introduccion.nbconvert

January 20, 2017

INVESTIGACIÓN REPRODUCIBLE
Mario Mañana Canteli
Universidad de Cantabria

In [1]:
```
import time

        now = time.strftime("%c")
        print "a Ultima version: " +(time.strftime("%d/%m/%Y")) + " " + (time.strftime("%H:%M:%S"))
```

```
a Ultima version: 20/01/2017 21:45:38
```

0.1 Introducción

El concepto de investigación reproducible (reproducible research) no es algo nuevo. Desde el desarrollo de la escritura, las personas involucradas en la transmisión del conocimiento han estado preocupadas por la forma en la que éste se transmitía. Debe entenderse aquí que estamos pensando en que había un interés real en transmitirlo. No se consideran, por tanto, situaciones en las que se buscaba de forma explícita codificar el conocimiento para limitar las personas que podían tener acceso a él. En este contexto, los libros y documentos funcionan como fotos fijas con capacidades limitadas para aportar información que no pueda ser expresada de forma descriptiva. Esta vía de comunicación resulta adecuada en ciertas áreas de conocimiento, pero no lo es tanto en otras. Con el desarrollo de la ciencia de la computación, la información que se podía transmitir comenzó a ser más compleja. A las descripciones de procedimientos se comenzaron a añadir otros elementos como diagramas, fotografías y tablas con datos. El soporte en papel no resulta adecuado con el volumen de datos es elevado. De igual forma, cuando los documentos requíeren aportar código fuente, resulta difícil incluirlo si el número de líneas de código es elevado. Se produce entonces un esfuerzo de síntesis, que termina por sintetizar la información a transmitir. En algunos casos, este esfuerzo de síntesis es útil, ya que resume de forma concisa trabajos que tienen, por su extensión y complejidad, un encaje difícil en un documento de extensión limitada. Comienza aquí el problema. ¿Qué pasa cuando el escritor, o conjunto de escritores, tienen un interés explícito en transmitir de forma completa toda la información disponible? Supóngase que un ensayo o experimento científico ha generado miles o millones de datos. En este caso, la solución pasa por hacer disponibles dichos datos en algún tipo de repositorio o incorporarlos al documento en un soporte digital (CD, DVD o similar) Por complicar el problema un poco más, supóngase que los datos son procesados con un programa o conjunto de programas que implementan un algoritmo. La persona interesada en replicar los resultados conseguidos, debe instalar dichos programas en su ordenador y ejecutar los códigos sobre el conjunto de datos. Comienza aquí el calvario... En algunos casos, las versiones disponibles de las herramientas software que deben compilar (por ejemplo C, fortran, etc.) o interpretar (octave, python, etc.) el código habrán cambiado, generando errores de compatibilidad con los códigos disponibles. En otras ocasiones, especialmente cuando se requiere utilizar diferentes herramientas de forma simultánea, aparecerán incompatibilidades entre versiones, entre versiones de las herramientas y del sistema operativo sobre el que se ejecutan o requisitos relativos a librerías que faltan o que interfieren con otras. Se requiere, por tanto, una aproximación holística al problema que permita abordarlo de forma integral. Adicionalmente, los lenguajes de programación permiten añadir comentarios al código, pero en la mayor parte de los casos, utilizando caracteres ASCII estándar. Esto signficia que no es posible utilizar texto enriquecido, lenguaje matemático y/o tablas o diagramas. El primer problema se resuelve con herramientas

de virtualización y/o contenedores. Este tipo de soluciones permite aportar al destinatario no solo el código fuente, sino también las herramientas de compilación o interpretación perfectamente sintonizadas. Ésto resulta especialmente útil cuando se interacciona dentro de un equipo de programadores. La segunda cuestión se resuelve combinando un documento de texto enriquecido (Word, Open Office, LateX) con el código. El problema con esta solución es que código y documentación se tratan por separado, complicando la lectura y facilitando la generación de errores.

La investigación reproducible versa precisamente sobre estos problemas. ¿Cómo transferir el conocimiento de una forma más integral? ¿Cómo combinar, en un único repositorio, todos los elementos necesarios e incluyendo incluso las herramientas?

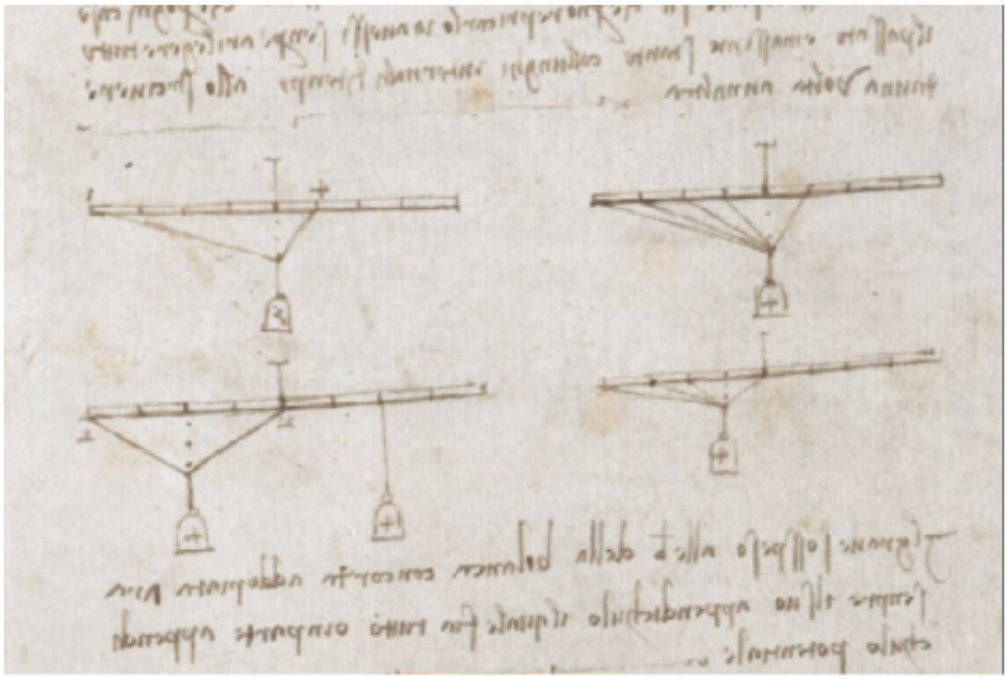
Comenzaremos por analizar el método clásico para ir resolviendo poco a poco los desafíos planteados...

0.2 El método clásico

Leonardo da Vinci escribió múltiples libros de notas en disciplinas muy distintas que van desde la medicina hasta la ingeniería, pasando por la biología. Uno de esos libros de notas es ('The Codex Arundel'), escrito entre 1478 y 1518.

```
In [2]: from IPython.display import Image
        Image(filename='./figuras/leonardo_01.png')
```

Out[2]:



La estructura de los libros de notas de Leonardo son similares a los que los científicos han venido utilizando en el último medio siglo.

Una combinación de texto con explicaciones, hipótesis, metodologías, así como resultados de experimentos que pueden incluir diagramas/fotos y resultados. La principal limitación de esta metodología reside en la dificultad para trabajar con grandes volúmenes de datos. No resulta sencillo compartir datos y procedimientos de cálculo utilizando un libro de notas clásico.

0.2.1 Historia de una crisis anunciada

"Growth in a Time of Debt" es un conocido artículo científico en los círculos financieros, también referenciado por los apellidos de sus autores "Reinhart–Rogoff". Este artículo fue publicado (sin revisión por pares) en 2010 por los economistas norteamericanos Carmen Reinhart (Universidad de Maryland) y Kenneth Rogoff (Universidad de Harvard) en la revista American Economic Review. A raiz de la publicación de dicho artículo, muchos políticos y comentaristas económicos comenzaron a citar dicho artículo en sus debates sobre la efectividad de la austeridad en políticas fiscales en paises con grandes niveles de deuda. De hecho, el artículo incide en el hecho de que cuando la deuda externa de un pais supera el 60% del PIB, dicho pais sufre graves problemas en su crecimiento. En el artículo se indica también que cuando la deuda supera el 90%, el crecimiento del PIB se reduce a la mitad. La importancia de este artículo en el contexto económico fue que se publicó en plena crisis financiera y daba soporte a las políticas de austeridad.

El propio Olli Rehn, comisario de asuntos económicos de la Unión Europea, utilizó este artículo como referencia para la intervención de paises con elevados niveles de deuda.

En el año 2013 algunos académicos mostraron que la metodología presentaba algunos errores que no justificaban sus hipótesis de partida. El primer análisis fue realizado por un estudiante de la Universidad de Massachusetts utilizando iPython Notebook. Lo interesante de la nueva metodología es la capacidad para revisar datos, procedimientos y conclusiones sobre un único documento.

0.2.2 Motivaciones: El principio de Claerbout

"An article about computational result is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result." - Claerbout and Karrenbach, Proceedings of the 62nd Annual International Meeting of the Society of Exploration Geophysics. 1992.

"When we publish articles containing figures which were generated by computer, we also publish the complete software environment which generates the figures" - Buckheit & Donoho, Wavelab and Reproducible Research, 1995.

...

0.3 Un nuevo paradigma

Imaginemos por un momento que pudiésemos plantear una alternativa a los libros de notas clásicos.

Imaginemos que cuando escribimos un artículo científico, un guión de una práctica o un informe técnico para una empresa o un colega, no tuviésemos que separar texto, datos y algoritmos de cálculo. Imaginemos también que tuviésemos una herramienta para compartir todo el material de una forma sencilla y cooperativa.

Beneficios

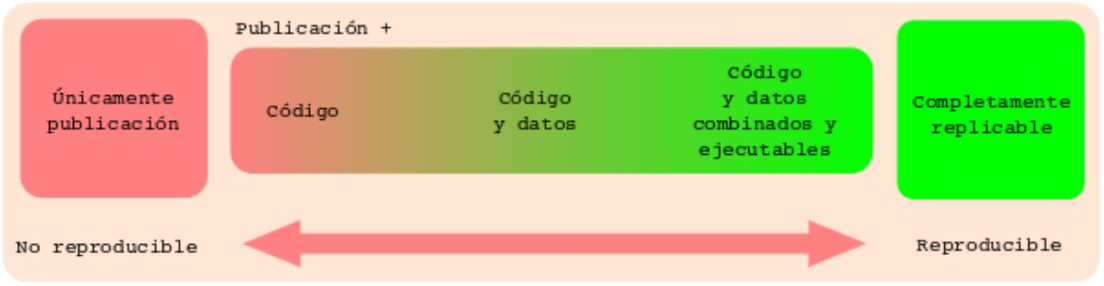
- Verificación y fiabilidad. Facilidad para encontrar errores. Los resultados que se obtienen hoy deben ser los mismos que los que se producirían mañana.
- Transparencia. La disponibilidad pública de datos y procedimientos permite mejorar los índices de citas de autores y sus instituciones.
- Eficiencia. Facilidad para reutilizar datos y procedimientos.
- Flexibilidad. Capacidad para hacer cambios sobre los procedimientos descritos.

Desde un punto de vista general, un trabajo se considera investigación reproducible si verifica las condiciones siguientes (Stodden, V., et al. 2013. "Setting the default to reproducible." computational science research. SIAM News 46: 4-6):

- Revisable. Dispone de una descripción general del mismo.
- Replicable. Existen herramientas (públicas y/o privativas) que pueden ser utilizadas para obtener el resultado final.
- Confirmable. Diferentes personas pueden obtener los mismos resultados de forma independiente.
- Auditable. Tanto los datos como el código son accesibles (el acceso a los datos puede ser público o privado.)
- Reproducible. Datos y código existen en el dominio público.

In [3]: Image(filename='./figuras/rr_grado.png')

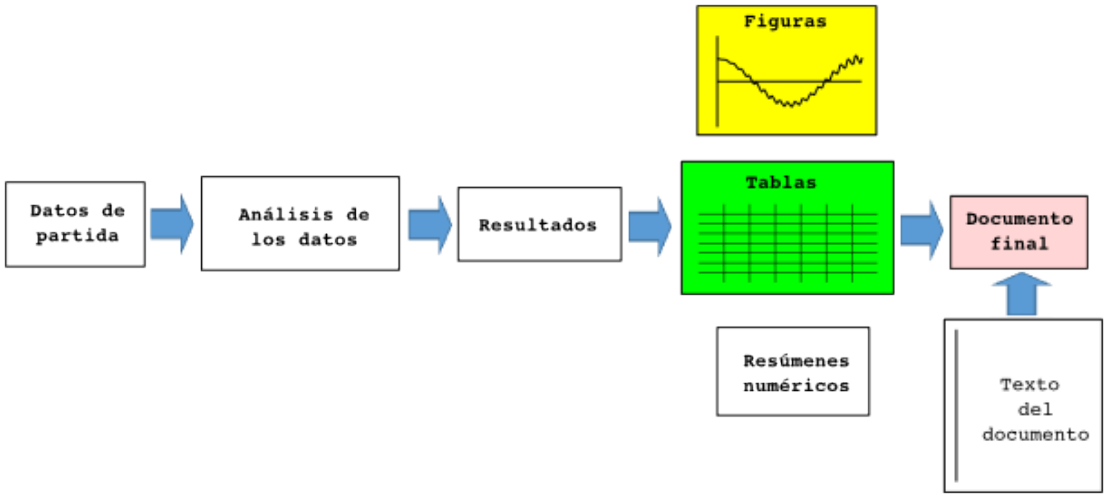
Out[3]:



La figura siguiente resume la metodología de generación de documentos, tanto basadas en un paradigma no reproducible como reproducible.

In [4]: Image(filename='./figuras/metodologia_nr.png')

Out[4]:



In [5]:

git.nbconvert

January 20, 2017

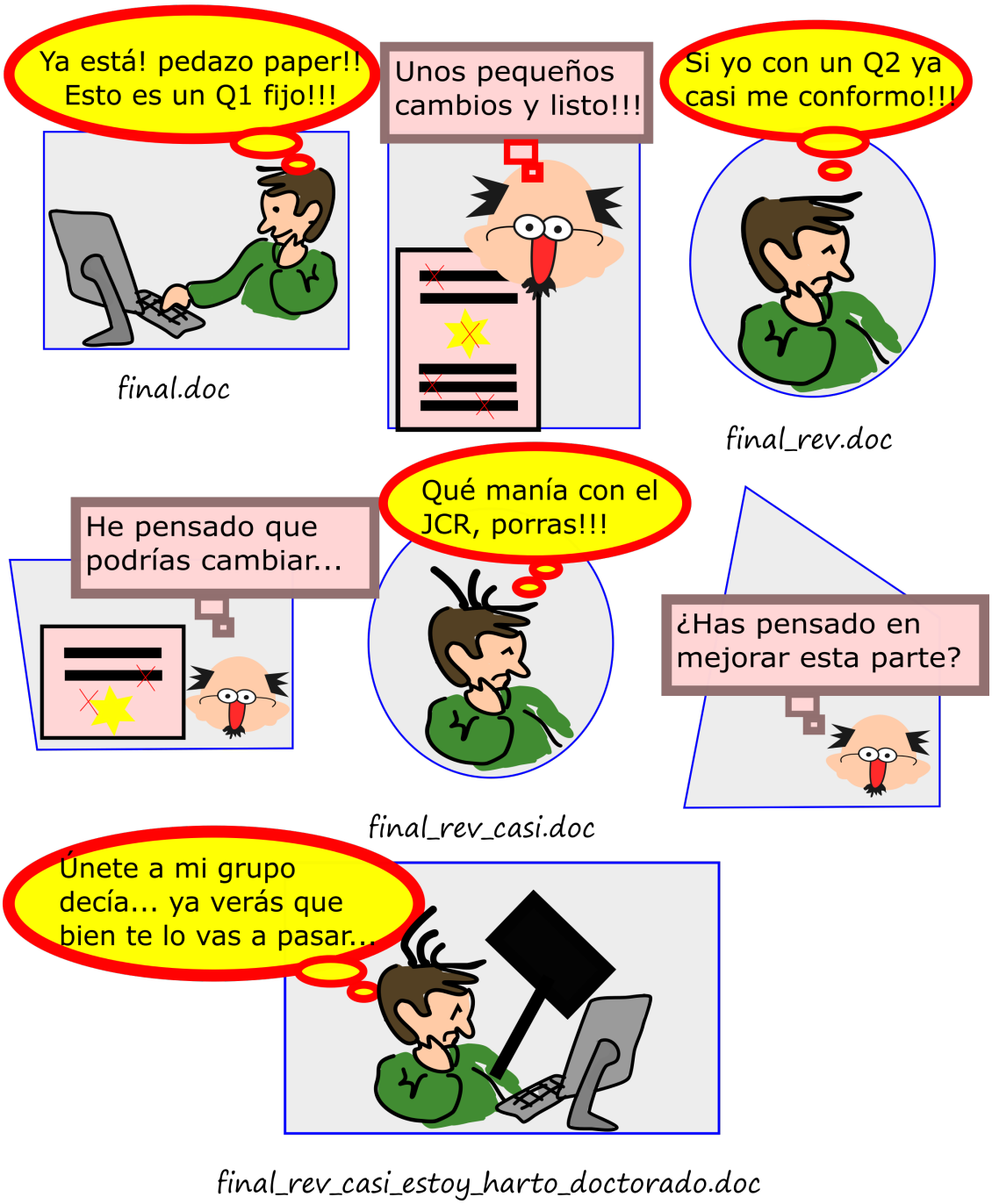
1 Compartir y gestionar documentos

1.1 Introducción

Todos los que han escrito un artículo/documento saben lo que supone el control de versiones.

```
In [1]: from IPython.display import Image
        Image(filename='./figuras/final_version.png')
```

```
Out[1]:
```



Uno de los aspectos más importantes en la generación de documentos basados en el concepto de investigación reproducible es su gestión y compartición. La gestión de la documentación puede realizarse mediante herramientas de control de versiones, como Visual SourceSafe, BitKeeper o Git. Las diferencias entre unas y otras herramientas tiene relación con aspectos como el tipo de arquitectura: distribuida, local, etc. o el tipo de licencia: propietaria, open source. En este documento se realiza una introducción a Git, herramienta open source de carácter distribuido para el control de versiones y el trabajo colaborativo.

Git (pronunciado "guit") es un sotware de control de versiones con una arquitectura definida en origen

por Linus Torvalds como respuesta al cambio de licencia del software de control de versiones BitKeeper, utilizado en origen para el desarrollo del kernel de Linux. Entre las características más importantes de Git cabe destacar:

- Soporte al desarrollo de código no lineal. Permite generar ramas de forma sencilla, así como combinarlas, realizar saltos entre versiones, etc.
- Gestión distribuida. La filosofía de Git es proporcionar a cada usuario una copia local del historial completo del desarrollo. Los cambios se propagan entre los repositorios locales.
- Gestión eficiente de grandes proyectos. El motor del sistema busca las diferencias entre ficheros y su funcionamiento está optimizado para este tipo de operaciones. Este características facilita la gestión eficiente de grandes proyectos.

1.2 Fundamentos de Git

La diferencia principal entre Git y otros sistemas de control de versiones tiene que ver con la forma en la que Git gestiona los documentos del proyecto. La mayor parte de los sistemas de control de versiones se basan en la búsqueda de cambios o diferencias en los ficheros gestionados. Git no utiliza este planteamiento, sino que obtiene fotos del sistema de ficheros. Cada vez que el usuario realiza un commit Git obtiene una instantánea del sistema de ficheros.

1.3 Los tres estados de Git

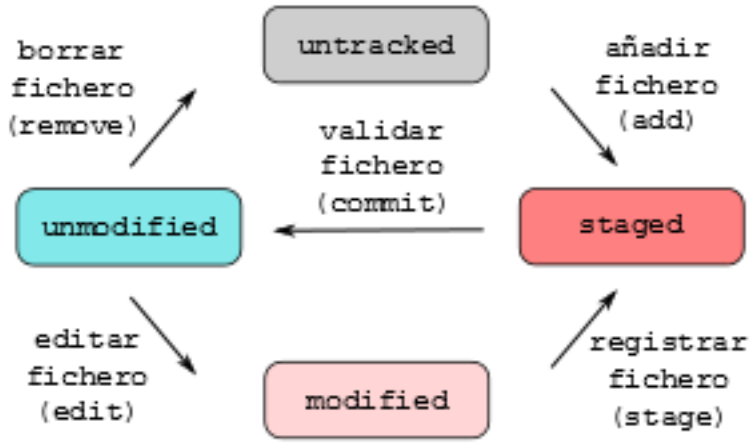
Entender cómo funciona Git consiste básicamente en entender que los ficheros gestionados por Git están en uno de los tres estados siguientes:

- modified
- staged
- committed

En la figura siguiente se muestra un esquema simplificado de los estados de un fichero en Git y los mecanismos para producir un cambio de estado.

```
In [2]: Image(filename='./figuras/git_estados_01.png', width=300)
```

Out[2]:




```
In [3]:
```

generacion.nbconvert

January 20, 2017

1 Generación de documentos

1.1 Windows. Utilización de ficheros .bat

1.2 Windows / Linux. Utilización de ficheros makefile

CLEANFILES=$(.aux=.tex=.out=.log=.pdf)
latexfiles:

```
ipython nbconvert --to latex introduccion.ipynb
ipython nbconvert --to latex git.ipynb
ipython nbconvert --to latex accesodatos.ipynb
ipython nbconvert --to latex otros_lenguajes.ipynb
ipython nbconvert --to latex ejercicio_dados.ipynb
```

buildpdf: latexfiles

```
pdflatex introduccion.tex
pdflatex git.tex
pdflatex accesodatos.tex
pdflatex otros_lenguajes.tex
pdflatex ejercicio_dados.tex
octave octave_1.m
pdflatex ./matweave/myexample
pdfjoin introduccion.pdf git.pdf accesodatos.pdf otros_lenguajes.pdf ./matweave/myexample.pdf ejercicio
evince ejercicio_dados-joined.pdf
```

clean:

```
rm -f *.aux
rm -f *.tex
rm -f *.out
rm -f *.log
rm -f *.pdf
```

In [1]:

accesodatos.nbconvert

January 20, 2017

1 iPython Notebook

1.1 Acceso a diferentes orígenes de datos

Python es un lenguaje de programación que permite el acceso rápido a diferentes orígenes de datos:

1. Ficheros de datos.
2. Bases de datos.
3. Internet.

1.2 Ficheros .csv

Los ficheros de texto de tipo .csv o similar pueden ser leídos fácilmente con el paquete csv.

In [1]:
```
import csv
import matplotlib.pyplot as plt
%matplotlib inline

nombrefichero='datos.csv'
data=[]
```

In [2]:
```
f = open(nombrefichero, 'rt')
try:
    reader=csv.reader(f, delimiter=';')
    header=reader.next()
    data=[row for row in reader]
finally:
    f.close()
```

In [3]:
```
print header
print "**********************"
```

```
['year', 'Ta']
**********************
```

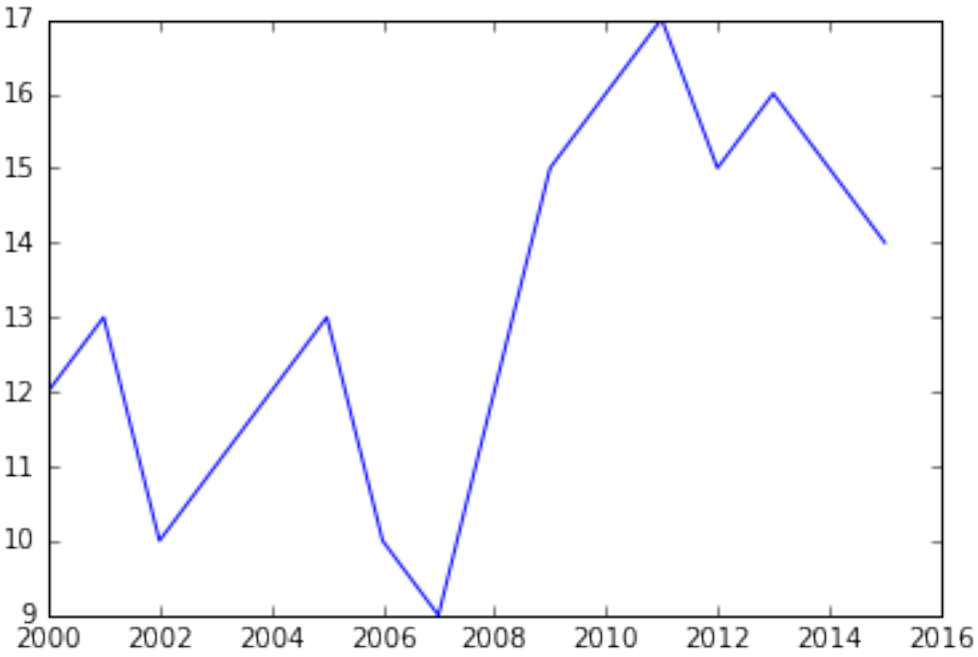
In [4]:
```
data
```

Out[4]:
```
[['2000', '12'],
 ['2001', '13'],
 ['2002', '10'],
 ['2003', '11'],
 ['2004', '12'],
 ['2005', '13'],
 ['2006', '10'],
 ['2007', '9'],
```

```
          ['2008', '12'],
          ['2009', '15'],
          ['2010', '16'],
          ['2011', '17'],
          ['2012', '15'],
          ['2013', '16'],
          ['2014', '15'],
          ['2015', '14']]
```

```
In [5]: x1=[x[0] for x in data]
        y1=[x[1] for x in data]
```

```
In [6]: plt.plot(x1, y1)
        plt.show()
```



1.3 Acceso a bases de datos sqlite3

```
In [7]: import sqlite3
```

```
In [8]: sqlite3.version
```

```
Out[8]: '2.6.0'
```

```
In [9]: sqlite3.sqlite_version
```

```
Out[9]: '3.6.21'
```

```
In [10]: ficherobbdd='bbddsqlite3'
```

```
In [11]: import sqlite3 as lite
         import sys

         con = None

         try:
             con = lite.connect(ficherobbdd)

             cur = con.cursor()
             cur.execute('SELECT SQLITE_VERSION()')

             data = cur.fetchone()

             print "SQLite version: %s" % data

         except lite.Error, e:

             print "Error %s:" % e.args[0]
             sys.exit(1)

         finally:

             if con:
                 con.close()

SQLite version: 3.6.21

In [12]: con = None

         try:
             con = lite.connect(ficherobbdd)

             cur = con.cursor()
             cur.execute("SELECT * FROM tabla1")

             rows = cur.fetchall()

             for row in rows:
                 print row

         except lite.Error, e:

             print "Error %s:" % e.args[0]
             sys.exit(1)

         finally:

             if con:
                 con.close()

(u'ana', 11, 165.5)
(u'maria', 12, 170.0)

In [13]:
```

otros_lenguajes

January 20, 2017

1 Interacción con otros lenguajes

1.1 Introducción

Uno de los aspectos positivos de jupyter es su capacidad para utilizar o interaccionar con otros kernels. Actualmente están disponibles kernels para R, Julia y Octave.

```
In [1]: %load_ext oct2py.ipython
```

```
In [7]: x = %octave linspace(0,2*pi,128);
        y = sum(x)
```

```
In [9]: print(y)
```

```
[ 0.          0.0494739   0.0989478   0.1484217   0.1978956   0.2473695
  0.2968434   0.3463173   0.3957912   0.4452651   0.494739    0.5442129
  0.5936868   0.6431607   0.6926346   0.7421085   0.7915824   0.8410563
  0.8905302   0.9400041   0.989478    1.0389519   1.0884258   1.1378997
  1.1873736   1.2368475   1.2863214   1.3357953   1.3852692   1.4347431
  1.484217    1.5336909   1.5831648   1.6326387   1.6821126   1.7315865
  1.7810604   1.8305343   1.8800082   1.9294821   1.978956    2.0284299
  2.0779038   2.1273777   2.1768516   2.2263255   2.2757994   2.3252733
  2.3747472   2.4242211   2.473695    2.5231689   2.5726428   2.6221167
  2.6715906   2.7210645   2.7705384   2.8200123   2.8694862   2.9189601
  2.968434    3.0179079   3.0673818   3.1168557   3.1663296   3.2158035
  3.2652774   3.3147513   3.3642252   3.4136991   3.463173    3.5126469
  3.5621208   3.6115947   3.6610686   3.7105425   3.7600164   3.8094903
  3.8589642   3.9084381   3.957912    4.0073859   4.0568598   4.1063337
  4.1558076   4.2052815   4.2547554   4.3042293   4.3537032   4.40317711
  4.45265101  4.50212491  4.55159881  4.60107271  4.65054661  4.70002051
  4.74949441  4.79896831  4.84844221  4.89791611  4.94739001  4.99686391
  5.04633781  5.09581171  5.14528561  5.19475951  5.24423341  5.29370731
  5.34318121  5.39265511  5.44212901  5.49160291  5.54107681  5.59055071
  5.64002461  5.68949851  5.73897241  5.78844631  5.83792021  5.88739411
  5.93686801  5.98634191  6.03581581  6.08528971  6.13476361  6.18423751
  6.23371141  6.28318531]
```

```
In [ ]:
```

MATweave: Integración de código MATLAB/Octave en LaTeX*

Mario Mañana Canteli
mananam@unican.es
Dpto. de Ingeniería Eléctrica y Energética
Universidad de Cantabria

16/02/2016

Abstract

MATweave define un procedimiento para integrar código MATLAB/Octave dentro de documentos LaTeX. El objetivo final es facilitar a los usuarios de MATLAB/Octave la generación integral de investigación reproducible.

1 Introducción

MATweave proporciona un método simple para la generación de informes que integran datos y código utilizando LaTeX como lenguaje de edición, y puede considerarse como un sistema básico para la generación de documentos basados en el paradigma de la investigación reproducible. El objetivo final de este conjunto de herramientas es obtener un entregable que integre documentación, datos y código que pueda, además, ser reutilizado por otros usuarios. MATweave está inspirado en Sweave **?**, diseñado originalmente para combinar programas en R y LaTeX.

1.1 Procedimiento

La idea que subyace detrás de MATweave es utilizar códigos específicos para indicar el inicio y fin del fragmento de código dentro del fichero LaTeX.

1. El primer procedimiento es utilizar códigos de comentario tipo bloque, introducidos en Octave 3.2 and MATLAB R14.

   Los comentarios tipo bloque en MATLAB/Octave se abren con `%{` y se cierran con `%}` permitiendo incluir código dentro:

   ```
   %{
   Esto es un comentario!
   %}
   for i = 1:10
     % Este codigo esta fuera del comentario.
     disp(i)
   end
   ```

   El 'truco' consiste en que un bloque de comentario en MATLAB/Octave es un comentario en LaTeX(debido a que comienza con el carácter `%`), pero *no* un bloque de comentario.

*Adaptado de Neil D. Lawrence

```
%{
En MATLAB/Octave esto sería un comentario, pero en \LaTeX sí se compilaría.
Esto significa que es posible escribir código \LaTeX, como por ejemplo
$\tau = 2\pi$, dentro de un script MATLAB.
Ahora debemos ser capaces de escribir MATLAB/Octave dentro de un
fichero \LaTeX.
%}
```

2. El segundo 'truco' es incluir el paquete `verbatim` y utilizarlo para definir un nuevo entorno MATLAB/Octave utilizando los comandos siguientes:

   ```
   \newenvironment{matlab}{\comment}{\endcomment}
   \newenvironment{octave}{\comment}{\endcomment}
   \newenvironment{matlabv}{\verbatim}{\endverbatim}
   \newenvironment{octavev}{\verbatim}{\endverbatim}
   ```

   Esto permitirá incluir código MATLAB/Octave que no será leído por LaTeXutilizando: `\begin{octave} ... \end{octave}`, así como código que será mostrado en un entorno verbatim utilizando `\begin{octavev} ... \end{octavev}`. Por supuesto es posible hacer esto mismo utilizando el entorno estándar `verbatim`. Sin embargo, utilizando un nuevo entorno es posible definir los fragmentos de código que se muestran y/o se ejecutan.

1.2 Ejemplo

El código siguiente se ha desarrollado utilizando el entorno `matlabv` definido anteriormente. El código fuente de este documento comienza con un código de bloque abierto MATLAB/Octave `%{`. Dicho código se cierra al comiendo de un bloque MATLAB/Octave, de forma que será compilado por dichos programas.

```
%}
% El codigo MATLAB/Octave comienza aqui
tau = 2*pi;
x = linspace(-3, 3, 100)';
y = 1/sqrt(tau)*exp(-0.5*x.^2);
plot(x, y, 'r-');
set(gca, 'fontname', fontName, 'fontsize', fontSize);
print -dpdf myGaussian.pdf
% El codigo MATLAB/Octave finaliza aqui
%{
```

La combinación de dos bloques significa que el código fuente de este documento puede ser compilado en LaTeXo en MATLAB/Octave.

2 Ejecutando MATweave

El procedimiento para generar el documento se resume a continuación:

- Ejecutar el comando 'source fichero.tex' en Matlab/Octave.
- Ejecutar el comando 'pdflatex fichero.tex' en la línea de comandos del sistema operativo. Si el documento incluye referencias bibliográficas es necesario realizar una secuencia:
  - 'pdflatex fichero.tex'
  - 'bibtex fichero.tex'

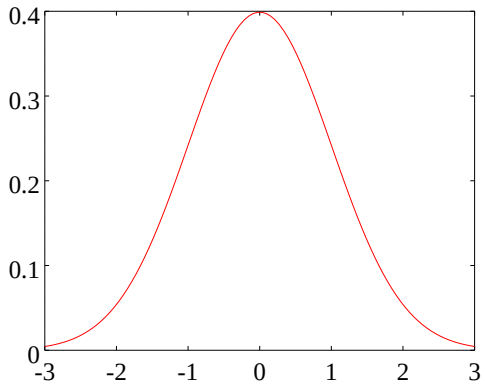


Figure 1: Curva de densidad de probabilidad Gausiana.

– 'pdflatex fichero.tex'

Como se ha comentado, el fichero LaTeX puede ser ejecutado en Octave con el comando `source`. Como ejercicio, guardar este documento como `myexample.tex` y ejecutar `source myexample.tex` en la línea de comandos de Octave. En MATLAB el procedimiento es un poco más complicado.

La salida del código incluye figuras en formato .pdf. La primera de las figuras se muestra en la Figura **??**.

El código MATLAB/Octave puede ser mostrado u ocultado en el documento según convenga en cada momento atendiendo a la necesidad de documentar un procedimiento de cálculo e incluso por motivaciones de depuración del código. A modo de ejemplo, el histograma mostrado en la Figura **??** ha sido codificado utilizando el entorno `\begin{matlab} ... \end{matlab}`.

Otra posibilidad que puede mejorar la visualización de código es utilizar el paquete `listings`. Otra posibilidad para mejorar la automatización del documento es utilizar MATLAB/Octave para generar un fichero .tex con la tabla que se pretenda visualizar en el documento LaTeX.

```
%}
% Codigo para generar una tabla con numeros aleatorios.
rows = 3;
cols = 4;
numSigFigs = 3;
resultMatrix = randn(3, 4);
fid = fopen('results.tex', 'w');
for i = 1:rows
  for j = 1:cols
    fprintf(fid, ['$' num2str(resultMatrix(i, j), numSigFigs) '$']);
    if j < cols
```

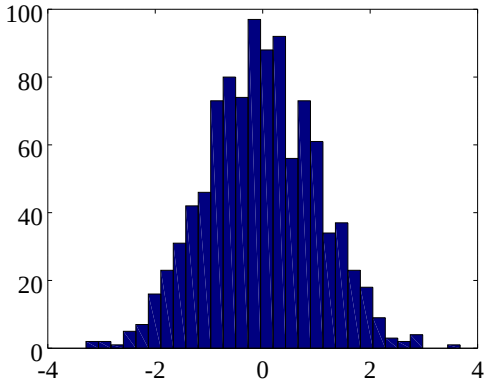


Figure 2: Histograma con 1000 muestras de una distribución Gausiana.

Table 1: Ejemplo de una tabla de números aleatorios con MATLAB/Octave.

A	B	C	D
-0.442	1.59	0.302	-1.5
-0.418	2.39	-0.123	0.632
-1.46	1.08	0.128	-2.35

```
      fprintf(fid, ' & ');
    end
  end
  if i < rows
    fprintf(fid, '\\\\\n');
  end
end
fclose(fid);
%{
```

El fichero de resultados con la tabla puede ser incorporado al documento mediante el código `\input{results.tex}` y mostrado como aparece en la Tabla **??**.

2.1 Consejos adicionales

Se considera una buena medida comenzar el fichero LaTeX con un conjunto de comandos que limpien la memoria, definan las rutas necesarias, borren las figuras, etc.

2.2 Nota para usuarios de Beamer

Aquellos usuarios de MATweave que generen presentaciones utilizando Beamer deben utilizar la opción `[fragile]` en aquellas transparencias que contengan el entorno MATLAB/Octave. Sin dicha opción Beamer no será capaz de manejar los entornos `verbatim` o `comment`.

Se necesita una versión MATLAB R14 o superior y Octave 3.2 o superior. Para compilar el fichero en Linux:

```
octave --eval source\ myexample.tex
```

o

```
matlab < myexample.tex
```

y después,

```
pdflatex myexample
```

3 Conclusiones

Se ha mostrado que la integración de código MATLAB/Octave con LaTeXpuede realizarse mediante procedimientos sencillos. El procedimiento no es tan elegante como la pareja Sweave con R, donde tanto las variables como las gráficas se integran en el texto de forma natural.

El objetivo de MATweave es facilitar la generación de documentos reproducibles.

Nota

Este documento fue generado utilizado MATweav utilizando Octave version 4.0.0 en una máquina `i686-w64-mingw32` . El documento fue generado el 16/02/2016 .

ejercicio_dados.nbconvert

January 20, 2017

```
In [1]: import pandas as pd
        import numpy as np
        import pylab as P
        %matplotlib inline
```

0.1 Resultados de la serie de lanzamientos

```
In [2]: url = 'https://raw.githubusercontent.com/mmanana/data/master/experimento_dados.csv'
        df = pd.read_csv(url,sep=";")

In [3]: df
```

Out[3]:

	Lanzamiento	Resultado
0	1	4
1	2	3
2	3	5
3	4	2
4	5	6
5	6	3
6	7	2
7	8	4
8	9	1
9	10	2

0.2 Valor medio de la serie de lanzamientos, así como la suma de resultados.

El valor medio se obtiene mediante la ecuación: $media = \frac{\sum x[n]}{N}$

```
In [4]: media=5.0*np.mean(df['Resultado'])
        print media

16.0

In [5]: suma=np.sum(df['Resultado'])
        print suma

32
```

0.3 Histograma de resultados

```
In [6]: histograma=np.histogram(df['Resultado'],bins=6)
        print histograma

(array([1, 3, 2, 2, 1, 1], dtype=int64), array([ 1.        ,  1.83333333,  2.66666667,  3.5       ,  4.3
        5.16666667,  6.        ]))
```

In [7]:
```
bins = 6
# the histogram of the data with histtype='step'
n, bins, patches = P.hist(df['Resultado'], bins, normed=1, histtype='bar', rwidth=0.8)

#
# now we create a cumulative histogram of the data
#
#P.figure()
P.show()
```