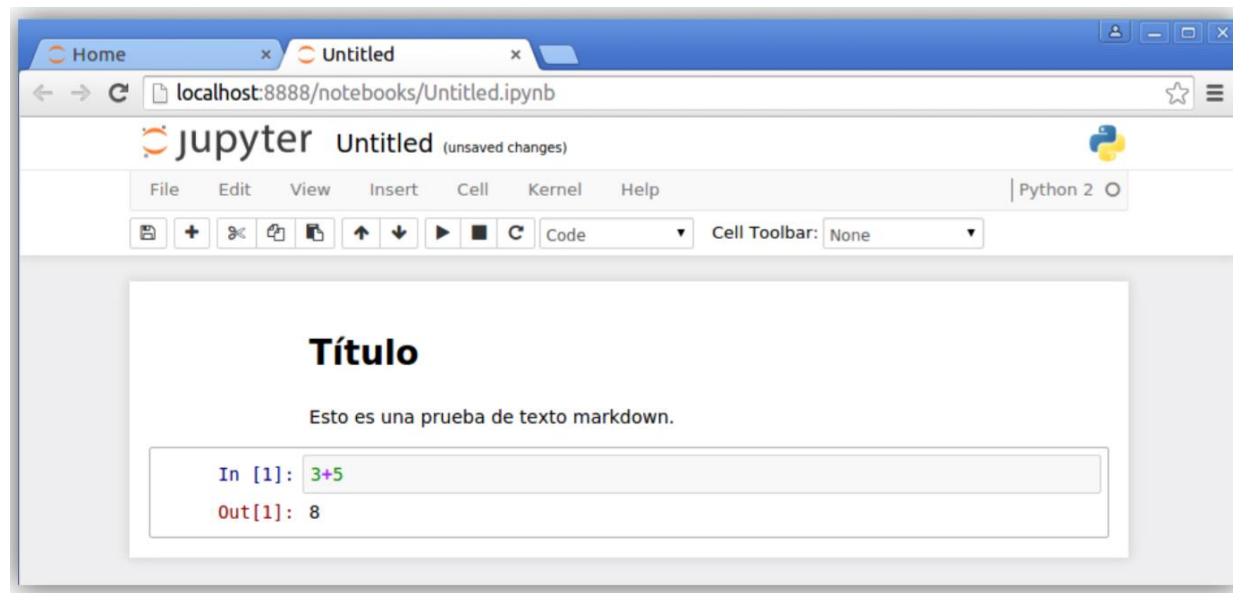


```
usuario@usuario-mds: ~  
File Edit Tabs Help  
usuario@usuario-mds:~$ python  
Python 2.7.10 |Anaconda 2.3.0 (64-bit)| (default, May 28 2015, 17:02:03)  
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
Anaconda is brought to you by Continuum Analytics.  
Please check out: http://continuum.io/thanks and https://binstar.org  
>>> 5*3  
15  
>>> 
```


IDEs disponibles

Notebook

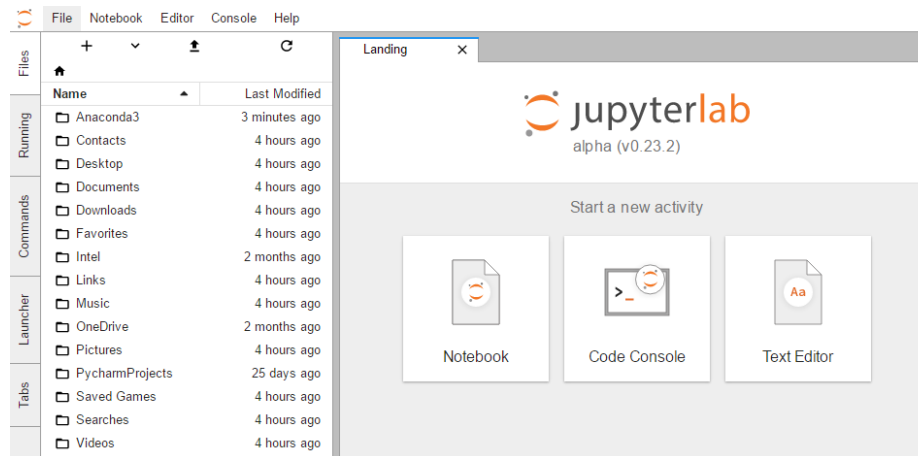
- Evolución de la consola interactiva de IPython, que ha derivado en el Proyecto Jupyter.
- Se trata de un intérprete de Python ejecutado directamente sobre un navegador Web
- Permite combinar “celdas” de código con texto enriquecido (HTML, imágenes, gráficos, etc.).
- Permite almacenar en un único documento comentarios, entradas, código y salidas.



IDEs disponibles

JupyterLab

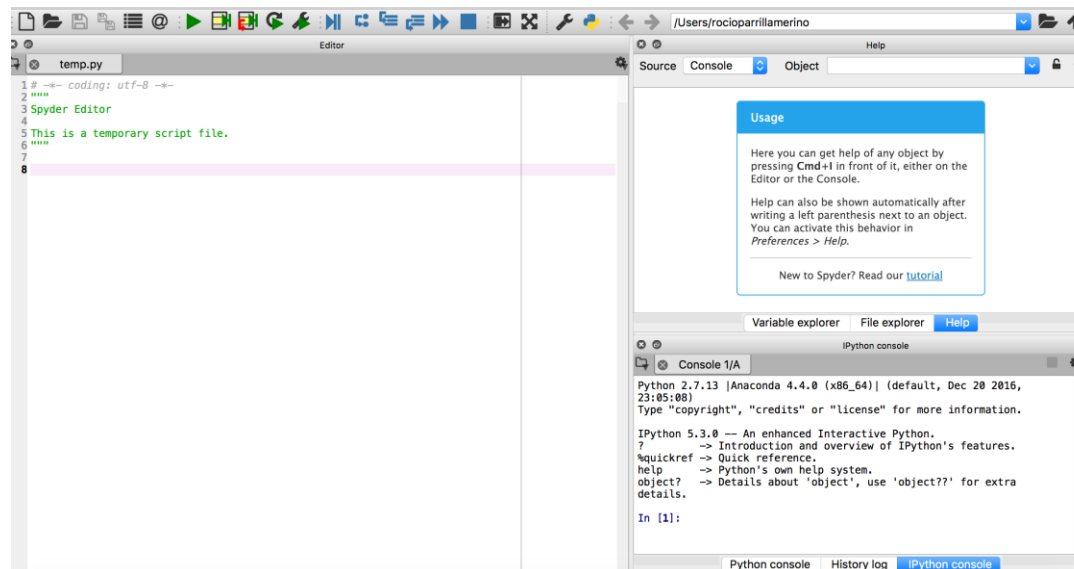
- Se considera la evolución de Jupyter Notebook.
- Mejora la interfaz, ofrece más opciones de personalización e interacción.
- Proporciona editores de texto y terminales de Python, entre otros componentes que pueden abrirse y visualizarse al mismo tiempo que los documentos de Notebook.
- Incluye también accesos directos a Google Drive y otros servicios en la nube.



IDEs disponibles

Spyder

- Editor completo de Python orientado a la programación científica / interactive (similar a R Studio o Matlab).
- Incluye funcionalidades como resaltado y completado de código, trabajo simultáneo con diversas consolas de Python (incluso con diferentes versions), inspección de variables en el entorno de trabajo, depuración de código, creación y gestión de proyectos...

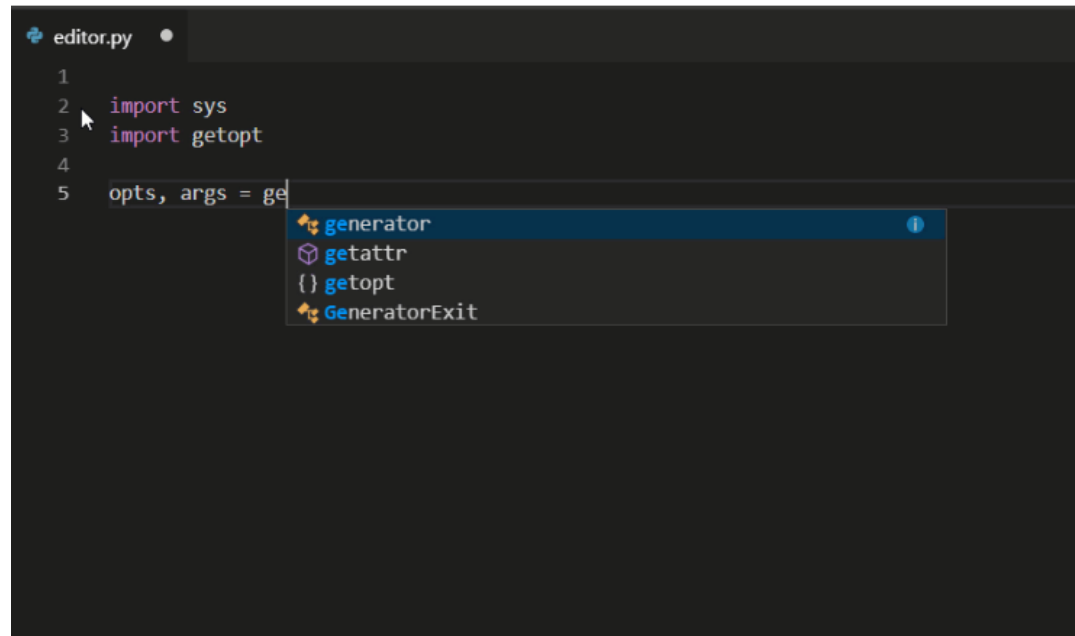


IDEs disponibles

Visual Studio Code

Visual Studio Code es un entorno de programación muy usado para trabajar con diversos lenguajes de programación, que se caracteriza por ser **rápido, ligero y muy fácil de utilizar**.

Para usarlo para programar en Python, debemos **instalar la extensión de Python para Visual Studio Code**.



```
editor.py •
1
2 import sys
3 import getopt
4
5 opts, args = ge
```

The screenshot shows the Visual Studio Code editor interface. The file 'editor.py' is open. The code contains the following lines: 1 (empty), 2 'import sys', 3 'import getopt', 4 (empty), and 5 'opts, args = ge'. A dropdown menu is visible below line 5, showing suggestions for the next part of the code: 'generator' (with a blue icon), 'getattr' (with a purple icon), 'getopt' (with a blue icon), and 'GeneratorExit' (with a blue icon).

Distribuciones disponibles

- Python puede descargarse e instalarse a partir de su core e incluir a posteriori los módulos que se quieran ir utilizando.
- Pero también existen diferentes distribuciones que “empaquetan” tanto el core como algunos de los módulos más típicos, entornos de programación y otras utilidades.
- Esto facilita enormemente la puesta en marcha de una nueva máquina como entorno de trabajo.

Python

- Se trata del core de Python.
- Incluye únicamente los paquetes básicos y el intérprete de consola de comandos.
- Se puede descargar para cualquier sistema operativo.



Distribuciones disponibles

Anaconda

- Se trata de la distribución de Python más extendida y reconocida de las existentes.
- Incluye más de 300 módulos preinstalados desde análisis de datos, hasta de desarrollo web pasando por librerías matemáticas...
- Incluye una consola (gráfica) para Python, iPython, Jupyter Notebooks y Spyder.



<https://store.continuum.io/cshop/anaconda/>

2 | Python básico

Sintaxis y elementos básicos

Asignaciones

```
a = 3
```

Tipos de datos

- Cadenas de caracteres *str*.
- Valores numéricos enteros *int*.
- Valores numéricos decimales *float*.
- Valores booleanos *bool*.

Operadores básicos

Estructuras de datos

Secuencias

- Tuplas: (1,3,4)
- Listas: [1, 3, 7]
- Cadena de caracteres : 'Hola mundo'

Diccionarios

{'clave1': valor1, 'clave2': valor2}

Conjuntos

{a, b, c}

Estructuras de control

Condicionales

```
if condición:  
    código  
else:  
    código
```

Bucles

```
for elemento in secuencia:  
    código que se ejecuta para cada elemento de la secuencia
```

```
while condición:  
    código que se ejecuta mientras la condición se verifica
```

Control de errores: try, except

```
try:  
    código que intenta ejecutarse  
except:  
    código que se ejecuta si lo anterior devuelve error
```

Funciones y módulos

Funciones

```
def mifuncion:  
    código  
    return(...)
```

Módulos

Por defecto, en un script de Python tienes acceso a todas las variables y funciones definidas en el propio fichero.

Es posible acceder a elementos definidos en otros ficheros mediante la importación de módulos.

La forma de incorporar elementos definidos en un módulo es mediante el uso de la sentencia ***import***.

3 | Introducción al análisis de datos con Python: Numpy y Pandas

¿Qué es NumPy?

¿Qué es NumPy?

- NumPy es un módulo de Python y es la abreviatura de Numerical Python.
- Ofrece estructuras de datos y funciones matemáticas que serían complejas de replicar utilizando el *core* de Python (computación matricial, operaciones matemáticas sobre grandes conjuntos de información, etc.).
- Es la base de un gran conjunto de módulos de Python, especialmente orientados al análisis de datos (entre ellos *pandas*), por lo que entender correctamente su funcionamiento facilita el aprendizaje posterior de estos paquetes.
- Junto con las funcionalidades incluidas en el módulo SciPy ofrece un *framework* completo para llevar a cabo procesos de computación científica en Python. De hecho, “dicen” que...

Python + NumPy + SciPy + Matplotlib → MATLAB

Capacidades principales

- Estructura de datos para el almacenamiento y recuperación de datos en forma de matrices n-dimensionales: *ndarray*.
- Funciones matemáticas estándar con rendimiento optimizado para poder ser aplicadas a matrices completas (o secciones de las mismas) sin necesidad de utilizar bucles.
- Herramientas para la lectura y escritura de información matricial a disco.
- Funciones para la aplicación de álgebra lineal, generación de números aleatorios, transformadas de Fourier...
- Herramientas para la integración de código nativo escrito en C, C++ o Fortran, permitiendo la integración en nuevos desarrollos de librerías *legacy* de cálculo.

Numpy: Conceptos básicos

Estructura básica: ***ndarray***.

- Un ndarray puede contener elementos de **CUALQUIER TIPO**
- Todos los elementos de un ndarray deben tener **EL MISMO TIPO**
- Pueden tener cualquier dimensión:

d = 1: vector

d = 2: matriz

- Operaciones aritméticas sobre ndarrays
- Álgebra lineal: submódulo **linalg**.

¿Qué es Pandas?

¿Qué es Pandas?

- Pandas es un módulo de Python, de alto rendimiento, orientado al análisis de datos.
- Inicialmente creada por Wes McKinney (autor del libro “Python for Data Analysis”, referencia en este curso).
- La primera versión se publicó en 2008 y la última disponible en marzo de 2020 (1.0.3).
- Con los años se ha convertido en el estándar (de facto) para el análisis de datos en Python.
- Una de las librerías con más evolución y seguimiento por parte de la comunidad (más de 200 contribuidores).

Características principales (I)

- Capacidad de almacenamiento y procesamiento de diferentes estructuras de datos: series temporales, información tabular, información matricial...
- Facilidad para la carga de información desde muy diferentes fuentes: ficheros CSV, bases de datos relacionales...
- Implementación de operaciones sobre estructuras de datos completas como selección de subconjuntos, agrupaciones, filtrado, ordenación...
- Capacidad para el tratamiento de *missing values* según las necesidades del usuario / desarrollador.
- Utilidad tanto para la carga y tratamiento de datos como para el análisis estadístico, exploratorio y modelado.

Características principales (II)

- Integración con otras librerías como statsmodel, SciPy, NumPy (en la que se basa) y *scikit-learn*...
- Alto rendimiento, que puede ser incluso mayor haciendo uso de Cython (que permite integrar extensiones escritas en C en programas escritos en Python).
- En esencia, trata de incorporar a Python, estructuras de datos y operaciones como las existentes en R:
 - Estructuras: vectores (con nombre), `data.frame`, `data.table`...
 - Operaciones: familia *apply*, agregación y agrupación con `data.table`...

Pandas: conceptos básicos

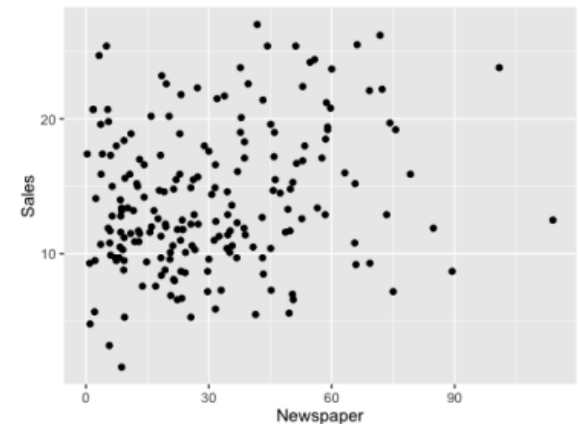
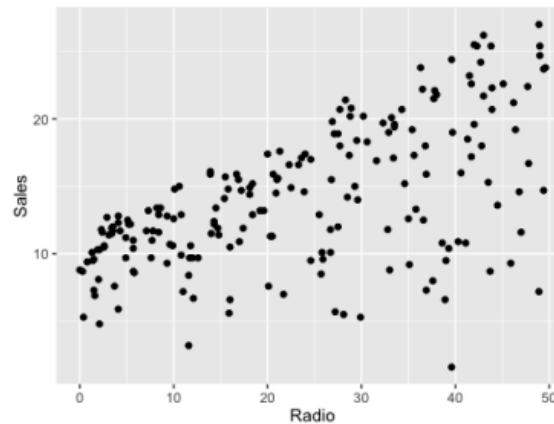
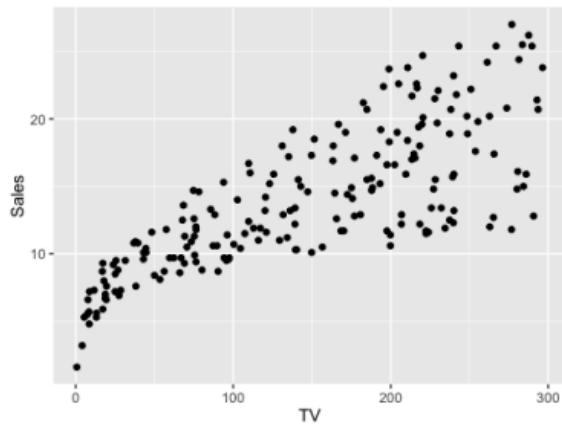
- Dos estructuras de datos:
 - Series:** Para información unidimensional.
 - DataFrame:** Para información tabular.
- Lectura y escritura de ficheros: *read_csv* y *write_csv*.
- Preparación y exploración de datos.

4 | Modelo de Regresión lineal

Introducción

Queremos aconsejar a una compañía sobre cómo mejorar las ventas de un determinado producto (en miles de unidades).

Para conseguirlo, nos proporcionan un set de datos que contiene las ventas del producto en 200 mercados diferentes, junto con el presupuesto de publicidad en televisión, radio y periódicos en cada uno de tales mercados (en miles de dólares).



Introducción

Para responder al cliente, lo ideal sería encontrar una función f de modo que

$$\text{Sales} = f(\text{TV}, \text{Radio}, \text{Newspaper})$$

y

$$\{\text{TV} \geq 0, \text{Radio} \geq 0, \text{Newspaper} \geq 0, \text{TV} + \text{Radio} + \text{Newspaper} \leq P\},$$

si P es el presupuesto total del que dispone nuestro cliente.

El problema es que encontrar una tal f no es fácil:

- Los datos disponibles estarán, generalmente, afectados de ruido.
- Sería de extrañar que no hubiese ninguna otra variable relevante en el estudio.
- Los datos siempre serán insuficientes para determinar f con seguridad.

Regresión

En este escenario, a las variables de presupuesto se les llama **variables de entrada** (inputs, predictores, features, variables explicativas o variables independientes), y suelen denotarse usando la letra X , con algún subíndice para distinguir a unas de otras.

Por su parte, a la variable dependiente Sales se le llama **variable de salida, respuesta o variable objetivo**, y suele denotarse por la letra Y .

Regresión

Si tenemos una variable respuesta cuantitativa Y y p predictores diferentes X_1, X_2, \dots, X_p , suponemos que existe una cierta relación entre $X = (X_1, X_2, \dots, X_p)$ e Y que es de la forma

$$Y = f(X) + \varepsilon,$$

donde f es una función fija (pero desconocida) y ε es un término de error que es independiente de X y que tiene media 0.

Un **problema de regresión** consiste en estimar una función hipótesis que aproxime a f de la mejor forma posible, al menos, en los datos disponibles.

Regresión

Existen diversos motivos por los que uno querría estimar dicha hipótesis que principalmente pueden clasificarse en relativos a:

- Realización de **predicciones**.
- **Inferencia de relaciones**.

Regresión lineal

En un modelo de **regresión lineal**, asumimos que existe una relación lineal entre X_1, X_2, \dots, X_p e Y de la forma

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

El caso más sencillo, es un modelo de **regresión lineal simple**.

En este caso, asumimos que existe una relación entre Y y una única variable predictora X :

$$Y \approx \beta_0 + \beta_1 X_1$$

Para realizar un modelo de regresión lineal en Python, usaremos el módulo ***Scikit-Learn***.

5 | Visualización estática y dinámica

Visualización estática

Matplotlib

Es un módulo de Python que permite la creación de visualizaciones 2D (y 3D) basadas en datos.

La librería fue creada por John Hunter (1968 - 2012) en 2002 – 2003.

Aunque cada vez están apareciendo más alternativas en Python (seaborn, plotly, bokeh, plotnine...), el desarrollo de matplotlib sigue siendo muy activo.

Matplotlib: características principales

- Ofrece un **conjunto amplio de funciones**(a través del submódulo matplotlib.pyplot) que permiten crear diferentes tipos de gráficos (puntos, barras, líneas, etc.).
- La llamada a estas funciones **NO genera un gráfico**, sino que modifican el estado de un objeto interno y común a todas las llamadas.
- La utilización de la función **show** ofrecida por el módulo será la que se encargue de renderizar el objeto interno y común en el estado que se encuentre.
- Permite, por tanto, **la creación iterativa de gráficos**.

Plotnine

Plotnine es una implementación de una gramática de gráficos en Python, se basa en **ggplot2**. Este módulo está construido sobre matplotlib.

Esta gramática permite componer gráficas al mapear explícitamente los datos a los objetos visuales que componen la gráfica.

Esta forma de crear visualizaciones es muy **potente** y **flexible**: permite personalizar tanto como queramos nuestros gráficos de una forma simple.

Plotnine

Los gráficos plotnine se construyen paso a paso agregando nuevos elementos uno encima del otro usando el operador `+`.

Para construir un gráfico plotnine necesitamos “conectar” el gráfico a un `DataFrame` específico usando el argumento `data`. Usaremos la función **ggplot** para ello.

Plotnine

Los gráficos plotnine se construyen paso a paso agregando nuevos elementos uno encima del otro usando el operador `+`.

Para construir un gráfico plotnine necesitamos “conectar” el gráfico a un `DataFrame` específico usando el argumento `data`. Usaremos la función **`ggplot`** para ello.

¿Qué necesitamos para hacer un gráfico con plotnine?

Plotnine

Los gráficos plotnine se construyen paso a paso agregando nuevos elementos uno encima del otro usando el operador `+`.

Para construir un gráfico plotnine necesitamos “conectar” el gráfico a un `DataFrame` específico usando el argumento `data`. Usaremos la función **ggplot** para ello.

¿Qué necesitamos para hacer un gráfico con plotnine?



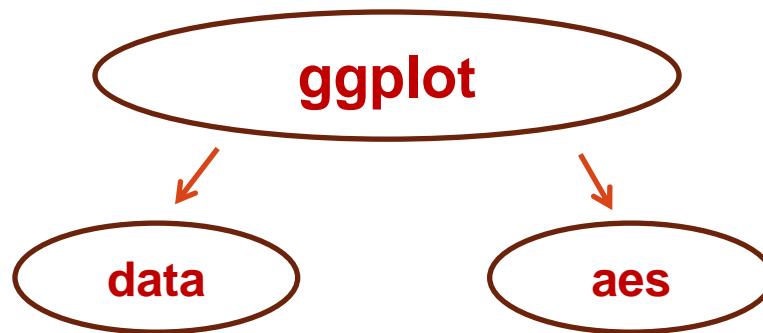
ggplot

Plotnine

Los gráficos plotnine se construyen paso a paso agregando nuevos elementos uno encima del otro usando el operador `+`.

Para construir un gráfico plotnine necesitamos “conectar” el gráfico a un DataFrame específico usando el argumento `data`. Usaremos la función **ggplot** para ello.

¿Qué necesitamos para hacer un gráfico con plotnine?

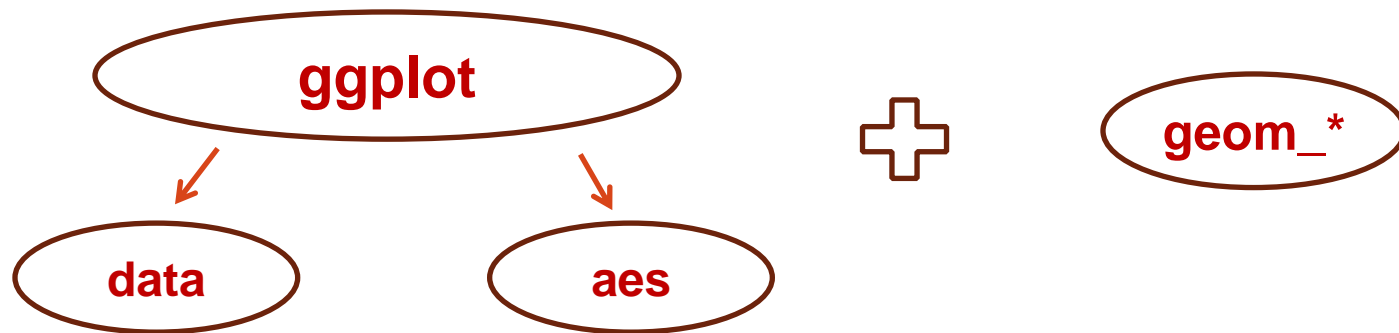


Plotnine

Los gráficos plotnine se construyen paso a paso agregando nuevos elementos uno encima del otro usando el operador +.

Para construir un gráfico plotnine necesitamos “conectar” el gráfico a un DataFrame específico usando el argumento **data**. Usaremos la función **ggplot** para ello.

¿Qué necesitamos para hacer un gráfico con plotnine?



Visualización dinámica

Plotly

Es una herramienta web para analizar y visualizar datos.
Además, existe una librería de Python para crear gráficos interactivos.

Existen tres principales objetos en Plotly:

- **Data:** este objeto contiene los datos que queremos dibujar. A la colección de datos junto con las especificaciones a dibujar se le llama ***trace***.
- **Layout:** es el objeto donde especificamos el título principal, los títulos de los ejes, los colores, formas, anotaciones, etc.
- **Figure:** el objeto *figure* es el resultado final de combinar los objetos *data* y *layout*.

Plotly

Es una herramienta web para analizar y visualizar datos.
Además, existe una librería de Python para crear gráficos interactivos.

Existen tres principales objetos en Plotly:

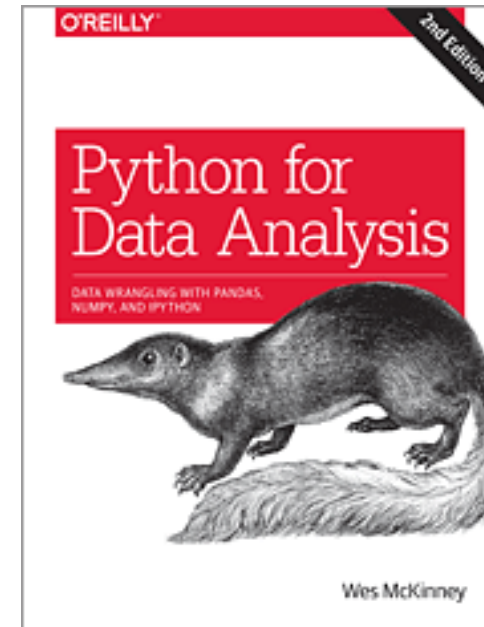
- **Data:** este objeto contiene los datos que queremos dibujar. A la colección de datos junto con las especificaciones a dibujar se le llama ***trace***.
- **Layout:** es el objeto donde especificamos el título principal, los títulos de los ejes, los colores, formas, anotaciones, etc.
- **Figure:** el objeto *figure* es el resultado final de combinar los objetos *data* y *layout*.

Otra forma de crear gráficos interactivos con plotly, es usar la función **`mpl_to_plotly`**, para “convertir” gráficos estáticos generados con matplotlib en gráficos dinámicos.

6 | Referencias

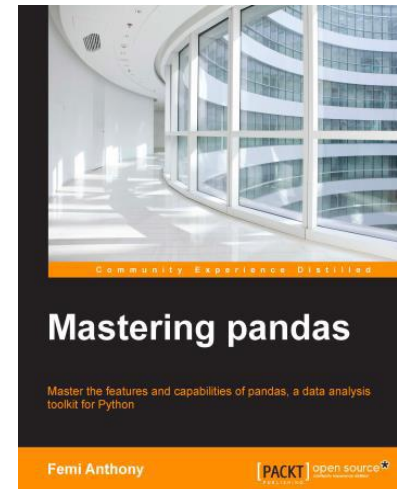
Referencias (I)

- “Python: A Simple Tutorial” de Matt Huenerfauth (Universidad de Pennsylvania).
<https://cs.brynmawr.edu/Courses/cs372/spring2012/slides/PythonTutorial.pdf>
- Curso básico de Python.
<https://www.tutorialpython.com/variables-en-python/>
- PEP8: Guía de estilo.
<https://www.python.org/dev/peps/pep-0008/>
- Python for Data Analysis (O’Reillys).



Referencias (II)

- Página oficial de NumPy: <http://www.numpy.org>
- Página oficial de SciPy: <http://www.scipy.org>
- Página oficial de Pandas: <https://pandas.pydata.org/>
- Mastering pandas (PACKT Publishing).
- Página oficial de matplotlib: <http://matplotlib.org/>



Referencias (III)



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

[https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PandasPythonForDataScience+\(1\).pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PandasPythonForDataScience+(1).pdf)

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Pandas_Cheat_Sheet_2.pdf



Afi Escuela
de Finanzas

© 2020 Afi Escuela de Finanzas. Todos los derechos reservados.