Revista digital
Matemática, Educación e Internet
(http://tecdigital.tec.ac.cr/revistamatematica/).
Vol 15, No 2. Marzo – Agosto 2015.

Artículo de sección
ISSN 1659 -0643

Textos reproducibles y dinámicos con PythonTeX

Jesús H. Cuevas A.
humberto.cuevas@itchihuahuaii.edu.mx
Departamento de Ciencias Básicas
Instituto Tecnológico de Chihuahua II
México

Juan J. Rodriguez A.
jrodriguezarc@gmail.com
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Recibido: Agosto 12, 2014 Aceptado: Noviembre 19, 2014

Resumen. En este artículo se describe el proceso de instalación y utilización del paquete PythonTeX en la elaboración de un texto reproducible y dinámico. Se indican los requisitos tecnológicos y el protocolo de su operación a partir de ejemplos en que se realizan operaciones con matrices y se hacen cálculos elementales de estadística descriptiva.

Palabras clave: investigación reproducible, textos dinámicos, PythonTeX, Python, LaTeX

Abstract. This paper describes the process of installing and use of the PythonTeX package in the developing a reproducible and dynamic text. The technological requeriments and operation protocol are indicated in examples where operations with matrix and basic statistical descriptive are made.

KeyWords: reproducible research, dynamic text, PythonTeX, Python, LaTeX

1.1 Introducción

La capacidad de reproducir experimentos y protocolos de trabajo es una de las características fundamentales del método científico. De igual forma, es habitual que cuando se realiza un nuevo descubrimiento o se generan nuevas aplicaciones en algún campo del saber por parte de un investigador o grupo colegiado de investigación, sus hallazgos son evaluados por pares independientes que intentan replicar de forma controlada dichos experimentos y pruebas.

Los investigadores que fungen como pares siguen el método descrito por los autores originales e intentan comprobar si obtienen resultados conmensurables y similares a través de reproducir procedimientos de forma rigurosa. Según Gandrud [8], los resultados que se obtienen de una investigación son reproducibles en la medida que existe información suficiente para que investigadores independientes consigan los mismos resultados con los procedimientos descritos en el reporte. Asimismo, Laine [9] agrega que la reproducibilidad implica que los investigadores compartan sus bases de datos, el código computacional utilizado y el procedimiento seguido en sus estudios.

Actualmente, revistas científicas de reconocido prestigio han formulado lineamientos de publicación que exigen la posibilidad de reproducir los resultados obtenidos en los estudios. Las razones de esta exigencia son múltiples, entre las más representativas se encuentran:

- Examinar la pertinencia del protocolo metodológico y computacional.
- Analizar la potencia de las pruebas y asociaciones entre variables.
- Verificar la coherencia de los resultados obtenidos respecto de las conclusiones emitidas.
- Demarcar la cultura científica de la no científica.

Es importante señalar que la acepción del término reproducibilidad que se utiliza en este artículo se refiere a la capacidad de desarrollar, implantar, monitorear y evaluar experimentos y estudios –simples o complejos– exclusivamente desde la óptica del seguimiento riguroso del método científico.

También se consideró pertinente establecer de forma breve las diferencias entre *ciencia* y *pseudociencia* que se adoptaron en dicho artículo. La demarcación entre ciencia y pseudociencia esta influida por diferentes corrientes epistemológicas, y en consecuencia, es difícil establecer criterios universales para validar una disciplina científica de la que no lo es; aun así, hay atributos orientadores en una y en otra. Así, en la ciencia se plantean hipótesis susceptibles de ser probadas empíricamente, su falsabilidad es un atributo inseparable y los procedimientos, instrumentos y estructura de su difusión siguen una ruta normada por el método científico. Asimismo, la reproducibilidad de sus procedimientos es parte inherente de un estudio científico. Por otra parte, en la pseudociencia regularmente se observa una falta de consistencia interna y externa en su método de operación e interacción con otras disciplinas; se abusa de metáforas como mecanismo para explicar eventos desconocidos y sin posibilidad de someterlos a prueba; se recurre al uso de posturas dogmáticas en sus planteamientos, lo que impide aplicar un proceso de falsación de sus hipótesis y la consecuente replicación de sus protocolos de trabajo. En este artículo se adoptan los principios teóricos y los parámetros de diferenciación entre ciencia y pseudociencia planteados por el filósofo y teórico de la ciencia, Karl S. Popper [10], y en relación al termino cultura científica, se le concibe como la valoración y ponderación que hace de la ciencia un individuo tanto en su actuar profesional como cotidiano.

La reproducibilidad en el sentido que se definió líneas atrás, también tiene cabida en el ámbito de la educación. Así, pueden crearse documentos que planteen situaciones–problema en las que se generen de forma aleatoria los datos de trabajo dentro de parámetros controlados por el profesor, introduciendo el código necesario para que el programa resuelva cada una de dichas situaciones–problema. Un profesor también puede generar un examen distinto –solución incluida– para cada estudiante, con solo

compilar el documento. De igual manera, pueden elaborarse manuales, libros, entre otros documentos dinámicos.

El componente computacional es fundamental en la investigación reproducible. Actualmente existen diversas herramientas de cómputo que se usan para tal fin, entre las más representativas se encuentran los lenguajes de programación R y Python; el primero de uso común en el campo de la estadística, y el segundo de amplia utilización como lenguaje de propósito general y científico. De igual forma el sistema de composición de textos LaTeX y los lenguajes de marcado Markdown y HTML son útiles cuando se requiere generar reportes dinámicos formales y publicarlos en formato digital.

El paquete Sweave desarrollado por Leisch [11] [12] [13] permite mezclar código de R y LaTeX en un mismo documento con el propósito de crear informes dinámicos que se actualizan en función del cambio en los datos y la realización del proceso de compilación. De esta manera, se combina la potencia en el análisis de datos y construcción de gráficos en R, con la versatilidad de LaTeX para generar documentos estructurados. En 2012, Xie [15] [16] [17] [14] [18] [19] desarrolló Knitr con el mismo propósito que Sweave, extendiendo las capacidades de este último y permitiendo la integración del código de R con LaTeX, Markdown, LyX, entre otros; además de reducir la curva de aprendizaje para usuarios nóveles.

En 2013, Poore [20] publicó el paquete PythonTeX que permite introducir y ejecutar código de Python en un documento LaTeX. El paquete permite aprovechar el potencial de Python como lenguaje de propósito general, así como las capacidades de librerías como Scipy, Numpy y Matplotlib. El paquete provee soporte para lenguajes de propósito general como Ruby, y de cómputo científico como Julia.

En virtud de la gran capacidad y utilidad de Python y LaTeX en el ámbito científico y educativo, a la cantidad de paquetes disponibles y la extensa comunidad de programadores que impulsan su desarrollo en distintos ámbitos, este artículo tiene el objetivo de describir el proceso de instalación y uso del paquete PythonTeX para que se permita elaborar textos reproducibles y dinámicos. Lo anterior es importante si se considera el impulso que se está dando a la generación de reportes de investigación que puedan replicarse, contribuyendo así a promover una cultura de transparencia y rigurosidad académico–científica.

1.2 Lenguaje Python y el paquete PythonTeX

El paquete PythonTeX es de reciente creación y continúa en fase de desarrollo. Para su funcionamiento requiere una distribución de LaTeX como Texlive o MikTeX. Según el manual de usuario [22], es necesario contar con los paquetes –y sus dependencias– *fancyvrb*, *etex*, *etoolbox*, *xstring*, *pgfopts*, *newfloat*, *currfile*, y *color* o *xcolor*. PythonTeX se ha probado en distintas distribuciones de LaTeX, sin embargo las pruebas más recientes se han efectuado principalmente con Texlive, por lo que se recomienda instalar la última versión disponible de esta distribución.

Aún cuando las versiones subsecuentes de Python 3 están recibiendo mucho apoyo de la comunidad internacional, se recomienda utilizar la versión 2.7 para garantizar la compatibilidad completa de PythonTeX. Para resaltar la sintaxis y presentar el código de forma elegante y clara, es necesario utilizar el

paquete pygments.

En las siguientes subsecciones se describe el protocolo de instalación y configuración de las herramientas necesarias, así como el proceso de compilación–interpretación con LaTeX y Python respectivamente.

1.2.1 Instalación

Antes de iniciar proceso de instalación, se deben descargar en el disco duro los programas y paquetes de cómputo que se enuncian a continuación:

- Distribución TexLive (de preferencia la más reciente). La distribución MikTeX también puede usarse, sin embargo es más propensa a errores.

- Editor de código especializado (v.gr Texstudio, Texmaker, Texworks, TeXShop, Kile, o cualquier otro compatible)

- Python 2.7. La versión 3.4.5 también puede utilizarse, sin embargo las pruebas más exhaustivas se han realizado con la versión 2.7.

- Biblioteca de herramientas y algoritmos matemáticos para Python «SciPy Stack» (SciPy, Numpy, Matplotlib)

- Paquetes de Python: ez_ setup, setuptools, PythonTeX y pygments

El proceso de instalación debe iniciar verificando que la configuración y operación de las instalaciones de Texlive y Python sea óptima, es decir, probar la edición de un documento típico de LaTeX y su compilación en un editor como los descritos anteriormente, así como la elaboración y posterior ejecución de un script de Python en el editor *Idle* que viene incluido en la instalación. Posteriormente debe instalarse la biblioteca de herramientas y algoritmos matemáticos para Python, especialmente interesan *SciPy*, *NumPy* y *Matplotlib*. La primera (*SciPy*) contiene módulos para optimización, álgebra lineal, integración, interpolación, procesamiento de señales e imagen,algunas funciones especiales, entre otras. La segunda (*NumPy*) integra funciones matemáticas especializadas en el tratamiento de vectores y matrices. La tercera (*Matplotlib*) potencia la generación de gráficos en Python a través de la interacción con NumPy. Enseguida es necesario instalar los paquetes ez_setup y setuptools. Con easy_ install se instalará pygments y creará una carpeta en el subdirectorio de Python.

Posteriormente, es necesario descomprimir PythonTeX y junto a pygments copiarlos en la carpeta que se utilizará para almacenar los archivos del trabajo. Este paso en el procedimiento es importante, ya que para funcionar, ambos paquetes deberán permanecer en la misma carpeta en la que se guardará el archivo tex. En el preámbulo de dicho archivo se deberá cargar el paquete PythonTeX. En la figura 1.1 se describe un protocolo de instalación y configuración de los paquetes.

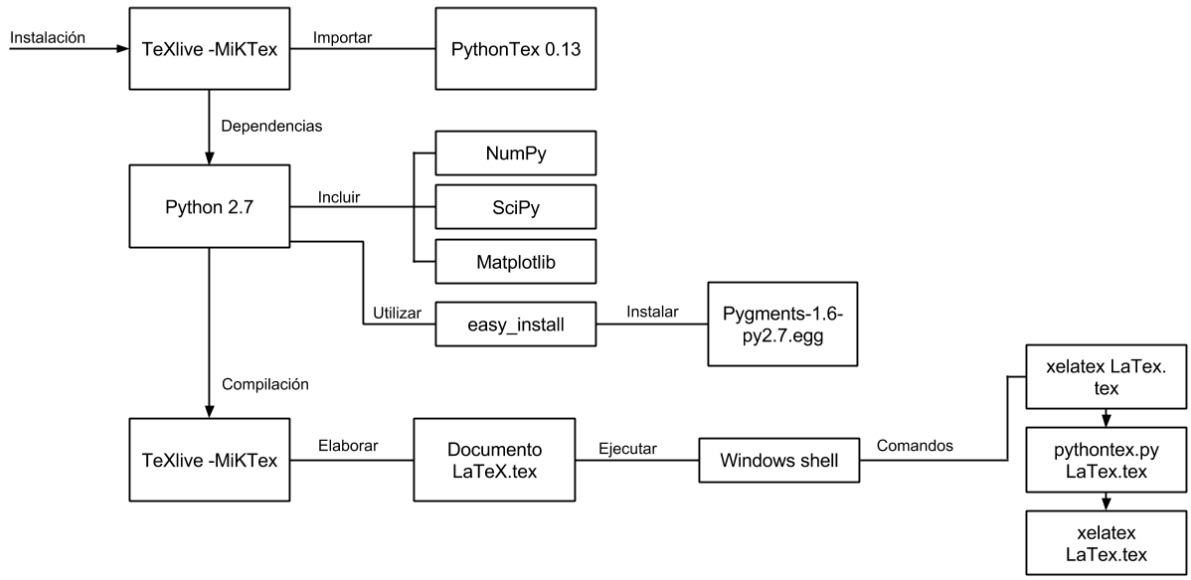

Figura 1.1: Proceso de instalación y ejecución del paquete PythonTeX

1.2.2 Protocolo de compilación–interpretación

Python es un lenguaje de programación que requiere de un interprete para poder ejecutarse. LaTeX por su parte es un lenguaje de marcado que necesita un compilador para traducir código fuente a máquina. El paquete PythonTeX permite la interacción entre ambos lenguajes y aprovechar su potencial.

Para generar textos reproducibles y dinámicos con la interacción de LaTeX y Python se recomienda seguir el siguiente protocolo:

1. Redactar un texto que incorpore LaTeX y Python en un mismo archivo.

2. Compilar el archivo en un editor. Este proceso extraerá el código de Python del texto y lo almacenará en otro archivo.

3. Iniciar la línea de comandos e ir al directorio de la carpeta en que se encuentran los paquetes PythonTeX, pygments y los archivos de trabajo.

4. Activar el paquete PythonTeX seguido del nombre del archivo .tex (pythontex archivo.tex) y observar los resultados que genera el proceso, en la pantalla se mostrará la cantidad de errores encontrados. Si no se presentan errores, se genera un archivo interpretado por Python en conjunto con los demás paquetes externos.

5. Regresar al editor y volver a compilar el archivo original. El texto final incorporará los resultados generados previamente en Python y se generará un archivo con extensión pdf.

Es imperativo seguir el protocolo descrito antes para obtener los resultados deseados. En la figura 1.2 se muestra un esquema del protocolo descrito anteriormente.

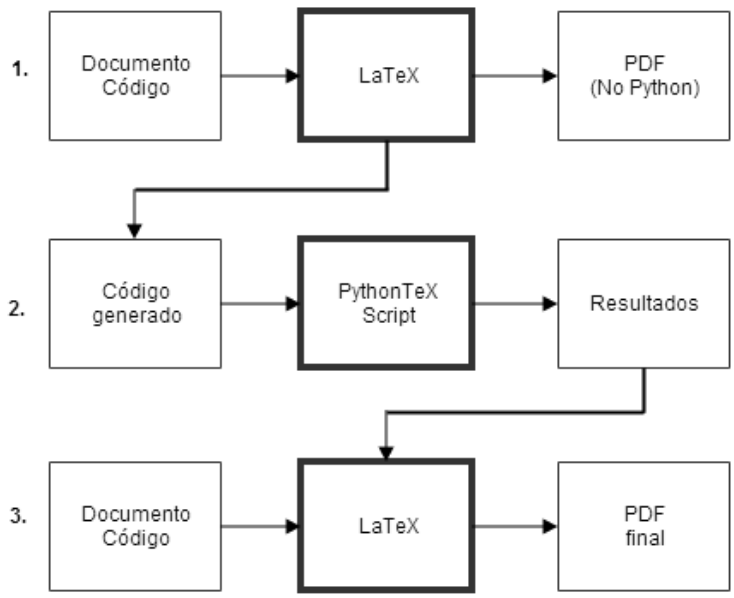

Figura 1.2: Proceso de compilación–interpretación

1.3 Ejemplo de su aplicación en tópicos matemáticos y estadísticos

A continuación se presenta un ejemplo con código de LaTeX y Python interactuando en un mismo archivo. El objetivo fue generar de forma aleatoria una matriz de orden 4 X 4 haciendo uso de la biblioteca de herramientas y algoritmos matemáticos numpy, además de resolverla y mostrar su resultado en la pantalla. El código básico para realizar lo anterior se muestra en las siguientes líneas:

```
1.  \documentclass[10pt,letterpaper]{article}
2.  \usepackage[utf8]{inputenc}
3.  \usepackage{pythontex}

4.  \begin{document}

5.      \begin{pycode}
6.         import numpy as np
7.         M = np.random.randint(-20,20, size=(4,4))
8.         R = np.linalg.det(M)
9.      \end{pycode}

10.     \pyc{print(R)}
11. \end{document}
```

Las primeras tres líneas de código constituyen el preámbulo de LaTeX. En la línea inicial se establece que es un documento tipo artículo con fuente de 10 puntos a ser impreso en hoja tamaño carta. La segunda, carga el paquete inputenc y establece la codificación *utf8* para que el archivo pueda ser transferido y leído en diversos sistemas operativos –UNIX, Linux, Windows, OSX– sin necesidad de efectuar ajustes especiales al texto. A partir de la cuarta línea es posible insertar texto, o en su defecto acompañarlo de código escrito en Python como en este ejemplo, abriendo un entorno denominado pycode para lograrlo. Puede notarse que en la primera línea se importa la biblioteca *numpy* y se asocia a la variable *np* para respetar la lógica de la programación orientada a objetos. Posteriormente se utiliza la función *random.randint* para crear de forma aleatoria la matriz de orden 4 X 4 con valores enteros del intervalo (-20 y 20) y la matriz se almacena en M. En la linea siguiente se usa la función *linalg.det* para calcular el determinante de M y el resultado se almacena en la variable R. Dicha variable es mostrada en la salida estándar de documento, y en este caso será visualizado en el pdf únicamente un número entero con el resultado de dicho determinante. En este punto se cierra el entorno *pycode*, la impresión del resultado debe quedar fuera del bloque de código y se anota en la siguiente linea con el auxilio de *pyc* como entorno de ejecución. La línea final es para culminar el código de LaTeX.

Es importante señalar que PythonTeX permite crear y llamar funciones auxiliares en medio del código LaTeX. A continuación se muestra un ejemplo que permite imprimir la multiplicación de dos matrices y su respectivo resultado:

```
#    Creación de funciones auxiliares:
1.  \begin{pycode}
2.     import numpy as np

3.     from sympy import *
4.     #  Función encargada de imprimir una matriz
5.     def printMatrix(M):
6.          print(r' $\displaystyle \begin{pmatrix}' )
7.          for i in range(len(M)):
8.             for j in range(len(M[0])):
9.                print(latex((M[i][j])))
10.               if (j != len(M)-1):
11.                  print ('&')
12.                  print (r'\\')
13.                  print(r'\end{pmatrix}$')

14.            #  Función encargada de multiplicar dos matrices
15.     def multMatrix(A,B):
16.        printMatrix(A.dot(B))
17.   \end{pycode}
```

El protocolo para crear funciones es idéntico al utilizado en cualquier interprete de Python, no obstante, para que dicho protocolo funcione de forma optima y dichas funciones puedan llamarse en cualquier sección de código LaTeX, es necesario que estas se ubiquen dentro del entorno *pycode*. También se debe tener presente importar las librerías especializadas –numpy y scipy– para usar funciones relacionadas

con operaciones algebraicas. Así, la función *printMatrix* permitirá imprimir en formato LaTeX cualquier matriz de dimensiones iguales o mayores de 1 x 1, evitando cometer errores de redundancia de código. Asimismo, la función *multMatrix* permite multiplicar dos matrices e imprimirlas directamente en el documento.

```
#    Multiplicación de 2 matrices:
1.   $A \cdot B =$ \pyc{printMatrix(mul_C)} $\cdot $
2.   \pyc{printMatrix(mul_D)} $ = $
3.   \pyc{multMatrix(mul_C,mul_D)}
```

A continuación se muestra un ejemplo que genera la salida de este programa:

$$A \cdot B = \begin{pmatrix} -18 & -7 \\ 2 & 14 \\ -20 & -8 \\ -16 & -1 \end{pmatrix} \cdot \begin{pmatrix} 8 & -716 & -6 \\ -10 & 1-1 & 9 \end{pmatrix} = \begin{pmatrix} -74 & 119 & -281 & 45 \\ -124 & 0 & 18 & 114 \\ -80 & 132 & -312 & 48 \\ -118 & 111 & -255 & 87 \end{pmatrix}$$

Figura 1.3: Multiplicación de 2 matrices aleatorias

PythonTeX también permite acceder a una gran cantidad de funciones que son útiles en otras disciplinas científicas, como el cálculo integral, diferencial y vectorial; matemática discreta; optimización de funciones, y estadística. En el caso de ésta última, destaca la librería especializada en construcción de gráficos en dos dimensiones *Matplotlib*. En el siguiente código, se muestran las instrucciones para calcular un modelo de regresión lineal básico, elaborar un histograma e imprimirlo dentro de un archivo LaTeX:

```
#  Calcular un modelo de regresión lineal
1.   \begin{pylabcode}
2.      from scipy import stats
3.      import numpy as np
4.      import pylab
5.
6.     #  Montar el modelo
7.     x = np.array([1, 2, 5, 7, 10, 15])
8.     y = np.array([2, 6, 7, 9, 14, 19])
9.     slope, intercept, r_value, p_value, slope_std_error = stats.linregress(x, y)
10.
11.    #  Calcule algunas salidas adicionales
12.    predict_y = intercept + slope * x
13.    pred_error = y - predict_y
14.    degrees_of_freedom = len(x) - 2
15.    residual_std_error = np.sqrt(np.sum(pred_error**2) / degrees_of_freedom)
16.
17.    #  Graficación
```

```
18.    pylab.plot(x, y, 'o')
19.    pylab.plot(x, predict_y, 'k-')
20.    pylab.show()
21.    outputfile = 'linear.png'
22.    savefig(outputfile)
23.    clf()
24.    \end{pylabcode}
25.
26.        % Impresión en LaTeX
27.    \begin{center}

28.        \includegraphics[width=0.5\linewidth]{./linear}

29.    \end{center}
```

Para elaborar gráficas de funciones en Python es necesario contar con librerías auxiliares. En el código anterior, se recurre al entorno *pylabcode*, donde se puede observar que en las primeras cuatro líneas se hace una importación de los paquetes que van a ser utilizados; posteriormente, se elabora un modelo de datos y se parametrizan valores tales como pendiente, intercepciones y los datos a utilizar para luego plasmarlo en el modelo de regresión lineal.

Enseguida, se establecen las etiquetas de los ejes y escala correspondientes al gráfico. El resultado final se imprime de la manera estándar de LaTeX –`\includegraphics`–. En la siguiente figura, se muestra el resultado final:

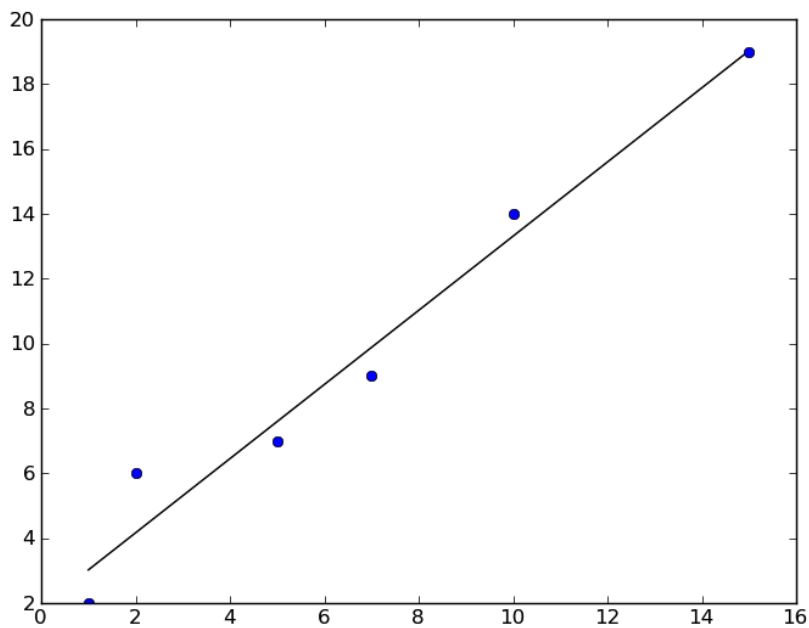

Figura 1.4: Regresión lineal básica

1.4 Conclusiones

Hoy en día la elaboración de textos reproducibles en el ámbito de la investigación científica es una realidad. En el caso de la enseñanza, además de la reproducibilidad, la adición del componente dinámico permitirá que el profesorado genere material de estudio variado y con mayor utilidad para sus estudiantes porque serán textos vivos, dinámicos, autorreproducibles y con una posibilidad mayor de mantener su vigencia. Se espera que en la comunidad de profesores y estudiantes, dicha tecnología contribuirá a demarcar con precisión la cultura científica de la que no lo es.

La reproducibilidad también tiene efecto en los protocolos de trabajo de los investigadores en ciencia y los editores y evaluadores de revistas especializadas en divulgación científica. En ambos introduce nuevos lineamientos y exigencias para someter y evaluar ante la crítica experta; la apertura, transparencia y suministro de las bases de datos y métodos de experimentación forman parte de dichas exigencias.

En resumen, en el ámbito de la investigación científica, se espera que este tipo de tecnología contribuya a mejorar la rigurosidad en los procesos de revisión y dictamen de reportes.

En este artículo se describió el proceso de instalación y el protocolo de operación del paquete Pythontex para elaborar textos reproducibles y dinámicos. Se presentaron ejemplos de código para realizar cálculos en álgebra lineal y estadística, así como una representación gráfica elemental.

Es importante subrayar que esta tecnología es reciente, comenzó a probarse en 2013 y se encuentra en desarrollo. Así, el protocolo de compilación–interpretación no es automático debido a la ausencia de un programa compilador eficiente, pero con la gran comunidad epistémica y de programadores en LaTeX y Python existente, es factible que en menos de un lustro el protocolo se automatice.

Ciertamente, existen otras alternativas para elaborar textos reproducibles y dinámicos. Algunas buenas aproximaciones son GNU TeXmacs, WxMaxima, Scientific Workplace, entre otros, que permiten redactar informes e introducir cálculos matemáticos en el mismo documento. No obstante, su poder de cómputo y las posibilidades de manipulación de código y prestaciones se quedan muy lejos de las capacidades de usar lenguajes de programación especializados como Fortran, C, C++, R y Python, en conjunción con lenguajes de marcado como LaTeX, Markdown, o HTML5. Así, el uso del paquete `knitr` como medio de interacción y RStudio como entorno de desarrollo integrado (IDE) han posibilitado lo anterior.

También es posible integrar R con C++ o Fortran y aprovechar su potencial en actividades que demanden cómputo científico sofisticado. Sin embargo, la dupla LaTeX–Python ofrece la posibilidad de aprovechar la potencia de un lenguaje de propósito general y científico muy difundido como el caso de Python, especialmente en textos que demandan el trabajo con cálculo simbólico usando como intermediario un Sistema de Álgebra Computacional (Computer Algebra System, CAS por sus siglas en inglés).

Agradecimientos. Los autores agradecen al Programa de Pasantías 2013 para la Movilidad Estudiantil con Fondos del Sistema (CONARE) y coordinado por la Rectoría del Instituto Tecnológico de Costa Rica, por el apoyo en la elaboración de esta publicación y la experiencia académica y cultural promovida.

Mención especial merecen el Director de la Escuela de Matemática, Dr. Luis Gerardo Meza Cascante y el Coordinador de la Carrera de Enseñanza de la Matemática Asistida Por computadora , M.C. Paulo García Delgado por todo el apoyo otorgado en la gestión académica.

Al Instituto Tecnológico de Chihuahua II en México, se le está agradecido por el apoyo para realizar una pasantía en las instalaciones del Departamento de Ciencias Básicas.

Se agradece también al M. C. Alexander Borbón Alpizar, profesor de escuela de matemática por su trabajo como asesor interno.

Bibliografía

[1] Allen, Jeff. "The Next Generation of R Markdown". http://user2014.stat.ucla.edu/abstracts/talks/902_Allen.pdf

[2] Boettiger, Carl "An introduction to knitcitations". http://cran.rproject.org/web/packages/knitcitations/vignettes/tutorial.pdf

[3] Chirigati, Fernando and Shasha, Dennis and Freire, Juliana. "Packing experiments for sharing and publication". *"Proceedings of the 2013 international conference on Management of data"*. ACM. 2013. pp. 977–980

[4] De Leeuw, Jan. "Reproducible research. the bottom line". Department of Statistics, UCLA

[5] Donoho, David L. "An invitation to reproducible computational research". Biostatistics. Vol 11, N 3, pp. 385–388 http://biostatistics.oxfordjournals.org/content/11/3/385.short

[6] Donoho, David L and Maleki, Arian and Rahman, Inam Ur and Shahram, Morteza and Stodden. "Reproducible research in computational harmonic analysis". Computing in Science & Engineering. Vol 11, N 1, pp. 8–18 http://scitation.aip.org/content/aip/journal/cise/11/1/10.1109/MCSE.2009.15

[7] Fomel, Sergey and Claerbout, Jon F. "Reproducible research". Computing in Science & Engineering". Vol 11, N 1, pp. 5–7 http://scitation.aip.org/content/aip/journal/cise/11/1/10.1109/MCSE.2009.14

[8] Gandrud, Christopher. *"Reproducible research with R and RStudio"*. CRC Press. 2013

[9] Laine, Christine and Goodman, Steven N and Griswold, Michael E and Sox, Harold C. "Reproducible research: moving toward research the public can really trust". Annals of Internal Medicine. Vol 146, N 6, 2007. http://annals.org/article.aspx?articleid=733696

[10] Popper, Karl *"Conjeturas y refutaciones"*. Paidós. 1996.

[11] Leisch, Friedrich. "Sweave: Dynamic generation of statistical reports using literate data analysis". Springer. 2002. http://link.springer.com/chapter/10.1007/978-3-642-57489-4_89

[12] Leisch, Friedrich. "Sweave, part I: Mixing R and LaTeX". R News. Vol 2, N 3. 2002.

[13] Leisch, Friedrich. "Sweave, part II: Package vignettes". R News. Vol 3, N 2. 2003.

[14] Xie, Yihui. "knitr A General Purpose Tool for Dynamic Report Generation in R". 2013. https://bitbucket.org/stat/knitr/downloads/knitr-manual.pdf

[15] Xie, Yihui". *"Package knitr"*. 2013. http://cran.r-project.org/web/packages/knitr/knitr.pdf

[16] Xie, Yihui. "knitr a comprehensive tool for reproducible research in R". Implementing Reproducible Research. CRC Press. http://books.google.com/books?hl=en&lr=&id=JcmSAwAAQBAJ&oi=fnd&pg=PA3&dq=knitr&ots=ylVdrPuMMK&sig=4BkhQuBGKSVfqI6ZWYQy8SMxWsI

[17] Xie, Yihui. "Dynamic Documents with R and knitr". CRC Press.2013

[18] Xie, Yihui "formatR Format R Code Automatically". R package version 0.4". Vol 3. 2012.

[19] Xie, Yihui. "An Introduction to knitr". Citeseer. 2012. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.259.8718

[20] Poore, Geoffrey M. "Reproducible Documents with PythonTeX". *"Proceedings of the 12th Python in Science Conference.* Ed. by Stéfan van der Walt 2013.

[21] Robinzonov, Nikolay. "A Gentle Introduction to LATEX". http://www.statistik.lmu.de/~poessnecker/Lehre/WiSe201415/Latexkurs/latex1415-beamer.pdf

[22] Poore, Geoffrey. "The pythontex package". github.com/gpoore/pythontex. 2013.