open_science.nbconvert

January 29, 2018

1 Open Science

In [1]:
```
from IPython.display import Image
import time

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

```
a Ultima version: 29/01/2018 01:41:17
```

1.1 Introducción

Wikipedia define la ciencia abierta o **Open Science** como una movimiento que tiene como objetivo principal conseguir la accesibilidad universal al conocimiento científico. Desde un punto de vista práctico, la ciencia en abierto incluye paradigmas como la investigación abierta y el **Open Notebook Science**, que persiguen proporcionar acceso público y abierto a los experimentos y resultados obtenidos al desarrollar éstos. Esta acceso universal es especialmente importante cuando la investigación está financiada con fondos públicos.

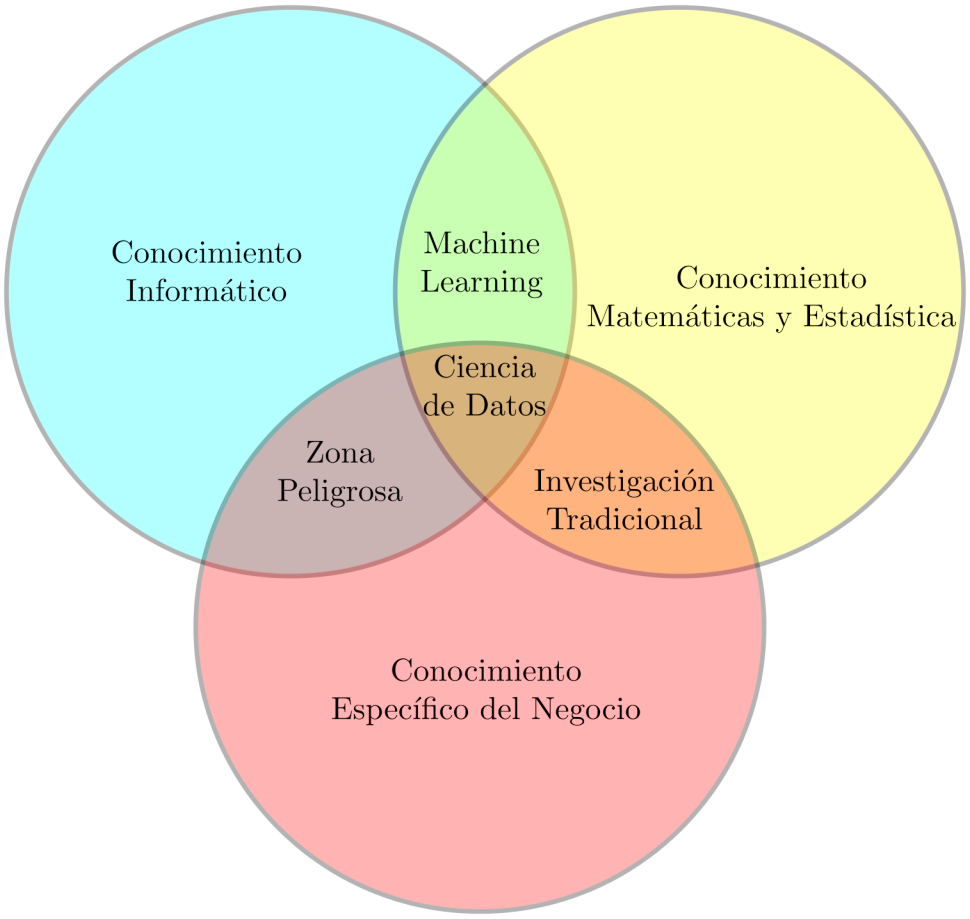
In [2]:
```
Image(filename='./figuras/220px-Open_Science_Logo.png',width=200)
```

Out[2]:

Uno de los referentes más representativos de la investigación en abierto es el proyecto europeo CERN y su gran colisionador de hadrones (LHC). Los datos obtenidos en sus trabajos de investigación son accesibles públicamente mediante la red LHC [http://opendata.cern.ch]

In [3]:
```
Image(filename='./figuras/data_science_diagram.png',width=500)
#print('Logo representativo de Ciencia Abierta')
```

Out[3]:



1.2 Referencias

[] K.N. Cukier and V.M Mayer-Schoenberger "The rise of Big Data" Journal Foreign Affairs. May/June 2013.

introduccion.nbconvert

January 29, 2018

1 Conceptos generales

```
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: 29/01/2018 01:41:21
```

1.1 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?

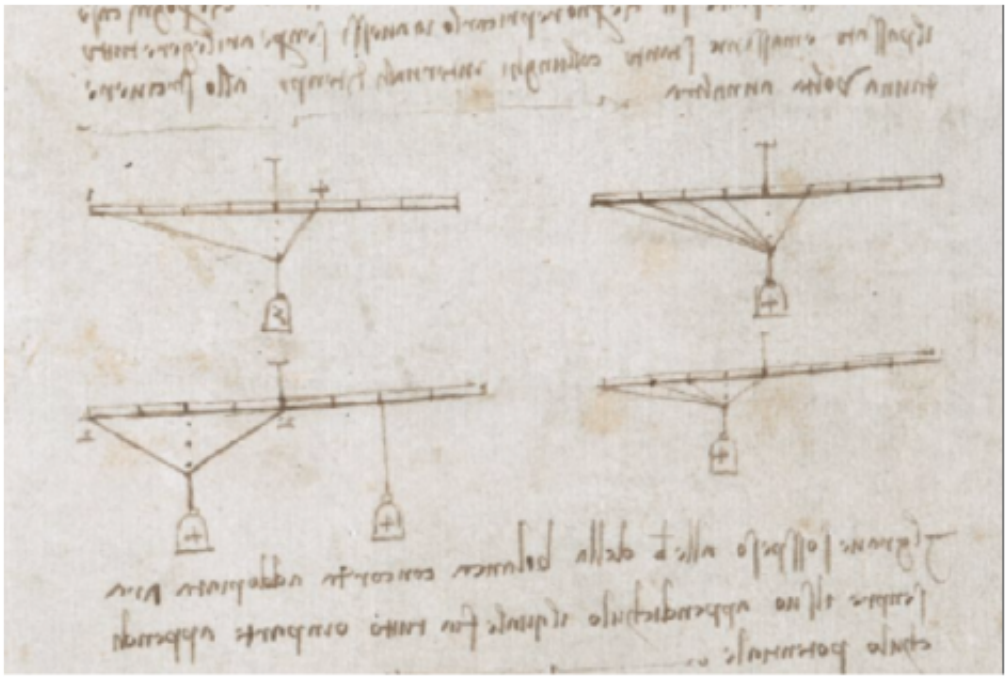
Comenzaremos por analizar el método clásico para ir resolviendo poco a poco los desafíos planteados...

1.2 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.

1.2.1 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.

1.2.2 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.

...

1.3 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

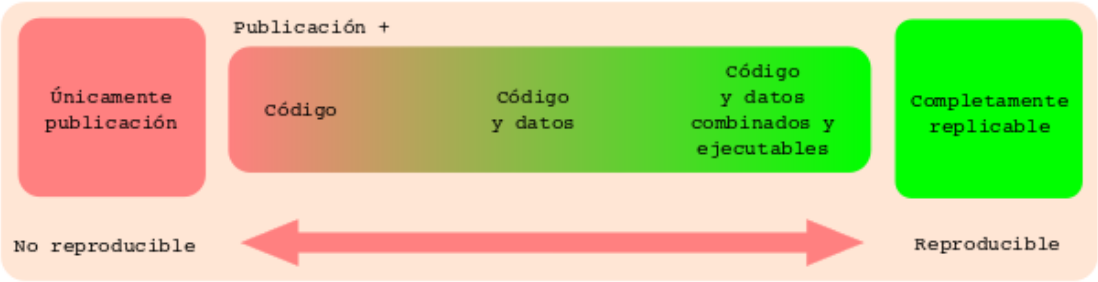
- Verificación y fiabilidad. Facilidad para encontrar errores. Los resultados que se obtienen hoy deben ser los mismos que los que se producirían mañana.
- Transparencia. La disponibilidad pública de datos y procedimientos permite mejorar los índices de citas de autores y sus instituciones.
- Eficiencia. 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')
```

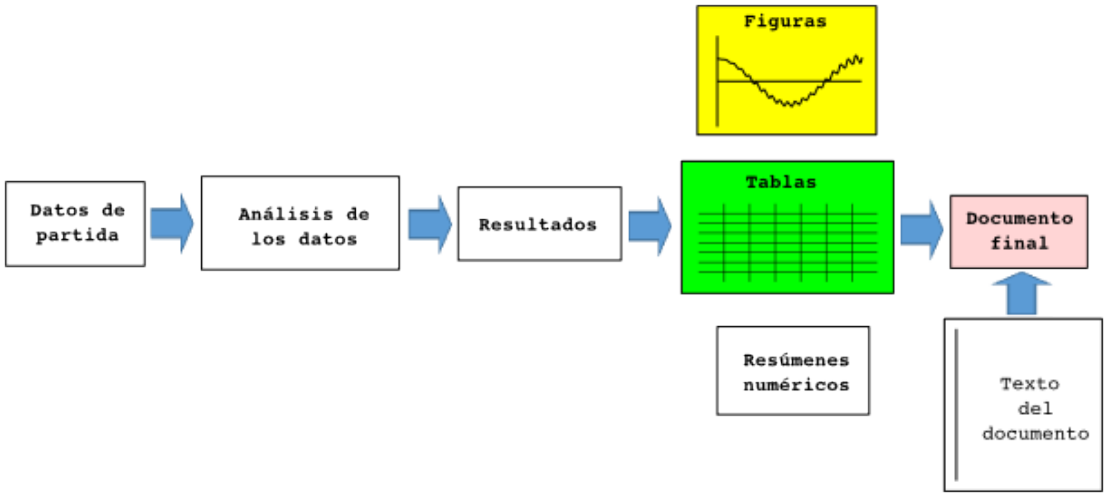
```
Out[3]:
```



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 29, 2018

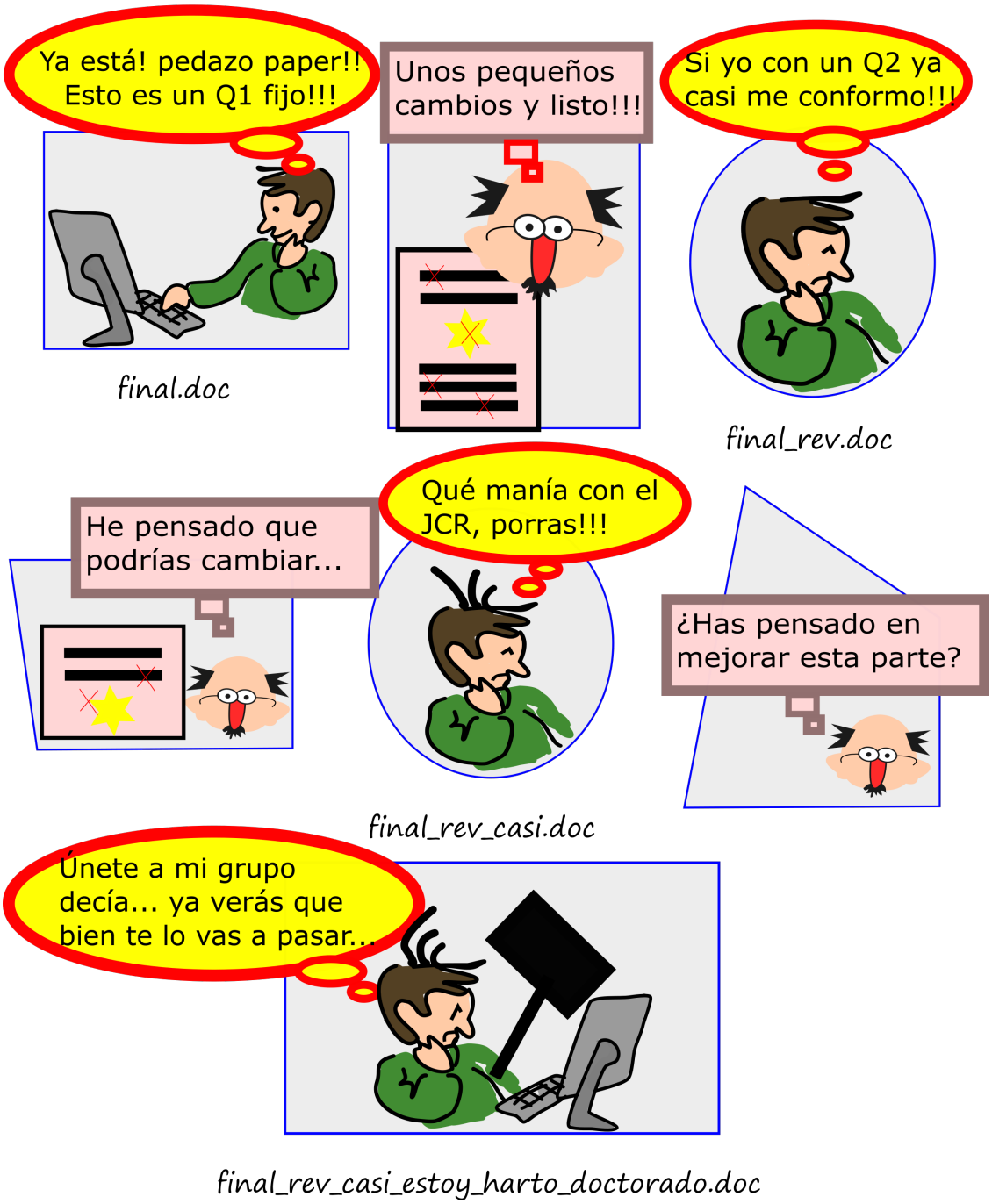
1 Compartir y gestionar documentos

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

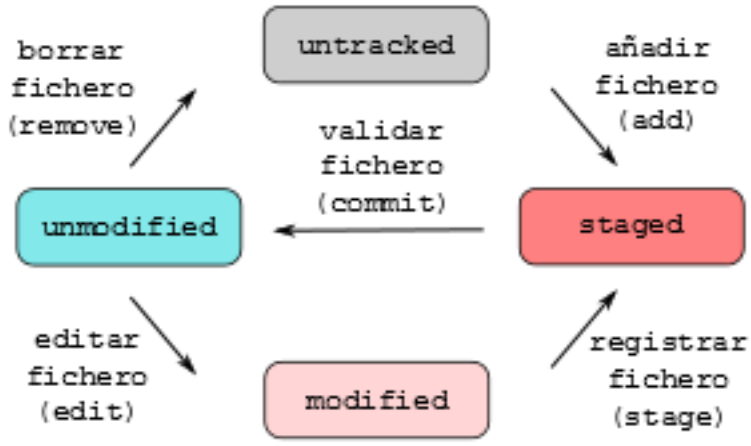
Entender cómo funciona Git consiste básicamente en entender que los ficheros gestionados por Git están en uno de los tres estados siguientes:

- 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]:
```

documentando_workflow.nbconvert

January 29, 2018

1 Documentando un flujo de datos

1.1 Enlaces (Links)

En general, las direcciones URL incluidas en el texto se transforman directamente en enlaces activos. Si se quiere definir un enlace de forma explícita:

`http://www.gtea.unican.es` `correo@unican.es`

Salida:

http://www.gtea.unican.es

correo@unican.es

Si se quiere definir un texto diferente como referencia:

Texto a visualizar

salida:

Texto a visualizar

Es posible también añadir un título title que se muestra al pasar el cursor sobre el texto a visualizar:

Texto a visualizar

salida:

Texto a visualizar

1.2 Enlaces a referencias (Reference Links)

Los enlaces pueden incluirse también mediante referencias numeradas al estilo de las utilizadas en las publicaciones científicas.

En un lugar de Cantabria es posible encontrar a la UC. Dentro de la UC es posible encontrar al grupo de investigación GTEA.

Salida:

Es posible definir este tipo de enlaces de otra forma.

Se pueden utilizar también referencias shortcut, que se enlazan "shortcut" al enlace denominado "shortcut" en el párrafo siguiente.

1.3 Formateo básico de texto

Es posible enfatizar el texto utilizando * o _

texto en cursiva y texto en cursiva

texto en negrita and **texto en negrita**

texto en cursiva y negrita and **texto en cursiva y negrita**

Se puede tachar texto utilizando HTML texto tachado

Un salto de línea al final de una línea genera un "line break".

Dos saltos de línea consecutivos genera un nuevo párrafo.

Se puede utilizar el carácter > al inicio de una línea para incluir citas literales.

> Texto de ejemplo

También es posible incluir texto literal sin ninguna interpretación de código.

```
Si se incluye texto entre dos tildes se mostrará sin interpretación. Por ejemplo
<b>HTML</b> Se mostrará como texto. Es una forma útil de mostrar código fuente, código
HTML o Markdown. <b>Esto no se mostrará como HTML</b>.
```

1.4 Listas

1.4.1 Listas no ordenadas

- Utiliza asteriscos * para listas no ordenadas
- otro elemento de la lista
- también se puede utilizar el signo +.
- o el menos -.

1.4.2 Listas ordenadas

1. Lista ordenada con números.

- Un signo + se mostrará aquí como 2.
- Y el asterisco como 3.

7. Cualquier número, +, -, o * también servirá.

    - Utilizar cuatro espacios o un tab para sublistas.
        1. Y así para subsublistas.

Tablas
Se pueden generar tablas utilizando linea vertical y símbolos menos incluso con formato:

Encabezado primera columna	Encabezado segunda columna
contenido 1,1	contenido 1,2
contenido 2,1	**contenido 2,2**

Se puede utilizar también código HTML
Encabezado
Incluir 1 o más símbolos menos o símbolos igual (— o ===) debajo del título.

2 Un gran encabezado

2.1 Un encabezado de segundo nivel

Para incluir una imagen colocar un símbolo "!" al inicio de un enlace:

```
![alternate text](https://sourceforge.net/images/icon_linux.gif)
```

El enlace mostrará un "texto alternativo" si no puede cargar la imagen.

2.2 Video embebidos

Para incrustar un video YouTube utilizar la macro `embed` (solo soporta YouTube):

```
[[embed url=http://www.youtube.com/watch?v=6YbBmqUnoQM]]
```

2.3 html

La documentación markdown permite también utilizar algunas etiquetas HTML.
texto en negrita
[TOC]

3 Section 1

3.1 Sub-section 1

4 Section 2

Es posible también resaltar texto Python

```
:::python
import abc


#!/usr/bin/python
import abc
```

My code
Muchas gracias a John Gruber y Aaron Swartz por crear **Markdown**.
Esta página está basada en algunos ejemplos creados por Greg Schueler, greg@vario.us

scipy.nbconvert

January 29, 2018

1 SciPy

Importación básica de paquetes

```
In [1]: import numpy as np
        import scipy as sp
        import matplotlib as mpl
        import matplotlib.pyplot as plt
        %matplotlib inline
        from scipy import optimize
```

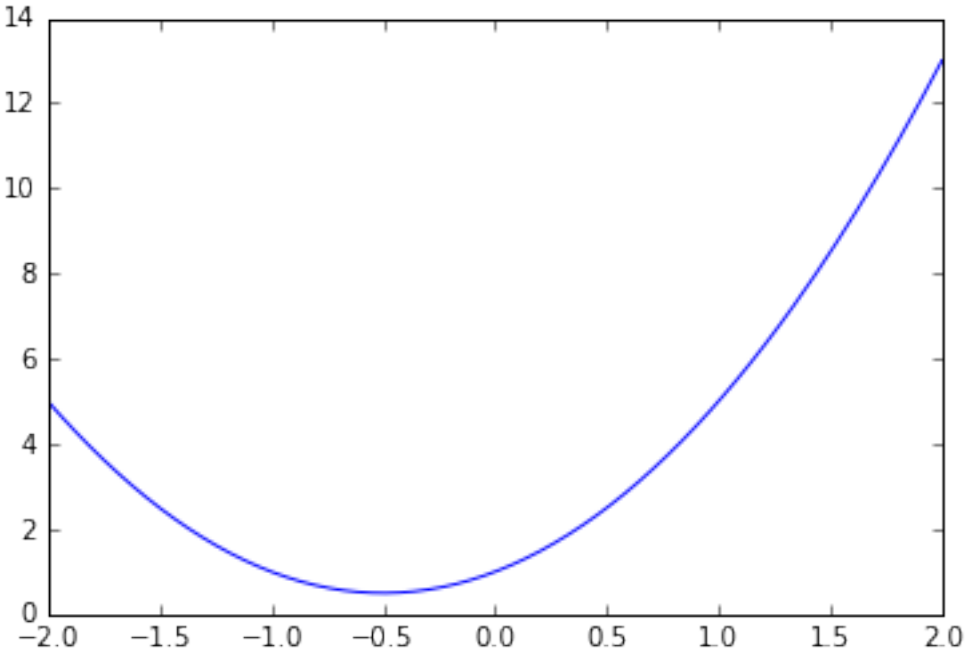
Un ejemplo de minimización

```
In [2]: def f(x):
            return (2*(x**2)+2*x+1.0)
```

```
In [3]: N=128
        x_data=np.linspace(-2.0,2.0,N)
        y_data=np.empty(N)
        for i,valor in enumerate(x_data):
            y_data[i]=f(valor)
```

```
In [4]: plt.plot(x_data, y_data)
```

```
Out[4]: [<matplotlib.lines.Line2D at 0xc43d530>]
```




```
In [5]: result = optimize.minimize_scalar(f)
        #print(result.success)

In [6]: print('minimo de f(x) se obtiene para x: ' + str(result.x))

minimo de f(x) se obtiene para x: -0.5
```

generacion.nbconvert

January 29, 2018

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 29, 2018

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']
**********************
```

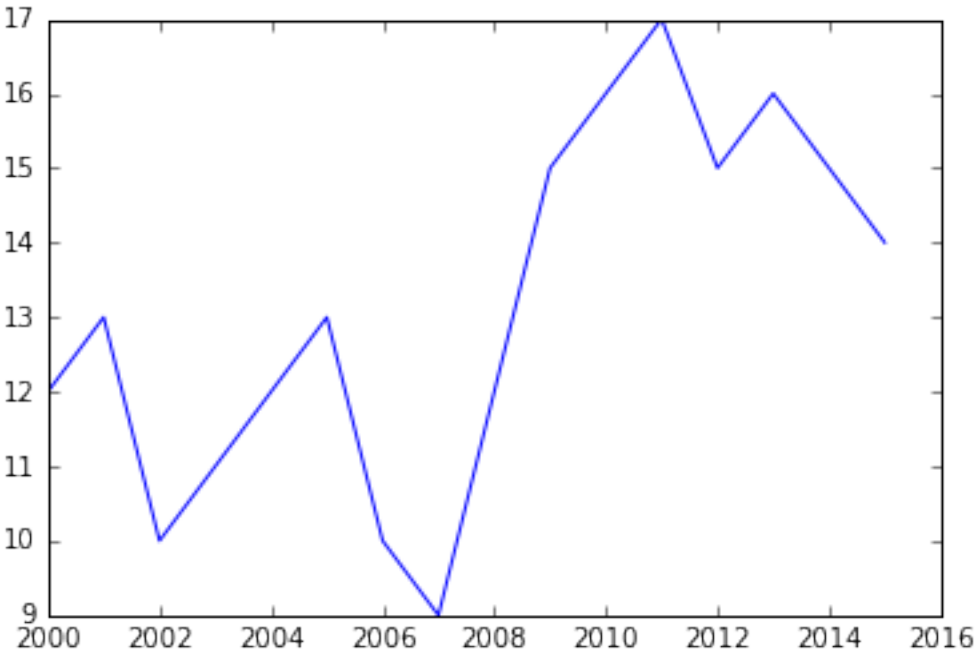
In [4]:
```
data
```

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]: '3.8.11'
```

```
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: 3.8.11

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]:
```

ms_sql.nbconvert

January 29, 2018

1 MS SQL. Acceso a datos con python

Este capítulo proporciona un ejemplo básico de acceso a datos almacenados en una base de datos utilizando python.

El paquete pypyodbc proporciona acceso a servidores MS SQL Server.

```
In [1]: import pypyodbc
        import pandas as pd
```

Se definen los parámetros básicos de la conexión. Uno de los inconvenientes de este método es que hace públicos usuario y password de la bbdd. Esto no es siempre conveniente.

```
In [2]: connection = pypyodbc.connect('Driver={SQL Server};'

                                      'Server=den1.mssql1.gear.host;'

                                      'Database=opendata;'

                                      'uid=opendata;pwd=Ox4gL32-?iQD')
```

```
In [3]: cursor = connection.cursor()
        SQLCommand = ("SELECT ALL [Fecha],[Temperatura],[Viento],[Estacion]  FROM [OPENDATA].[dbo].[dat

        cursor.execute(SQLCommand)
        results = cursor.fetchone()
```

```
In [4]: for d in cursor.description:
            print (d[0])
```

```
fecha
temperatura
viento
estacion
```

```
In [5]: datos = cursor.fetchall()
        for row in cursor.fetchall():
            for field in row:
                print( field)
            print(" ")
```

```
In [6]: print( datos)
```

```
[(datetime.datetime(2017, 2, 2, 0, 0), 12.0, 3.0, u'WS1       '), (datetime.datetime(2017, 2, 3, 0, 0),
```

```
In [7]: type( datos)
```

```
Out[7]: list

In [8]: print( datos[0][1])

12.0

In [9]: df=pd.DataFrame(datos)
        print(df)

0   1  2           3
0 2017-02-02  12  3  WS1
1 2017-02-03  15  4  WS2
2 2017-02-04  16  5  WS2
3 2017-02-05  18  4  WS2

In [10]: print(df[1])

0    12
1    15
2    16
3    18
Name: 1, dtype: float64

In [11]: df.describe()

Out[11]:            1         2
         count   4.00  4.000000
         mean   15.25  4.000000
         std     2.50  0.816497
         min    12.00  3.000000
         25%    14.25  3.750000
         50%    15.50  4.000000
         75%    16.50  4.250000
         max    18.00  5.000000

In [12]: cursor.close()
         connection.close()

In [13]:
```

python_json.nbconvert

January 29, 2018

1 JSON

JSON es el acrónimo de JavaScript Object Notation y hace referencia a un formato de texto orientado hacia el intercambio de datos. Se considera una alternativa al formato **XML**. La definición completa está basada en la norma ECMA-262, aunque frecuentemente se implementa únicamente un subconjunto de la misma. A medida que se ha ido extendiendo, han aparecido APIs o parsers para casi todos los lenguajes de programación. En este capítulo se revisa la implementación utilizando **python**.

```
In [1]: import json
        import urllib
```

```
In [2]: url = u'http://maps.googleapis.com/maps/api/geocode/json?address=Universidad de Cantabria, Santa
```

```
In [3]: response = urllib.urlopen(url)
        data = json.loads(response.read())
```

```
In [4]: print( data)
```

```
{u'status': u'OK', u'results': [{u'geometry': {u'location': {u'lat': 43.471405, u'lng': -3.804062}, u'vi
```

‘” {u’status’: u’OK’, u’results’: [{u’geometry’: { u’location’: { u’lat’: 43.471405, u’lng’: -3.804062}, u’viewport’: { u’northeast’: { u’lat’: 43.47275398029149, u’lng’: -3.802713019708498}, u’southwest’: { u’lat’: 43.47005601970849, u’lng’: -3.805410980291502}}, u’location_type’: u’APPROXIMATE’}, u’address_components’: [{u’long_name’: u’Santander’, u’types’: [u’locality’, u’political’], u’short_name’: u’Santander’}, {u’long_name’: u’Cantabria’, u’types’: [u’administrative_area_level_2’, u’political’], u’short_name’: u’S’}, {u’long_name’: u’Spain’, u’types’: [u’country’, u’political’], u’short_name’: u’ES’}, {u’long_name’: u’39005’, u’types’: [u’postal_code’], u’short_name’: u’39005’}], u’place_id’: u’ChIJz1mfL7lLSQ0Rczt72xAnk2w’, u’formatted_address’: u’Av los Castros s/n, 39005 Santander, Cantabria, Spain’, u’types’: [u’establishment’, u’point_of_interest’, u’university’] }] } ‘”

```
In [5]: print( data['status'])
```

```
OK
```

```
In [6]: print( data['results'][0]['geometry']['location']['lat'])
```

```
43.471405
```

```
In [7]: print( data['results'][0]['geometry']['location']['lng'])
```

```
-3.804062
```

ejercicio_dados.nbconvert

January 29, 2018

```
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
```

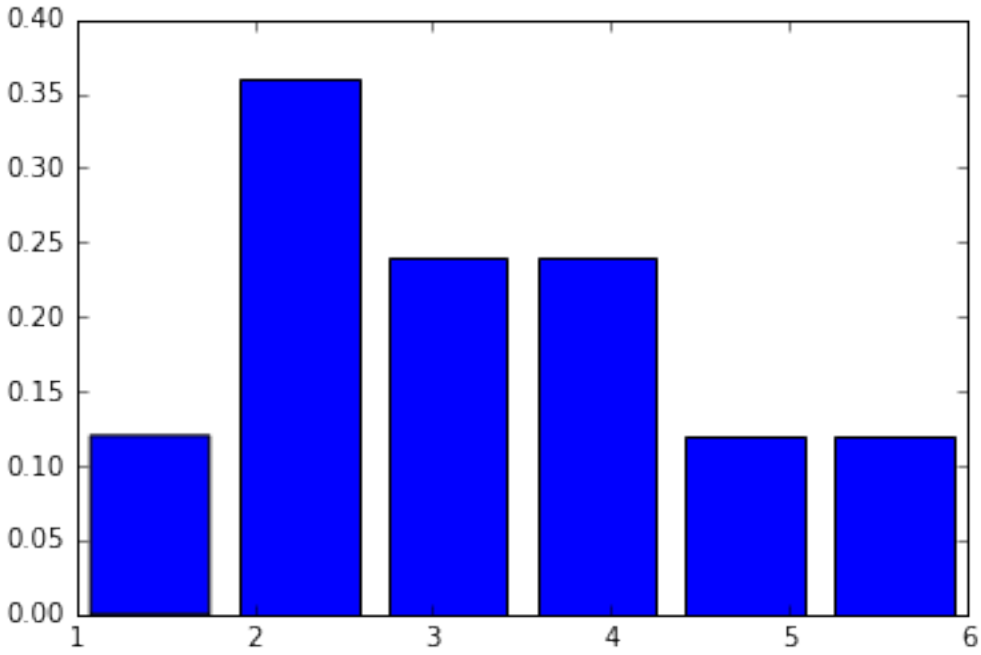
0.3 Histograma de resultados

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

(array([1, 3, 2, 2, 1, 1]), array([ 1.        ,  1.83333333,  2.66666667,  3.5       ,  4.33333333,
        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()
```



open_data.nbconvert

January 29, 2018

0.1 Open data

El concepto de **open data** o `datos en abierto` hace referencia a una filosofía que busca hacer públicos conjuntos de datos, sin aplicar ningún tipo de restricción, incluyendo derechos de autor, patentes u otras formas de restricción. Puede entenderse como un paradigma similar al del software libre pero que está centrado sobre conjuntos de datos.

0.1.1 Aspectos legales

El movimiento **open data** surge, desde una perspectiva histórica, en la necesidad de aumentar la transparencia en las administraciones públicas. Las leyes básicas que regulan este concepto son las siguientes:

- Directiva 2013/37/UE del Parlamento Europeo y del Consejo, de 26 de junio de 2013, por la que se modifica la Directiva 2003/98/CE relativa a la reutilización de la información del sector público. https://www.boe.es/diario_boe/txt.php?id=DOUE-L-2013-81251
- Ley 19/2013, de 9 de diciembre, de transparencia, acceso a la información pública y buen gobierno. https://www.boe.es/buscar/doc.php?id=BOE-A-2013-12887
- Ley 21/2014, de 4 de noviembre, por la que se modifica el texto refundido de la Ley de Propiedad Intelectual, aprobado por Real Decreto Legislativo 1/1996, de 12 de abril, y la Ley 1/2000, de 7 de enero, de Enjuiciamiento Civil. https://www.boe.es/diario_boe/txt.php?id=BOE-A-2014-11404
- Ley 18/2015, de 9 de julio, por la que se modifica la Ley 37/2007, de 16 de noviembre, sobre reutilización de la información del sector público. https://www.boe.es/diario_boe/txt.php?id=BOE-A-2015-7731

0.1.2 Índices Open Data

El **open data indicies** es un conjunto de indicadores que permiten valorar el grado de transparencia de un portal gubernamental de datos abiertos. Con carácter general, este índice proporciona información relativa a 11 aspectos incluidos en la definición de **open data**. De forma complementaria, el **open data barometer** añade dos índices adicionales.

0.1.3 Repositorios de Open Data

La consolidación del paradigma Open Data está generando un gran crecimiento en el número de repositorios en los que los investigadores pueden tanto publicar sus datos como descargar datasets existentes. Este crecimiento del número de repositorios no ha ido acompañado, sin embargo, de una homogeneización en los procedimientos de búsqueda y localización tanto de repositorios como de datasets.

https://public.opendatasoft.com

In [1]: