統計計算與模擬期末報告

統計碩一

110354017 趙立騰

目錄

一、摘要

現今有許多模擬分配的方法，每種方法都有不同的優劣性，其中 MCMC 是近幾年很常見的模擬方式，利用已知的條件分配去模擬目標分配，因為 MCMC 是一種從分配中抽出相依樣本的方法，因此想知道多條具有不同起始值的 MCMC 和一條很長的 MCMC 何者對於目標分配的估計較優。

在這次模擬中我使用 MCMC 中的 Metropolis-Hasting 去進行，去測試一條、兩條、三條、五條與十條不同起始值的 MCMC 去對 N(0,1)的分配去做模擬，並觀察改變σ對於接受率的影響，最後再加入 burn-in 的方法，去比較 burn-in sample 與原樣本之間的優劣，從最後的結果可以看出的確一條長的 MCMC 對於模擬分配會比多條 MCMC 的組合還好，尤其是加入 burn-in 的方法後更為明顯。

二、研究動機與目的

在課堂中學過許多不同的模擬方式，從一開始的亂數生成、拔靴法到後來的蒙地卡羅法等等，每一種方法都有它的優點也有它的缺點，沒有哪一種方法是最好的。在許多模擬方法中，MCMC 是近幾年很常見的模擬方式，在課堂

中老師有提到若是使用多條 MCMC 的組合是否會比只使用一條 MCMC 的結果來的好，因此這次我將使用許多條不同起始值的 MCMC 來與單一條 MCMC 的模擬狀況來進行比較何者對於模擬目標分配的結果較優。

三、研究方法

(一)Metropolis-Hasting

在 1950 年 Metropolis 等人發表了有關 MCMC(Markov Chain Monte Carlo)的論文(J. of Chemical Physics 1953)，Hasting 則拓展了這項研究(Biometrika 1970)，是一種近年裡相當常用的模擬方法，利用從已知目標分配中抽取相依樣本，生成出一條有時間可逆性的馬可夫鍊，在達到平穩狀態後可以收斂到目標分配。

Metropolis-Hasting 是 MCMC 中的其中一種方法，會利用已知的條件分配$q(y|x_t)$(Proposal)來模擬目標分配$\pi(x)$

Proposal:$q(y|x) = \emptyset(y; x, \sigma^2)$

$$q(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}}$$

Steps:

從q(y|x_t)生成 y

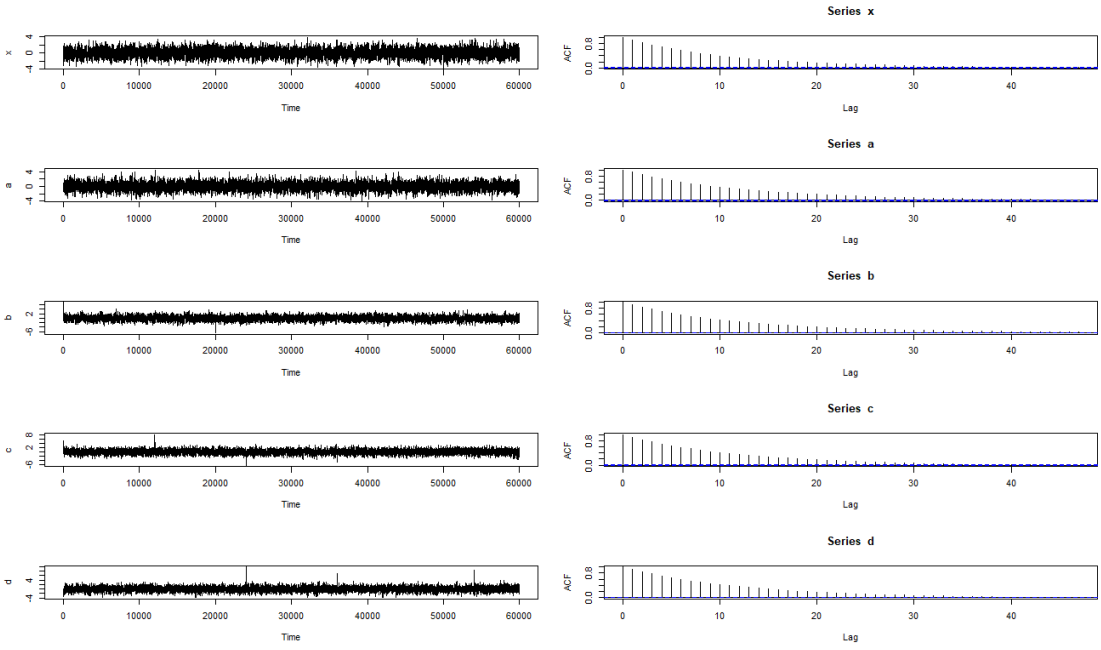
$$x_{t+1} = \begin{cases} y, with\ probability\ \alpha(x_t, y) \\ x_t, with\ probability\ 1-\alpha(x_t, y) \end{cases}$$

$$\alpha(\mathrm{x}, y) = min(\frac{\pi_y q(x|y)}{\pi_x q(y|x)}, 1)$$

$$min\left(\frac{\pi_y q(x|y)}{\pi_x q(y|x)}, 1\right) = min\left(\frac{\frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}}{\frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}} * \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-y)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-x)^2}{2\sigma^2}}}, 1\right)$$

$$\alpha(\mathrm{x}, y) = min(e^{-\frac{y^2 - x^2}{2}}, 1)$$

首先將模擬次數設定在 60000 次，起始值由 U(-10,10)隨機選取，σ設定在 0.5，來比較一條、兩條、三條、五條和十條 MCMC 的結果。



上圖由上而下分別是一條、兩條、三條、五條和十條的MCMC模擬結果。可以從左圖看出若是由多條MCMC組合的結果可能會導致原本已經到平穩狀態的Markov chain又回到不平穩狀態，又要經過數次迭代才會到平穩狀態，再來看他們的平均數及標準差。

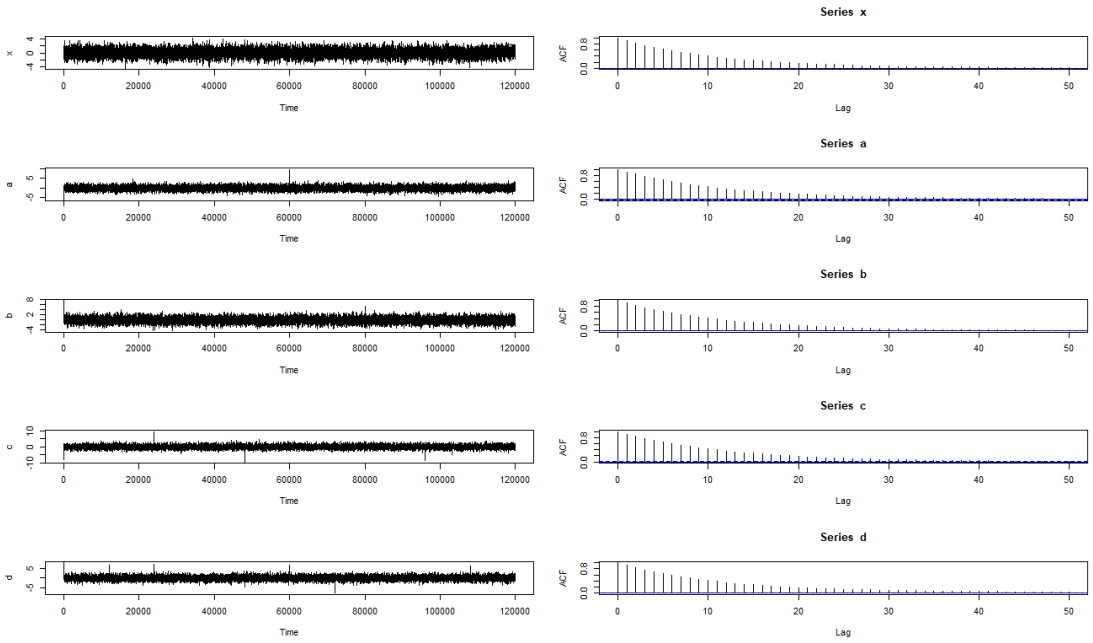
	mean	sd
一條	-0.03000677	1.013746
兩條	-0.006713935	1.015068
三條	-0.06137526	1.012203
五條	0.003218699	1.015342
十條	-0.006534694	1.031474

從這個表格可以看出我們的想法是沒錯的，大致上越多條MCMC的組合的估計結果會較不好，因為容易受不同的起始值影響收斂狀況。

	一條	兩條	三條	五條	十條
接受率	0.84295	0.8436	0.84515	0.84288	0.84417

上圖為五種方法的接受率平均值，幾乎看不出差異

再來將總模擬次數設定在 120000 次，起始值與σ的設定都相同再來看結果。

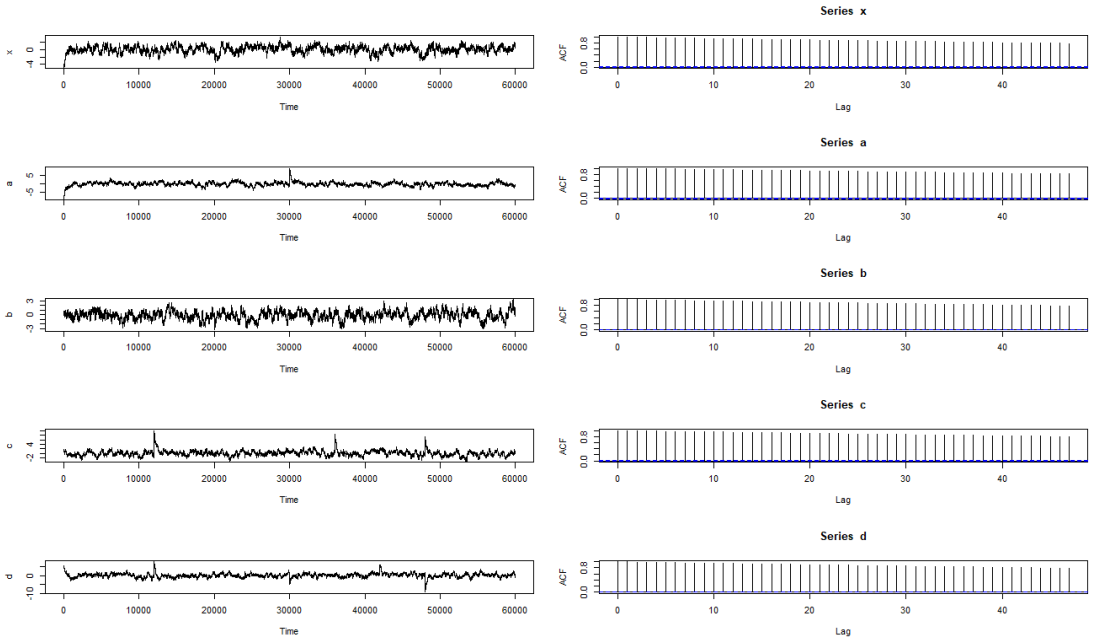


	mean	sd
一條	0.03023792	0.9952425
兩條	0.001647414	1.014424
三條	0.0162717	1.012688
五條	0.0119797	1.031036
十條	-0.03624707	1.030696

與上次的結果相似，可以看出不同的起始值確實對模擬是會有影響的。

	一條	兩條	三條	五條	十條
接受率	0.84552	0.84232	0.84259	0.84415	0.84458

接下來想比較σ對模擬的影響，將總模擬次數設定在 60000 次，起始值選取方法與前面相同，這次將σ設定為 0.1 來看與上次的結果有何差異



	mean	sd
一條	0.08287461	0.9847966
兩條	-0.1126046	1.163012
三條	-0.1305335	0.9640429
五條	0.08333121	1.208668
十條	0.1150274	1.168011

	一條	兩條	三條	五條	十條
接受率	0.96842	0.96645	0.96868	0.96585	0.967467

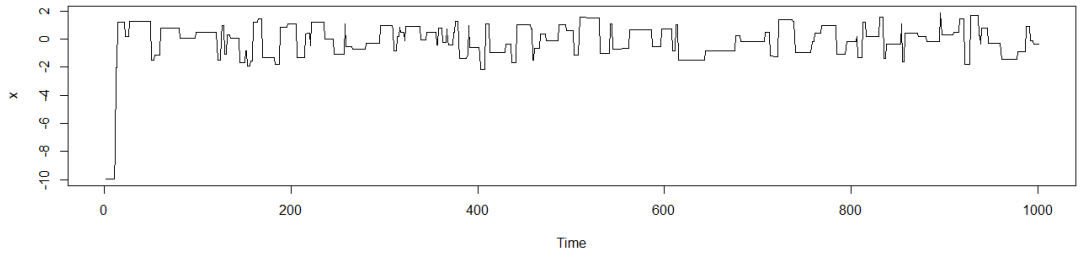
可以看出五種方法的接受率一樣沒有甚麼差異，但是跟 $\sigma = 0.5$時的接受率相比全部都上升了，換句話說也就是更容易接受由 proposal $q(y|x_t)$所生成的 y 了，整個 time series 的圖也看起來更加的像 simple random walk。

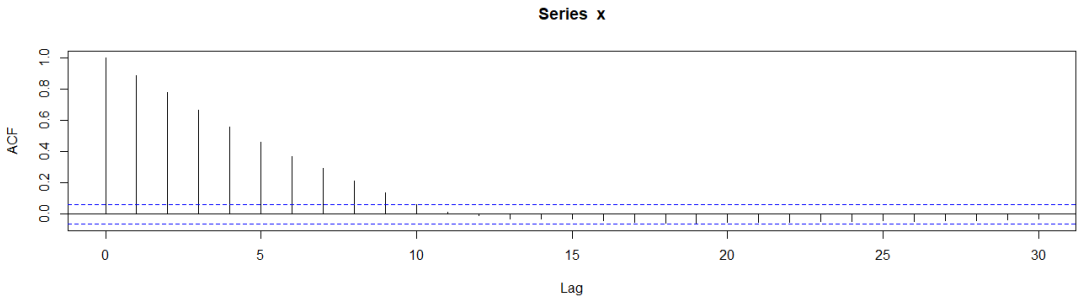
	一條	兩條	三條	五條	十條
接受率 $(\sigma = 0.5)$	0.843	0.844	0.845	0.843	0.844
接受率 $(\sigma = 0.1)$	0.968	0.966	0.968	0.965	0.967

σ的值會與接受率息息相關，因為 y 的分配與σ有關。

若σ變小，則會更傾向接受 proposal 所生成的值。反之，若σ變大，則會更傾向於拒絕 proposal 所生成的值。因此σ從 0.5 降為 0.1 時才會有接受率上升的情況。

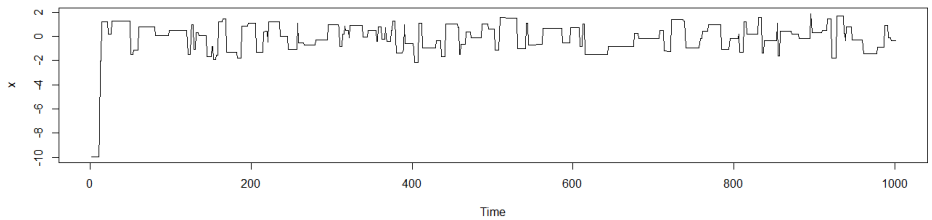
若我把σ設定為 10，就會有以下的情況發生(為了方便觀察，僅生成 1000 筆)

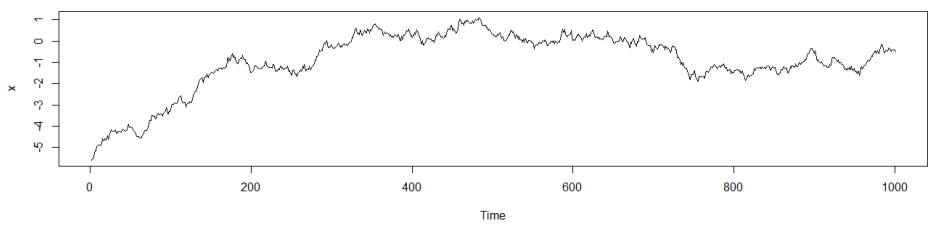




可以看出若$\sigma = 10$時，會有許多地方$x_{t+1} = x_t$，因為此時的接受率只有 0.13 左右而已，從 acf 圖也可以看出樣本間的自相關性降低許多。

下圖為同樣模擬 1000 筆資料，$\sigma = 0.1$與$\sigma = 10$的 time series 的比較。

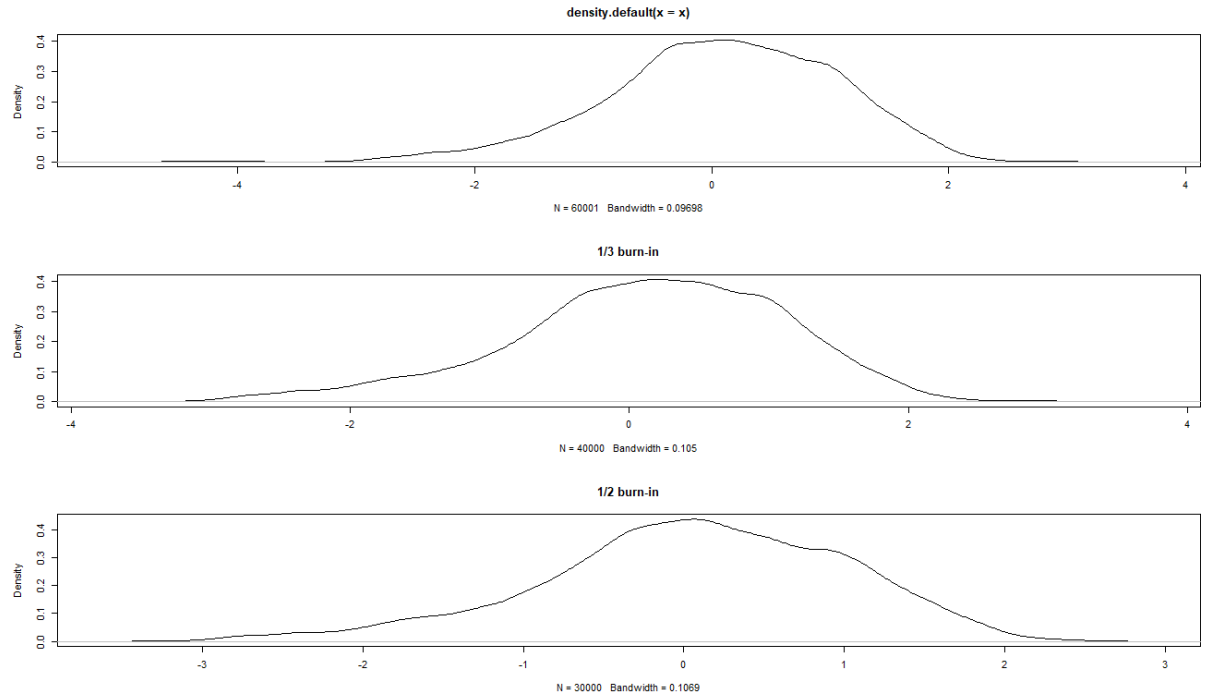




這次使用剛剛的樣本，起始值從 U(-10,10)取出、σ=0.5，去比較去除掉 1/3 和 1/2 的 burn-in sample 有何差異，先從模擬次數為 60000 次的樣本來看。

	mean	1/3 burn-in	1/2 burn-in
一條	-0.030007	-0.02876561	-0.0082483
兩條	-0.006714	-0.02101607	-0.03126599
三條	-0.061375	-0.08474482	-0.06927226
五條	0.0032187	0.01866641	0.003417443
十條	-0.006535	0.002431952	0.01839236

	sd	1/3 burn-in	1/2 burn-in
一條	1.013746	1.016939	1.005305
兩條	1.015068	0.9994246	0.9955323
三條	1.012203	0.9976588	0.9977953
五條	1.015342	1.0075	1.01391
十條	1.031474	1.041578	1.014615



從 mean 跟 sd 分別來看都可以看出，有 burn-in 方法加入後大致上都是更接近目標分配，除了十條組合的 MCMC 丟掉 1/3 的 x 後標準差反而變大了，可能是沒有丟掉比較像 outlier 的起始值，如果要解決這個問題可能要先將每一條 MCMC 都先找出 burn-in sample 再將多條 MCMC 組合起來，可能就可以解決這個問題。

再來觀察模擬次數為 120000 次的樣本，同樣是使用前面生成的數列來看，起始值從 U(-10,10)抽出，$\sigma = 0.5$

	sd	1/3 burn-in	1/2 burn-in
一條	0.9952425	0.9992829	0.9890905
兩條	1.014424	1.018849	1.023184
三條	1.012688	1.007745	0.9897966
五條	1.031036	1.025822	0.9979134
十條	1.030696	1.030784	1.030321
	mean	1/3 burn-in	1/2 burn-in
一條	0.03023792	0.02316221	0.02715285
兩條	0.001647414	0.01884047	0.01845908
三條	0.0162717	0.01044976	-0.0024306
五條	0.0119797	0.03449415	0.02653844
十條	-0.03624707	-0.03619184	-0.0509371

得到的結果與前面相似，也可以看出沒有一定是丟掉 1/3 的樣本還是丟掉 1/2 的樣本可以得到比較好的結果，以後如果會使用這個方法的話依舊可以把兩種方法都考慮進去，再去比較何者的模擬結果較好。

四、結論

從上面的模擬中可以看出若是一條不是很長的 MCMC，起始值的影響是很大的。

若把 60000 筆模擬分成 10 條 6000 筆的 MCMC，只要起始值與目標分配有點差距，就會需要一些時間才能使馬可夫鍊達到平穩狀態，因此最後才會得到一條很長的 MCMC 的模擬結果比多條 MCMC 的組合的模擬結果佳。

而σ對此模擬方法最大的影響就是在接受率的部分，比較小的σ會有較高的接受率，會更容易接受 proposal 所抽出的 y。

最後再加入 burn-in 方法後可以看出，幾乎所有的模擬結果都更好了，因為 burn-in sample 可以去除掉起始值的影響，而其中並沒有得到去除 1/3 或是 1/2 的 burn-in sample 誰會有較好的模擬結果，如果想要使用這個方法的話可以把兩種狀況都考慮進去再去做比較。

五、參考文獻

Chib, S., Greenberg, E.(1995).”Understanding the Metropolis-Hasting Algorithm.”

Hastings, W. (1970). “Monte Carlo sampling methods using Markov chains and their application.”

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). “Equations of state calculations by fast computing machines.”

Metropolis, N. and Ulam, S. (1949). “The Monte Carlo method.”

Robert, C.P.(2015).”The Metropolis-Hasting algorithm.”