UNIVERSIDAD DE GUANAJUATO

Cadenas de Markov con método de Montecarlo aplicado a restricciones cosmológicas.

División de ciencias e ingenierías - Astrofísica computacional

Gabriel Missael Barco

15 de Junio del 2020

1. Cadena de Markov.

No siempre el conjunto de parámetros que mejor se ajusta a mis datos es el que es mas probable para mi modelo, es decir, el mejor ajuste podría tener parámetros que son casi imposibles. La respuesta a ambas preguntas (parámetros con mejor ajuste o mas probables) muchas veces es la misma, pero esto no siempre es así.

Para encontrar los valores mas probables, puedo tomar el conjunto de valores que mas se aproximan conjuntamente al mínimo de manera recurrente en mi minimización.

Por ahora consideremos el **likelihood**, que es una construcción que realizamos para establecer la probabilidad de que nuestra hipótesis sobre un modelo o teoría sea cierta dada que estamos observando unos datos determinados. Es la probabilidad de que el modelo sea el que describa a los datos.

No tiene una forma única, depende del tipo de datos que estoy manejando, es decir, que tipo de distribución de probabilidad tienen estos. Por ejemplo, si los datos son gaussianos, entonces podemos pensar que el likelihood también es gaussiano. Tenemos, en general que:

$$-ln(L(\vec{x},\vec{y}|\vec{\theta}) \propto \frac{1}{2}\sum_i (\frac{(y_i - \lambda(x_i,\vec{\theta}))^2}{\sigma_i^2}) \tag{1}$$

Es útil expresarlo de esta manera para eliminar el exponencial y tener datos mas 'digeribles'. Pero ¿qué pasa si queremos maximizar un likelihood de muchos parámetros para un modelo bastante complejo y si además no es gaussiano? Para ello nos sirve el **método Montecarlo**.

Este método consiste en generar un muestreo aleatorio para los parámetros (por ejemplo, en la ecuación anterior, las $\vec{\theta}$ son los parámetros libres que podemos muestrear), y después los rechaza o los acepta dependiendo del likelihood que estos datos tengan. Esta discriminación se hace de acuerdo a likelihood anteriores, y en general, el algoritmo es:

1. Elige un punto de partida $\vec{p}_{old}$.
2. Genera un punto nuevo en la vecindad de $\vec{p}_{old}$, dado una distribución gaussiana $p_{new}i = N(p_{old}i, \sigma_i)$. Es importante notar que cada parámetro puede tener un sigma distinto.

3. Si el likelihood de la nueva muestra es mas alta que la anterior, guardamos el nuevo, esto es: si $L(\vec{p}_{new}) > L(\vec{p}_{old})$ entonces $\vec{L}_{old} = L_{new}$

4. Si el likelihood de la nueva muestra es menor a la anterior, entonces tomamos un numero aleatorio $U = N(0, 1)$; si el cociente de los likelihood $C = (p_{new})/(p_{old})$ es mas grande que dicho numero aleatorio, entonces la aceptamos.

5. Si no cumple ninguna de las dos, entonces lo rechazamos.

6. Regresa al paso 2, hasta dar N pasos.

Después de haber realizado los N pasos, vemos la distribución resultante (las cadenas) de los parámetros, y por lo tanto, su likelihood.

Podemos aplicar este algoritmo a un modelo de una recta. Conocemos la ordenada al origen b y la pendiente m. Construimos nuestro modelo de recta y luego aplicamos el algoritmo a este modelo, y podemos observar como la cadena *camina* hacia la zona de máxima probabilidad, que ene este caso coincide con los valores m y b del modelo de la recta. Podemos observar un ejemplo de esto en la Figura 1, donde $m = 12$ y $b = 10$, con 500 pasos.

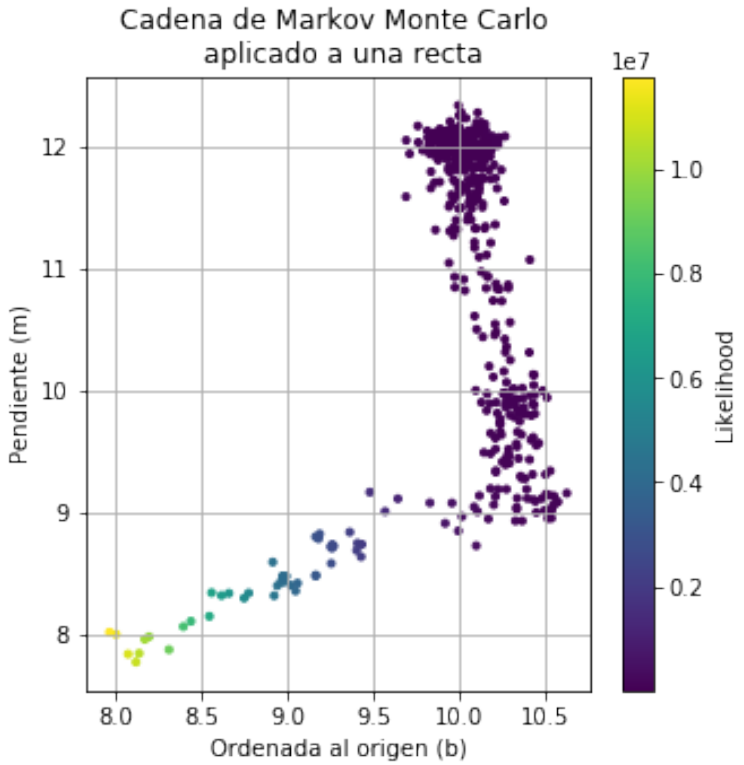

Figura 1: Ejemplo de una cadena de Markov aplicado al modelo de una recta.

Puede encontrarse el código de esta primera aproximación en `https://github.com/DCIDA2019/da2020-GabrielMissael/blob/master/Semana08_Casa/Clase_remota1.ipynb`. Podemos observar como la cadena va avanzando en cada paso hacia la zona de máxima probabilidad, y una vez que llega a esa zona, se queda ahí, explorando únicamente regiones alrededor de esa zona.

2. Múltiples cadenas de Markov.

Podemos pensar en una cadena MCMC como un descenso con el gradiente; con esto en mente, es fácil darse cuenta de un posible problema: terminar en un área de máxima probabilidad **local**,

lo cual no es buen resultado. Para arreglar esto, es necesario tener cuidado con el tamaño del paso del caminador de las cadenas (es decir, cual es la distancia máxima que puede recorrer la cadena en cada parámetro en cada paso); si es muy grande no se obtienen resultados detallados, y si es muy pequeño, no se explora todo el espacio de parámetros de manera adecuada.

Además de eso, lo que se suele hacer es generar **múltiples** cadenas, que en principio deberían llegar al mismo lugar del espacio de parámetros; en ese caso, podemos decir (tentativamente) que las cadenas han **convergido**, aunque hay que ser cuidadosos con ese detalle (se verá mas adelante los criterios de convergencia). Aplicando múltiples cadenas MCMC al un modelo de una recta (Figura 2) podemos observar como todas las cadenas avanzan hasta llegar a la misma zona de alta probabilidad.

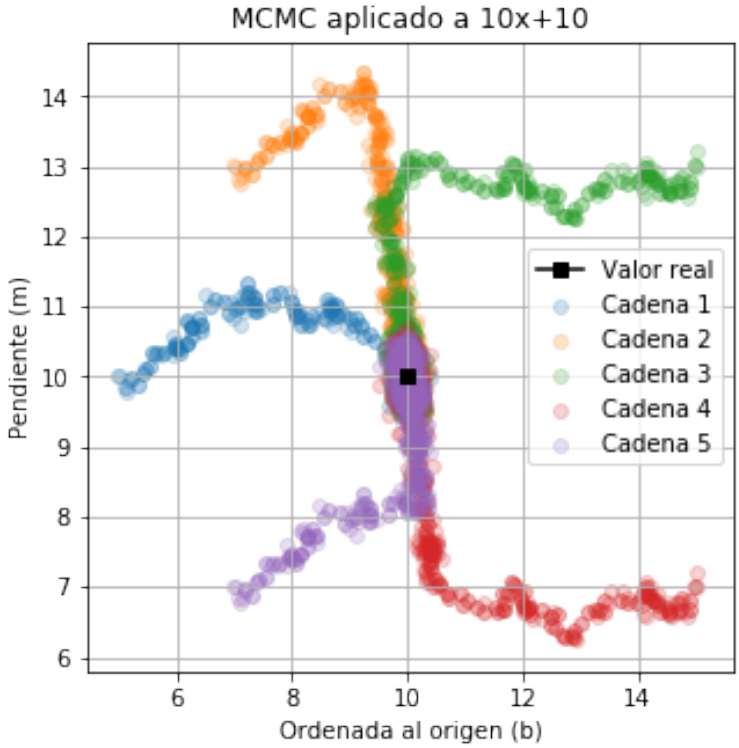

Figura 2: Ejemplo de múltiples cadenas aplicadas a una recta

El código utilizado para generar estas cadenas y esta gráfica se encuentra en este notebook en github: `https://github.com/DCIDA2019/da2020-GabrielMissael/blob/master/Semana09_Casa/Actividad_semana9.ipynb`.

3. Justiciación con Teorema de Bayes

Como ya mencionamos, en los ejemplos anteriores, con una o múltiples cadenas utilizamos el likelihood, pero esto únicamente como una primera aproximación, en realidad se utiliza algo mas, no solamente el likelihood.

Primeramente recordemos el Teorema de Bayes:

$$P(\vec{\theta}|\vec{x}) = \frac{P(\vec{x}|\vec{\theta})P(\vec{\theta})}{P(\vec{x})} \tag{2}$$

En esta expresión $P(\vec{\theta}|\vec{x})$ es el **posterior**, o bien, la probabilidad de que nuestros parámetros $\vec{\theta}$ sean correctos dado un conjunto de datos $\vec{x}$. Por otro lado, $P(\vec{\theta})$ es el **prior**, o bien, lo que pensamos a priori acerca de los parámetros antes de haber visto cualquier conjunto de

datos. Finalmente, $P(\vec{x}|\vec{\theta})$ es el **likelihood**, que es como pensamos que nuestros datos están distribuidos. (Esto es lo que habíamos estado usando)

Todo bien hasta aquí, el problema es $P(\vec{x})$, que también se conoce como la *evidencia* de que esos datos sean generados por nuestro modelo. En general, se obtiene cómo:

$$P(\vec{x}) = \int_{\theta} P(x, \theta) d\theta \tag{3}$$

Realizar esta integral es realmente difícil, inclusive para problemas no tan complejos, generar las cadenas MCMC calculando esta $P(\vec{x})$ tiente un costo computacional extremadamente alto, ya que lo que nos interesa calcular a cada paso es el **posterior**. La clave de este algoritmo es que no se calcula $P(\vec{x})$ en ningún punto.

El posterior es utilizado en dos partes del algoritmo. Primero al comparar el posterior de dos puntos directamente; dado que ambos posteriors están divididos por el mismo $P(\vec{x})$, podemos 'omitirlo' y compáralos directamente. La segunda ocasión cuando se determina un cociente entre los posterior, que luce como:

$$C = \frac{\frac{P(\vec{x}|\vec{\theta})P(\vec{\theta})}{P(\vec{x})}}{\frac{P(\vec{x_0}|\vec{\theta})P(\vec{\theta})}{P(\vec{x})}} = \frac{P(\vec{x}|\vec{\theta})P(\vec{\theta})}{P(\vec{x_0}|\vec{\theta})P(\vec{\theta})} \tag{4}$$

Donde podemos observar que el termino problemático $P(\vec{x})$ desaparece, y por lo tanto podemos realizar las cadenas MCMC utilizando el posterior de manera indirecta.

4. Criterios de convergencia y otras consideraciones.

Primero tenemos la fracción de aceptación, que es la razón entre los puntos aceptados y los rechazados en la cadena. Si es muy pequeño, se esta avanzando con mucha dificultad (i.e no converge); en cambio si es muy alta, se esta recorriendo el espacio de parámetros de manera casi uniforme. Se considera apropiado que esa fracción de aceptación esté entre $[0{,}2, 0{,}5]$

También es recomendable corre cadenas largas, para obtener una mejor nube de probabilidad y asegurar convergencia, aunque en realidad el numero de pasos depende totalmente del tipo de problema que estemos atacando, por ejemplo, del numero de parámetros y sus características.

Después de tener la cadena, es necesario cortar los pasos iniciales hasta que se llegue a la zona de alta probabilidad, lo que se denomina como **burn in**, que son los pasos en los que la cadena se dirige hacia la zona de alta probabilidad.

Existen varias condiciones cuantitativas para determinar si una cadena ha convergido o no. Un método cuantitativo para conocer si el método a convergido es el **diagnostico Gelman-Rubin**, el cual implementamos en el ejemplo de la recta y mas adelante en la implementación con supernovas.

Es importante señalar que, en teoría, si el código esta bien programado y las σ de los datos están elegidas de manera apropiada, teóricamente las cadenas siempre convergen en un tiempo infinito.

Una vez que tenemos la cadena generada y una vez que retiramos el burn in de esta, para elegir que valor reportar, debemos tomar la media y tomar la desviación estándar. Esto es idealmente, en caso de obtener una distribución gaussiana. En caso contrario, depende de que resultado obtengamos; esta decisiónn' la podemos apoyar graficando todos los histogramas 2d y 1d de todos los parámetros para visualizar que decisiones serian óptimas.

5. Métodos empíricos y visualización.

Primeramente, para remover el burn in de las cadenas, es necesario observar el momento en el que todos los parámetros varían alrededor de un centro (la media) con una distancia máxima (caracterizada por la desviación estándar). Esto debe realizarse de manera manual, o bien, poner el corte después de un número considerable de pasos para asegurarse que siempre se retiran estos puntos indeseados.

Para ayudar a decidir el corte del burn in de las cadenas, se sugiere gráficar los parámetros vs los pasos. Por ejemplo, en la Figura 3 se puede observar, una vez mas en el ejemplo de la recta, cuando ya se removió el burn in de manera adecuada en esta cadena en especifico.

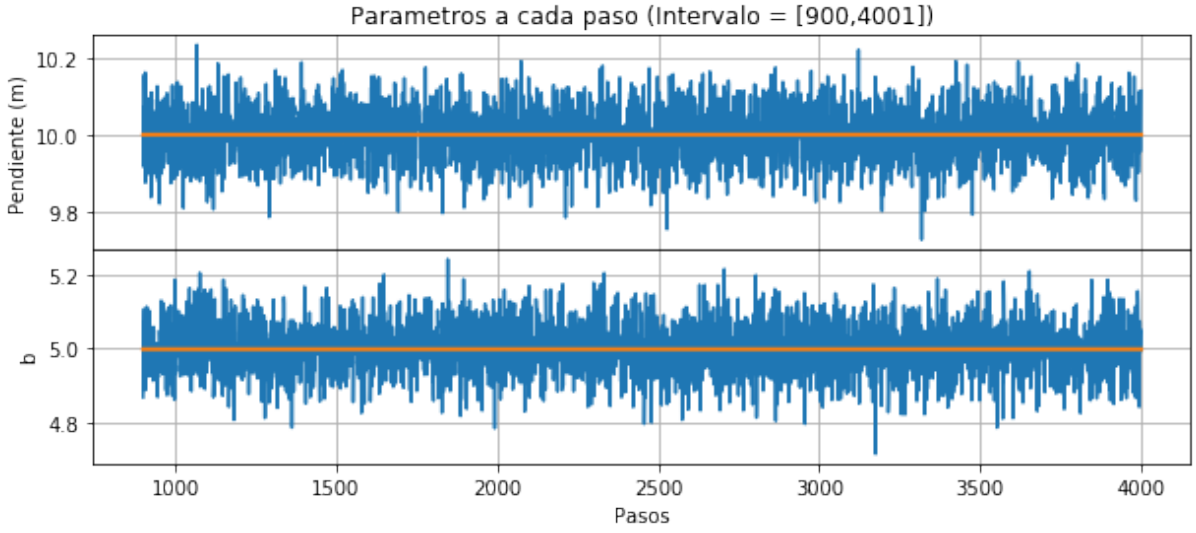

Figura 3: Observar como varían los parámetros ayuda a saber si el algoritmo ha convergido.

Una vez hecho esto, podemos realizar un histograma 2d de los puntos para cada par de parámetros, por ejemplo, en el caso de la recta, se obtuvo el histograma de la Figura 4.

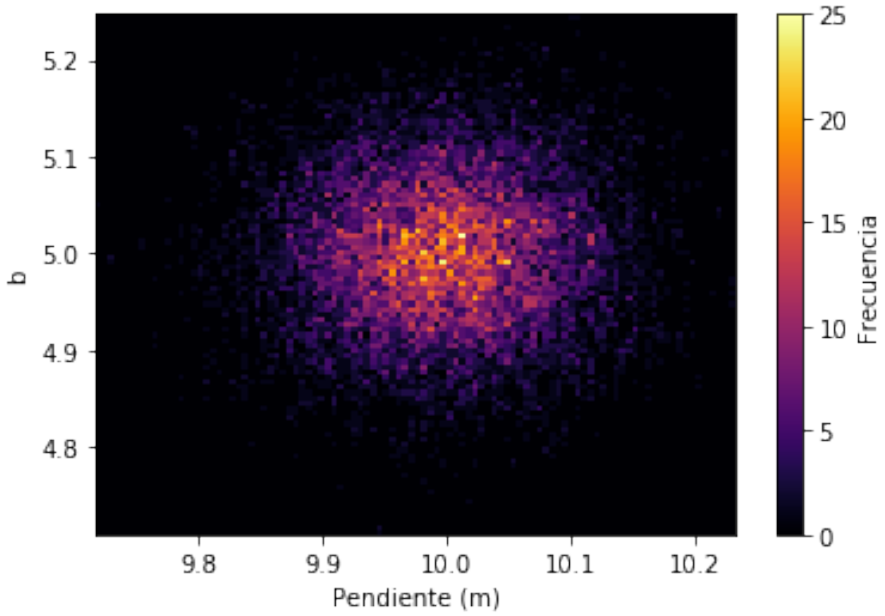

Figura 4: Histograma 2-dimensional entre dos parámetros.

Estos histogramas 2d nos sirven para observar la interdependencia de unos parámetros con

otros. En general, esperamos observar círculos o elipses, estas ultimas inclinadas positiva o negativamente. Estos histogramas 2d dependen directamente de los histogramas 1d, o mas bien, de las distribuciones obtenidas con las cadenas MCMC de cada uno de los parámetros. Si algún parámetro tiene un distribución errática no definida, no se podrá observar, en general, su dependencia con los otros parámetros. Dado que existe un histograma 2d para cada par de parámetros, se suelen graficar los resultados con una gráfica triangular. Para al caso de la recta, esto se observa en la Figura 5.

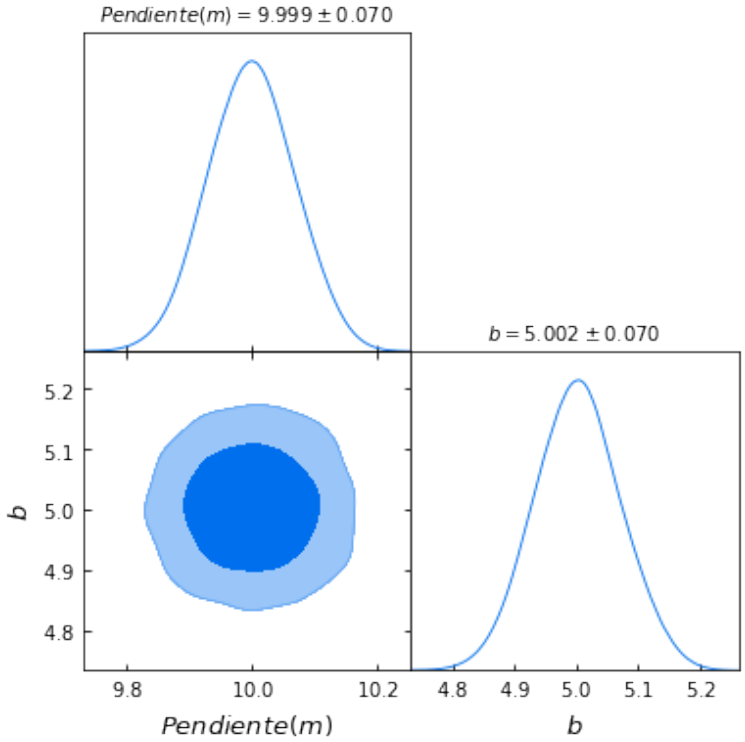

Figura 5: Gráfica triangular de dos parámetros.

En este tipo de gráficas, se muestran las distribuciones obtenidas de cada parámetro, así como un histograma 2d de cada para de parámetros.

El código fuente de estas gráficas y de los criterios de convergencia mencionados anteriormente se encuentra en `https://github.com/DCIDA2019/da2020-GabrielMissael/tree/master/Semana10_casa`

6. Implementación de MCMC a supernovas.

Una vez realizado y depurado el código utilizando el ejemplo de la recta, se procedió a implementar esto a observaciones de supernovas obtenidas por las colaboraciones SDSS-II y SNLS [1], para obtener restricciones cosmológicas del modelo Lamda-CMD. Para ello, primeramente fue necesario reproducir dicho modelo.

6.1. Observaciones y modelo de supernovas.

El modelo a reproducir es el que se observa en la Figura 6.

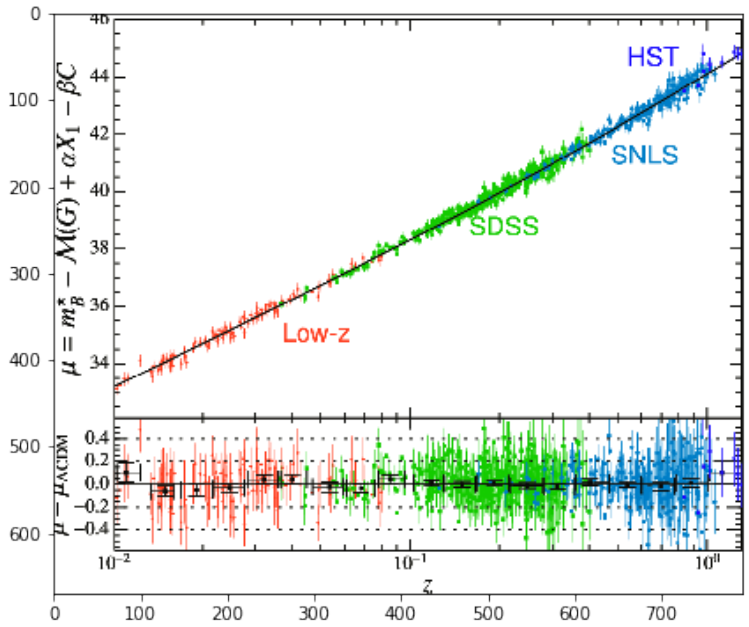

Figura 6: Datos y modelo originales

Para ello, obtuvimos los datos de las supernovas utilizados en dicho modelo [1], y posteriormente se utilizaron las librerías **Cosmolopy** para obtener la distancia luminosa de las galaxias y **Scipy** para ajustat el modelo a los datos proporcionados. Se obtuvo algo muy similar al original, véase Figura 7. Para esto, se considera el modelo:

$$\mu = m_B * -(M_B - \alpha \times X_1 + \beta \times C) \tag{5}$$

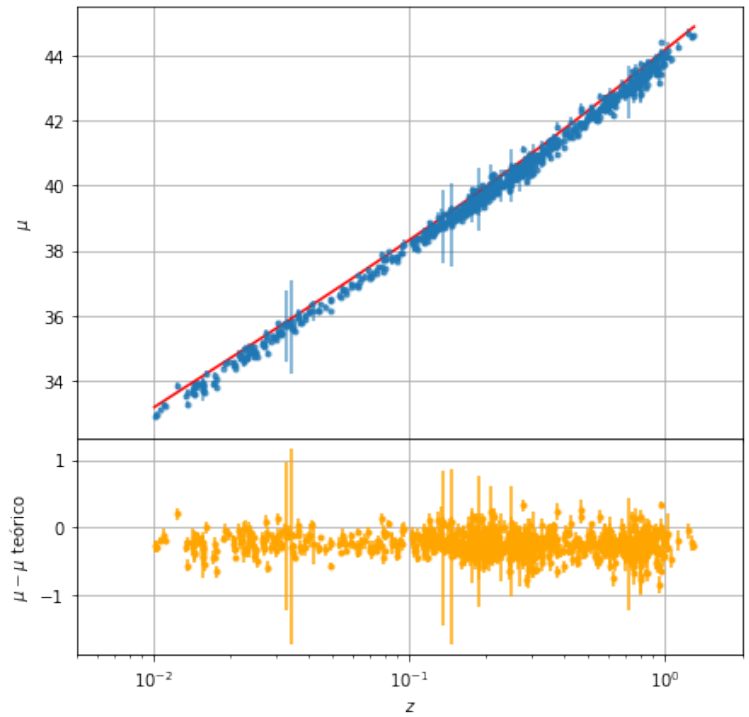

Figura 7: Datos y replicación de modelo con Scipy

El código de esto ultimo puede encontrarse en `https://github.com/DCIDA2019/da2020-GabrielMissa/blob/master/Semana07/Actividades_Clase.ipynb`.

6.2. Resultados

Una vez listo esto, se aplicaron las cadenas MCMC al modelo para obtener los parámetros que mejor se ajustan y además son mas probables a las observaciones de supernovas y su corrimiento al rojo mencionadas. En particular, se usaron 8 cadenas, cada una de ellas con 10,000 pasos. El inicio de las cadenas fue obtenido de manera aleatoria alrededor de los parámetros obtenidos con el ajuste usando **Scipy** mencionado anteriormente.

Ya no es posible visualizar las cadenas completas, ya que viven en el número de dimensiones igual al número de parámetros; sin embargo, es posible proyectarlas en solo dos parámetros, como se puede observar en la Figura 8

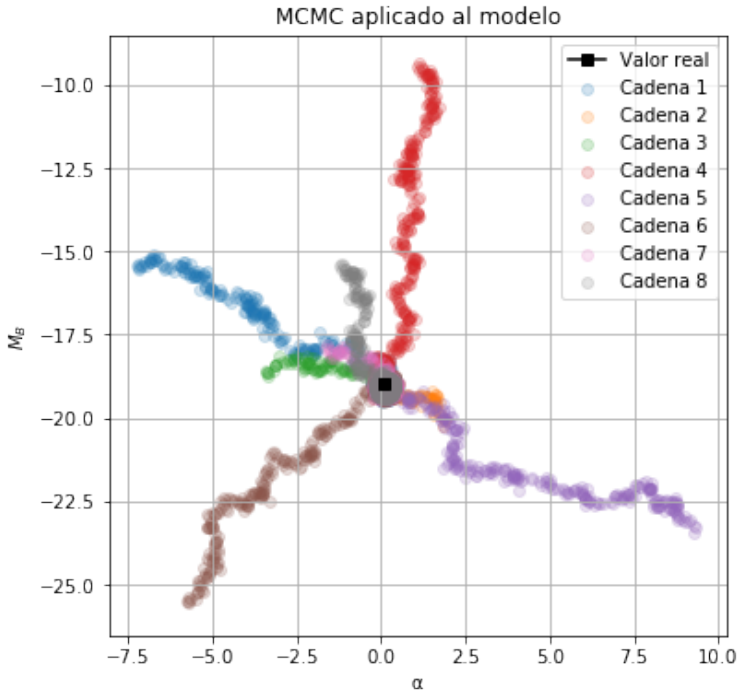

Figura 8: Visualización de las cadenas con dos de los parámetros.

Se eligió el corte del burn in en el paso N=2000 en todas las cadenas, esto después de haber observado la variación de los parámetros en cada una de ellas a lo largo de los 10000 pasos. Para esto, es visualizo la variación como se observa en la Figura 9 como ejemplo.

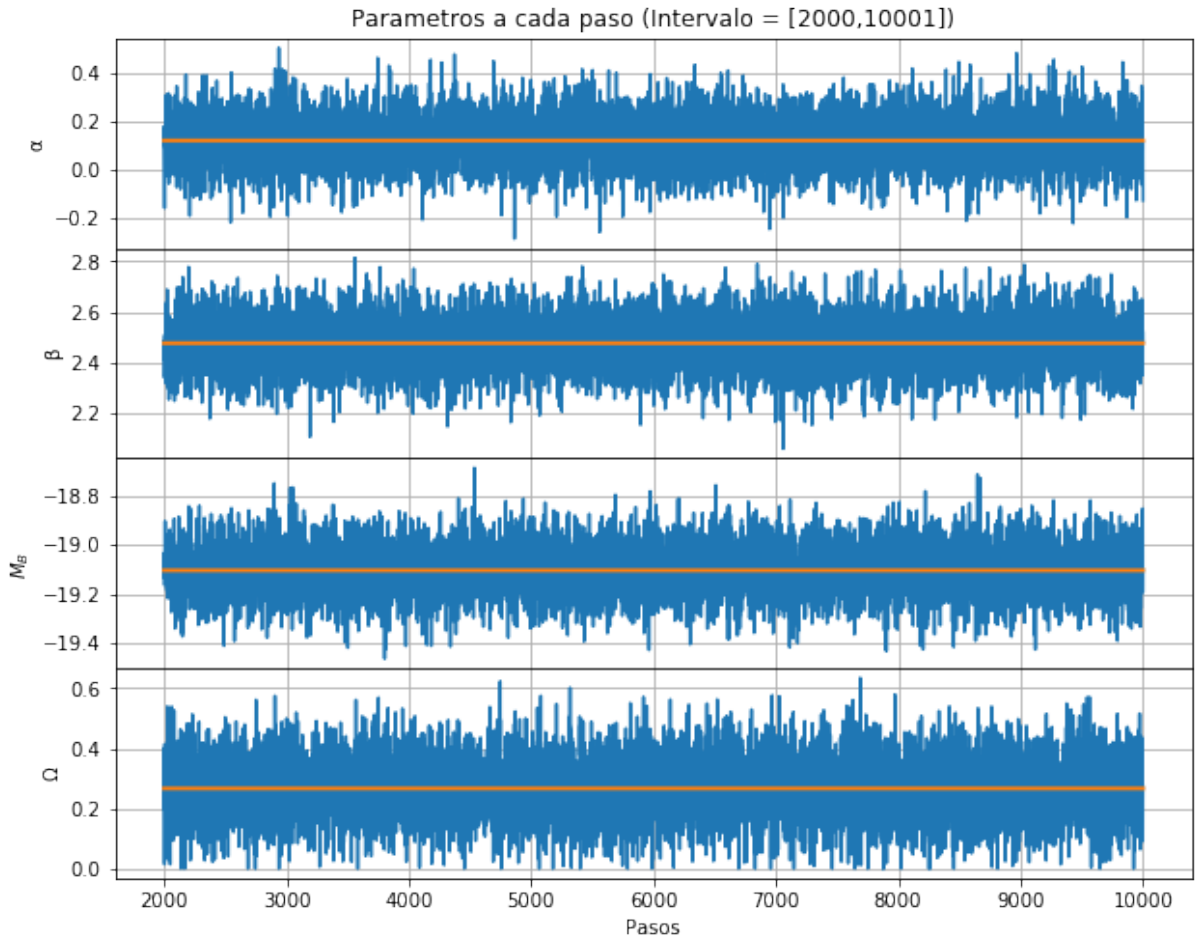

Figura 9: Variación de parámetros a cada paso (Sin Burn In)

Finalmente, se unieron las 8 cadenas y se realizó la gráfica triangular que se puede observar en la Figura 10.

En resumen, los valores de los parámetros obtenidos son:

Parámetro	Valor	Desviación estándar
α	0.1213	$\pm(0{,}002786, 0{,}002739)$

Tabla 1: Valores de los parámetros obtenidos con las cadenas MCMC

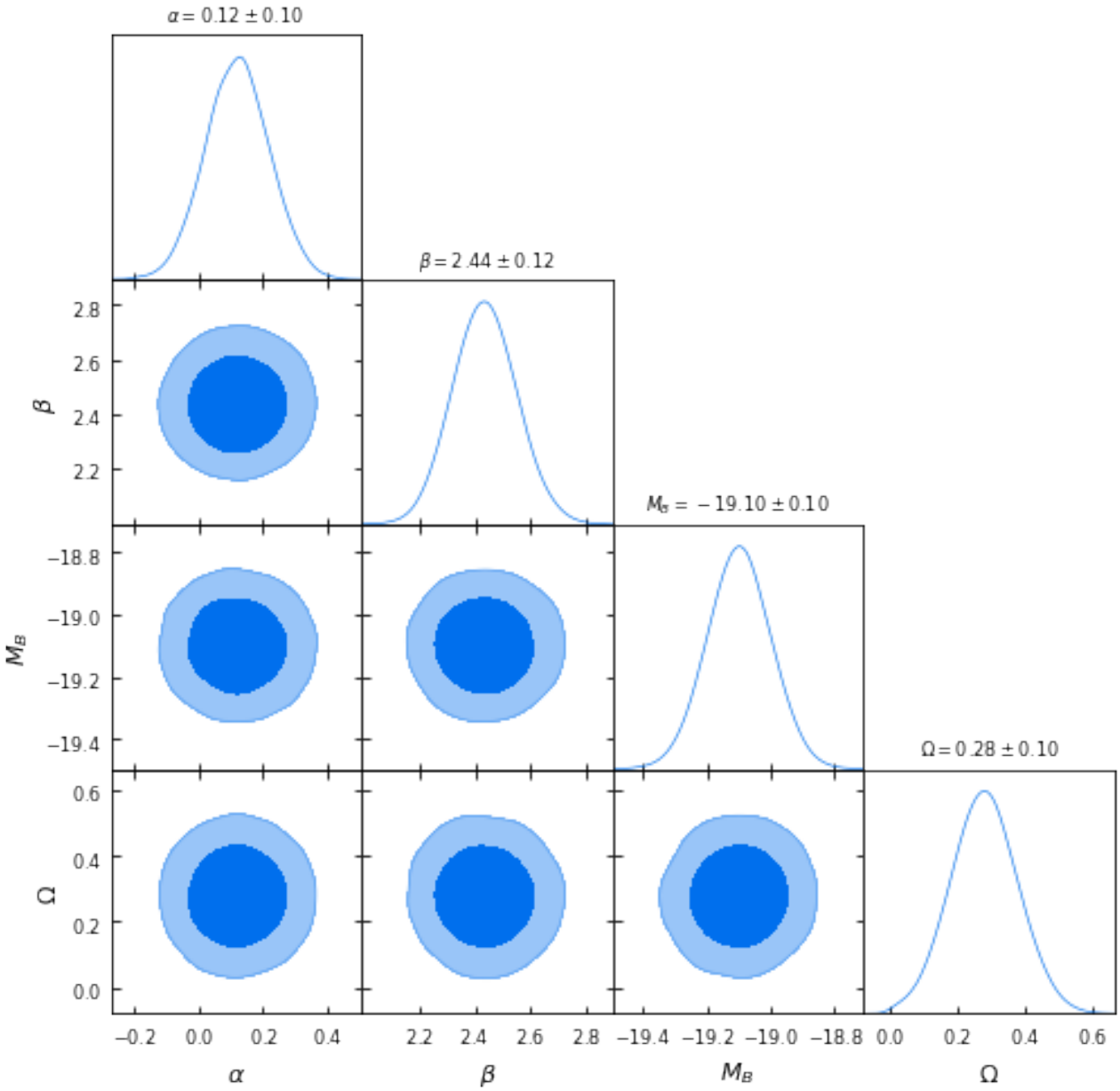

Figura 10: Gráfica triangular de todos los parámetros.

6.3. Comparación con librería EMCEE.

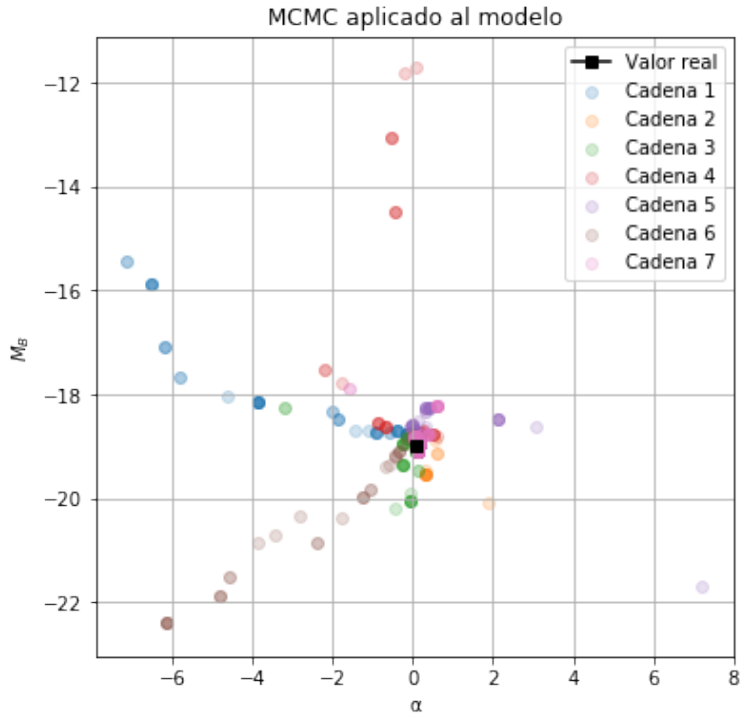

Figura 11: Visualización de las cadenas generadas con EMCEE para dos parámetros.

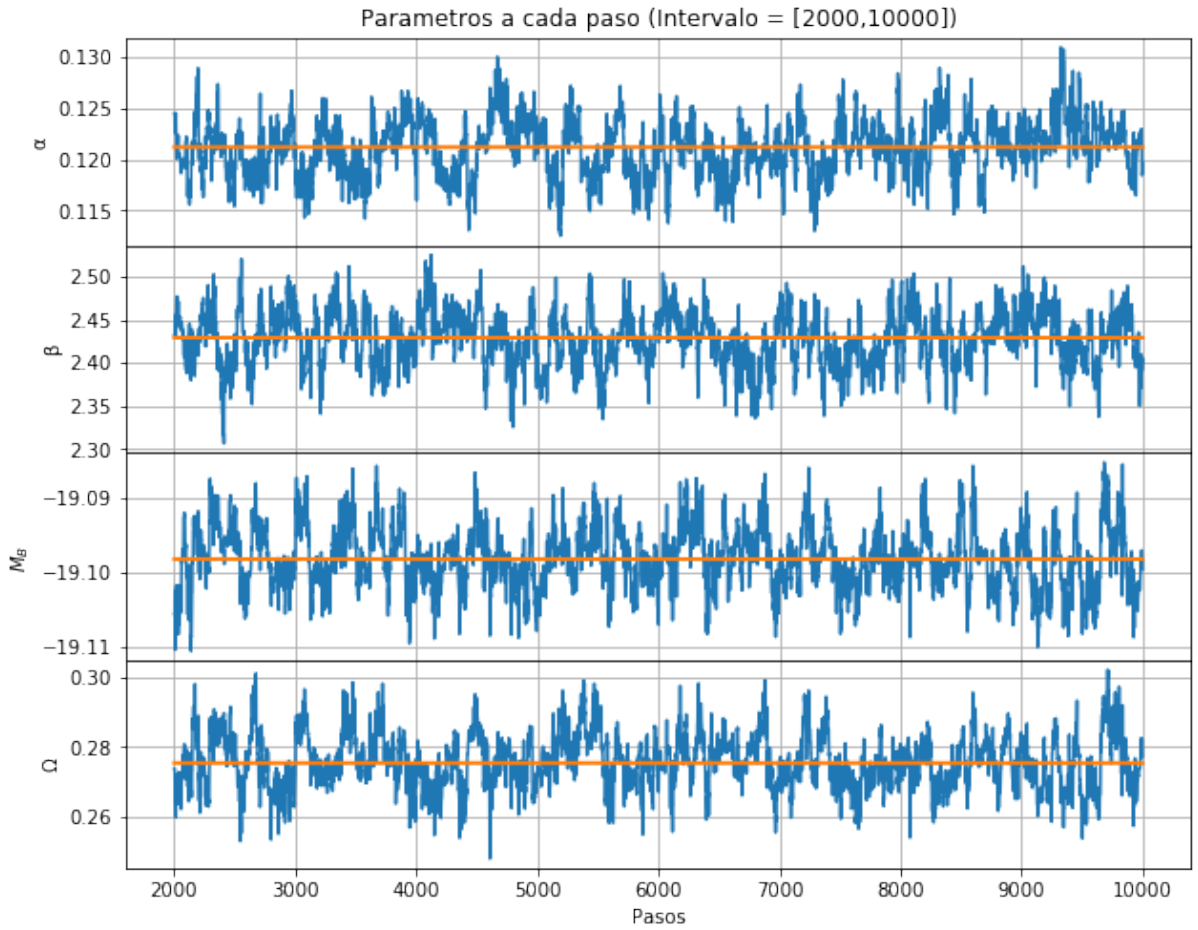

Figura 12: Variación de parámetros con EMCEE

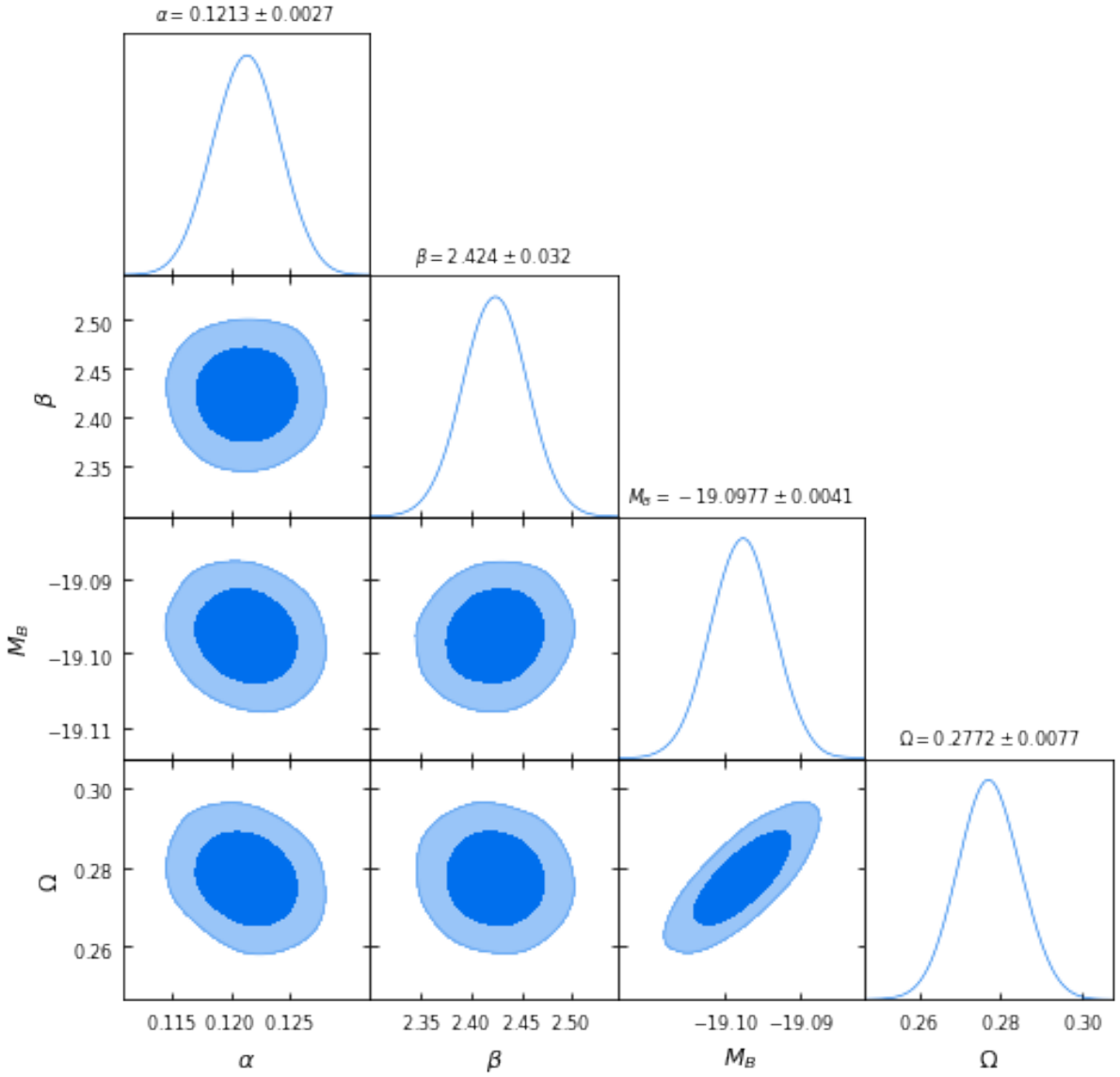

Figura 13: Gráfica triangular obtenida con EMCEE

Referencias

[1] M. Betoule, R. Kessler, J. Guy, et all. *Improved cosmological constraints from a joint analysis of the SDSS-II and SNLS supernova samples*, Junio 5, 2014

[2] Thomas Wiecki. *MCMC sampling for dummies* `https://twiecki.io/blog/2015/11/10/mcmc-sampling/` Consultada el 14 de junio del 2020.