TecCom Studies
Estudios de tecnología y comunicación

Número 8 - Año 6 (Enero-Diciembre 2017)
Facultad de Ciencias de la Información
Universidad Complutense de Madrid
Artículo bajo la licencia *Creative Commons*

Programación y uso de Python en el desarrollo del periodismo de datos

Autor: Yolanda García Ruiz, Victoria López, Guadalupe Miñana (Grupo Grasia, Social BigData)
Universidad / Institución / Centro: Universidad Complutense de Madrid
Páginas: 25-33
Descriptor: Programación, Phyton, datos, periodismo.
País: España
Ciudad: Madrid
Contacto: ygarciar@fdi.ucm.es

Resumen: Uno de los propósitos del periodismo de datos y del periodismo de investigación es la creación de historias visuales a través de la información y el tratamiento de los datos. En este sentido, Python es un lenguaje ideal para todos aquellos que deseen iniciarse en el mundo de la programación y el análisis de datos. Este lenguaje tiene la ventaja de que incluye una gran cantidad de librerías con herramientas que resuelven aspectos muy variados dentro de un proyecto periodístico. Por ejemplo, el acceso a la información contenida en medios digitales, el procesamiento de grandes volúmenes de datos que son difíciles de procesar usando medios tradicionales, así como la generación de gráficos que permitan a los usuarios visualizar los datos e interactuar con ellos.

En este artículo presentamos el lenguaje Python junto con el entorno de desarrollo Jupyter notebook y las librerías básicas para la explotación y el análisis de información.

Palabras clave: Python, Periodismo de Datos, Análisis de Datos, Programación

Abstract: One of the purposes of data journalism is the creation of visual stories from information and data analysis. Python is a suitable language for all who wish to begin in the world of programming and data analysis. This programming language includes a large number of libraries with tools forhandling varied taskswithin a journalistic project. For example, searching information in digital media, processing huge volumes of data as well as building visualizations that allow users the interaction with data and its better understanding.

In this article we introduce the programming language Python, its development environment Jupyter notebook and the main libraries in Python for the analysis of information.

Keywords: Python, Data Journalism, Data Analysis, Programming

1. PYTHON COMO HERRAMIENTA EN EL PERIODISMO DE DATOS

Python (Python Software Foundation, 2017) es un lenguaje de programación muy popular en el contexto del software libre. Gran parte de esta popularidad se debe a que se trata de un lenguaje de alto nivel y muy expresivo cuya sintaxis es limpia y sencilla. Es un lenguaje vivo y en continua evolución ya que está respaldado por una amplia comunidad de desarrolladores, proveedores de las funcionalidades y librerías necesarias para responder a las nuevas y crecientes demandas tecnológicas. Es por ello por lo que es cada vez más utilizado por profesionales de todos los campos y con un perfil no tan tecnológico como nos podríamos imaginar. Desde investigadores, estadistas e ingenieros hasta periodistas, economistas y profesionales de ciencias de la salud coinciden en que Python es un lenguaje amigable, fácil de leer, sencillo de aprender y que proporciona un universo de posibilidades.

Actualmente existen herramientas en Python para el desarrollo de aplicaciones en casi cualquier contexto. Permite desarrollar aplicaciones y páginas web, analizar y visualizar datos, realizar estadísticas, gestionar bases de datos (*SQL* y *NoSQL*) y un amplio etcétera. Gran parte de estas herramientas forman parte de la biblioteca estándar del lenguaje, mientras que otras han sido desarrolladas por la comunidad de desarrolladores de Python y es necesario instalarlas para poder usarlas. En el contexto del análisis y procesamiento de grandes volúmenes de datos (*Big Data*), Python es una alterativa de éxito ya que dispone de todo un ecosistema de librerías que incluyen las herramientas necesarias en todas las etapas de análisis, desde la extracción de datos y su preparación y limpieza hasta la representación gráfica y su posterior análisis.

2. EL ENTORNO DE DESARROLLO JUPYTER

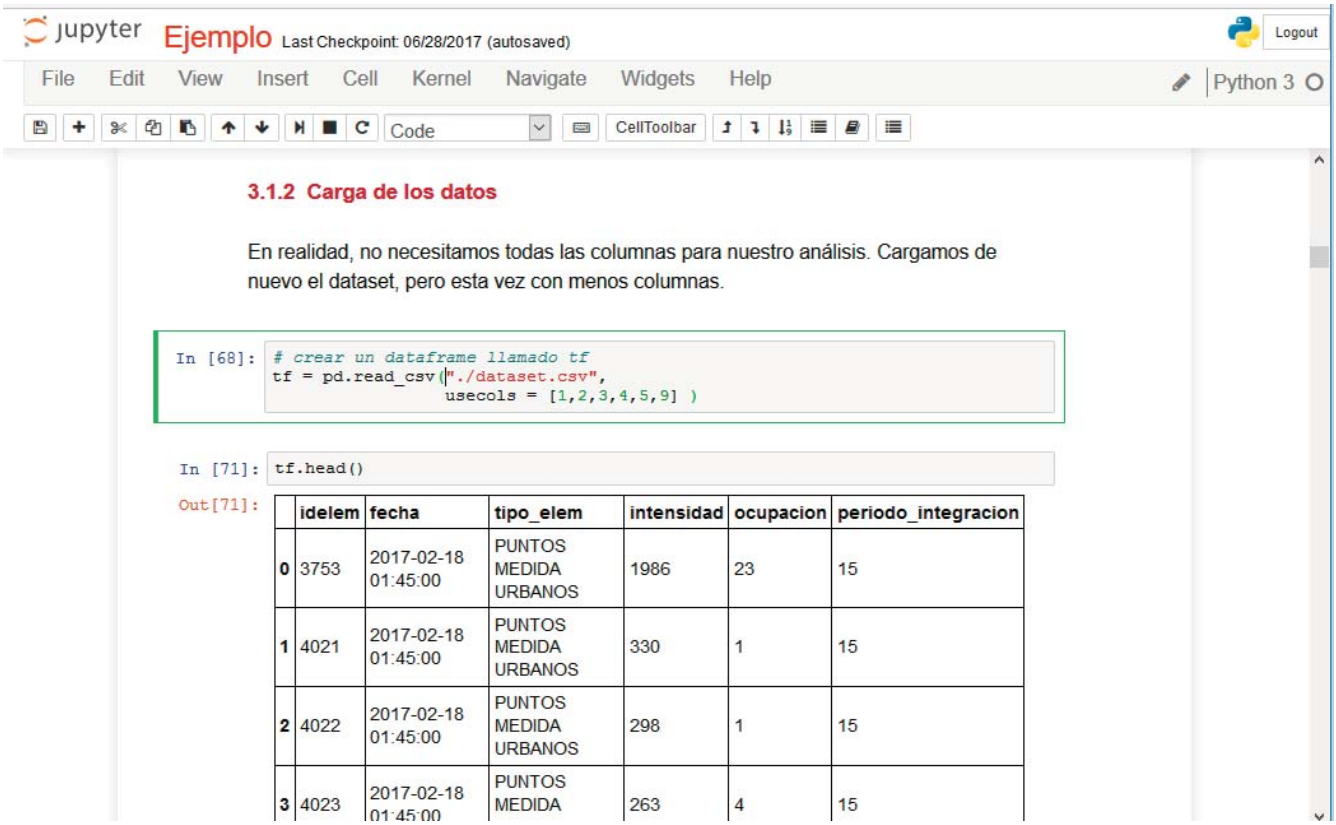
En cualquier lenguaje de programación, los entornos de desarrollo constituyen un elemento esencial. En general, los entornos de desarrollo incluyen utilidades para ayudar al usuario tanto en la tarea de escribir programas como de ejecutarlos. En el caso concreto de Python, dada su popularidad, existen cientos de entornos de desarrollo, tanto gratuitos como de pago, de escritorio o de tipo online. Aquí introducimos el entorno de desarrollo *Jupyter Notebook* (Jupyter. org, 2017) por ser una herramienta indispensable para el trabajo de los científicos de datos. Contiene todas las herramientas científicas estándar de Python, y permite realizar todas las tareas típicas en el

contexto del análisis de datos: importación y exportación, manipulación y transformación, visualización, creación de modelos estadísticos, aprendizaje automático y mucho más.

*Jupyter Notebook*es una aplicación web de software libre que permite integrar instrucciones escritas en el lenguaje de programación Python, y por tanto código ejecutable, junto con contenidos multimedia o textuales que en esencia, ayudan a documentar y facilitan la comprensión de los algoritmos realizados (ver ***Figura 1***) . La ejecución de los bloques de código se puede realizar paso a paso o todos de golpe, lo que permite obtener todos los resultados parciales. El resultado final es un fichero que contiene una serie de piezas de código, notas y resultados denominado *notebook.*

Jupyter está integrado en plataformas como GitHub (García-Ruiz, 2017) o Databricks (DataBriks, 2017) que facilitan la compartición de conocimiento y resultados a través de los *notebooks*. Esto permite difundir los notebooks, cuyos resultados pueden ser reproducidos y validados por terceros en distintos entornos. Son muchos los profesionales que usan este entorno de desarrollo, desde científicos y docentes hasta periodistas de

Figura 1. Notebook de Jupyter - Lectura de datos



	idelem	fecha	tipo_elem	intensidad	ocupacion	periodo_integracion
0	3753	2017-02-18 01:45:00	PUNTOS MEDIDA URBANOS	1986	23	15
1	4021	2017-02-18 01:45:00	PUNTOS MEDIDA URBANOS	330	1	15
2	4022	2017-02-18 01:45:00	PUNTOS MEDIDA URBANOS	298	1	15
3	4023	2017-02-18 01:45:00	PUNTOS MEDIDA	263	4	15

Fuente: Elaboración propia.

datos (Buzz Feed and the BBC, 2016; Propublica, 2016; Los Angeles Times, 2016) .

Jupyter se encuentra integrado en un paquete de utilidades para la edición y desarrollo de programas en Python denominado Anaconda (Continuum Analytics, 2017) . La instalación de Anaconda nos dará acceso directo a la aplicación Jupyter Notebook. También es posible probar este entorno de desarrollo desde la página https://try. jupyter. org/sin necesidad de tener instalado el software Anaconda. Dicha página proporcionaalgunos ejemplos para poder ejecutar y permite crear nuevos notebooks.

3. LAS LIBRERÍAS BÁSICAS DE PYTHON PARA ANÁLISIS DE DATOS

3.1. Acceso a los datos y procesamiento

Los datos son considerados la fuente del periodista de datos y la herramienta con la que narrar historias. Una tarea importante y en muchos casos engorrosa y pesada es la recolección de los datos. Esto es debido a que en muchos casos la información se encuentra dispersa en diferentes fuentes y en diferentes formatos (volúmenes masivos y complejos de información estructurada y no estructurada) . En algunos casos la información se encuentra enlazada y accesible a todos los usuarios en páginas web, como es el caso de los portales de datos abiertos. En estos casos se suele ofrecer la información en distintos formatos (CSV, JSON, XML o Excel) . En otros casos, la información se encuentra dispersa en páginas web y solo en algunas ocasiones es accesible (y de forma parcial) a través de APIs[1]. En el caso de ausencia de APIs, es necesario recurrir a un conjunto de técnicas que se utilizan para obtener de forma automática el contenido que hay en páginas web a través de su código HTML, es lo que se conoce como *scraping*.

En general, es necesario disponer de métodos computacionales que permitan extraer tal volumen de información. Aunque son varias las librerías de Python que proporcionan herramientas para importar y exportar datos, la librería *Pandas*aporta una gran potencia en este aspecto, resolviendo así muchos de los problemas encontrados a la hora de recolectar datos en diferentes formatos. Empezó a desarrollarse en el año 2008 por Wes McKinney (McKinney, 2012) en un intento por conseguir una herramienta flexible y de alto rendimiento para el tratamiento de datos financieros. Pandas es actualmente una de las librerías de alto nivel de Python más utilizadas en las primeras etapas del proceso de análisis de datos, especialmente apropiada en aplicaciones estadísticas.

Una gran parte del tiempo empleado en la tarea de análisis de datos se invierte en la preparación de los mismos. En la mayoría de las ocasiones, es necesario que los datos tengan un determinado formato. Sin embargo, los datos que se encuentran en ficheros o en bases de datos, no tienen por qué cumplir, a priori, ningún requisito, por lo que será necesario proceder a su limpieza, transformación, clasificación, procesamiento, etc. Todo ello

[1] API (Application Programming Interface) , conjunto de reglas que las aplicaciones ponen a disposición de terceros para dar acceso de forma segura a cierta informaciónrestringida. Ejemplos de aplicaciones con APIs: Twitter, Flickr.

con el objetivo de generar nuevos datos con un gran valor comunicativo.

La librería Pandas proporciona estructuras de datos de alto nivel que permiten representar series de datos (*series*) y datos en forma tabular(*dataframes*) , así como herramientas diseñadas específicamente para un tratamiento de datos rápido y sencillo. Está construida sobre los fundamentos de *NumPy*, librería básica de Python para realizar cálculo científico, resultando especialmente útil cuando hay que trabajar con datos heterogéneos representados de forma tabular.

La librería Pandas presenta un conjunto de funciones recogidas en lo que se conoce como el I/O API de Pandas. En muchas ocasiones los datos se encuentran almacenados en ficheros de texto y representados en forma de tabla. El formato más popular es el CSV, donde los valores de cada una de las filas están separados por el símbolo coma. En otras ocasiones, los valores de cada fila están separados por otro símbolo, un tabulador o un espacio. Las funciones más utilizadas en Pandas para leer de forma flexible ficheros de texto plano son *read_csv* y *read_table*(ver **Figura 1**) . Otra forma muy habitual de guardar datos en forma tabular es mediante hojas de cálculo en un libro o fichero de Microsoft Excel. Pandas incluye la función *read_excel* para realizar la lectura de datos y crear dataframes. Todas estas funciones admiten un gran número de parámetros que permiten modificar el comportamiento por defecto. Eligiendo los parámetros adecuados es posible efectuar la lectura de un subconjunto de los datos (importando solo aquellos datos necesarios para el análisis posterior) , sustituir unos valores por otros, gestionar los valores nulos, aplicar funciones de conversión en el procedimiento de carga, por ejemplo para la lectura adecuada de fechas, para cambiar un texto en minúsculas a mayúsculas, y en general todo lo relacionado con el tratamiento de textos, etc.

Para extraer los datos alojados en sitios web es necesario acceder a su código HTML, para lo cual es necesario realizar una petición siguiendo el protocolo HTTP Request/Response. La librería *Request* (Reitz, 2017) de Python ofrece utilidades para realizar este tipo de peticiones. Los documentos HTML pueden ser tratados directamente en Pandas. Por un lado, la función*to_html* realiza la transformación de dataframes en tablas HTML de forma automática. La operación inversa también es posible; la función *read_html* es capaz de recorrer el código HTML de una página web en busca de los datos contenidos en tablas. Pero, ¿qué ocurre si lo que nos interesa son datos que no están representados en tablas HTML?. La librería *BeautifulSoup* (Richardson, 2017) junto con *Selenium* (Muthukadan, 2014) aportan los métodos necesarios para navegar entre las etiquetas HTML y recolectar los datos de forma sencilla. La función *find_all* (ve rFigura 2) se utiliza para recuperar todos los elementos de un determinado tipo y una clase concreta.

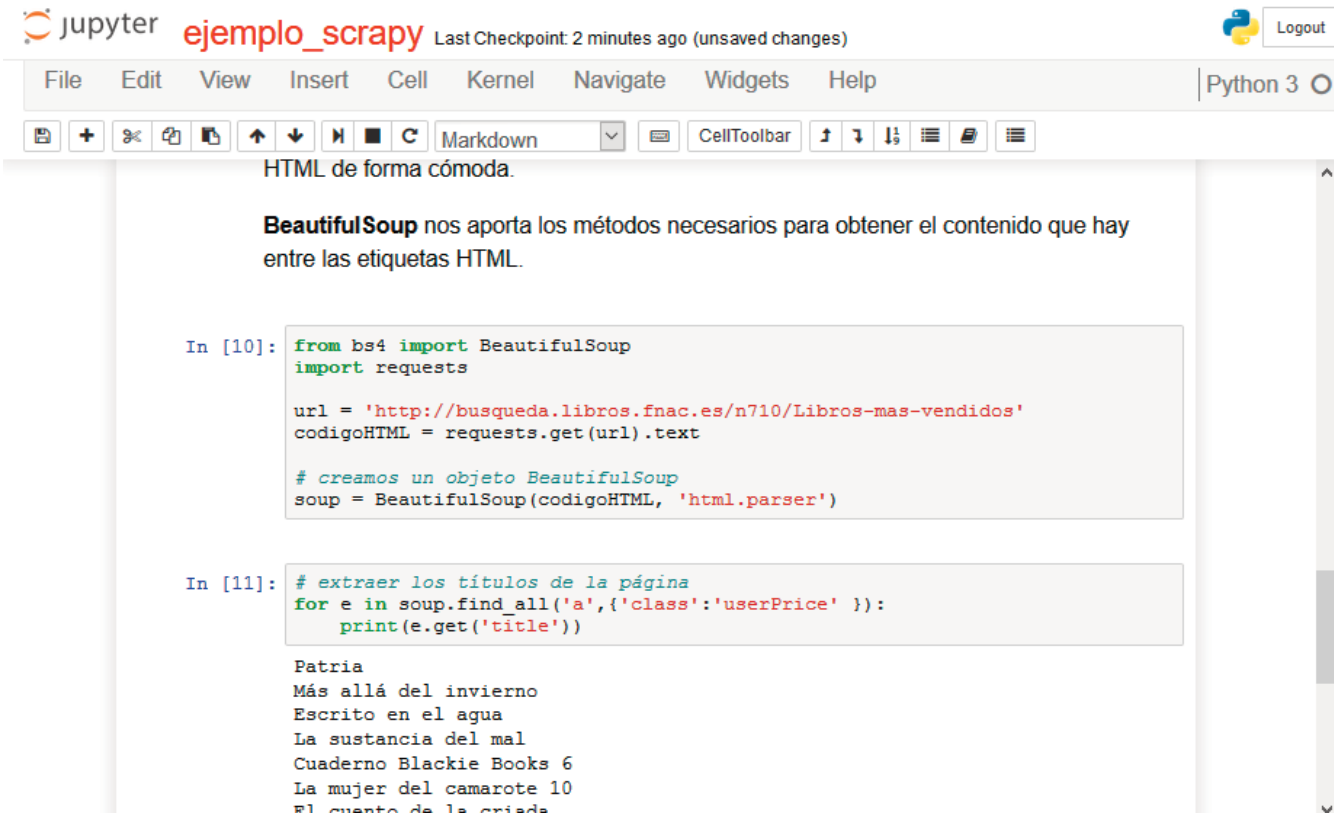
Uno de los estándares para el intercambio de datos en la Web es el formato JSON (JavaScript Object Notation) . Al igual que ocurre con XML, permite representar cualquier estructura de datos de una forma jerárquica, la cual puede ser fácilmente representada en forma de árbol. Este tipo de documentos se encuentra de forma habitual en portales de datos abiertos. También son proporcionados por sitios web a través de sus APIs

(Twitter, 2017) . Pandas dispone de la función *read_json* para crear dataframes a partir de un documento JSON. La operación recíproca, el método *to_json*permite guardar dataframes en formato JSON.

Pero la gestión de la información mediante estructuras de ficheros no es siempre la forma más adecuada de almacenamiento de datos. Es por todos conocido que se pierden propiedades deseables como son la integridad, la no redundancia, etc. Son muchas las aplicaciones que usan bases de datos como forma de almacenamiento, ya sean bases de datos relacionales, también conocidas como bases de datos SQL, como las aparecidas recientemente no relacionales o NoSQL.

En Python existe un mecanismo estándar para el acceso a bases de datos relacionales. Este mecanismoes similar para todos los sistemas gestores de bases de datos, y lo único que cambiaes la librería a utilizar para realizar la conexión. Por ejemplo, para el caso de MySQL, se utilizará la librería *mysql. connector*. Esta librería tiene que ser instalada antes de su importación, ya que no viene incluida en Anaconda. La creación de un dataframe a partir de los datos almacenados en una base de datos relacional es muy simple utilizando

Figura 2. Ejemplo de *scraping* de una página web con Beautiful Soup.



Fuente: Elaboración propia.

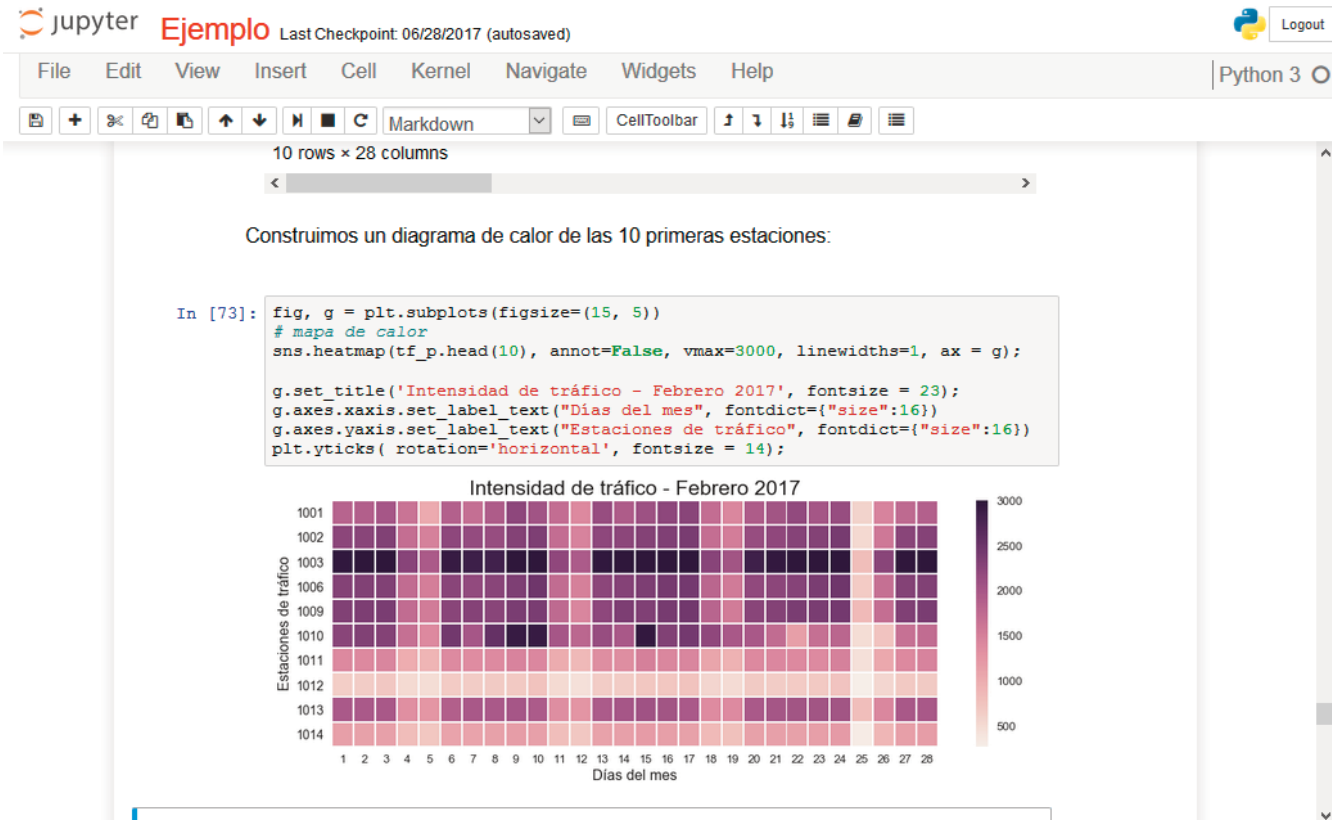
las funciones que proporciona Pandas. La función *read_sql* permite ejecutar una consulta SQL pasada como argumento. El resultado, será un *dataframe* con la misma estructura que el resultado de la consulta.

MongoDB (MongoDB, 2017) es una base de datos NoSQL muy popular, la cual está orientada a documentos. Estos documentos son de tipo JSON, cuyo tratamiento ya se ha comentado anteriormente. La librería Pandas cuenta con las herramientas necesarias para, una vez establecida la conexión con MongoDB, crear dataframes a partir de los datos contenidos en la base de datos, así como para almacenar los datos contenidos en los dataframes de Pandas en MongoDB. En el servidor MongoDB pueden existir muchas bases de datos independientes y cada base de datos puede tener múltiples colecciones. Así mismo, cada colección puede contener múltiples documentos. Para recuperar todos los documentos de una colección, la librería proporciona la función *find.*

3.2. Visualización de los datos

Matplotlib (The Matplotlib development team, 2017) es una de las librerías en Python

Figura 3. Diagrama de calor generado con las librerías Matplotlib y Seaborn



Fuente: Elaboración propia.

más utilizadas para la visualización de datos. Una de las características que hace que tenga éxito esla facilidad de uso; con muy pocas líneas de código se puede generar una gran variedad de gráficos (histogramas, áreas, visualizaciones de líneas, barras, diagramas de dispersión, etc.) que ayudan a comprender los datos. Por otro lado, la librería *Basemap* (The matplotlib development team, 2016) , como complemento de Matplotlib, permite construir visualizaciones con mapas.

Más orientada a la visualización de información estadística, la librería *Seaborn* (Waskom, 2017) proporciona herramientas capaces de realizar de forma automática las transformaciones necesarias a los datos, para posteriormente construir gráficos atractivos y con un alto nivel de información. Normalmente, ambas librerías, Seaborn y Matplotlib, se utilizan al mismo tiempo ya que la combinación de ambas permite construir gráficos de una mayor calidad (ver Figura 3) .

Para crear visualizaciones interactivas, Python cuenta con la librería *Bokeh* (Bokeh Development Team, 2016) . Se trata de una librería al estilo D3. js (Bostock, 2017) , capaz de manejar y representar de forma elegante grandes volúmenes de datos. Entre otras características, Bokeh proporciona herramientas para crear cuadros de mando y aplicaciones con datos (Rosling, 2006) .

En general, los gráficos generados por todas estas librerías pueden ser grabados en diferentes formatos, ya sea pdf, jpg o html, lo que permitirá incluirlos en cualquier proyecto web.

4. CONCLUSIONES

Existe una necesidad entre los profesionales del periodismo de datos de formarse para trabajar con grandes volúmenes de datos en todas las etapas, desde la búsqueda y depuración de los datos, pasando por su visualización, hasta la generación de aplicaciones con datos.

Este artículo presenta al lenguaje Python y sus librerías junto con el entorno de desarrollo Jupyter Notebook, como una herramienta sencilla y eficiente capaz de cubrir por si sola, todas y cada una de las etapas de análisis. Pandas, como librería básica para la gestión de datos en diferentes formatos, desde los más habituales como son CSV, txt o Microsoft Excel, hasta las Bases de datos, pasando por los ficheros en formato HTML y el estándar para el intercambio de datos JSON. Request, BeautifulSoup y Selenium para la extracción de datos alojados en laweb. Las librerías de visualización, desde las más sencillas como Matplotlib y Seaborn, hasta aquellas que aportan más riqueza e interactividad como Bokeh.

5. BIBLIOGRAFÍA:

Bokeh Development Team. (2016) . *Bokeh: Python library for interactive visualization*. Retrieved from http://bokeh. pydata. org/en/latest/

BOSTOCK, M. (2017) . *Data-Driven Documents*. Retrieved from https://d3js. org/

BUZZ FEED and the BBC. (2016) . *Data and Analysis: Detecting Match-Fixing Patterns In Tennis*. Retrieved from https://github.

com/BuzzFeedNews/2016-01-tennis-betting-analysis/blob/master/notebooks/tennis-analysis. ipynb

Continuum Analytics. (2017) . *Anaconda: The Most Popular Data Science Ecosystem*. Retrieved from https://www. continuum. io/downloads

Data Briks. (2017) . *DataBriks: The Unified Analytics Platform*. Retrieved from https://community. cloud. databricks. com/login. html

GARCÍA-RUIZ, Y. (2017) . *Programación y uso de Python en el desarrollo del periodismo de datos*. Retrieved from https://github. com/ygarciar/PeriodismoDeDatosUCM/blob/c0016a4d5e06a572cb73b620f3265b657ef72229/Tema_1/pd. ipynb

Jupyter. org. (2017) . *Try Jupyter*. Retrieved from https://try. jupyter. org/

Los Angeles Times. (2016) . *L. A. vacant building complaints analysis*. Retrieved from https://github. com/datadesk/la-vacant-building-complaints-analysis/blob/master/la-vacant-building-complaints-analysis. ipynb

MCKINNEY, W. (2012) . *Python for Data Analysis.* EEUU: O'Reilly.

MONGO DB. (2017) . *MongoDB*. Retrieved from https://www. mongodb. com/es

MUTHUKADAN, B. (2014) . *Selenium with Python*. Retrieved from http://selenium-python. readthedocs. io/

NLTK project. (2017) . *NLTK project*. Retrieved from NLTK

Propublica. (2016) . *COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) Analysis*. Retrieved from https://github. com/propublica/compas-analysis/blob/master/Compas%20Analysis. ipynb

Python Software Foundation. (2017) . *Python Software Foundation*. Retrieved from https://www. python. org/

REITZ, K. (2017) . *Requests: HTTP for Humans*. Retrieved from http://docs. python-requests. org/en/latest/

RICHARDSON, L. (2017) . *Beautiful Soup*. Retrieved from https://www. crummy. com/software/BeautifulSoup/

ROSLING, H. (2006) . *A Reproduction of Gapminder*. Retrieved from https://demo. bokehplots. com/apps/gapminder

The matplotlib development team. (2016) . *Basemap*. Retrieved from http://matplotlib. org/basemap/

The Matplotlib development team. (2017) . *Matplotlib*. Retrieved from http://matplotlib. org/

Twitter. (2017) . *The Search API*. Retrieved from https://dev. twitter. com/rest/public/search

WASKOM, M. (2017) . *seaborn: statistical data visualization*. Retrieved from http://seaborn. pydata. org/index. html