PS8 Answer key

Time Series

Problem 1

In order to replicate the table asked, we have to use the codes *gbs_ato* and *mle_ato*. The first one uses Bayesian analysis, whereas the second one estimates the model by maximum likelihood.

Bayesian analyisis: Gibbs sampling

Let's take a deeper look at this code. At the beginning, the raw data is generated. That is, a regression model with autocorrelated errors is simulated. Note that **this is NOT part of the Gibbs sampling procedure**. We are just generating the data which we will later analyse. After that, we set the priors hyperparameters.

```
    @================= PARAMETERS FOR PRIOR DISTRIBUTIONS====@

R0=EYE(2)*1000000;
BETA0=0|0;        @FOR beta0,beta1@

R00=EYE(1)*1000000;
phi0=0;         @FOR  phi @

nu0=0;
d0=0;                    @FOR SIG2_V OF INDIVIDUAL COMPONENT@
```

Figura 1: Priors hiperparameters

The prior distributions are the ones explained in class: $\beta|\phi,\sigma^2 \sim N(\beta_0, R_0)$, $\phi|\beta,\sigma^2 \sim N(\phi_0, R_{00})$ and $\frac{1}{\sigma^2}|\beta,\phi \sim \Gamma(nu0/2, d0/2)$. We see that, in this case, the first two priors have zero mean. On the other hand, the variance matrix is huge (for example, R0 is 1 million times the 2×2 identity matrix). That is, we are encoding high prior uncertainty. This will make the data more important to determinate the posterior parameters. Last but not least, note that the prior of σ^2 is improper (that is, there is no $\Gamma(0,0)$ distribution), but the posterior will be proper so this is fine. Note also that this are NOT the priors used by Kim and Nelson. We could change this by simply changing these prior hyperparameters.

We then set the number of repetitions of our Gibbs sampling Monte Carlo simulation. In this case, we will use the first 2000 reps to ensure that the algorithm converges. We then use the following 4000 reps to generate the posterior distribution.

Then the algorithm begins. It sequentially generates draws of the parameters using the posterior distributions conditional on the other parameters. The advantage of this is that we don't need to derive the joint distribution: we only need the conditionals in order to proceed.

When we run the code, we get as output several statistics if the posterior distribution (mean, SD and median), as well as a lot of plot that we will analyse. Note that **the program won't generate all the plots at once**. It will generate sets of plots corresponding to each parameter. To generate the following set, just go to command bar and press any key. Posterior descriptive statistics are shown in the table.

The plots are shown below. For each parameter, the program will generate three plots. We use

Parameter	Mean	Sd	Median
β_0	-0.1041	0.1749	-0.1103
β_1	1.0444	0.0730	1.0432
ϕ	0.5375	0.0731	0.5345
σ^2	0.9784	0.1150	0.9642

Cuadro 1: Posterior descriptive statistics

the ones generated for ϕ, but the other ones have similar characteristics.

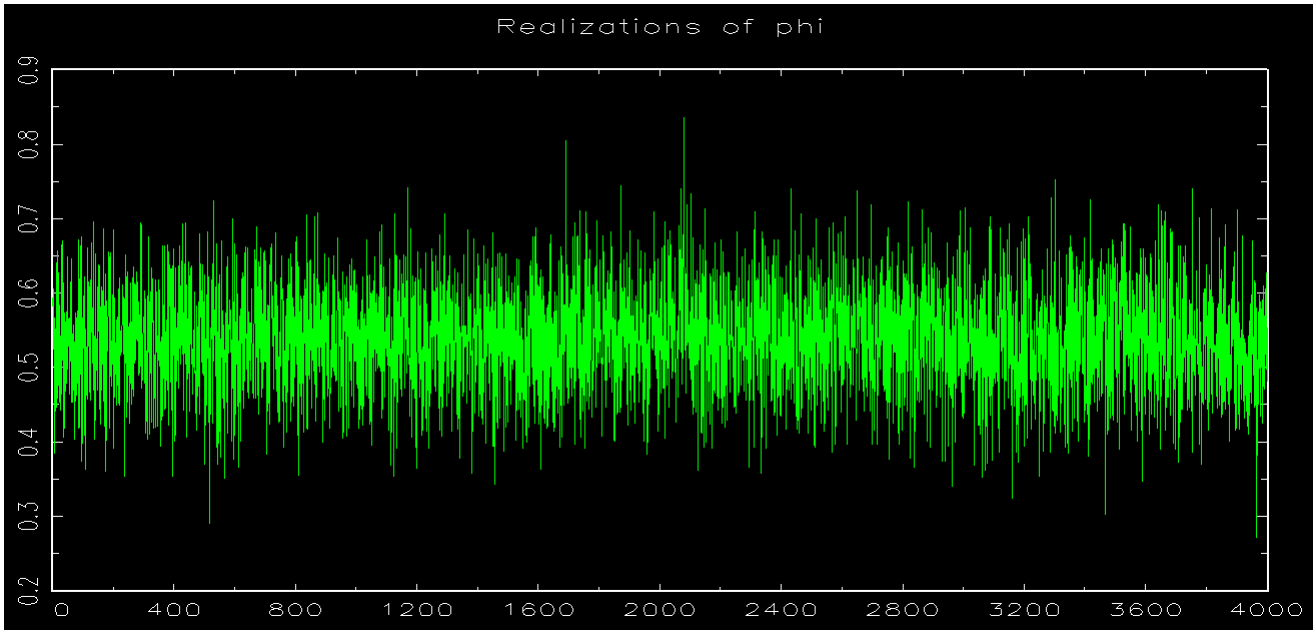


Figura 2: Realizations of ϕ

The first plot shows realization for ϕ for each iteration. The second plot shows the histogram based only in the first 1000 iterations. The idea here is that, if the algorithm converged, the histogram based in the first 1000 reps has to have similar properties as the one that uses the all

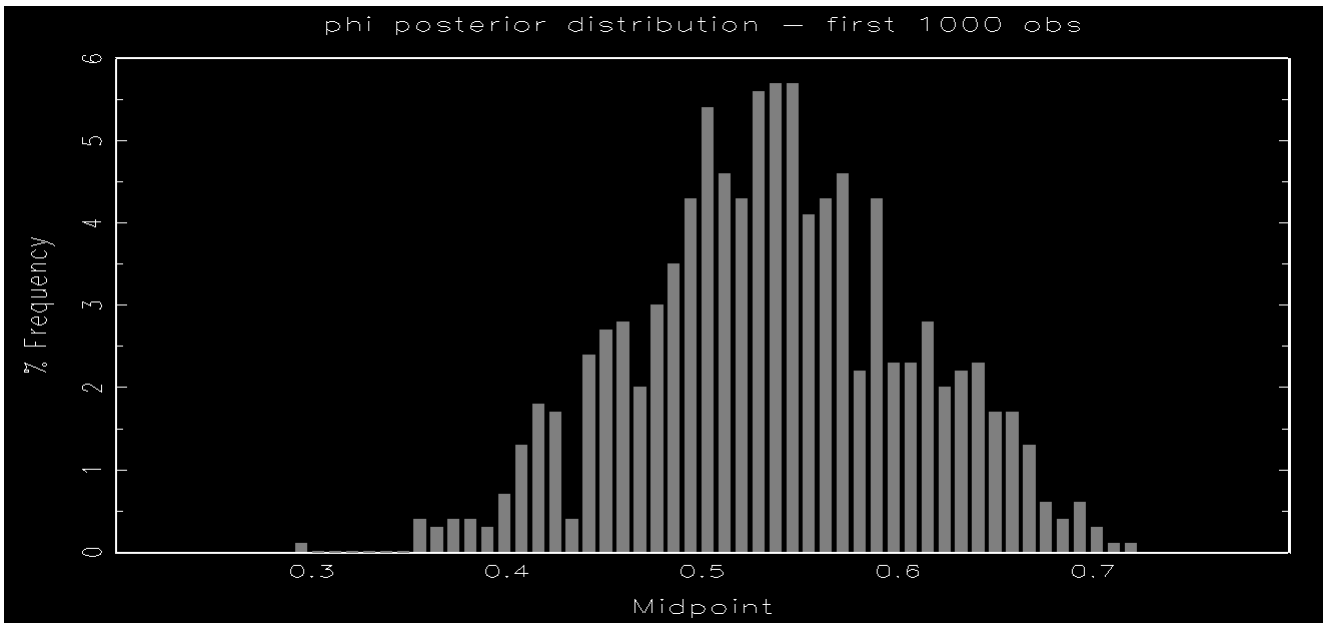


Figura 3: Histogram of ϕ. Only first 1000 realizations used

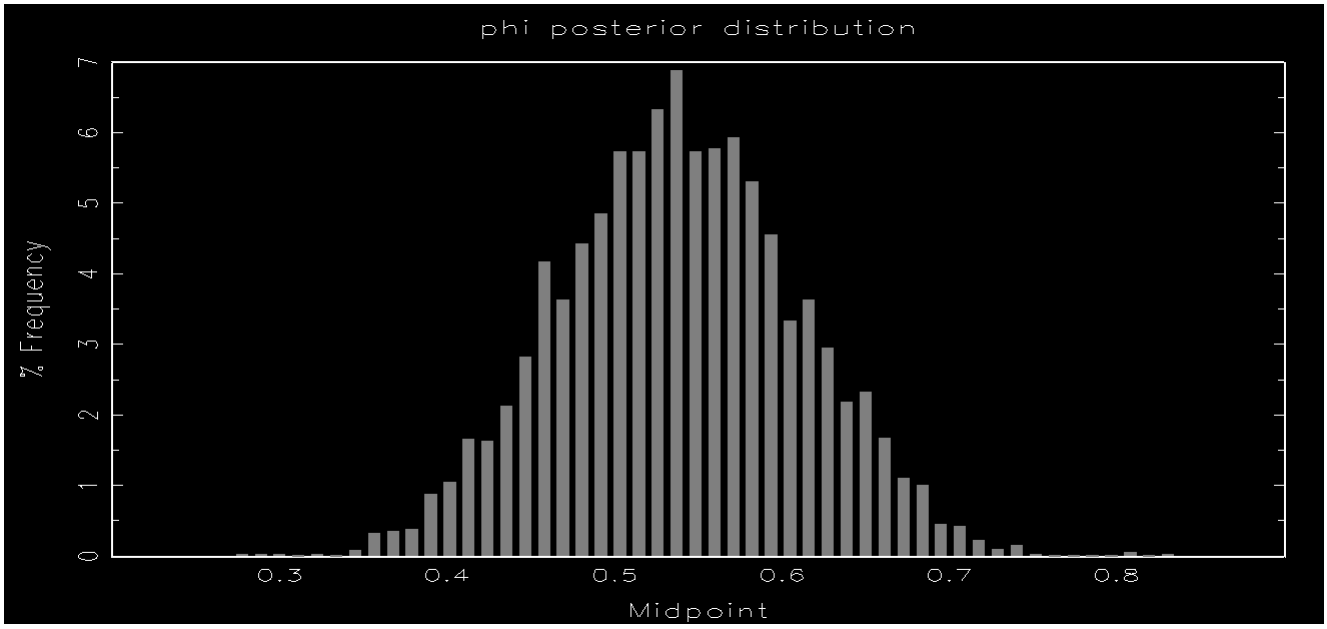


Figura 4: Histogram of ϕ.All 4000 realizations used

the iterations. As we can see from the graph, this actually happens. There is some difference in the scale of the axis, but that's caused by the fact in 4000 iterations we have a higher chance of getting outliers that we have in 1000 (that is, with only 1000 repetitions we may not have a proper estimation of the probability of outliers to appear). The plots for the other random variables is similar.

Maximum Likelihood

We use the file *mle_ato* to estimate the parameters for this model. This code is similar in spirit to the ones used in PS5 and PS6. The code prints in the command bar the estimated parameters and its standard errors as output. We compare our results from both methods in

```
==FINAL OUTPUT=================================================
code is--------  0.00000
likelihood value is -208.03856
Estimated parameters are:
 -0.09536   1.04041   0.52924   0.95560

Calculating Hessian..... Please be patient!!!!
Standard errors of parameters are:
  0.16889   0.07247   0.07032   0.10995
================================================================
```

Figura 5: Maximum likelihood estimation output

the following table We note that results are quite similar. This is due to the fact that we use

	Bayesian			MLE	
Parameter	Mean	Sd	Median	Estimation	Std error
β_0	-0.1041	0.1749	-0.1103	-0.09536	0.16889
β_1	1.0444	0.0730	1.0432	1.04041	0.07247
ϕ	0.5375	0.0731	0.5345	0.52924	0.07032
σ^2	0.9784	0.1150	0.9642	0.95560	0.10995

Cuadro 2: Bayesian and MLE compared

diffuse priors. In thaat case, we know that Bayesian analysis will be similar to MLE. You can try using tighter priors (i.e with a smaller variance matrix, for example, $R0 = I_2$ [1]) and see how results differ.

Problem 2

The analysis in this problem is similar in spirit to the one in the previous section. Now, we have to set prior for 6 parameters (the constant, 4 AR paramters and the variance). The Gibbs sampler works almost the same way as before. There is an extra restriction on the generation of the ϕ: the generated vector has to be stationary.

As before, the code computes the posterior parameters V and $BETA1$. The Choslesky decomposition of V is then calculated [2]

[1] Write $R0 = EYE(2)$ in the code instead of what it is in the definition

[2] This comes from the folloing fact: if z is $N(0, I_k)$, then $x = \beta_1 + C'z$ is $N(\beta_1, C'C)$. Therefore, if we wanted to generate a multivariate normal draw of a $N(\beta_1, V)$, we can achieve this by generating a draw of a $N(0, I_k)$ and then use the property discussed before. In order to do this, we need a matrix C such that $C'C = V$. This is precisely what Cholesky decomposition does.

```
PROC GEN_BETA;
     LOCAL V, BETA1, BETA_F, C, ACCEPT, COEF, ROOT, ROOTMOD;

     V = inv(INV(R0) + SIG2^(-1)*X'X);
     BETA1 =  V*(INV(R0)*BETA0 + SIG2^(-1)*X'Y);
     C = chol(V);

     ACCEPT = 0;
     DO WHILE ACCEPT ==0;

          BETA_F = BETA1 + C'rndn(5,1); @GENERATE BETA@

          COEF = -REV(BETA_F[2:5])|1;
          ROOT = POLYROOT(COEF);
          ROOTMOD = ABS(ROOT);

          IF MINC(ROOTMOD) GE 1.0001;
              ACCEPT = 1;
          ELSE;
              ACCEPT = 0;
          ENDIF;

    ENDO;

RETP(BETA_F);
ENDP;
```

Figura 6: Joint generation of $[\phi_1, \phi_2, \phi_3, \phi_4]$

The new part is what comes next. Basically, we generate a draw, and see if with those ϕ the model is stationary or not. If it wasn't stationary, we generate another draw. We keep doing this until we get a stationary draw.

As before, the code shows us posterior statistics and several plots. We present some of these. Run the code yourself to see them all.

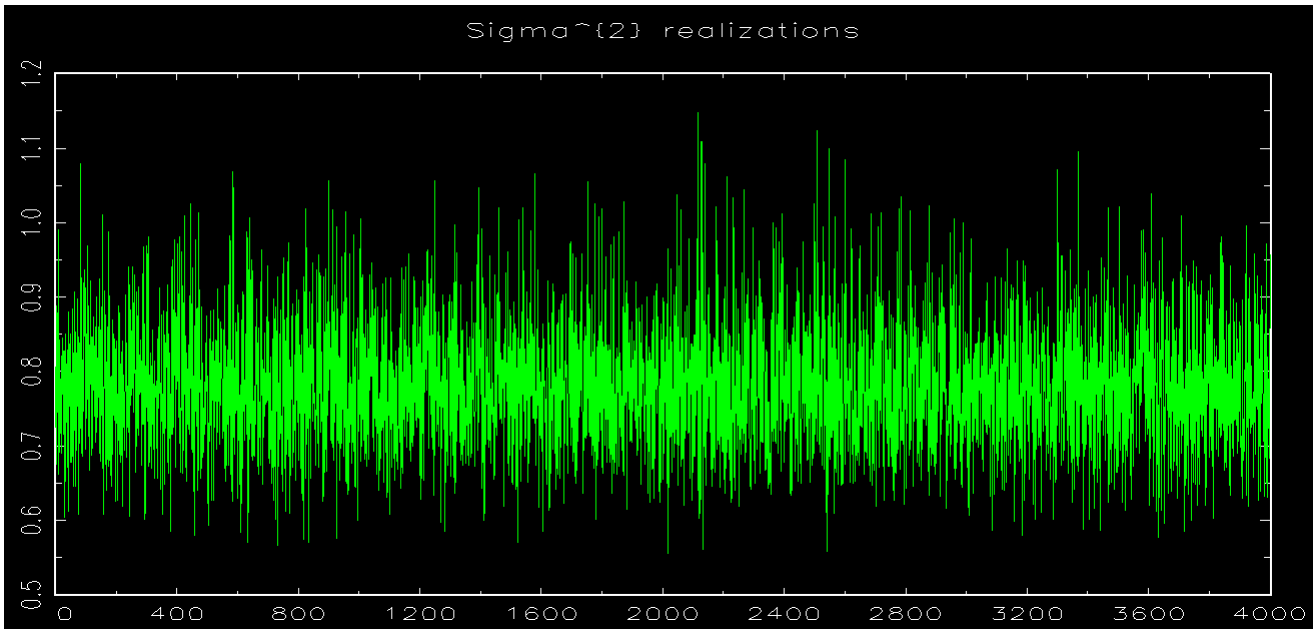
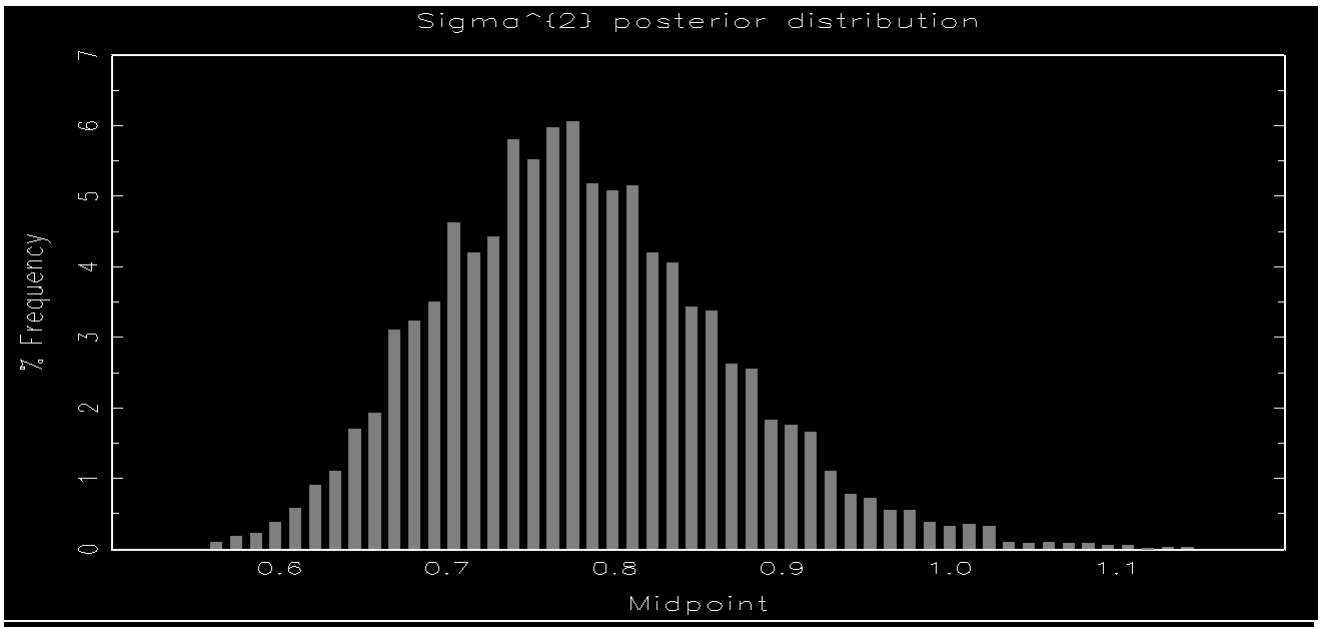
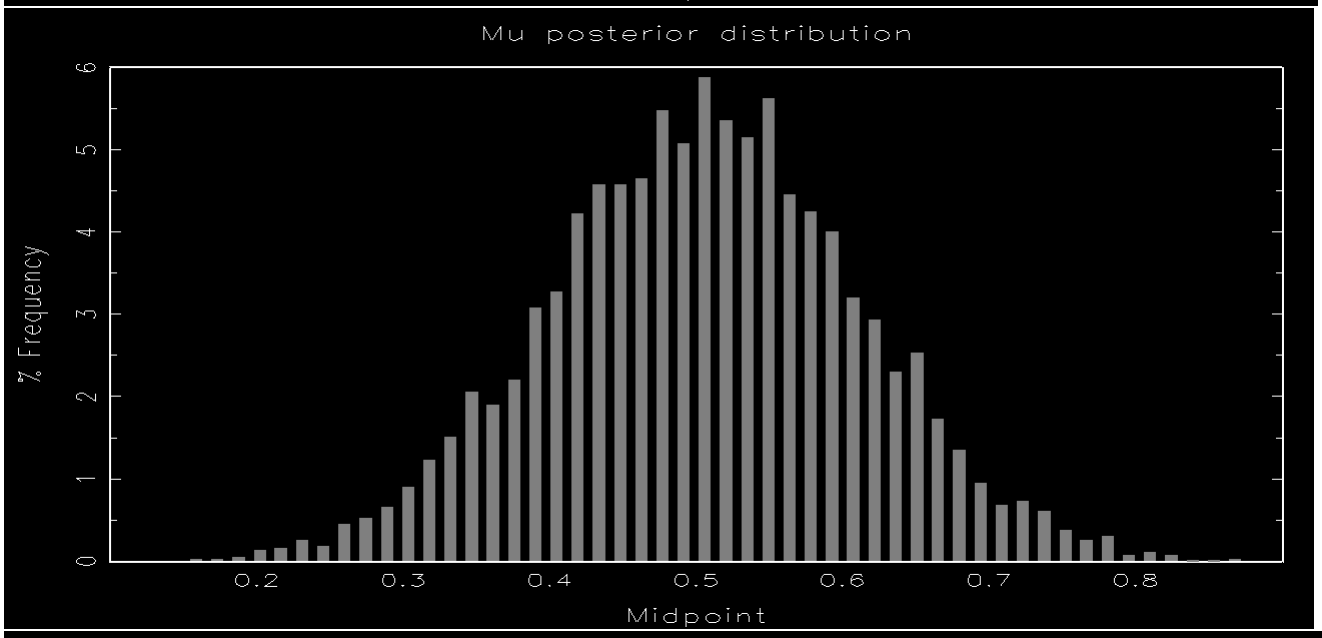
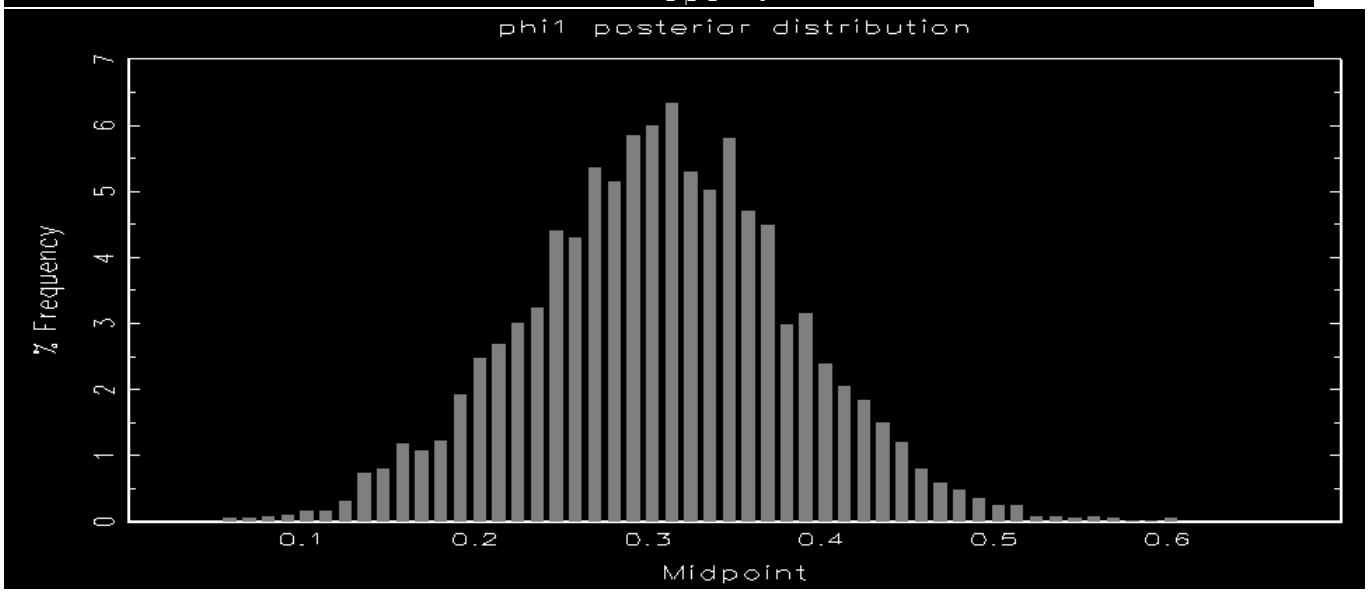


Figura 7: σ^2 realizations

On the other hand, the code *mle_ar4* estimates the model by ML. Comparison among different methods is shown in the table below.


Sigma^{2} posterior distribution
% Frequency
0
1
2
3
4
5
6
7
0.6
0.7
0.8
0.9
1.0
1.1
Midpoint


Mu posterior distribution
% Frequency
0
1
2
3
4
5
6
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Midpoint


phi1 posterior distribution
% Frequency
0
1
2
3
4
5
6
7
0.1
0.2
0.3
0.4
0.5
0.6
Midpoint

	Bayesian			MLE	
Parameter	Mean	Sd	Median	Estimation	Std error
μ	0.50192	0.10774	0.50709	0.50502	0.10429
ϕ_1	0.30750	0.07657	0.30591	0.30650	0.07578
ϕ_2	0.09574	0.08000	0.09770	0.09722	0.07901
ϕ_3	-0.08794	0.08031	-0.08766	-0.09051	0.07915
ϕ_4	-0.03057	0.07739	-0.03132	-0.03167	0.07586
σ^2	0.78072	0.08121	0.77591	0.74922	0.08102

Cuadro 3: Bayesian and MLE compared

Once again, we can see that results of MLE and Bayesian methods are similar when we use diffuse priors.

Problem 3

In this exercise we will use the code *gibs_ms0*. The procedure is explained in class notes (which are heavily based in Kim and Nelson(1998), ch. 9). In this case, we have to form a prior of $\sigma^2, \mu_0, \mu_1, p, q$ and $\tilde{S_T}$, that is, the states in each period. We will use a Gibbs sampling approach in order to obtain the posteriors.
It can be shown that the posterior distribution can be obtained from:

$$g(\sigma^2, \mu_0, \mu_1, p, q, \tilde{S_T}|\tilde{y_T}) = g(\sigma^2, \mu_0, \mu_1|\tilde{S_T}, \tilde{y_T})g(p, q|\tilde{S_T})g(\tilde{S_T}|\tilde{y_T})$$

Intuitively, this follows from the following facts:

- Conditional on knowing in which state we are, the inference is similar the one done in the first problem: we have only one equation, in this case with only a constant, and update our priors using the date. To put it briefly, conditional on knowing the state, we are doing the bayesian analogous to a simple regression.
- p, q are the information about state transitions. Therefore, if we know the states (that is, we condition on them), then nothing else is needed to infer p, q. That is, we assume that if we knew the actual states, every other information is redundant.

With this decomposition, we can use Gibbs sampling to obtain the marginals from the conditionals. We follow the following steps:

1. Set starting values for $\sigma^2, \mu_0, \mu_1, p, q$
2. Using $g(\tilde{S_T}|\tilde{y_T}, \sigma^2, \mu_0, \mu_1, p, q)$ generate $\tilde{S_T}$ (the whole block [3]).

[3]This is what this code does. there is another procedure explained in lecture notes

3. Conditional on the previous realization of $\tilde{S_T}$, use $g(p, q|\tilde{S_T})$ to generate transition probs.

4. Conditional on the realizations of $\tilde{S_T}$, use $g(\sigma^2, \mu_0, \mu_1|\tilde{S_T}, \tilde{y_T})$ to generate realizations of the parameters

5. Using this parameteres, go back to 2). Iterate to convergence.

The code follows that structure. The "action" happens at the bottom of the code, where the procs [4] to generate the draws are defined. At the top, the code just uses this generators to produce and collect draws. We specify the Gibbs sampler iterations just as we did before. The code may take some minutes to run if the number of iterations is too high.
We have to scroll up in the command bar to see the output. At the end, the code prints what we need, and also the whole vector of estimated state probabilities.Descriptive statistics of the posteriors of the parameters are shown in the table.

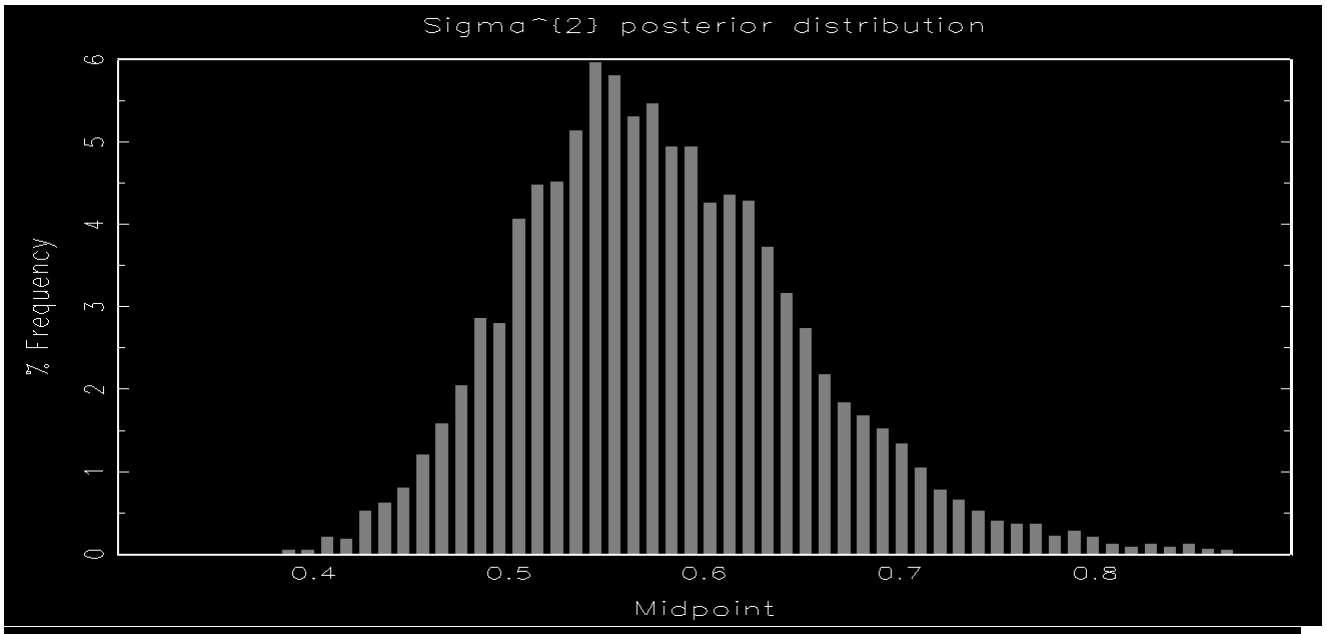
Parameter	Mean	sd	Median
σ^2	0.5788	0.0764	0.5699
μ_0	-0.3755	0.2610	-0.3614
μ_1	1.3524	0.2287	1.3493
p	0.9273	0.0307	0.9314
q	0.7367	0.0869	0.7444

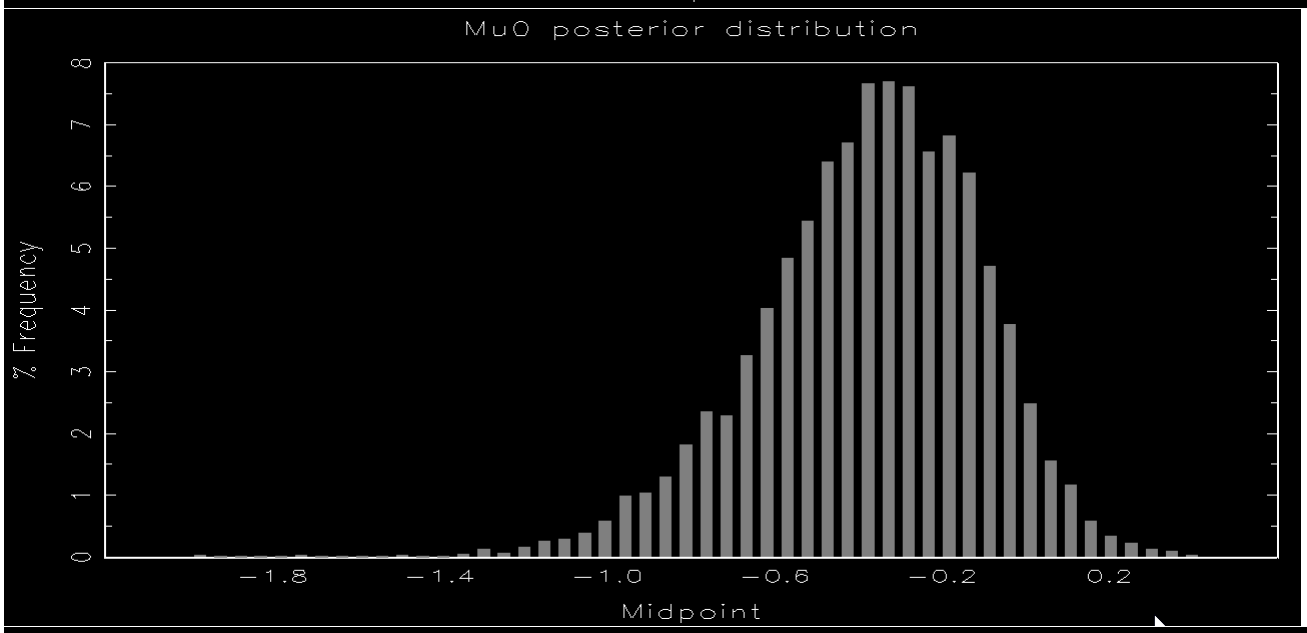
Cuadro 4: Descriptive stats of posteriors

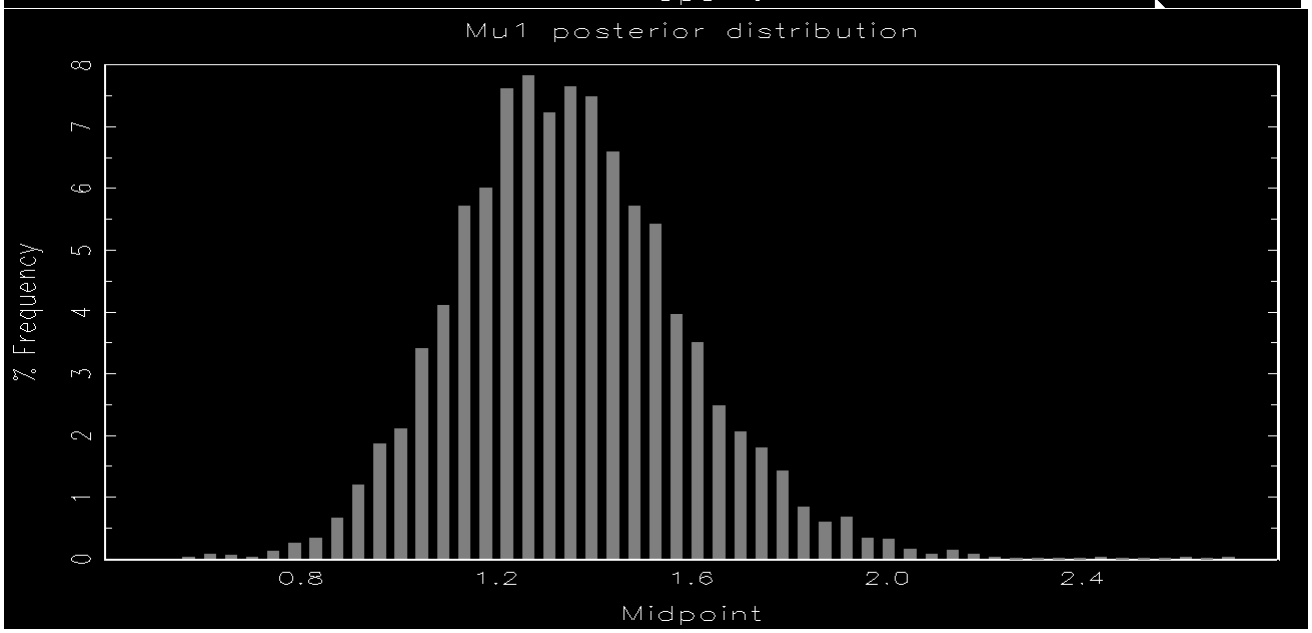
We have histograms of the posterior distributions. The code also plots the inferred probabilities of being in each state. These are calculated by taking the mean of the realizations of S_t for all the iterations. We can see in the graphs that σ^2 seems to have a skewed distribution. This is reasonable, given that the posterior of σ^2 has Inverted Gamma distribution. p, q have Beta distribution, whereas μ_0 and μ_1 have normal distributions. We can see all this by looking at the procs that generate them[5]

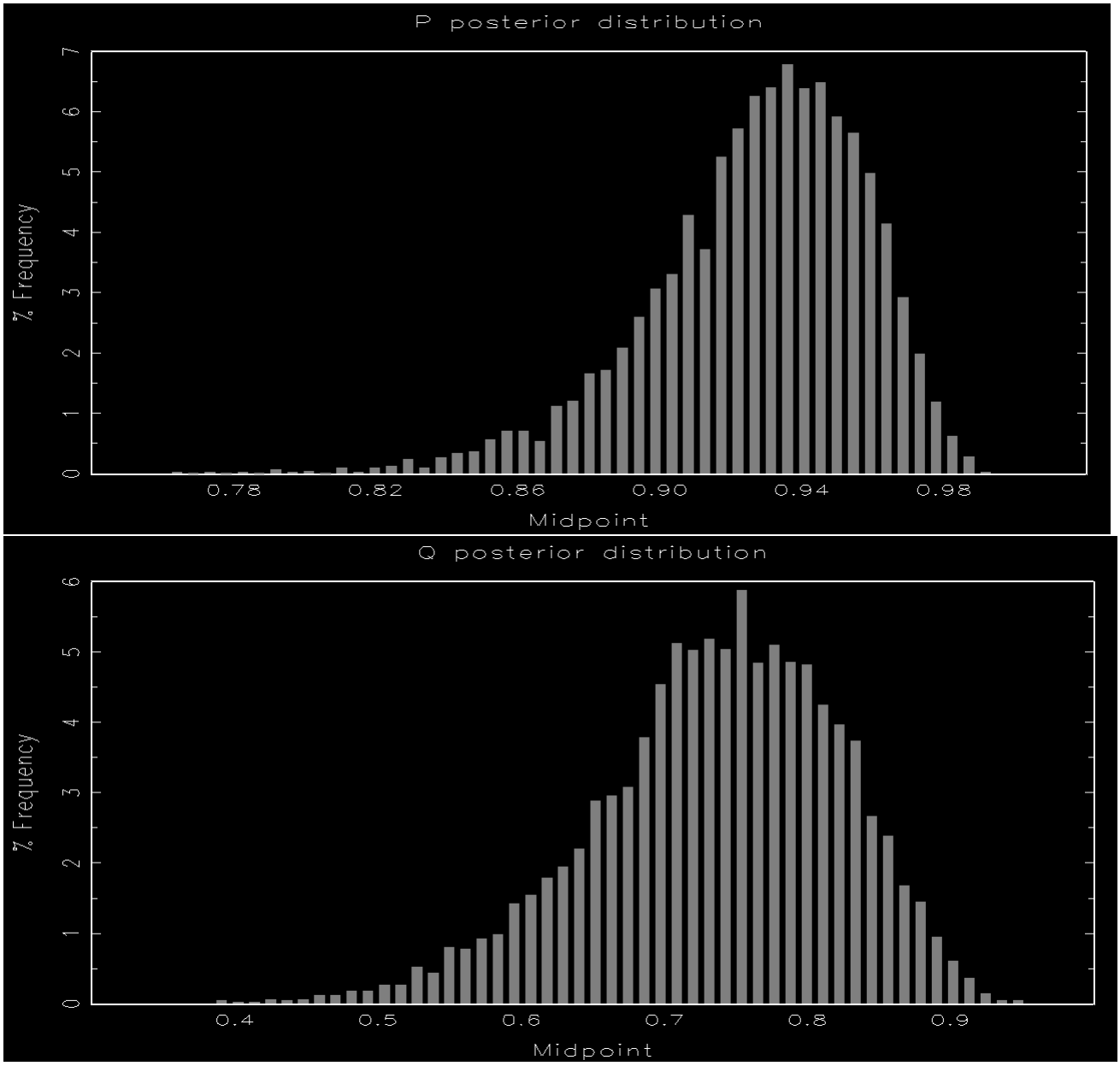
[4]Think of these as sub codes inside the main code. These are similar to MATLAB functions

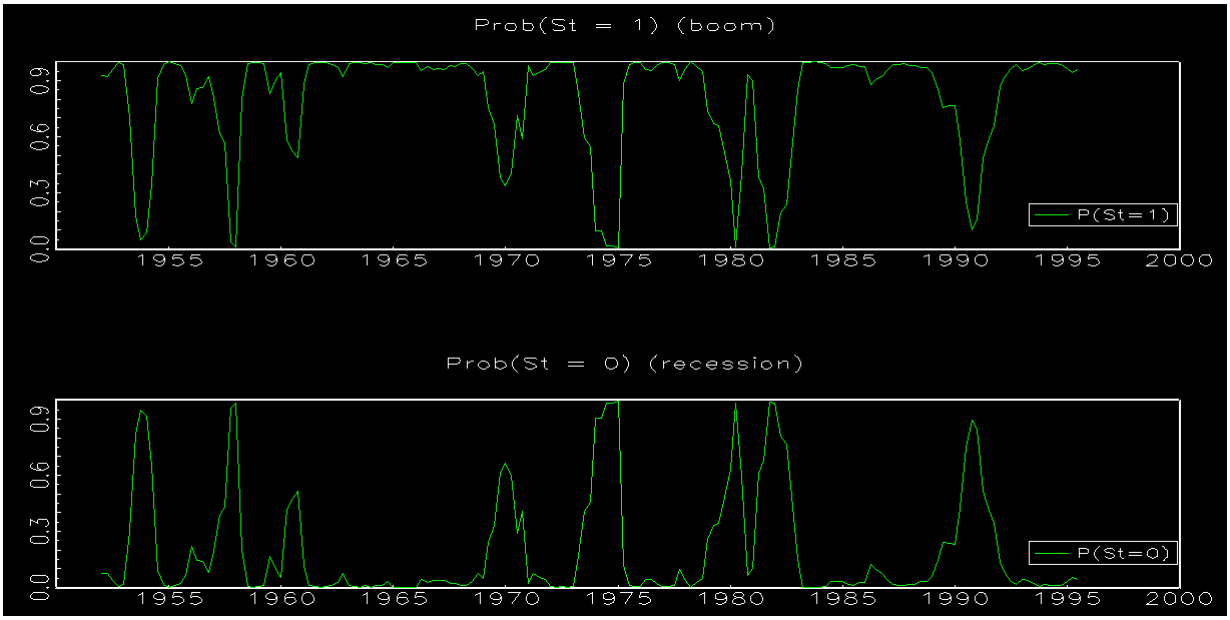
[5]The generation of p, q doesn't occurs at the bottom of the code, but at the top. It doesn't have a separate proc because Gauss have a command to generate draws from a Beta distibution, so we just use it. σ^2 is inverted gamma (no command for this) and μ_0, μ_1 are bivariate normal, therefore we have to generate them together with a particular variance matrix


Sigma^{2} posterior distribution
% Frequency
0
1
2
3
4
5
6
0.4
0.5
0.6
0.7
0.8
Midpoint


Mu0 posterior distribution
% Frequency
0
1
2
3
4
5
6
7
8
−1.8
−1.4
−1.0
−0.6
−0.2
0.2
Midpoint


Mu1 posterior distribution
% Frequency
0
1
2
3
4
5
6
7
8
0.8
1.2
1.6
2.0
2.4
Midpoint


P posterior distribution
% Frequency
0
1
2
3
4
5
6
7
0.78
0.82
0.86
0.90
0.94
0.98
Midpoint
Q posterior distribution
% Frequency
0
1
2
3
4
5
6
0.4
0.5
0.6
0.7
0.8
0.9
Midpoint


Prob(St = 1) (boom)
0.0 0.3 0.6 0.9
1955 1960 1965 1970 1975 1980 1985 1990 1995 2000
P(St=1)
Prob(St = 0) (recession)
0.0 0.3 0.6 0.9
1955 1960 1965 1970 1975 1980 1985 1990 1995 2000
P(St=0)