Contents

¿Cómo crear libros interactivos con Jupyter Book?

Con las herramientas de Jupyter, podemos crear libros interactivos que combinan texto, HTML y código de programación que los lectores pueden ejecutar y modificar a voluntad.

A continuación vamos a ver un ejemplo de un libro de programación interactivo construido con Jupyter Book utilizando materiales de cursos de Python y Data Science.

Jupyter Book

Jupyter Book es una herramienta open-source para crear libros y documentos PDF de calidad, con contenidos interactivos de código fuente de programación informática.

Jupyter Book permite a los usuarios

- escribir contenidos en ficheros markdown (.md) y Jupyter Notebooks (.ipynb),
- incluir elementos de programación (ej. código de programación) en ambos tipos, markdown y notebooks,
- incluir referencias bibliográficas y ecuaciones matemáticas, y
- usar un comando para ejecutar código embebido, cache el resultado y, finalmente,
- convertir el contenido de los ficheros markdown y notebooks en:
    - un libro interactivo para verlo en un browser (web-based interactive book) y
    - una publicación PDF de calidad.

Algunas de las funcionalidades principales de Jupyter Book son las siguientes:

- ✔ **Escribir contenidos en Markdown**
  El contenido puede estar en Jupyter markdown, o bien en una versión de markdown extendido con funciones de publicación avanzada. Por ejemplo, pueden incluirse fácilmente contenidos en formato editorial como referencias bibliográficas, ecuaciones matemáticas y gráficos.
- ✔ **Añadir código fuente directamente como ficheros Jupyter Notebooks**
  Esto permite incluir tanto el código como el resultado (output) de la ejecución directamente en el libro. Del mismo modo, también pueden escribirse notebooks enteramente en markdown para ejecutar cuando se genera el libro con el comando 'build' como veremos más adelante.
- ✔ **Ejecutar y guardar en memoria cache el contenido del libro**
  Para ficheros Notebooks `.ipynb`, permite ejecutar código e insertar los últimos outputs. Esto permite además, guardar en cache y re-utilizar los outputs posteriormente.
- ✔ **Insertar los outputs en el libro**
  Se pueden insertar outputs a medida que se genera la documentación y añadirlos in-line con el resto de la página.
- ✔ **Añadir interactividad al libro**
  Por ejemplo, se puede alternar la visibilidad de una celda, incluir outputs interactivos desde Jupyter y ejecutar el código conectando con servicios online como Binder, Google Colab y Deepnote.
- ✔ **Generar libros en HTML, PDF y otros formatos**
  Esto incluye single- y multi-page websites, así como PDF.
- ✔ **Un libro se genera con un simple comando**
  Por ejemplo, `jupyter-book build mybook/`

Colabora

> **¡Colabora con Jupyter Book!**
>
> Jupyter Book es una open community que agradece el feedback, input y las muestras de interés.
>
> **Abir una cuestión (issue)**
> para proporcionar feedback y nuevas ideas o comunicar problemas.
> **Votar por nuevas funcionalidades**
> añadiendo 👍 a las cuestiones (issues) que deseas ver resueltas.
> **Contribuir a Jupyter Book**
> siguiendo nuestra guía de participación e identificando una cuestión (issue) para trabajar en su resolución. Ver la tabla de votación de funcionalidades como ejemplo.
> **En español puedes contactar con Publiconsulting Media**
> para comentarios e ideas en español.

Agradecimientos

Jupyter Book cuenta con el soporte de una comunidad abierta, muchos de los cuales han participado en the Jupyter community. Jupyter Book y muchas de las herramientas que utiliza están promovidas por the Executable Book Project, que a su vez cuenta con el apoyo de the Alfred P. Sloan foundation.

1. Instalación

Esta es una guía para la creación de libros con la herramienta Jupyter Book. Para utilizar Jupyter Book, el contenido del libro debe guardarse en ficheros de tipo Markdown y Jupyter Notebooks. `jupyter-book` convierte estos ficheros a formato HTML y PDF. De esta manera, el libro puede leerse en cualquier browser, subirlo a un servicio cloud o server host, así como imprimir y distribuir el fichero PDF generado.

La utilización de Jupyter Book está especialmente indicada para crear libros interactivos que permiten ejecutar código fuente de programación utilizando Jupyter Notebooks. Esto es especialmente recomendable para libros sobre programación, aprendizaje automático (Machine Learning) e Inteligencia Artificial (AI) en general.

1.1. El entorno Anaconda

Aunque no es estrictamente necesario para crear un libro con Jupyter Book, es recomendable tener instalado el entorno Anaconda (o miniconda, una versión reducida que ocupa mucho menos espacio):

- Installing Anaconda on Mac OS X
- Setting up Python on Windows with Miniconda by Anaconda

1.2. Jupyter Book

Para utilizar Jupyter Book, se puede comenzar con

- la guía de utilización getting started guide,
- el menu de navegación de este libro (a la izquierda, en una ordenador de sobremesa), y también
- el libro ejemplo que se explica en los siguientes capítulos.

> **Warning**
>
> Jupyter Book `0.7` es una versión que reescribe completamente la versión `0.6` y tiene numerosos cambios. Ver the legacy upgrade guide para actualizaciones y legacy.jupyterbook.org para ver la documentación de anteriores versiones.
>
> Es importante saber que Jupyter Book todavía está en una versión pre-1.0 y, por lo tanto, previsiblemente irá incorporando cambios importantes en su API en los próximos meses.

Para instalar `jupyter-book` desde pip, se utiliza el siguiente comando:

```
pip install -U jupyter-book
```

> **A note for Windows users**
>
> Jupyter Book is now also tested against Windows OS 😀
>
> However, there is a known incompatibility for notebook execution, when using Python 3.8.
>
> See Working on Windows

Para saber más sobre Jupyter Notebooks esta es la documentación oficial (en inglés)

Sobre cómo crear el entorno de desarrollo para Jupyter Notebook, en español, puede verse Instalar entorno de desarrollo Python Anaconda para Aprendizaje Automático

---

1.3. Recomendación: Jupyter Lab

Jupyter Lab es un entorno que en el futuro próximo sustituirá al clásico Jupyter Notebook (notebook, terminal, text editor, file browser, rich outputs, etc.). Listo para usuarios.

JupyterLab es la nueva propuesta interactiva del Proyecto Jupyter que incorpora una más potente y flexible interfaz de usuario.

JupyterLab permite incorporar extensiones utilizando paquetes npm que usan APIs públicas de Jupyter. Para encontrar extensiones JupyterLab, buscar el término npm en jupyterlab-extension o buscar en GitHub jupyterlab-extension. Para más información sobre extensiones, ver la documentación de usuario.

Ver la última versión de la documentación en ReadTheDocs.

1.3.1. Instalación

JupyterLab puede instalarse usando `conda` o `pip`. Para instrucciones detalladas, consultar la installation guide.

1.3.2. conda

```
conda install -c conda-forge jupyterlab
```

1.3.3. pip

```
pip install jupyterlab
```

1.3.3.1. Instalación con versiones anteriores de Jupyter Notebook

Cuando se usan versions de Jupyter Notebook anteriores a la versión 5.3, después de instalarse JupyterLab debe ejecutarse el siguiente comando:

```
jupyter serverextension enable --py jupyterlab --sys-prefix
```

1.3.4. Ejecutar JupyterLab

Para iniciar JupyterLab se usa el comando:

```
jupyter lab
```

JupyterLab se abrirá automaticamente en el browser. Ver la documentación para mayor información.

Si encuentra un error tipo "Command 'jupyter' not found", puede ser que la variable de entorno `PATH` no tenga la información correcta. Como alternativa, se puede arrancar JupyterLab usando `~/.local/bin/jupyter lab` sin cambiar la variable de entorno `PATH`.

1.3.5. Browsers soportados

Las últimas versiones de los siguientes browsers funcionan con Jupyter Lab:

- Firefox
- Chrome
- Safari

Ver la documentación para más información.

1. Mi primer libro

Una vez que tenemos instalado Anaconda Python, tal y como hemos visto en el capítulo inicial, ya podemos crear nuestro primer libro.

1.1. Libro ejemplo

Para ello, vamos a partir de un breve ejemplo de un libro interactivo creado con Jupyter Book.

Algunas de las funcionalidades incluidas en este libro son:

- Jupyter notebooks
- Referencias bibliográficas
- Ecuaciones matemáticas, numeradas para referenciar
- Figuras, numeradas para referenciar con leyendas explicativas y referencias cruzadas

Los ficheros se encuentran alojados en GitHub en la carpeta docs. El contenido está escrito en MyST markdown, una extensión de Jupyter notebook markdown que permite incluir markup científico. Igualmente, podrían estar escritos en Jupyter notebooks.

Construir el libro ejemplo

Para construir este libro ejemplo, vamos a hacerlo en Terminal, con los comandos siguientes:

1. En primer lugar, vamos a crear un entorno virtual de trabajo donde instalar las librerías necesarias de Python para ejecutar el código fuente de nuestro primer libro. Una forma sencilla de hacerlo es a través de un fichero environment.yml, como veremos a continuación.
2. Clonamos el repositorio de Github que contiene los ficheros del libro ejemplo:

```
git clone https://github.com/executablebooks/quantecon-mini-example
cd quantecon-mini-example
```

3. Instalamos las librerías Python necesarias utilizando este fichero `environment.yml` file. Además, también se instalará la última versión de Jupyter Book. Este es el contenido de nuestro fichero evironment.yml:

Ver la página getting started para más información.

```
name: qe-mini-example
channels:
  - default
dependencies:
  - python=3.7
  - sphinx=2.4.4
  - pip
  - pandas
  - matplotlib
  - pip:
    - quantecon
    - jupyter-book
```

La instalación se lleva a cabo en Terminal, con los comandos:

```
conda env create -f environment.yml
conda activate qe-mini-example
```

De esta forma, conseguimos instalar las dependencias necesarias, entre ellas Python 3.7 y la última versión de Jupyter Book, así como también hemos creado y activado un entorno virtual, llamado aquí qe-mini-example, que nos permite tener un contexto aislado donde crear el libro ejemplo, sin interferencias con otros posibles entornos instalados en nuestro ordenador.

1. Ejecutar Jupyter Book para generar el libro ejemplo

```
jupyter-book build mini_book/
```

2. Ver el resultado en cualquier browser haciendo doble-click en el fichero `html` siguiente:

```
mini_book/_build/html/index.html
```

Ahora, podemos hacer modificaciones a los ficheros fuente que forman el libro ejemplo y que se encuentran en `mini_book/docs`. Para ver el resultado de nuestras modificaciones, se vuelve a generar el libro con el mismo comando `build`:

```
jupyter-book build mini_book/
```

1.1.1. Más información

Ver el libro completo en QuantEcon con todo el contenido de un Jupyter Book con diversos casos de uso.

Para más información sobre el proyecto Jupyter Book, ver toda la documentación en The Executable Book Project.

2. Tipos de contenidos

Existen diversos modos de escribir contenido en Jupyter Book. Este capítulo presenta algunas maneras de hacerlo, tanto en ficheros Jupyter Notebooks (.ipynb) como en ficheros markdown (.md)

2.1. Ficheros Markdown (.md)

Tanto si el contenido del libro se escribe en Jupyter Notebooks (`.ipynb`) como en estándar markdown (`.md`), utilizamos **MyST Markdown**.

2.1.1. ¿Qué es MyST?

MyST significa "Markedly Structured Text". Se trata de una ligera variante de un tipo de markdown llamado "CommonMark", con algunas extensiones que permiten añadir **roles** y **directivas** en el ecosistema de documentación Sphinx.

2.1.2. ¿Qué son roles y directivas?

Roles y directivas son dos potentes herramientas de Jupyter Book similares a las funciones, pero escritas en un lenguaje de markup. Ambas buscan un propósito similar, pero los **roles se escriben en una sola línea** mientras que las **directivas ocupan varias líneas**.

2.1.2.1. Uso de directivas

La forma más simple de insertar una directiva en el contenido de un libro es la siguiente:

````
```{midirectiva}
Contenido de mi directiva
```
````

Esto sólo funciona si existiera una directiva de nombre `midirectiva` (que no existe, por supuesto). Existen muchas directivas predefinidas asociadas con Jupyter Book. Por ejemplo, para insertar una caja con una nota, se puede hacer con la directiva siguiente:

````
```{note}
Aquí hay una nota
```
````

Que se visualiza como:

> **Note**
>
> Aquí hay una nota

Cuando se genera el lbro con el comando `jupyterbook build`.

Para moás información sobre directivas, ver la documentación sobre MyST.

2.1.2.2. Uso de roles

Los roles son similares a las directivas, pero se escriben en una sola línea. Para insertar un rol en un libro se hace siguiendo este formato:

```
Algún texto {rolename}`Este es mi texto!`
```

Como con las directivas, los roles sólo funcionan si el nombre `rolename` es válido. Por ejemplo, un nombre válido es `doc` y se usa para hacer referencia a otra página del libro. Así, puede utilizarse el encaminamiento relativo a la página de introducción, mediante la sintaxis `{doc}`../chapter0/intro`` que se visualizará como este texto con enlace a dicha página: ¿Cómo crear libros interactivos con Jupyter Book?.

Para más información sobre roles, ver la documentación sobre MyST.

2.1.2.3. Uso de referencias bibliográficas

Se pueden citar referencias que estén almacenadas en un fichero `bibtex`. Por ejemplo, la sintaxis siguiente: `{cite}`holdgraf_evidence_2014`` se visualizará como: [HdHPK14].

Para insertar la bibliografía se utiliza la directiva `{bibliography}`. Esta directiva debe ser utilizada para todos los roles `{cite}` para su correcta visualización. En el libro ejemplo las referencias están almacenadas en el fichero `references.bib` y la bibliografía se incluye con:

````
```{bibliography} references.bib
```
````

Que resulta en la siguiente visualización:

> **[HdHPK14]**
> Christopher Ramsay Holdgraf, Wendy de Heer, Brian N. Pasley, and Robert T. Knight. Evidence for Predictive Coding in Human Auditory Cortex. In *International Conference on Cognitive Neuroscience*. Brisbane, Australia, Australia, 2014. Frontiers in Neuroscience.

2.1.2.4. Ejecución de código en un fichero markdown

Jupyter Book usa *jupytext* para ejecutar código de programación en ficheros markdown. Para más información, ver la documentación Jupytext

En primer lugar, se añaden Jupytext metadata al fichero markdown que contenga el código que queramos ajecutar. Por ejemplo, para añadir Jupytext metadata a esta página `markdown.md`, utilizando un kernel python3, hay que ejecutar el comando:

```
jupyter-book myst init docs/chapter2/markdown.md --kernel python3
```

que genera el header siguiente en el fichero `markdown.md`:

```
jupytext:
  cell_metadata_filter: -all
  formats: md:myst
  text_representation:
    extension: .md
    format_name: myst
    format_version: 0.12
    jupytext_version: 1.6.0
kernelspec:
  display_name: Python 3
  language: python
  name: python3
```

A continuación se añade la directiva `{code-cell}` con el código que queramos ejecutar:

````
```{code-cell}
a = "Este es un texto"
b = "en código Python"
print(f"{a} {b}")
```
````

Cuando se genera el libro con el comando `jupyterbook build`, el contenido del bloque `{code-cell}` se ejecutará en el kernel Jupyter (en este caso python3) y el output se visualizará junto con el resto del libro.

```
a = "Este es un texto"
b = "en código Python"
print(f"{a} {b}")
```

```
Este es un texto en código Python
```

Para más información sobre ejecución de código en ficheros markdown, ver la documentación MyST-NB.

2.2. Jupyter Notebooks (.ipynb)

Los ficheros Jupyter Notebooks (.ipynb) también pueden añadirse a nuestro libro, permitiendo la visualización y ejecución de bloques de código y sus outputs.

2.2.1. Celdas markdown

Permiten insertar imágenes, video, código HTML, etc. Ejemplo, esta imagen:

También podemos insertar add_{math} y

$$math^{blocks}$$

o también

$$\text{mean} la_{tex}$$

$$mathblocks$$

2.2.2. Celdas MyST markdown

Las celdas MyST markdown funcionan en Jupyter Notebooks de modo similar. Para más información sobre MyST markdown, ver the MyST guide in Jupyter Book, y the MyST markdown documentation.

2.2.3. Celdas de código y outputs

Jupyter Book permite la visualización de los bloques de código y outputs de los ficheros .ipynb. Por ejemplo, esta es una muestra de código que utiliza Matplotlib:

```
from matplotlib import rcParams, cycler
import matplotlib.pyplot as plt
import numpy as np
plt.ion()
```

```
# Fixing random state for reproducibility
np.random.seed(19680801)

N = 10
data = [np.logspace(0, 1, 100) + np.random.randn(100) + ii for ii in range(N)]
data = np.array(data).T
cmap = plt.cm.coolwarm
rcParams['axes.prop_cycle'] = cycler(color=cmap(np.linspace(0, 1, N)))


from matplotlib.lines import Line2D
custom_lines = [Line2D([0], [0], color=cmap(0.), lw=4),
                Line2D([0], [0], color=cmap(.5), lw=4),
                Line2D([0], [0], color=cmap(1.), lw=4)]

fig, ax = plt.subplots(figsize=(10, 5))
lines = ax.plot(data)
ax.legend(custom_lines, ['Cold', 'Medium', 'Hot']);
```

 _build/jupyter_execute/docs/chapter0/notebooks_2_0.png

Las posibilidades son muy diversas e incluyen el uso de outputs que permiten la interactividad en la lectura del libro
Para más información, ver the Jupyter Book documentation

3. Publicar el libro online

Una vez generado nuestro libro, podemos subirlo a la nube, alojarlo en su servidor web y hacer que esté disponible para su lectura online. Puesto que el libro que hemos creado es, técnicamente, una página web estática, la mejor manera de lograrlo es a través de un servicio de alojamiento de **páginas web estáticas**. Hay muchas maneras de hacer esto y la siguientes secciones repasan las más habituales.

3.1. Crear un repositorio online para nuestro libro

Vamos a utilizar Github para crear nuestro repositorio online y subir el contenido del libro que hemos creado.

1. Primero, hacer log-in en nuestro GitHub y crear un nuevo repositorio en la página “create a new repository”: https://github.com/new
2. Segundo, añadir nombre y descripción de nuestro repositorio online. Asegurarse de que el repositorio es ‘public’ y no seleccionar la inicialización con el fichero README. Finalmente, hacer click en “Create repository”.
3. Tercero, clonar el repositorio que acabamos de crear, y que está vacío, en nuestro ordenador:

```
git clone https://github.com/<my-github-name>/<my-repository-name>
```

4. Cuarto, copiar todos los ficheros y carpetas del libro a la carpeta local recién clonada. Por ejemplo, si el libro fue generado con `jupyter-book create mylocalbook` y el repositorio se llama `myonlinebook`, el comando a ejecutar sería:

```
cp -r mylocalbook/* myonlinebook/
```

5. Quinto, sincronizar el repositorio local y el repositorio remoto. Para ello es necesario ejecutar los siguientes comandos:

```
cd myonlinebook
git add ./*
git commit -m "subiendo mi libro ejemplo"
git push
```

Ver más información sobre cómo utilizar varios servicios de alojamiento online