Advance Econometrics

Jose Luis Martínez Martínez

March 2025

Problem 1: MA(1): MLE and indirect inference

Suppose that the true data generating process (DGP) is:

$$x_t = u_t + \theta u_{t-1} \tag{1}$$

where $u_t \sim$ i.i.d.$\mathcal{N}(0, 1)$.

In this exercise we'll be ask for:

a) Estimate θ directly via the true model's log-likelihood function: $\hat{\theta}_{MA(1)}$

b) Estimate θ directly via an AR(1) model's log-likelihood function: $\hat{\theta}_{AR(1)}$

c) Estimate θ indirectly via the implied estimate: $\tilde{\theta}_{\text{MA(1)}}$

Where the corresponding estimators are going to be always respective means of a S = 10,000 Monte Carlo Simulation.

(To do this we'll be using a grid separated by 0.1 of $\theta \in [-0.9, 0.9]$ so we end up with 19 different true θ's in total).

Just to introduce the problem, we present a possible result for the case where $\theta = 0.7$, and see how true vs. misspecified model perform, compering both for a T = 100 sample size:

Estimator	Value
$\hat{\theta}_{MA(1)}$	0.774
$\hat{\theta}_{AR(1)}$	0.467

Table 1: Computed ML estimator via MA(1) vs. AR(1) Log-likelihood Functions (Conditional and Unconditional, respectively).

Results suggest that a AR(1) misspecation for a MA(1) process can lead to huge

fail. We will see latter that this mismatch persists even asymptotically, specially for true values of θ near unit root (as the case we have just present).

Now, what we are going to do is to performe same excersise but including the implied AR(1) estimator seen in class digression: 'The other way around' in Computer_lab_I_UA2025.pdf).

For what I have understood, this estimator is computed 'inside' an AR(1) model specification BUT it is robust to it, meaning that the result (the computed value for the theta) will hold even if that is not the true model.

Following table resumes the results, let's comment it a bit:

θ_{true}	$\hat{\theta}_{\text{MA(1)}}$	$\tilde{\theta}_{\text{MA(1)}}$	$\hat{\theta}_{\text{AR(1)}}$
-0.9000	-0.9006	-0.7945	-0.4885
-0.8000	-0.7997	-0.7662	-0.4788
-0.7000	-0.7017	-0.7177	-0.4606
-0.6000	-0.6004	-0.6395	-0.4315
-0.5000	-0.4996	-0.5255	-0.3911
-0.4000	-0.3988	-0.4167	-0.3360
-0.3000	-0.3001	-0.3060	-0.2675
-0.2000	-0.1990	-0.2020	-0.1873
-0.1000	-0.0979	-0.1011	-0.0956
0.0000	-0.0003	-0.0016	0.0004
0.1000	0.0997	0.1007	0.0962
0.2000	0.2002	0.2015	0.1873
0.3000	0.2989	0.3045	0.2685
0.4000	0.4006	0.4167	0.3363
0.5000	0.4990	0.5316	0.3914
0.6000	0.6001	0.6315	0.4318
0.7000	0.6994	0.7110	0.4598
0.8000	0.8012	0.7665	0.4792
0.9000	0.9025	0.7970	0.4895

Table 2: Estimation of θ for MA(1) and AR(1); where values are the computed mean of a S = 10000 Monte Carlo sample of T = 100 draws each.

So, what we can see (latter with the plots more visually) is that, obviously the best performer is the one that match the true models: $\hat{\theta}_{\text{MA(1)}}$.

The other two has a highlight weakness (even thought in the case of the implied one this weakness is notoriously smaller):

The closer the true value to the unit root the worst is the estimator perfor-

mance. Why this can be the case? Well, we are trying to estimate a MA(1) with indirect methods, in this case with an AR(1) model. We know that the bridge from MA to AR is given by the Invertibility property: if a MA process is invertible then it has an AR infinity representation. Maybe this is the intuition behind why the more non-invertible a MA process is the less 'similar' to an AR is (the more faraway from its representation it is) so the biggest the mismatch and the poorest the estimation. Obviously when $\theta = 0$ both models are exaclty the same so all perfome pretty well.

It is fair to mention that for the implied estimator this issue becomes a problem for values really close to the unit root. Also, as we will see, for the implied this problem disappears asymptotically (as the estimator converge to true value. Makes sense due to its mispecification robustness); this is not the case for the AR estimator.

To finish with this exercise we are going to plot the results commented above but including two more sample sizes: T = 1,000 and T = 10,000. In order to see Asymptotic Properties in each estimator.

The results are presented in the following charts:

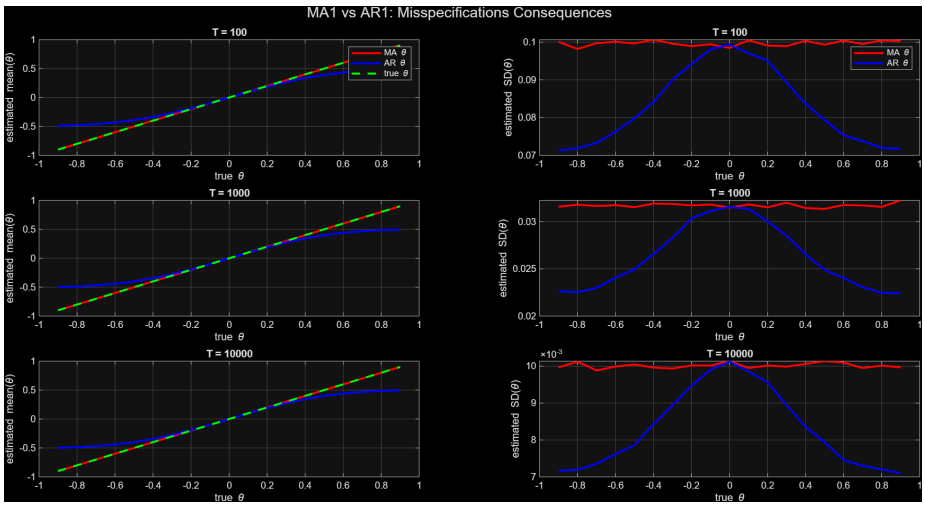


Figure 1: MA(1) vs. AR(1) Maximun Likelihood Estimator

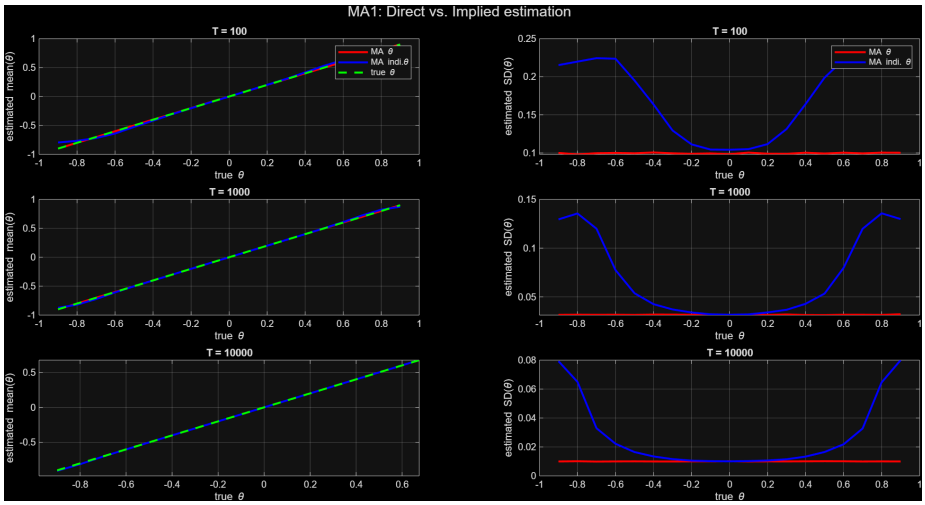


Figure 2: MA(1) direct vs. MA(1) implied Estimator

I think almost everything has been said. Figure 1. sets the asymptotically lasting problem of the AR(1) estimator, while Figure 2. confirms the robustness to mispesification of the implied estimator.

Related to the Standard Deviation my intuition is less clear, but it is obvious the persistance of this 'close-to-unit-root' thing that we set before: in both cases the closer to cero the more alike the behaviour is with respect to the direct estimator.
But to be true, I really don't know what it is happening because AR(1) and Implied estimator SD share shape, but not sign (both obviously positive but inverted form).
It is like saying that in AR(1) estimator the closer to unit root you are the more wrong you are about the true value but the more accurate you are about that error (fail) (fair to mention that values are really tiny any way).
In the case of Implied estimator the only intuition that I can get is that unit roots generates distortions in the estimator, driven possible by more jumps between the bounds (as this estimator is not continuous) and more presence of ones (value for $\theta = |1|$) as we can see in the data that we generate with the code.

Problem 2: Metropolis-Hastings & a fancy AR(2)

Consider the following AR(2) model:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \varepsilon_t, \quad \text{with} \quad \varepsilon_t \sim \text{i.i.d. } \mathcal{N}(0, 1 + \alpha^2)$$

First of all we are going to simulate data form this AR(2) process for $\alpha = 0.45$; but with different init values:

i) $y_0 = y_1 = 0$

ii) $y_0 = y_1 = 25$

iii) $y_0 = y_1 = 100$

Let's plot the results and discuss a bit about them:

Figure 3: AR(2) process with different initial values

Figure 3, show us how, independently of the init value, this three series converge to same process (ending up behaving equal). Why this is the case? This happens because we artificially generate the data to match with a certain value of the parameter ($\alpha = 0.45$); then no matter what the initial value is we always end up generating a process with the characteristics that we impose, meaning: a Stationary Process with mean in cero and a variance equal to $1+0.45^2$. Essentially because all the randomness came from the error term ($\varepsilon_t \sim$ i.i.d. $\mathcal{N}(0, 1+\alpha^2)$ and so all the variability behaves like this.

The following tasks are related to Metropolis-Hastings Technics. So we are going to explain a bit about it's design (the particular application we are going to be dealing with) and in general terms why it could be useful:

Metropolis-Hastings(MH):

Metropolis-Hastings belongs to the family of MSMC (as Gibb Samplings does).

It is a way of generating samples from a know Distribution. From what I understood it differs from Gibb Sampling in that with MH we don't have to know a proper(correct) way to *'take draws'* from that certain Know Distribution.
This numerical tool suits pretty well with the Bayesian Approach and is ideal to produce samples from Posterior Distributions, when their properties are not always straight forward (eg. Priors not always conjugated).

How it works?:

Well, at a really simple level, it basically computes a probability of retaining a certain draw or to kick it. This creates a MS process that ends up emulating a stationary time series, giving as an estimator (with mean and standard errors). This probability is what replace the *'know how to take a draw'* thing in the Gibb-Sampling, mentioned above.

Now, how is that probability computed?:

This is a good question and in our modest application it ends up being a simply Ratio; but this naive description bring us another question:

Ratio between what?:

And here is where we introduce one key feature of the MH methods: we need a way to generate *'candidate draws'*.
There are more methods but a convenient one is to use a Random Walk (one advantage is that it reduce complexity of the acceptance probability computation); with this method we can generate draws from previous draws.

We end up with New and Old draws:

$$\Phi^{New} = \Phi^{Old} + e \text{ where } \quad e \sim \mathcal{N}(0, \Sigma)$$

Then, the ratio we are aiming for is simply the one formed between the target Distribution evaluated at the New parameter(draw) vs. the target Distribution evaluated at the Old parameter(draw).

As you can imagine the elicitation of the initial value (the first draw that we impose (not random)) and the Variance of the Random walk can be pretty determinant for the convergence of the series and to perform a good estimation.

For the Random Walk MH we end up with the following acceptance probability:

$$\alpha = \min\left(\frac{\pi(\Phi^{New})}{\pi(\Phi^{Old})}, 1\right)$$

where $\pi(\Phi^{Draw}) \equiv$ Target Distribution.

In our case (the formula that we will be implementing) it will end up being:

$$\alpha = \min\left(\exp\left(\ln H(Y_t|\Phi^{New}) - \ln H(Y_t|\Phi^{Old})\right), 1\right)$$

where $\ln H(Y_t|\Phi^{New}) \equiv$ Prior $\times$ Likelihood ($\equiv$ Target Distribution).

Now, the exercise proposes us a particular case, where we want to impose a Prior that limits the draws elicitation to those that holds the stationary assumption (convenient for the AR(2) process that we are working with).
This Prior is going to be more a sort of Restriction; we mean: 'Posterior' distribution is not going to be changed (it will be simply the Likelihood) but we will reject any draw that violates stationary. We can accomplish this by imposing the following indicative Prior:

$$p(\alpha) \sim \mathbf{1}(1 - \alpha z - \alpha z^2;\, |z| > 1 \,\forall\, z)$$

This will give probability cero to all the non-stationary candidates and letting the ones that are stationary to compete with the old ones in a normal Likelihood Ratio framework.

From now on, we are going to implement this mechanism in the computer for a couple of different cases (Priors):

i) No informative Prior:
Here the Posterior = (Conditional)Likelihood Function:

$$L(\alpha) = -\frac{T-2}{2}\log(2\pi) - \frac{T-2}{2}\log(1+\alpha^2) - \sum_{t=3}^{T}\frac{(y_t - \alpha y_{t-1} - \alpha y_{t-2})^2}{2(1+\alpha^2)}$$

ii) Non Stationary Restriction:

Here we encounter the prior mentioned above:

$$p_s(\alpha) \sim \mathbf{1}(1 - \alpha z - \alpha z^2;\, |z| > 1 \,\forall\, z)$$

The Posterior ends up:

$$H_s(\alpha) = L(\alpha) \times p_s(\alpha)$$

iii) Normal Distributed Prior:

Here we encounter the following prior:

$$p_N(\alpha) \sim N(0, 0.01^2)$$

Which basically means that we are sure that our parameter is cero.

The Posterior ends up:

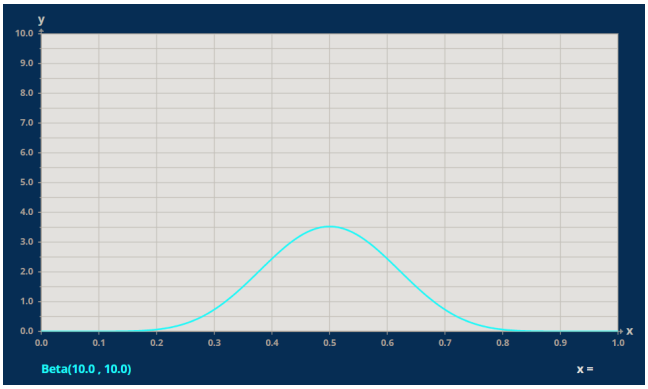
$$H_N(\alpha) = L(\alpha) \times p_N(\alpha)$$

iv) β Distributed Prior:

Here we encounter the following prior:

$$p_\beta(\alpha) \sim \beta(10, 10)$$

In this case this prior seems to account for the believe that the parameter is between 0 and 0.5 ($\alpha \in (0, 0.5)$). This prior somehow put this parameter around the 0.5, but I'm not sure if this is the best way to impose that restriction. Maybe we want to leave some space for the data to say something. I say this because a $\beta(10, 10)$ looks like this:



Is not like supper restrictive in the 0 to 0.5 boundary.

The Posterior ends up:

$$H_\beta(\alpha) = L(\alpha) \times p_\beta(\alpha)$$

The results are displayed in the following graphs. Each one has the MS chain generated by the MH (left-hand side) and a related non-parametric kernel density estimation (right-hand side) which will represent our Posterior Distribution. Both for three different chain size: 100, 1,000, 10,000.

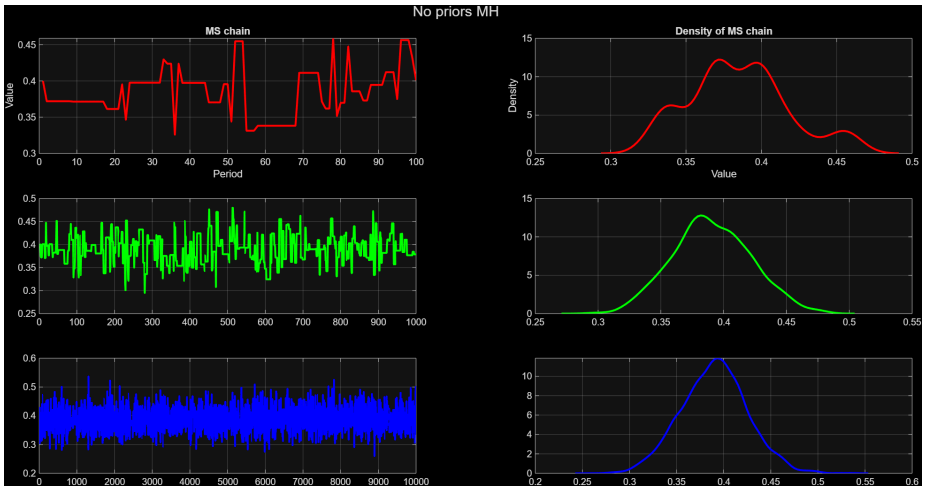


Figure 4: MH without Prior Believes (i)

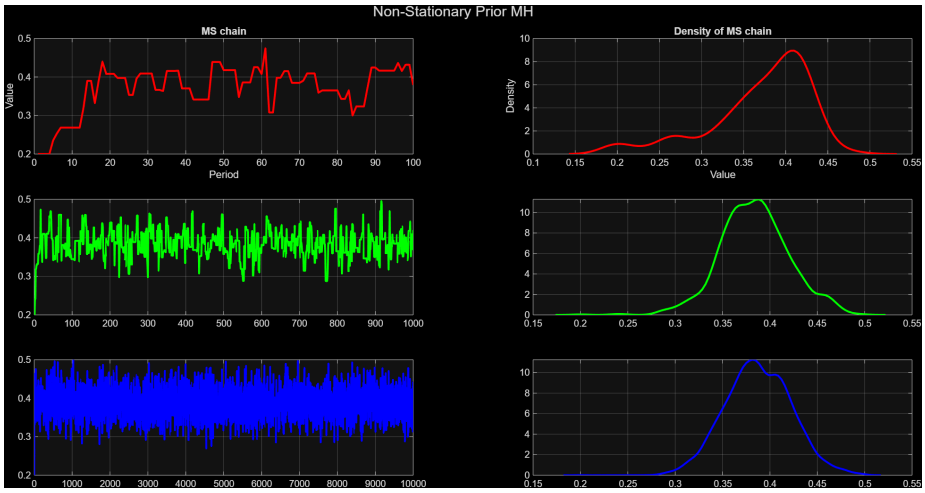


Figure 5: MH with No-Stationary Prior (ii)

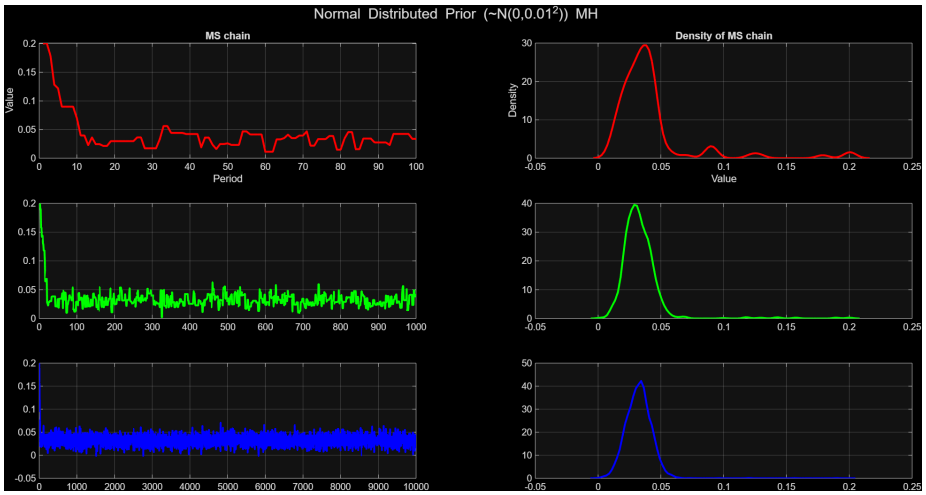


Figure 6: MH with very restricted Normal Prior (iii)

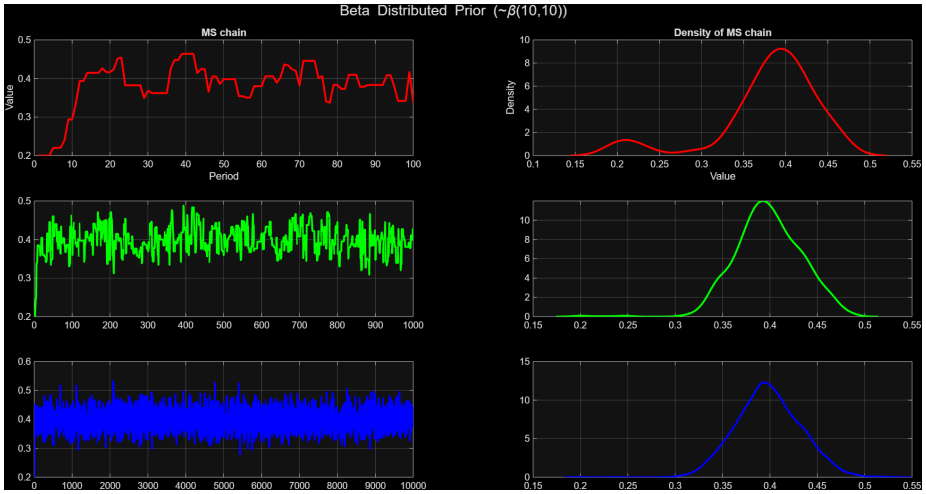


Figure 7: MH with $\beta(10, 10)$ Prior (iv)

Results are pretty reasonable:
Only thing that annoys me is that no one matches the true parameter value ($\alpha = 0.45$) at mean, it always end up around 0.4. (I really don't know the reason, maybe T = 125 is too small for the data to speak the truth (maybe the priors that we impose are not really helping in aiming the true value)).
(Also I run the code many times and always get similar results: pseudo-Monte Carlo Simulation)

Okey sorry, this misselection occurs because I didn't burn-out any draws, so the initial value is somehow dragging towards 0.2 the estimation, that's why the result doesn't get around 0.45. Here I report the results with burn-out:

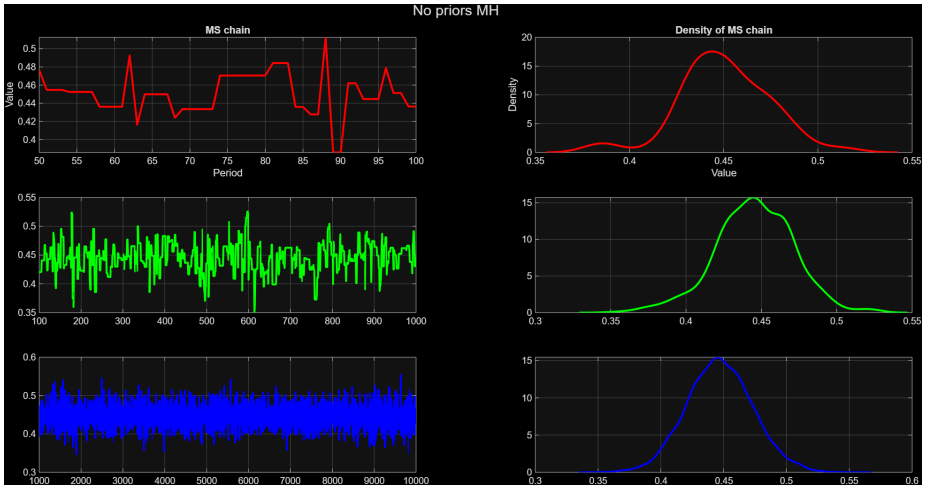


Figure 8: MH without Prior Believes (i)

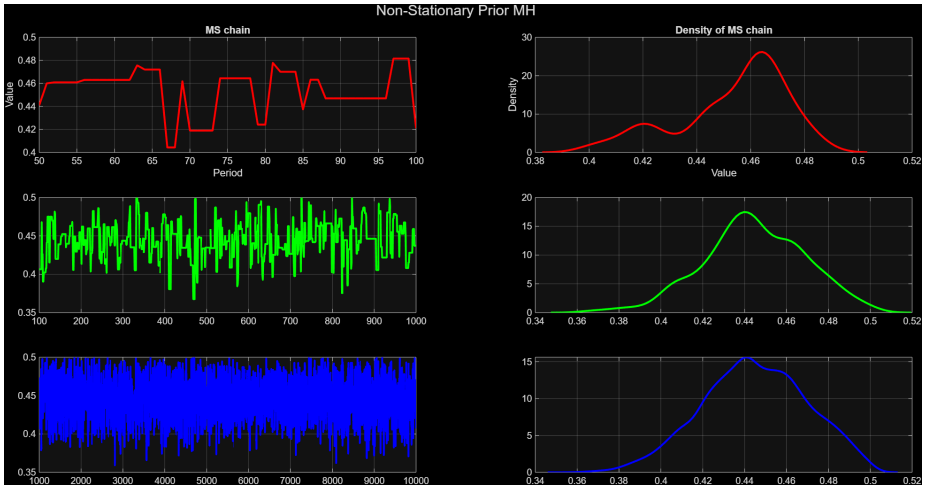


Figure 9: MH with No-Stationary Prior (ii)

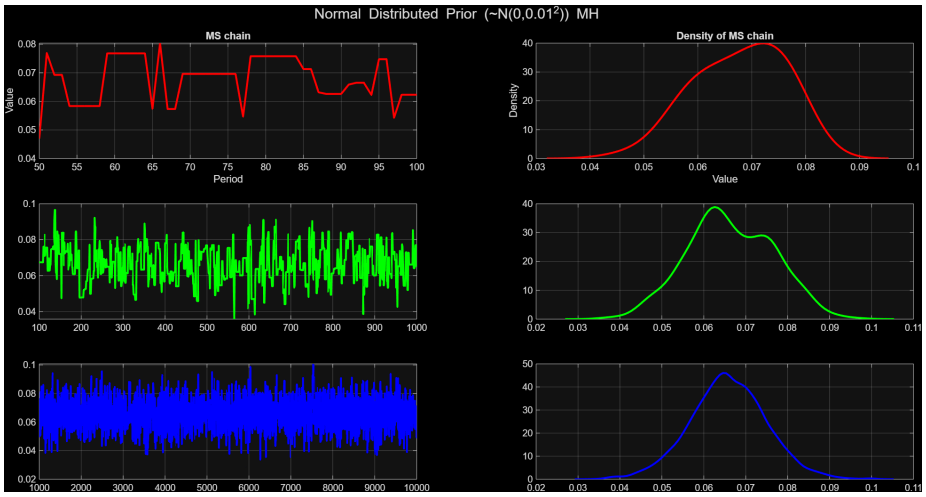


Figure 10: MH with very restricted Normal Prior (iii)

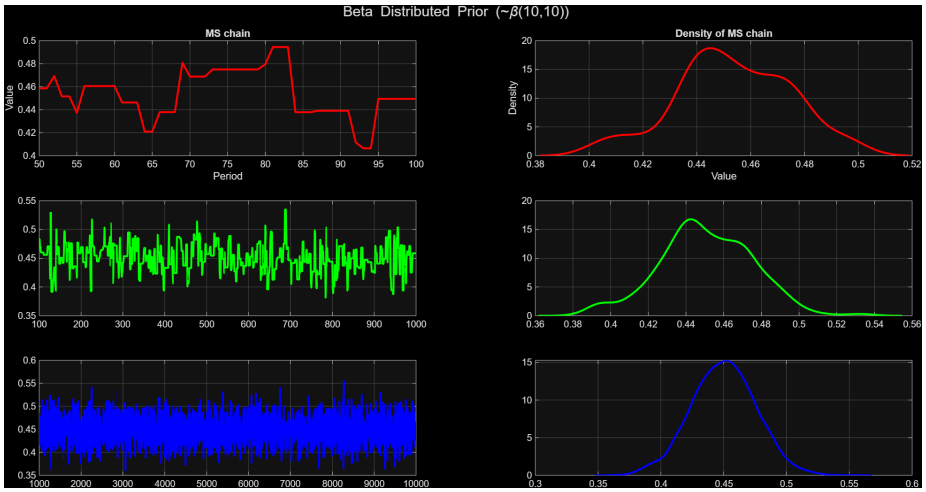


Figure 11: MH with $\beta(10, 10)$ Prior (iv)

All densities are around the true parameter value and the bigger the size of the chain the more accurate. I think the most worth-to-mention case is the one with the *'restrictive Normal Prior'*, this represents how strong believes can really affect the results. So even data is totally telling as something different this *'for sure'* kind of priors force things to be around what we believe (but in this case is so no true that is not capable of center the hole distribution around cero).

Just to highlight how priors lose strength against data increase I'm going to report results for this Strong believe, with an increase of sample size from T = 125 to T = 500:

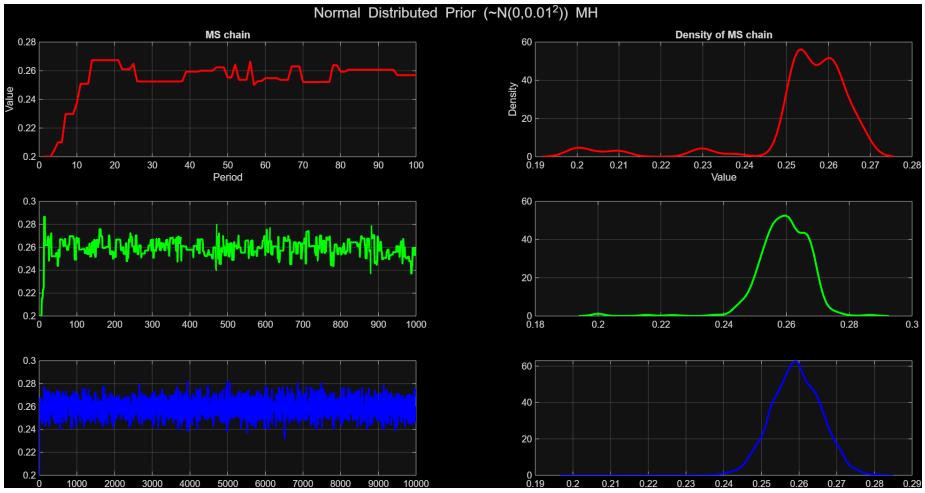


Figure 12: Very restricted Normal Prior with an increase in the Data size

As we can see distribution is moving towards the real value of the parameter, showing that our Prior is more a binding constraint than a trustfully thought. (Figure 12. does not accounts for burn-outs)

Now, and to finish the exam, Let's make a tiny discussion about the elicitation for the Candidate Generator Ranodom Walk Variance (Σ):

The main criteria was the comment you mention in the statement:
"(same stuff as we have seen in class, trying to get an acceptance rate as the one I mentioned, between 30%-60%)"

	Init Value	Σ	Acceptance Rate		
			$s = 100$	$s = 1000$	$s = 10000$
No-Prior	0.2	0.005	0.5100	0.4950	0.4974
Stationary Restriction	0.2	0.005	0.4600	0.4930	0.4998
Normal Prior	0.2	0.0005	0.4700	0.4310	0.4550
Beta Priors	0.2	0.005	0.5100	0.4780	0.4753

Table 3: Acceptance rates for different priors, initial values, and sample sizes

Also in the book "Applied Bayesian econometrics for central bankers" the au-

thors say:

"The variance of the candidate generating density Σ can be set as the OLS coefficient covariance matrix $\hat{\Omega}$ times a scaling factor λ i.e $\Sigma = \hat{\Omega} \times \lambda$. Note that $\hat{\Omega}$ provides a rough guess of how volatile each parameter is. The scaling factor lets the researcher control the acceptance rate (a higher value for λ would mean a lower acceptance rate)."

So even thought I didn't compute the OLS coefficient covariance matrix, we can infer from this words that the smaller the Variance the higher the Acceptance rate so in order to accomplish that 30% - 60% criteria I needed to really low the Variance reaching (even closer) almost cero in the case of the Normal Prior.
Not know if this is correct but it was the only way to reach those acceptance rates. Also 0.025 get acceptance rates close to 30% but didn't reach it.