Kalman and Bayesian Filters in Python

Roger R Labbe Jr

February 14, 2017

Contents

Preface

Introductory textbook for Reproducible Research in Python. The book has been written using Jupyter Notebook so you may choose between reading the book in pdf format or in your browser and also run and change the code. What a terrific way to learn?

The style format of this book is based on the fantastic book **Kalman and Bayesian Filters in Python** by Roger R Labbe Jr. Thank you very much Roger for sharing your book and also your source files. I will do the same.

0.1 Motivation for this Book

I'm a Researcher and also a Lecturer at the University of Cantabria (Spain) trying to apply the concepts of Reproducible Research.

There are some excellent references in the field, such as

This book is interactive and I encourage you to use it in that way. It has been written using Jupyter Notebook.

This book is free in both freedom and free beer. The book and source code is hosted on free servers at GitHub, and it uses only free and open software such as IPython and MathJax.

0.2 Reading Online

GitHub

The book is hosted on GitHub, and you can read any chapter by clicking on its name. GitHub statically renders Jupyter Notebooks. You will not be able to run or alter the code, but you can read all of the content.

The GitHub pages for this project are at

`https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`

binder

binder serves interactive notebooks online, so you can run the code and change the code within your browser without downloading the book or installing Jupyter. Use this link to access the book via binder:

`http://mybinder.org/repo/rlabbe/Kalman-and-Bayesian-Filters-in-Python`

nbviewer

The nbviewer website will render any Notebook in a static format. I find it does a slightly better job than the GitHub renderer, but it is slighty harder to use. It accesses GitHub directly; whatever I have checked into GitHub will be rendered by nbviewer.

You may access this book via nbviewer here:

http://nbviewer.ipython.org/github/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/table_of_contents.ipynb

0.3 PDF Version

The PDF version of the book can be downloaded from:

https://drive.google.com/file/d/0By_SW19c1BfhSVFzNHc0SjduNzg/view?usp=sharing

0.4 Downloading the book

This book is interactive and I recommend using it in that form. If you install IPython on your computer and then clone this book you will be able to run all of the code in the book inside your browser. You can perform experiments, see how filters react to different data, see how different filters react to the same data, and so on. I find this sort of immediate feedback vital. You do not have to wonder "what happens if". Try it and see!

The GitHub pages for this project are at

```
https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
```

You can clone it to your hard drive with the command

```
git clone --depth=1 https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.git
```

This will create a directory named `Kalman-and-Bayesian-Filters-in-Python`. The `depth` parameter just gets you the latest version. Unless you need to see my entire commit history this is a lot faster and saves space. Navigate to the directory, and run Jupyter Notebook with the command

```
jupyter notebook
```

This will open a browser window showing the contents of the base directory. The book is organized into chapters. Each chapter is named *xx-name*.ipynb, where *xx* is the chapter number. .ipynb is the Notebook file extension.

Admittedly this is a cumbersome interface to a book. I am following in the footsteps of several other projects that are re-purposing Jupyter Notebook to generate entire books. I feel the slight annoyances have a huge payoff - instead of having to download a separate code base and run it in an IDE while you try to read a book, all of the code and text is in one place. If you want to alter the code, you may do so and immediately see the effects of your change. If you find a bug, you can make a fix, and push it back to my repository so that everyone in the world benefits. And, of course, you will never encounter a problem I face all the time with traditional books - the book and the code are out of sync with each other, and you are left scratching your head as to which source to trust.

0.5 Installation and Software Requirements

Installation of the required software is described in detail in the Installation appendix, which you can read online here:

http://nbviewer.ipython.org/github/rlabbe/Kalman-and-Bayesian-Filters-in-Python/blob/master/Appendix_A_Installation.ipynb

You will need IPython 3.0 or greater, Python 2.7+ or Python 3.4+, NumPy, SciPy, SymPy, and Matplotlib. You will also need my open source library FilterPy.

The easiest way to get all of those except for FilterPy is by installing a free Scientific Python distribution, such as the Anaconda distribution https://store.continuum.io/cshop/anaconda/ (the link says "shop", but it is free). I am not naming them out of favoritism, I am merely documenting my environment. Should you have trouble running any of the code, perhaps knowing this will help you.

Once you have Python installed install FilterPy by typing

```
pip install filterpy
```

at the command line.

0.6 My Libraries and Modules

I am writing an open source Bayesian filtering Python library called FilterPy. Installation instructions are given above.

FilterPy is hosted GitHub at (https://github.com/rlabbe/filterpy) but the `pip` installed version should serve your needs.

Code that is specific to the book is stored with the book in the subdirectory **/code**. It contains code for formatting the book. It also contains python files with names like *xxx*_internal.py. I use these to store functions that are useful for a specific chapter. This allows me to hide Python code that is not particularly interesting to read - I may be generating a plot or chart, and I want you to focus on the contents of the chart, not the mechanics of how I generate that chart with Python. If you are curious as to the mechanics of that, just go and browse the source.

Some chapters introduce functions that are useful for the rest of the book. Those functions are initially defined within the Notebook itself, but the code is also stored in a Python file in **/code** that is imported if needed in later chapters. I do document when I do this where the function is first defined, but this is still a work in progress. I try to avoid this because then I always face the issue of code in the directory becoming out of sync with the code in the book. However, Jupyter Notebook does not give us a way to refer to code cells in other notebooks, so this is the only mechanism I know of to share functionality across notebooks.

There is an undocumented directory called **/experiments**. This is where I write and test code prior to putting it in the book. There is some interesting stuff in there, and feel free to look at it. As the book evolves I plan to create examples and projects, and a lot of this material will end up there. Small experiments will eventually just be deleted. If you are just interested in reading the book you can safely ignore this directory.

The directory **/code** contains a css file containing the style guide for the book. The default look and feel of Jupyter Notebook is rather plain. I have followed the examples set by books such as Probabilistic Programming and Bayesian Methods for Hackers [2]. I have also been very influenced by Professor Lorena Barba's fantastic work, available here [3]. I owe all of my look and feel to the work of these projects.

0.7 License

0.8 Contact

mananam@unican.es

0.9 Resources

- [1]

Chapter 1

Introduction

```
In [1]: import time

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

Ultima version: 09/10/2016 00:39:39
```

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?

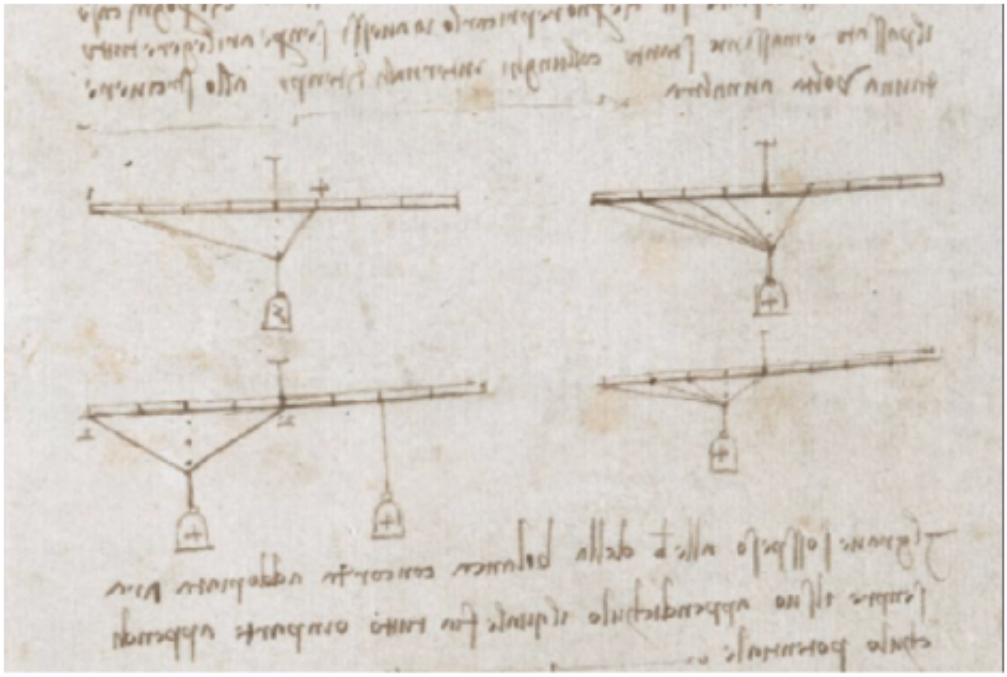
Comenzaremos por analizar el método clásico para ir resolviendo poco a poco los desafíos planteados...

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.

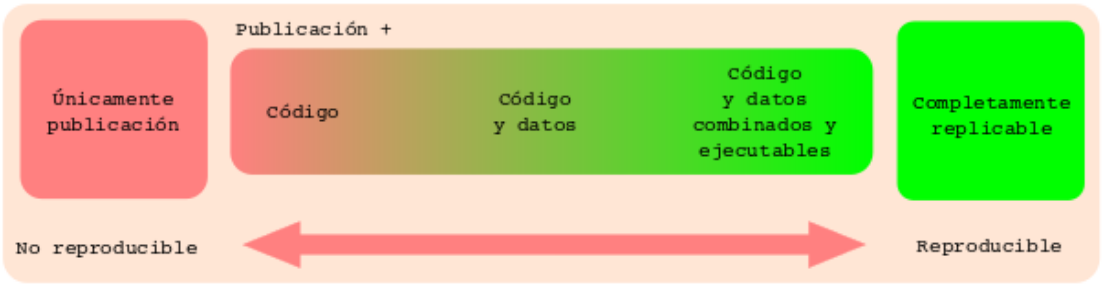
- Transparencia. La disponibilidad pública de datos y procedimientos permite mejorar los índices de citas de autores y sus instituciones.
- Eficiencia. Facilidad para reutilizar datos y procedimientos.
- Flexibilidad. Capacidad para hacer cambios sobre los procedimientos descritos.

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')
```

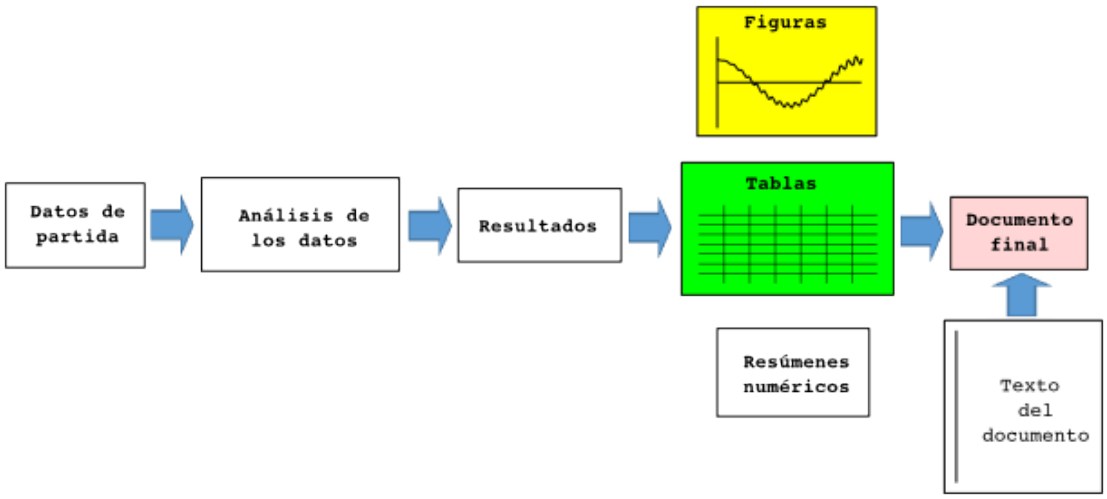
```
Out[3]:
```



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

```
In [5]: Image(filename='./figuras/metodologia_nr.png')
```

```
Out[5]:
```


Figuras
Datos de partida
Análisis de los datos
Resultados
Tablas
Documento final
Resúmenes numéricos
Texto del documento

In [ ]: