El lenguaje de programación Python como herramienta multiuso en el análisis de datos y cálculo actuarial

FRANCISCO GÁRATE SANTIAGO
Actuario

Las nuevas tecnologías han cambiado por completo la profesión actuarial, y por tanto, el conocimiento de lenguajes de programación se ha convertido en algo indispensable para los actuarios. Sin embargo, han sido otros ámbitos dentro de las entidades aseguradoras los que han desarrollado más activamente el uso de la analítica y explotación de sus bases de datos, principalmente en áreas comerciales y de prevención del fraude. Así, términos como minería de datos *(datamining)* o inteligencia de mercado *(business intelligence)* son conceptos con los que las aseguradoras están familiarizadas. Recientemente, y gracias a normativas como la de Solvencia II, la calidad y la protección de los datos han pasado a ser un objetivo prioritario de todas las entidades. Para ello, han implementado o potenciado sus repositorios y diccionarios de datos con el fin de que cualquier miembro de la organización que quiera obtener un dato tenga la certeza de que estará usando la misma información que el resto de la organización, aunque sea para distintos fines.

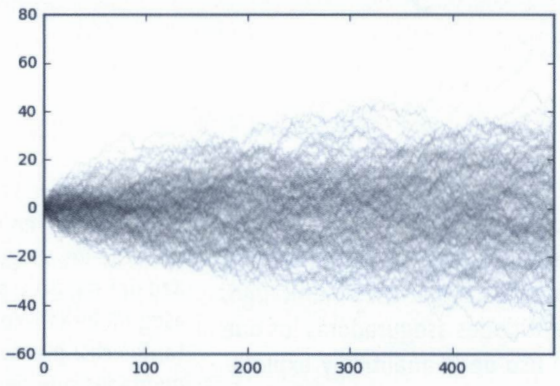
La necesidad de que la evaluación interna de riesgos y de la solvencia forme parte integrante de la estrategia comercial de las entidades y deba tenerse en cuenta de forma continua en las decisiones estratégicas, debe servir a las entidades aseguradoras, y en particular a la profesión actuarial, para fijarse en las soluciones de inteligencia de datos o *Big Data* como el próximo salto tecnológico. Las entidades poseen una gran cantidad de datos que, si fueran accesibles adecuadamente, servirían no sólo para su modelización de riesgos (tarificación, suscripción, reservas, catastrófico, prevención del fraude...) o en su ámbito financiero (VaR, ALM, CAPM...), sino también para obtener información de valor que ayude a la organización a tomar mejores decisiones estratégicas en aspectos tales como la mejora de la experiencia del cliente o el desarrollo de modelos predictivos que ayuden a la creación de nuevos productos o segmentaciones que ofrezcan soluciones a medida y minimicen sus riesgos. De hecho, ya se advierte desde el sector que el acceso a las nuevas fuentes de información en búsqueda de un mayor principio de equidad y eliminaciones de asimetrías de información en la contratación del seguro, pueden poner en riesgo no sólo el derecho a la privacidad de los asegurados, sino el principio de solidaridad del seguro y, por consiguiente, la propia actividad aseguradora tal como funciona hoy en día, pasando de tarifas casi planas y solidarias hacia tarifas hipersegmentadas que dejarían fuera del mercado a muchos individuos por motivos definidos por algoritmos difícilmente explicables.

Estas nuevas técnicas requieren por un lado interconectar múltiples fuentes de datos de diversa naturaleza y origen, con una necesidad de almacenarlos en entornos diferentes a los que se generaron y de estar a disposición desde el mismo momento en que se generan, y por otro lado profesionales con gran conocimiento y especialización en múltiples disciplinas tanto científicas, sociológicas y del ámbito normativo que sepan hacer las preguntas correctas a los datos y analizar las respuestas obtenidas. Así, el ecosistema denominado Big Data es cada día más complejo, donde aparecen continuamente nuevos entornos de trabajo con aplicaciones y plataformas analíticas cada vez más especializadas y que solucionan limitaciones técnicas anteriores. Ya no sólo se habla de bases de datos no relacionales o NoSQL, aquellas que no requieren necesariamente estructuras fijas tipo SQL con índices, tablas o columnas, sino de entornos completos y aplicaciones diseñadas específicamente para gestionar, optimizar y analizar fácilmente la información en función del objetivo buscado.

En relación a la interfaz de consulta, ya sea para acceder con los datos o para implementar los algoritmos que interactúen y analicen dicha información, existen principalmente apenas cuatro lenguajes de programación con los que podemos desenvolvernos en este ecosistema: Python, R, Scala y Julia, cinco si tenemos en cuenta a SAS. Cada lenguaje posee sus propios pros y contras, y entrar a valorar cual es mejor no es tarea fácil ya que dependerá de infinidad de factores. Así, con carácter general, estos lenguajes serán fácilmente accesibles para aquellos familiarizados previamente con algún lenguaje de programación. Julia

EJEMPLO. Ejemplo de tan solo cuatro líneas de código, haciendo uso de NumPy y SciPy, que muestra 200 trayectorias que resulta de hacer 500 sucesivos pasos aleatorios *(random walks)* empezando desde 0

```
In [1]: import numpy as np
        import pandas as pd
        datos = pd.DataFrame(np.random.randn(500,200).cumsum(axis=0))
        datos.plot(legend=False,alpha=0.05)
Out[1]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9f21286c50>
```

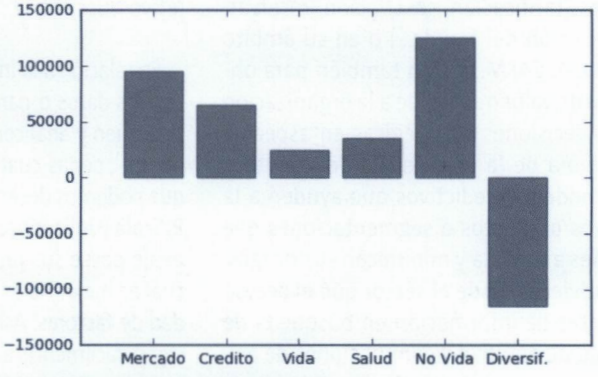


es similar en cuanto a su filosofía a Matlab, R es entre los lenguajes mencionados el más parecido a C, y Scala, como lenguaje de programación orientado a objetos, sería el más cercano a Java. En cambio, Python sería el lenguaje ideal para aquellos que se inicien o posean conocimientos básicos de programación.

Python, R, Scala y Julia son lenguajes abiertos y libres, y por consiguiente gratuitos, no así otros lenguajes o programas estadísticos como Stata, SPSS, Statistica o SAS. Este hecho, ha contribuido al significativo aumento de usuarios y desarrolladores dentro de la comunidad de código abierto y ha hecho posible la existencia de una

EJEMPLO. Ejemplo de cálculo del BSCR aplicando la matriz de correlación correspondiente a los diferentes submódulos de riesgo del SCR y mostrando gráficamente el resultado de cada uno de ellos incluyendo el efecto diversificación

```
In [2]: import matplotlib.pyplot as plt
        import numpy as np
        SubSCR = np.array([95000.25,65000.50,25000.25,35000.45,125000.75])
        MatCorBSCR = np.array([[1, 0.25, 0.25, 0.25, 0.25],[0.25, 1, 0.25, 0.25, 0.5],
                               [0.25, 0.25, 1, 0.25, 0],[0.25, 0.25, 0.25, 1, 0],[0.25, 0.5, 0, 0, 1]])
        BasicSCR = np.sqrt(np.dot(np.dot(SubSCR, MatCorBSCR),SubSCR.transpose()))
        BasicSCR_Vector = np.insert(SubSCR, 5, BasicSCR-sum(SubSCR))
        print BasicSCR
        x = np.arange(6)
        plt.bar(x, BasicSCR_Vector, align='center', alpha=0.5)
        plt.xticks(x, ('Mercado', 'Credito', 'Vida', 'Salud', 'No Vida','Diversif.'))
        plt.show()
230462.853573
```



EJEMPLO. Ejemplo de implementación de una función de geolocalización que devolvería las coordenadas gps (latitud y longitud) para las direcciones postales incluidas en un fichero csv externo, en este caso con la siguiente información:
Plaza de la Villa; 1; 28005; Madrid
Plaza de Catalunya; 1; 08002; Barcelona

```
In [3]: import urllib2, json, csv

        def osmap(direccion):
            request = urllib2.Request("http://nominatim.openstreetmap.org/search?q=%s&format=json"
                                      % (direccion), headers={"Accept" : "application/json"})
            response = urllib2.urlopen(request).read()
            data = json.loads(response)
            gps = []
            gps.append(float(data[0]['lat']))
            gps.append(float(data[0]['lon']))
            return gps

        with open('direcciones.csv','r') as fichero:
            direcciones = csv.reader(fichero,delimiter=';')
            for linea in direcciones:
                print osmap(','.join(linea).replace(' ','+'))

        [40.415161, -3.71040241939636]
        [41.3861238, 2.1691827]
```

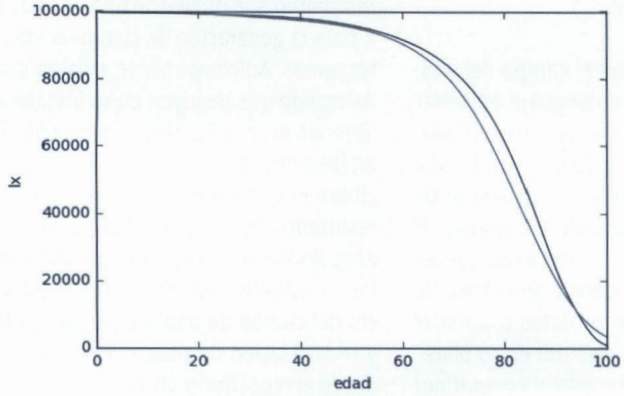
La función hace uso de la API de OpenStreetMap que devuelve los valores en formato json. Posteriormente, con librerías como Arcpy o Cartopy, podrían elaborarse cartografías o mapas, así como usarse otras librerías o funciones útiles para análisis geoespaciales (concentración de riesgos, desastres naturales, etc.)

gran cantidad de librerías abiertas para casi cualquier tipo de implementación imaginable, especialmente en el caso de R y Python. Del mismo modo, dichos lenguajes se entienden perfectamente con otras plataformas de código abierto y de uso común en el análisis de volúmenes masivos de datos (como Hadoop, Apache Beam, MongoDB o CDH) o Sistemas de Información Geográfica libres (GIS). Por otro lado, SAS integra dentro de su herramienta estadística su propio lenguaje, con funcionalidades muy potentes y enfocadas al ámbito empresarial,

EJEMPLO. Ejemplo, haciendo uso de la librería pyliferisk, que mostraría en una gráfica de líneas la comparativa entre de dos tablas de mortalidad

```
In [4]: import matplotlib.pyplot as plt
        import pyliferisk.lifecontingencies as lc
        from pyliferisk.mortalitytables import SPAININE2004, GKM95
        import numpy as np
        tarifa=lc.MortalityTable(nt=SPAININE2004,i=0.02)
        experiencia=lc.MortalityTable(nt=GKM95,i=0.02,perc=75)
        x = np.arange(tarifa.w)
        y = tarifa.lx[:tarifa.w]
        z = experiencia.lx[:tarifa.w]
        plt.plot(x,y, color = 'blue')
        plt.plot(x,z, color = 'red')
        plt.ylabel('lx')
        plt.xlabel('edad')

Out[4]: <matplotlib.text.Text at 0x7f41175baed0>
```



R y Python serían lenguajes muy recomendables para su uso como herramienta de análisis de datos, cálculo actuarial y modelización de riesgos

con un soporte profesional y garantizando una gran estabilidad y compatibilidad de versiones a costa de incorporar novedades con cierto retraso.

Los lenguajes interpretados (como R, Python o Julia) consisten en líneas de instrucciones o scripts que son interpretados en tiempo real, por tanto no requieren ser compilados (como si sucede con C o C++) y pueden ser ejecutados en cualquier sistema operativo e incluso bajo entornos web. Estas características hacen que sean lenguajes muy rápidos y cómodos a la hora de implementar cualquier solución. En contra, a excepción de Julia, una vez en funcionamiento no podrían compararse en cuanto a velocidad de cálculo a otros lenguajes como C, C++ o Fortran, aunque habría que valorar la capacidad de ciertos lenguajes (entre ellos Java, Scala, Python y R) para el procesamiento en paralelo *(cluster computing)* que reducen considerablemente los tiempos de proceso cuando se trabaja con grandes volúmenes de datos. Esta característica puede realizarse de forma totalmente transparente con plataformas tipo *Apache Spark* que permite procesar los datos por lotes, o *Apache Storm*, capaz de procesar los datos conforme se van generando en tiempo real.

En resumen, R y Python serían lenguajes muy recomendables para su uso como herramienta de análisis de datos, cálculo actuarial y modelización de riesgos. Ambos lenguajes son potentes y de sintaxis simple, lo que significa que son relativamente fáciles de escribir y de ser entendidos por los demás sin necesidad de tener amplios conocimientos de programación, además de ser perfectamente compatibles entre sí.

R es el lenguaje por excelencia en el campo del análisis de datos, considerado como el lenguaje estándar en los cálculos estadísticos, lo cual le hace extremadamente recomendable para cálculos actuariales. Además, recientemente Microsoft ha actualizado su servidor de SQL (MS SQL Server 2016) integrándole el lenguaje R para análisis avanzados de datos, permitiendo procesamientos paralelos directamente dentro de la base de datos SQL sin necesidad de mover los datos o trabajar con muestras, pudiéndose además ejecutar en su plataforma Azure de servicios en la nube *(cloud computing)* e interactuar con MS Excel. Sin embargo, para implementaciones que requieran el desarrollo de algoritmos no sólo estadísticos, una mayor y fácil conectividad con bases de datos (desde SQL y similares hasta API externas como Oracle, SAP, IBM DB2, Access, etc.), el uso de estándares para generar reportes (pdf, json, xml, xbrl...) o incluso interacciones con el propio sistema operativo u otros lenguajes de programación, sin duda Python es el lenguaje más recomendable.

El lenguaje Python fue creado por el holandés Guido van Rossum en 1990, y a día de hoy es el lenguaje multiuso más popular y de mayor crecimiento. La mentalidad con qué fue creado hace que sea muy intuitivo y uno de los lenguajes más fáciles de aprender, ideal para desarrollar cualquier tipo implementación en poco tiempo, tanto para realizar operaciones básicas como para el desarrollo de algoritmos de aprendizaje automáticos *(machine learning)*. Pero simplicidad no significa limitaciones, Python es usado, por ejemplo, por los científicos de la NASA o el CERN, y ha sido el lenguaje protagonista en el hallazgo del bosón de Higgs como puede consultarse en la web de este último organismo.

Python posee la filosofía de "baterías incluidas", es decir, incluye en su instalación básica una gran cantidad de módulos y funciones disponibles para que desde el primer momento se pueda empezar a trabajar. Así, por ejemplo, no haría falta instalar ningún paquete adicional para realizar conexiones mysql, leer ficheros csv, usar API externas en formato xml o json, conectarse a servidores de correo smtp o ejecutar comandos del propio sistema operativo. Pero, además de las librerías incluidas de serie, actualmente existen miles de librerías disponibles en su repositorio oficial de paquetes PyPi, que solucionan o simplifican complejas tareas y las cuales pueden instalarse fácilmente con un simple comando, lo que demuestra la gran comunidad de desarrolladores de este lenguaje. De todas ellas, las consideradas como imprescindibles para el análisis de datos son fundamentalmente cuatro: *numpy* y *scipy*, para los cálculos matemáticos y estadísticos, la librería *pandas* para el análisis de datos que replica las funcionalidades de R, y la librería *matplotlib* que provee de una interfaz similar a Matlab y R para la generación de cualquier tipo de gráficos o histogramas. Adicionalmente, existen distribuciones como *Anaconda* que incluyen en su instalación las principales librerías necesarias para desenvolverse en el campo del análisis de datos, así como entornos de escritorio que facilitan el desarrollo de programas y visualización de los resultados (al estilo de *RStudio* en R), destacando entre ellos *Rodeo*. A continuación, se detalla una selección de las principales librerías actualmente disponibles dentro del campo de análisis de datos, cálculos actuariales y modelización de riesgos, las cuales pueden instalarse desde el repositorio oficial de paquetes de python (pypi.python.org):

NOMBRE	DESCRIPCIÓN	SITIO WEB
scipy	SciPy (Scientific Python) es una biblioteca de herramientas y algoritmos matemáticos para Python	https://www.scipy.org
numpy	NumPy (Numeric Python) es una extensión de Python, que le agrega mayor soporte para el uso de vectores y matrices.	http://www.numpy.org
pandas	Pandas (Python Data Analysis) es una librería para estructuras de datos, fácil de usar y con herramientas de análisis de datos. Existen librerías complementarias (como PrettyPandas) que ayuda a crear tablas visualmente más atractivas.	http://pandas.pydata.org
matplotlib	Matplotlib es una librería muy completa para la generación de gráficos e histogramas a partir de datos contenidos en listas o arrays.	http://matplotlib.org
statsmodels	Librería de Python que proporciona clases y funciones para la estimación de los diferentes modelos estadísticos, así como la realización de test y estudios de datos estadísticos (Regresión lineal, GLM, etc.)	http://www.statsmodels.org/stable/
rpy2	RPy2 permite llamar a implementaciones escritas en R dentro de un proceso de Python. Existen librerías similares para otros lenguajes como C, C++ o Fortran. R, por su parte, también posee un paquete para ejecutar código escrito en python (rPython).	https://rpy2.readthedocs.io
ipython	Este paquete permite la funcionalidad de mostrar y ejecutar código en un entorno web local como si de un bloc de notas o ventana de escritorio se tratará. IPython forma parte de Jupyter, el cual engloba otros lenguajes como R, Julia y Scala. Especialmente útil para entornos académicos o para pruebas sin necesidad de instalación: https://try.jupyter.org.	https://ipython.org
openpyxl	Librería que permite la lectura y escritura de documentos Excel 2010.	https://openpyxl.readthedocs.io
pyliferisk	Pyliferisk es una librería que facilita implementaciones y cálculos actuariales de vida.	https://github.com/franciscogarate/pyliferisk
Arcpy	Módulo de análisis geoespacial para ArcGIS (Sistemas de Información Geográfica), que facilita el manejo de información espacial y la generación de cartografía a través del lenguaje de programación Python.	https://www.arcgis.com
python-xbrl	Librería para generar documentos XBRL (metalenguaje similar a xml), estándar utilizado por reguladores y agencias gubernamentales para el reporte de la información financiera. Arelle (http://arelle.org/) es un plataforma de código abierto que provee de una API de python utilizada para las validaciones de información financiera, con soporte expreso para la DPM database y XBLR utilizada por EIOPA.	https://github.com/greedo/python-xbrl
dora	Una de las tareas más pesadas en el análisis de datos es la limpieza de los datos, ya que no siempre todos los datos son íntegros y libres de errores u omisiones. Esta librería contiene las principales funciones para una adecuada limpieza de los datos si fuera necesaria.	https://github.com/NathanEpstein/Dora
cx_Freeze	Python no requiere ser compilado para ejecutarse, aunque puede empaquetarse para ser distribuido o cerrar el código a futuras modificaciones. Esta librería permite realizar ejecutables compatibles con Windows.	https://github.com/anthony-tuininga/cx_Freeze
graph-tool	Librería para la aplicación de la teoría de grafos y análisis estadístico de redes, utilizada para cálculos de algoritmos topológicos o de lugar (cálculos de PageRank, arboles expandidos mínimos, medidas de centralidad, K-Cores, etc.). Una de las aplicaciones de la teoría de grafos es el análisis de las conexiones o nodos de las redes sociales para establecer patrones comunes en las relaciones humanas.	https://graph-tool.skewed.de
scikit-learn	Scikit-learn es la librería actualmente más destacada que cubre la parte del desarrollo de algoritmos y aprendizaje automático *(machine learning)*. La combinación con otras librerías, como graph-tool, podría utilizarse para la utilización de modelos predictivos basados en la segmentación de riesgos: hábitos de vida, educación, ingresos, estado civil, etc.	http://scikit-learn.org