FIN 970: Homework 1

Alex von Hafften

February 22, 2022

1 Problem 1: GMM Estimation of a Linear Regression Model

Write a code to implement a GMM estimation of a linear regression model, $Y_t = \beta' X_t + u_t$. The code should produce the point estimates and the Newey-West standard errors of β and the regression R^2. We will use this code in later assignments to evaluate statistical significance of predictability evidence

Solution: See `gmm.jl` for implementation (also in appendix). Also see `gmm_test.jl` for a test of the GMM estimation using simulated data (also in appendix).

2 Problem 2: Bayesian Estimation of an Autoregressive Model

Consider an $AR(1)$ model for $y^T = \{y_t\}_{t=1}^T$:

$$y_{t+1} = \mu + \rho y_t + \sigma \varepsilon_{t+1}$$

where $\varepsilon \sim_{iid} N(0, 1)$

1. Consider independent conjugate priors for the model parameters,

$$\mu \sim N(m, s^2), \rho \sim N(\tilde{\rho}, \omega^2), \sigma^2 \sim IG(\alpha/2, \beta/2)$$

Show that the conditional posteriors are given by

$$\mu | y^T, \rho, \sigma \sim N, \rho | y^T, \mu, \sigma \sim N, \sigma^2 | y^T, \mu, \rho \sim IG$$

Find the parameters of the posterior distributions in terms of the parameters of the prior and the data.

Solution: The priors for the model parameters imply:

$$\begin{aligned}
&\mu \sim N(m, s^2) \\
\implies f(\mu) &= \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{1}{2s^2}(\mu - m)^2 \right) \propto \exp\left(-\frac{1}{2s^2}(\mu^2 - 2\mu m) \right) \\
&\rho \sim N(\tilde{\rho}, \omega^2) \\
\implies f(\rho) &= \frac{1}{\sqrt{2\pi\omega^2}} \exp\left(-\frac{1}{2\omega^2}(\rho - \tilde{\rho})^2 \right) \propto \exp\left(-\frac{1}{2\omega^2}(\rho^2 - 2\rho\tilde{\rho}) \right) \\
&\sigma^2 \sim IG(\alpha/2, \beta/2) \\
\implies f(\sigma^2) &= \frac{(\beta/2)^{(\alpha/2)}}{\Gamma(\alpha/2)} (\sigma^2)^{-\alpha/2-1} \exp\left(-\frac{\beta}{2\sigma^2} \right) \propto \sigma^{2(-\alpha/2-1)} \exp\left(-\frac{\beta}{2\sigma^2} \right)
\end{aligned}$$

From the $AR(1)$ structure, we know that

$$
\begin{aligned}
y_t|\mu,\rho,\sigma,y_{t-1} &\sim N(\mu+\rho y_{t-1},\sigma^2) \\
f(y_t|y_{t-1},\mu,\rho,\sigma) &= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\Bigg(-\frac{1}{2\sigma^2}(y_t-\mu-\rho y_{t-1})^2\Bigg) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\Bigg(-\frac{1}{2\sigma^2}(y_t^2+\mu^2+\rho^2y_{t-1}^2-2\mu y_t-2\rho y_{t-1}y_t+2\rho\mu y_{t-1})\Bigg) \\
&\propto \frac{1}{\sigma}\exp\Bigg(-\frac{1}{2\sigma^2}(y_t^2+\mu^2+\rho^2y_{t-1}^2-2\mu y_t-2\rho y_{t-1}y_t+2\rho\mu y_{t-1})\Bigg)
\end{aligned}
$$

Furthermore, assuming y_0 is given (so $f(y_0|\mu,\rho,\sigma)=1$):

$$
\begin{aligned}
f(y^T|\mu,\rho,\sigma) &= f(y_T|\mu,\rho,\sigma,y_{T-1})\cdot\ldots\cdot f(y_1|\mu,\rho,\sigma,y_0)f(y_0|\mu,\rho,\sigma) \\
&= \prod_{t=1}^{T} f(y_t|\mu,\rho,\sigma,y_{t-1}) \\
&\propto \prod_{t=1}^{T}\frac{1}{\sigma}\exp\Bigg(-\frac{1}{2\sigma^2}(y_t^2+\mu^2+\rho^2y_{t-1}^2-2\mu y_t-2\rho y_{t-1}y_t+2\rho\mu y_{t-1})\Bigg) \\
&= \frac{1}{\sigma^T}\exp\Bigg(\sum_{t=1}^{T}-\frac{1}{2\sigma^2}(y_t^2+\mu^2+\rho^2y_{t-1}^2-2\mu y_t-2\rho y_{t-1}y_t+2\rho\mu y_{t-1})\Bigg) \\
&= \frac{1}{\sigma^T}\exp\Bigg(-\frac{1}{2\sigma^2}(\sum_{t=1}^{T}y_t^2+T\mu^2+\rho^2\sum_{t=1}^{T}y_{t-1}^2-2\mu\sum_{t=1}^{T}y_t-2\rho\sum_{t=1}^{T}y_{t-1}y_t+2\rho\mu\sum_{t=1}^{T}y_{t-1})\Bigg) \\
&= \frac{1}{\sigma^T}\exp\Bigg(-\frac{T}{2\sigma^2}(\overline{y_T^2}+\mu^2+\rho^2\overline{y_{T-1}^2}-2\mu\overline{y_T}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})\Bigg)
\end{aligned}
$$

where

$$
\begin{aligned}
\overline{y_T^2} &\equiv \frac{1}{T}\sum_{t=1}^{T}y_t^2 \\
\overline{y_{T-1}^2} &\equiv \frac{1}{T}\sum_{t=1}^{T}y_{t-1}^2 \\
\overline{y_T} &\equiv \frac{1}{T}\sum_{t=1}^{T}y_t \\
\overline{y_{T-1}} &\equiv \frac{1}{T}\sum_{t=1}^{T}y_{t-1} \\
\overline{z_T} &\equiv \frac{1}{T}\sum_{t=1}^{T}y_ty_{t-1}
\end{aligned}
$$

Applying Bayes' Rule for μ, we know that:

$$
\begin{aligned}
f(\mu|y^T,\rho,\sigma) &\propto f(\mu)f(y^T|\mu,\rho,\sigma)\\
&\propto \frac{1}{\sigma^T}\exp\Bigg(-\frac{1}{2s^2}(\mu^2-2\mu m)\Bigg)\exp\Bigg(-\frac{T}{2\sigma^2}(\overline{y_T^2}+\mu^2+\rho^2\overline{y_{T-1}^2}-2\mu\overline{y_T}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})\Bigg)\\
&\propto \exp\Bigg(-\frac{1}{2s^2}(\mu^2-2\mu m)\Bigg)\exp\Bigg(-\frac{T}{2\sigma^2}(\mu^2+2\rho\mu\overline{y_{T-1}}-2\mu\overline{y_T})\Bigg)\\
&= \exp\Bigg(-\frac{1}{2s^2}(\mu^2-2\mu m)-\frac{T}{2\sigma^2}(\mu^2+2\rho\mu\overline{y_{T-1}}-2\mu\overline{y_T})\Bigg)\\
&= \exp\Bigg(-\frac{1}{2}\Bigg(\mu^2\Bigg(\frac{1}{s^2}+\frac{T}{\sigma^2}\Bigg)-2\mu\Bigg(\frac{m}{2s^2}+\frac{T(\rho\overline{y_{T-1}}-\overline{y_T}))}{2\sigma^2}\Bigg)\Bigg)\Bigg)\\
&= \exp\Bigg(-\frac{1}{2\Big(\frac{1}{s^2}+\frac{T}{\sigma^2}\Big)^{-1}}\Bigg(\mu^2-2\mu\Bigg(\frac{m}{2s^2}+\frac{T(\rho\overline{y_{T-1}}-\overline{y_T}))}{2\sigma^2}\Bigg)\Bigg(\frac{1}{s^2}+\frac{T}{\sigma^2}\Bigg)^{-1}\Bigg)\Bigg)
\end{aligned}
$$

Thus, $\mu|y^T,\mu,\rho,\sigma \sim N(\tilde{m},\tilde{s}^2)$ where $\nu_\mu \equiv \frac{\sigma^2}{s^2}$ and:

$$
\begin{aligned}
\tilde{s}^2 &\equiv \Bigg(\frac{1}{s^2}+\frac{T}{\sigma^2}\Bigg)^{-1}\\
&= \Bigg(\frac{\nu_\mu}{\sigma^2}+\frac{T}{\sigma^2}\Bigg)^{-1}\\
&= \frac{\sigma^2}{\nu_\mu+T}\\
\tilde{m} &\equiv \Bigg(\frac{m}{2s^2}+\frac{T(\rho\overline{y_{T-1}}-\overline{y_T}))}{2\sigma^2}\Bigg)\Bigg(\frac{1}{s^2}+\frac{T}{\sigma^2}\Bigg)^{-1}\\
&= \Bigg(\frac{m\nu_\mu+T(\rho\overline{y_{T-1}}-\overline{y_T}))}{2\sigma^2}\Bigg)\frac{\sigma^2}{\nu_\mu+T}\\
&= m\frac{\nu_\mu}{\nu_\mu+T}+(\rho\overline{y_{T-1}}-\overline{y_T})\frac{T}{\nu_\mu+t}
\end{aligned}
$$

Applying Bayes' Rule for ρ, we know that:

$$
\begin{aligned}
f(\rho|y^T,\sigma,\mu) &\propto f(\rho)f(y^T|\mu,\rho,\sigma) \\
&\propto \frac{1}{\sigma^T}\exp\Bigg(-\frac{1}{2\omega^2}(\rho^2-2\rho\tilde{\rho})\Bigg)\exp\Bigg(-\frac{T}{2\sigma^2}(\overline{y_T^2}+\mu^2+\rho^2\overline{y_{T-1}^2}-2\mu\overline{y_T}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})\Bigg) \\
&\propto \exp\Bigg(-\frac{1}{2\omega^2}(\rho^2-2\rho\tilde{\rho})-\frac{T}{2\sigma^2}(\rho^2\overline{y_{T-1}^2}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})\Bigg) \\
&= \exp\Bigg(-\frac{1}{2}\Bigg(\rho^2\Bigg(\frac{1}{\omega^2}+\frac{T\overline{y_{T-1}^2}}{\sigma^2}\Bigg)+2\rho\Bigg(\frac{\tilde{\rho}}{\omega^2}+\frac{T\overline{z_T}}{\sigma^2}-\frac{T\mu\overline{y_{T-1}}}{\sigma^2}\Bigg)\Bigg)\Bigg) \\
&= \exp\Bigg(-\frac{1}{2\Big(\frac{1}{\omega^2}+\frac{T\overline{y_{T-1}^2}}{\sigma^2}\Big)^{-1}}\Bigg(\rho^2+2\rho\Bigg(\frac{\tilde{\rho}}{\omega^2}+\frac{T\overline{z_T}}{\sigma^2}-\frac{T\mu\overline{y_{T-1}}}{\sigma^2}\Bigg)\Bigg(\frac{1}{\omega^2}+\frac{T\overline{y_{T-1}^2}}{\sigma^2}\Bigg)^{-1}\Bigg)\Bigg)
\end{aligned}
$$

Thus, $\rho|y^T,\mu,\rho,\sigma \sim N(\tilde{\tilde{\rho}},\tilde{\omega}^2)$ where $\nu_\rho \equiv \frac{\sigma^2}{\omega^2}$ and:

$$
\begin{aligned}
\tilde{\omega}^2 &\equiv \Bigg(\frac{1}{\omega^2}+\frac{T\overline{y_{T-1}^2}}{\sigma^2}\Bigg)^{-1} \\
&= \Bigg(\frac{\nu_\rho}{\sigma^2}+\frac{T\overline{y_{T-1}^2}}{\sigma^2}\Bigg)^{-1} \\
&= \frac{\sigma^2}{\nu_\rho+T\overline{y_{T-1}^2}} \\
\tilde{\tilde{\rho}} &= \Bigg(\frac{\tilde{\rho}\nu_\rho}{\sigma^2}+\frac{T\overline{z_T}}{\sigma^2}-\frac{T\mu\overline{y_{T-1}}}{\sigma^2}\Bigg)\frac{\sigma^2}{\nu_\rho+T\overline{y_{T-1}^2}} \\
&= \frac{\nu_\rho}{\nu_\rho+T\overline{y_{T-1}^2}}\tilde{\rho}+\frac{T}{\nu_\rho+T\overline{y_{T-1}^2}}[\overline{z_T}-\mu\overline{y_{T-1}}]
\end{aligned}
$$

Applying Bayes' Rule for σ, we know that:

$$
\begin{aligned}
f(\sigma|y^T,\rho,\mu) &\propto f(\rho)f(y^T|\mu,\rho,\sigma) \\
&\propto \frac{1}{\sigma^T}\sigma^{2(-\alpha/2-1)}\exp\Bigg(-\frac{\beta}{2\sigma^2}\Bigg)\exp\Bigg(-\frac{T}{2\sigma^2}(\overline{y_T^2}+\mu^2+\rho^2\overline{y_{T-1}^2}-2\mu\overline{y_T}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})\Bigg) \\
&= \sigma^{2(-(\alpha+T)/2-1)}\exp\Bigg(-\frac{1}{2\sigma^2}[\beta+T(\overline{y_T^2}+\mu^2+\rho^2\overline{y_{T-1}^2}-2\mu\overline{y_T}-2\rho\overline{z_T}+2\rho\mu\overline{y_{T-1}})]\Bigg)
\end{aligned}
$$

Thus, $\sigma | y^T, \mu, \rho, \sigma \sim IG(\tilde{\alpha}/2, \tilde{\beta}/2)$ where:

$$\tilde{\alpha} = \alpha + T$$
$$\tilde{\beta} = \beta + T(\overline{y_T^2} + \mu^2 + \rho^2 \overline{y_{T-1}^2} - 2\mu \overline{y_T} - 2\rho \overline{z_T} + 2\rho\mu \overline{y_{T-1}})$$

2. The Excel file `Longyielddata.xlsx` contains the annual data for long-term 10-year US government bond yields from Global Financial Data (GFD) database. Choose the priors for the model parameters. For example, we can use fairly uninformative priors:

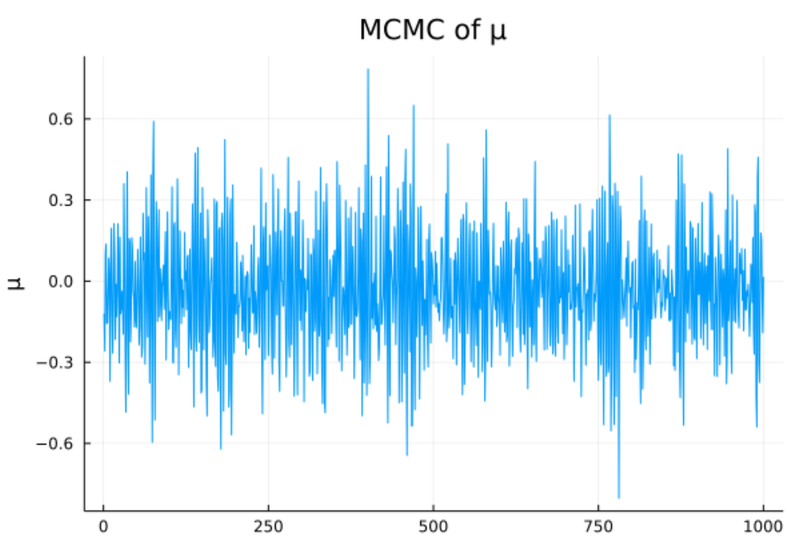
	Prior Mean	Prior Std. Dev.
μ	0.3	0.5
ρ	0.95	0.2
σ^2	1	1

Design and implement an MCMC algorithm to draw a long sample from the joint posterior distribution of the three parameters, where the sampling of each parameter is implemented via Gibbs sampler.

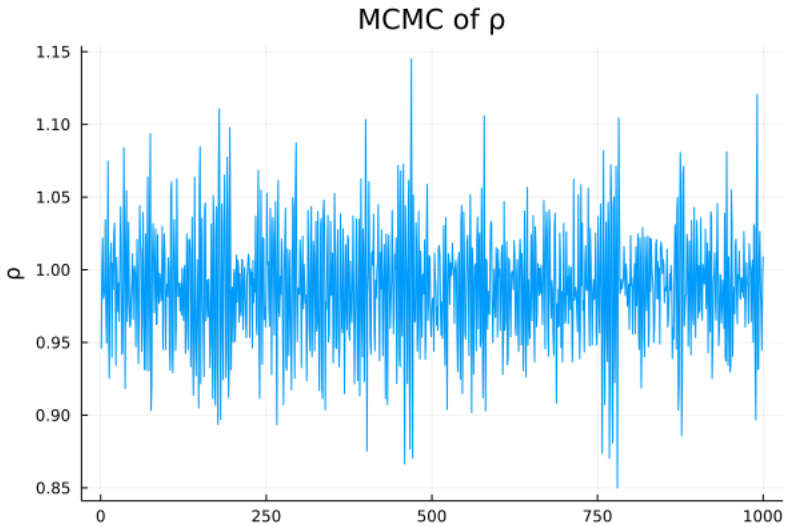
Solution: See `gibbs.jl` for the implementation simulates a MCMC using a Gibbs sampler (also in appendix). See `gibbs_run.jl` for code that runs a MCMC and plots the results (also in appendix).

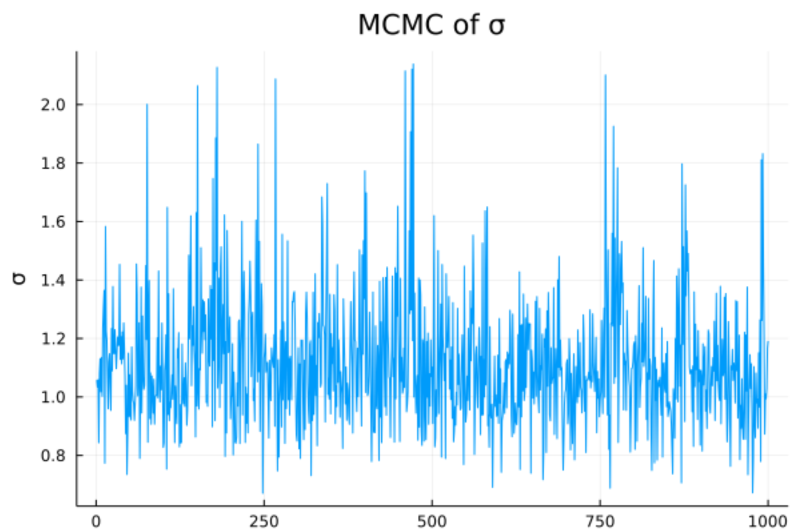
3. Run a long MCMC chain and discard appropriate number of initial draws due to burn-in. Plot the three chains. Do the chains mix well? Does it look like they come from a stationary distribution? How large is the persistence in the chains? Plot the autocorrelation plots for the chains, and the scatter plots of each parameter against the others. You might find that the most problematic is the persistence and cross-correlation between μ and ρ, and there's a good reason for it. Notice that μ is an intercept, and not the unconditional mean of y_t: $E(y) = \mu/(1-\rho)$. So every time we change ρ, intuitively, the draw for μ has to adjust to target the unconditional mean of the process. How would you change the problem to break up this almost mechanical correlation between the parameters?

Solution: I simulate MCMC of length 1000 with a 100 periods for burn in.[1] The chains are plotted below:



[1]Mean of inverse gamma distribution is $\beta/(\alpha-1)$ and variance is $\beta^2/((\alpha-1)^2(\alpha-2))$. For unit mean and unit variance/standard deviation, $\alpha = 3$ and $\beta = 2$.

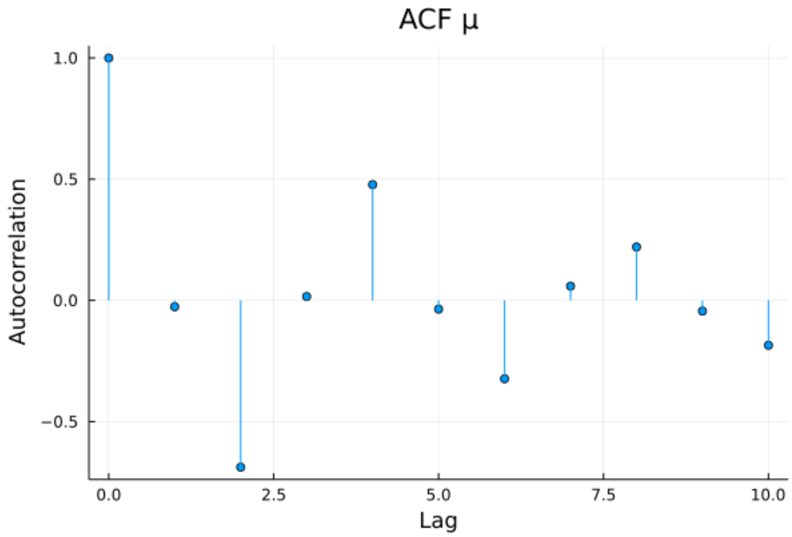


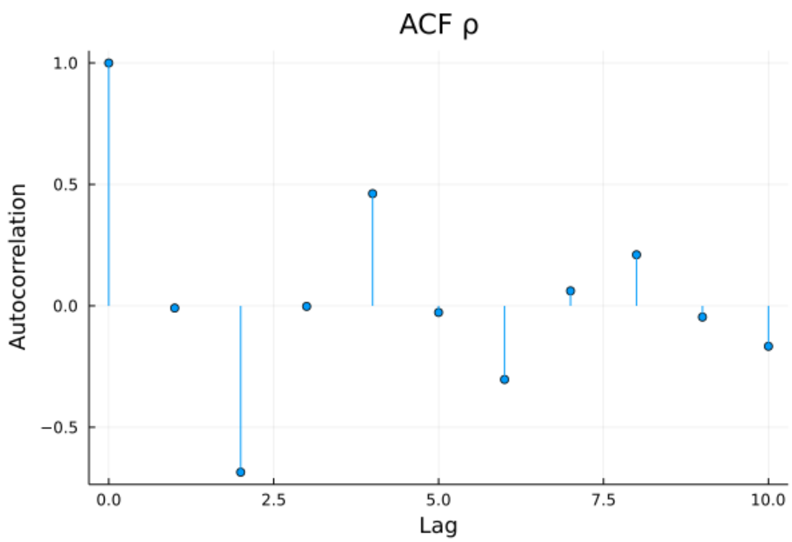


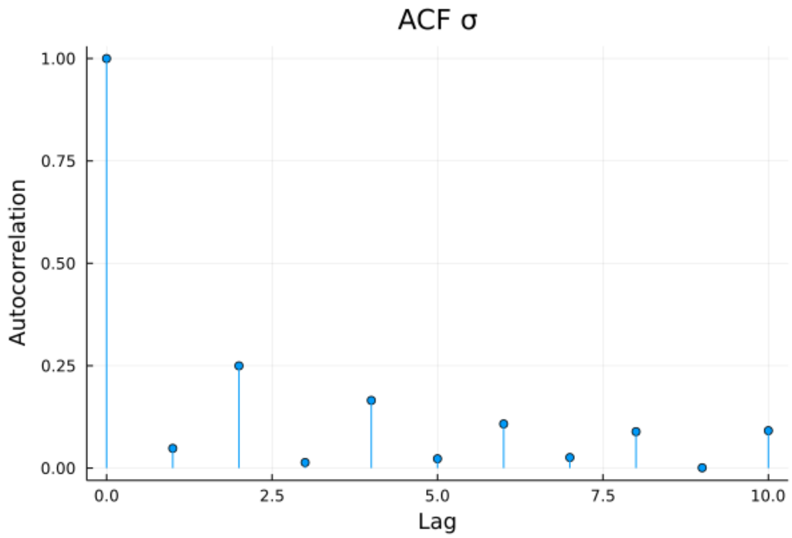
Yes, I think the chains mix pretty well. And yes, it looks like they come from a stationary distribution. The AR(1) coefficients for each chain are below. All coefficients are very close to zero, so the persistence is quite low.

	Constant	Lagged Value
μ	-0.0264	0.0040
ρ	1.0035	-0.0151
σ	1.0504	0.0612

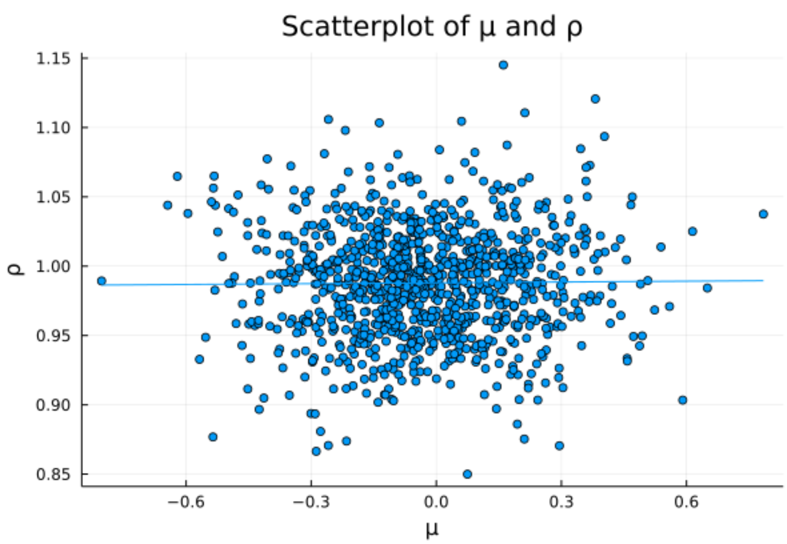
The autocorrelation functions are plotted below. These plots tell a different story. We see that the first lag almost zero correlation but the second is strongly (negatively for μ and ρ and positively for σ) correlated. The third is back to zero correlation, but the fourth is strongly positively correlated. This pattern repeats.

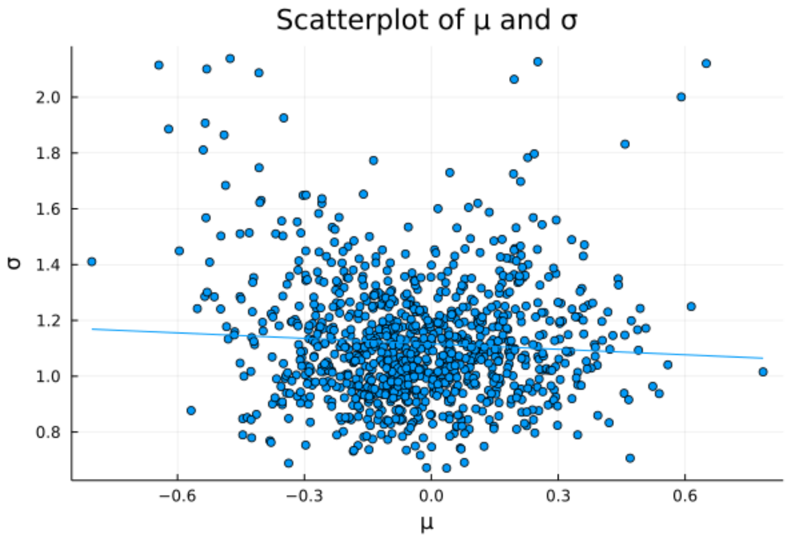


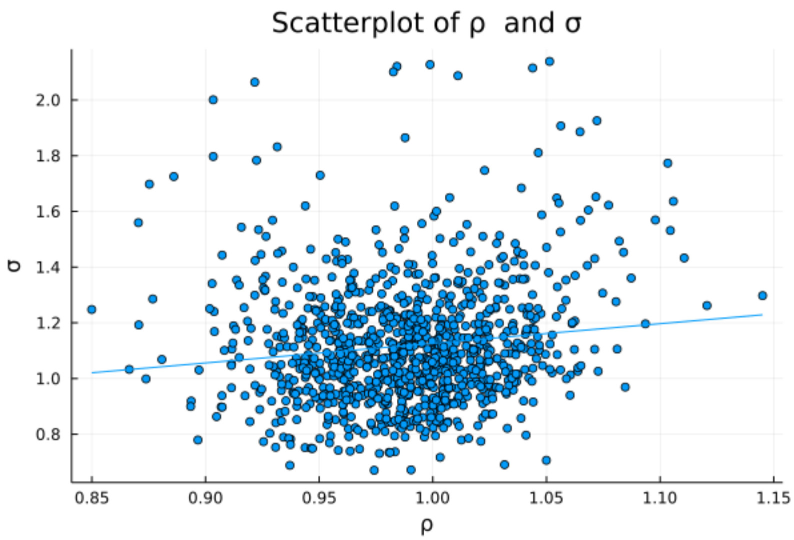


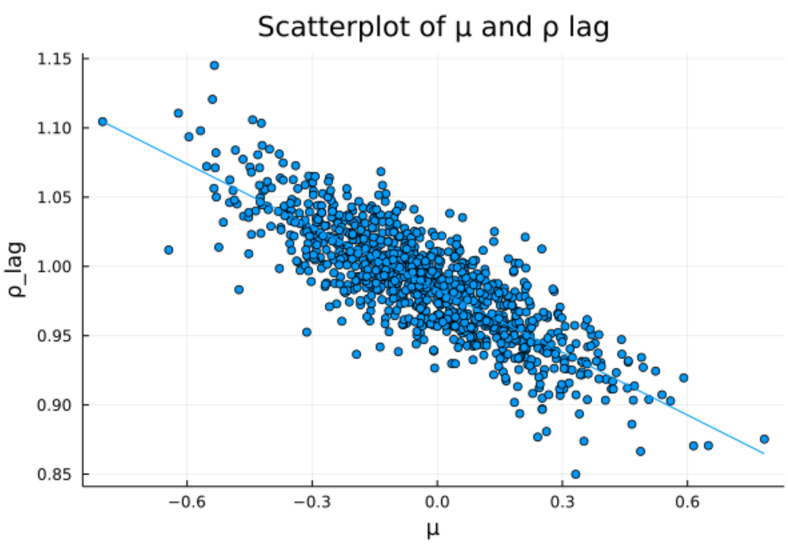


The scatterplots of the simulated parameters are below. These scatterplots suggest that the chains mix well because they look essentially like a cloud of points. However, we see a similar pattern to the ACF plots when we look at μ relative to lagged ρ.









We could fix this by simulating the chain and then only saving a simulation something like every twenty rounds.

4. Report the posterior means and standard deviations of the parameters - how different the posterior means are from the prior, and from the standard OLS estimates in the data? How do the prior and posterior standard deviations compare? Overall, do you think your results look reasonable?

	Prior Mean	Prior Std. Dev.	Post. Mean	Post. Std. Dev.
μ	0.3	0.5	-0.0267	0.2328
ρ	0.95	0.2	0.9886	0.0412
σ	1	1	1.1186	0.2448

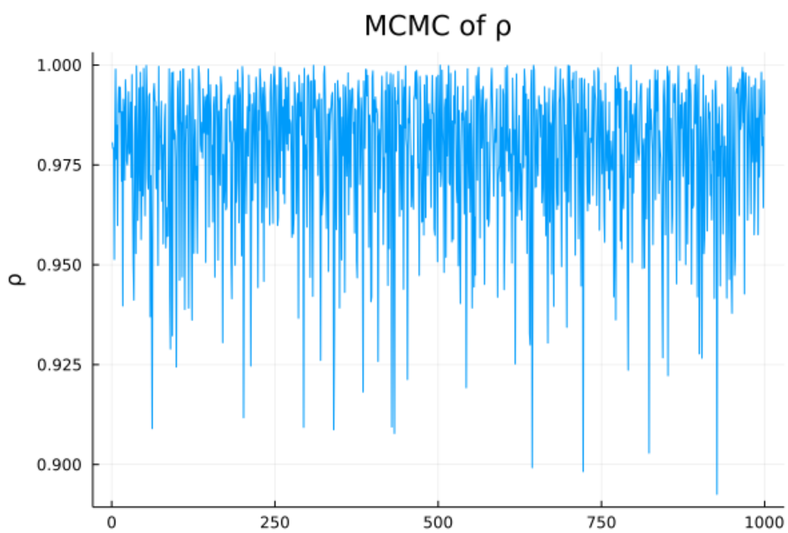
The posterior mean for μ is much different than the prior mean (-0.03 vs 0.3). The posterior mean for ρ is pretty close to the prior (0.99 vs 0.95). The posterior mean for σ is pretty close to the prior (1 vs 1.12). The posterior standard deviation for all parameters is much small than the prior standard deviation. That makes sense because we chose a relatively loose prior. I'm suspicious of these results due to the strong correlation between draws as discussed in question 3.

5. If you carefully examine your chain for ρ, you might find that some draws are above 1. This does not look reasonable, as we have good reasons to believe that interest rates are stationary. Let's correct that. First, let's use a truncated Normal prior for ρ to ensure that this parameter is always in $(-1, 1)$:

$$f(\rho) \propto N(\tilde{\rho}, \omega^2) \text{ if } \rho \in (-1, 1); f(\rho) = 0 \text{ otherwise.}$$

Further, instead of using Gibbs to draw ρ, now let's use Independence Metropolis-Hastings where the proposal density is equal to the Normal conditional posterior of ρ you found in part 1). Design and implement the new MCMC algorithm. Notice that a new algorithm (known as a Rejection Sampling) should be a very simple and intuitive modification of the previous one.

Solutions See `gibbs.jl` for implementation. If the option `truncate_rho` is set to `true`, then the normal distribution for ρ is truncated to be between -1 and 1. A resulting MCMC for ρ looks like:



6. Examine the chain, report the posterior means and standard deviations. Overall, do you think your results look reasonable?

...

7. Having simulation output from the MCMC chains makes it easy to compute, and assess significance of, any complicated non-linear functions of the parameters and the data. Suppose we are interested in $N = 5$ year forecast of yields from the model. Show that the forecast is given by,

$$E_t y_{t+N} = \mu \frac{1 - \rho^N}{1 - \rho} + \rho^N y_t$$

Fix t. You can compute the implied forecast $(E_t y_{t+N})^i$ for each parameter draw from the chain θ^i, for i from 1 to M. Having the distribution of time-t forecasts $\{(E_t y_{t+N})^i\}_{i=1}^M$, you can numerically compute their mean, and 2.5% - 97.5% confidence band. Now do it for all t, and plot the posterior mean and the confidence band for the yield forecasts from the model.

...

3 Problem 3: Latent Drift Model

Consider the following specification for consumption dynamics:

$$\Delta c_{t+1} = \mu + x_t + \sigma_c \eta_{t+1},$$
$$x_{t+1} = \rho x_t + \sigma_x e_{t+1},$$

where η and e are independent (over time and from each other) shocks with mean zero and variance one.

1. We are interested in estimating the parameters of the model: $\mu, \sigma_c, \rho, \sigma_x$. Consider the following four moments: $E(\Delta c_t), Var(\Delta c_t), Cov(\Delta c_t, \Delta c_{t-1}), Cov(\Delta c_t, \Delta c_{t-2})$. Show that these four moments exactly identify the four unknown parameters. Describe how you would design and implement a GMM estimation of the model parameters based on these four moments. How would you infer the unobserved x_t based on these estimates?

   Solutions: We can rewrite x_t:

$$\begin{aligned}
x_t &= \rho x_{t-1} + \sigma_x e_t \\
&= \rho(\rho x_{t-2} + \sigma_x e_{t-1}) + \sigma_x e_t \\
&= \rho^2 x_{t-2} + \sigma_x(\rho e_{t-1} + e_t) \\
&= \rho^2(\rho x_{t-3} + \sigma_x e_{t-2}) + \sigma_x(\rho e_{t-1} + e_t) \\
&= \rho^3 x_{t-3} + \sigma_x(\rho^2 e_{t-2} + \rho e_{t-1} + e_t) \\
&= \rho^t x_0 + \sigma_x \sum_{j=0}^{t-1} \rho^j e_{t-j} \\
&= \sigma_x \sum_{j=0}^{t-1} \rho^j e_{t-j}
\end{aligned}$$

   assuming $x_0 = 0$. Thus,

$$\begin{aligned}
E[x_t] &= \sigma_x \sum_{j=0}^{t-1} \rho^j E[e_{t-j}] \\
&= 0 \\
E[x_t^2] &= \sigma_x^2 \sum_{j=0}^{t-1} \rho^{2j} E[e_{t-j}^2] \\
&= t\sigma_x^2 \rho^{2j} \\
Var[x_t] &= E[x_t^2] - [E[x_t]]^2 \\
&= t\sigma_x^2 \rho^{2j}
\end{aligned}$$

$$\begin{aligned}
E[\Delta c_t] &= E[\mu + x_{t-1} + \sigma_c \eta_t] \\
&= \mu + E[x_{t-1}] + \sigma_c E[\eta_t] \\
&= \mu
\end{aligned}$$

$$
\begin{aligned}
E[(\Delta c_t)^2] &= E[(\mu + x_{t-1} + \sigma_c \eta_t)(\mu + x_{t-1} + \sigma_c \eta_t)] \\
&= E[\mu^2 + x_{t-1}^2 + \sigma_c^2 \eta_t] \\
&= \mu^2 + (t-1)\sigma_x^2 \rho^{2j}
\end{aligned}
$$

$$
\begin{aligned}
Var[\Delta c_t] &= E[(\Delta c_t)^2] - [E[\Delta c_t]]^2] \\
&= (t-1)\sigma_x^2 \rho^{2j}
\end{aligned}
$$

$$
\begin{aligned}
Cov(\Delta c_t, \Delta c_{t-1}) &= Cov(\mu + x_{t-1} + \sigma_c \eta_t, \mu + x_{t-2} + \sigma_c \eta_{t-1}) \\
&= Cov(\mu + (\rho x_{t-2} + \sigma_x e_{t-1}) + \sigma_c \eta_t, \mu + x_{t-2} + \sigma_c \eta_{t-1}) \\
&= \rho Cov(x_{t-2}, x_{t-2}) \\
&= \rho (t-2) \sigma_x^2 \rho^{2j}
\end{aligned}
$$

$$
\begin{aligned}
Cov(\Delta c_t, \Delta c_{t-2}) &= Cov(\mu + x_{t-1} + \sigma_c \eta_t, \mu + x_{t-3} + \sigma_c \eta_{t-2}) \\
&= Cov(\mu + (\rho x_{t-2} + \sigma_x e_{t-1}) + \sigma_c \eta_t, \mu + x_{t-3} + \sigma_c \eta_{t-2}) \\
&= Cov(\mu + (\rho(\rho x_{t-3} + \sigma_x e_{t-2}) + \sigma_x e_{t-1}) + \sigma_c \eta_t, \mu + x_{t-3} + \sigma_c \eta_{t-2}) \\
&= \rho^2 Cov(x_{t-3}, x_{t-3}) \\
&= \rho^2 \rho (t-3) \sigma_x^2 \rho^{2j}
\end{aligned}
$$

2. The shocks η and e are assumed to be independent from each other. Can we estimate the correlation between the shocks in the data? If so, show what data moments would identify it.

   ...

4 Appendix

4.1 Problem 1

The code for GMM estimation of linear model:

```
# Alex von Hafften
# FIN 970: Asset Pricing
# HW 1 Problem 1
# Professor Ivan Shaliastovich

# This code estimates a linear regression via gmm.
# Produces point estimates, Newey-West standard errors, and R^2

using Optim, LinearAlgebra

# struct to hold gmm estimation
mutable struct GMM
    # data
    Y::Array{Float64}       # dependent variable
    X::Array{Float64}       # independent variables
    Z::Array{Float64}       # instruments

    # sizes
    T::Int64                 # number of observations
    r::Int64                 # number of moments
    k::Int64                 # number of parameters
    L::Int64                 # number of lags for newey-west SEs

    # first stage
    beta_1::Array{Float64}     # point estimates
    S_1::Array{Float64}     # newey-West variance-covariance matrix esimate
    se_1::Array{Float64}    # newey-West standard errors

    # second stage
    W::Array{Float64}       # optimal weighting matrix
    beta_2::Array{Float64}     # point estimates
    S_2::Array{Float64}     # newey-West variance-covariance matrix esimate
    se_2::Array{Float64}    # newey-West standard errors

    # post estimation
    R_2::Float64             # r-squared
end

# estimate linear regression with gmm
function gmm(Y::Array{Float64}, X::Array{Float64}, Z::Array{Float64}, L::Int64)

    # get parameters
    T = size(X)[1]
    k = size(X)[2]
    r = size(Z)[2]

    # gmm objective function
    function gmm_obj(beta::Array{Float64}, W)
        moments = 1/T*Z'*(Y .- X*beta)
```

```
        return moments'*W*moments
    end

    # first stage coefficient estimates
    beta_1 = optimize(beta -> gmm_obj(beta, I), zeros(k)).minimizer

    # newey-west variance covariance estimate
    function newey_west_S(beta::Array{Float64}, L::Int64)

        h = (Y.-X*beta).*Z
        S = 1/T * h' * h

        for i = 1:L
            h_1 = h[(i+1):end, :]
            h_2 = h[1:(end-i), :]
            G_q = 1/T .* h_1' * h_2
            S += (1 - i/(L+1)) .* (G_q + G_q')
        end

        return S
    end

    # first stage newey west var-cov matrix and ses.
    S_1 = newey_west_S(beta_1, L)
    se_1 = sqrt.(diag(S_1))

    # optimal weighting matrix
    W = inv(S_1)

    # second stage point estimates
    beta_2 = optimize(beta -> gmm_obj(beta, W), zeros(k)).minimizer

    # second stage newey west var-cov matrix and ses.
    S_2 = newey_west_S(beta_2, L)
    se_2 = sqrt.(diag(S_2))

    # R-squared
    R_2 = 1 - var(Y .- X*beta_2)/var(Y)

    return GMM(Y, X, Z, T, r, k, L, beta_1, S_1, se_1, W, beta_2, S_2, se_2, R_2)
end
```

The code for testing:

```
# Alex von Hafften
# FIN 970: Asset Pricing
# HW 1 Problem 1
# Professor Ivan Shaliastovich

# This code tests the gmm estimation of a linear regression in ./gmm.jl.

cd("/Users/alexandervonhafften/Documents/UW Madison/problem_sets/fin_970/ps1")

using Distributions
```

```
include("gmm.jl")

# simulate data
n = 1000

X_1 = rand(Normal(0, 1), n)
X_2 = rand(Normal(0, 1), n)
X_3 = rand(Normal(0, 1), n)
epsilon = rand(Normal(0, 1), n)

Y = 1 .+ 2 .* X_1 .+ 3 .* X_2 .+ 4 .* X_3 + epsilon
X = hcat(ones(n), X_1, X_2, X_3)
Z = copy(X)

# estimate gmm
gmm_estimate = gmm(Y, X, Z, 1)
```

4.2 Problem 2

The code for simulating MCMC using Gibbs sampler:

```
# Alex von Hafften
# FIN 970: Asset Pricing
# HW 1 Problem 2
# Professor Ivan Shaliastovich

# This code use Gibbs sampling to estimate AR(1) process.
# Normal drift, normal persistance, inverse-gamma variance.

cd("/Users/alexandervonhafften/Documents/UW Madison/problem_sets/fin_970/ps1")

using XLSX, DataFrames, Parameters, Distributions

@with_kw struct Priors
    # for drift
    m = 0.3
    s = 0.5

    # for persistance
    rho_tilde = 0.95
    omega     = 0.2

    # for variance
    alpha = 6.0
    beta  = 4.0
end

@with_kw struct Data_Moments
    # data
    data = DataFrame(XLSX.readtable("Long_yield_data.xlsx", "data")...)
    yields = Float64.(data.IGUSA10D)
    y = yields[2:end]
    y_lag = yields[1:(end-1)]
```

```
    T = size(y)[1]

    # data moments
    y_2_bar = 1/T * sum(y.^2)
    y_lag_2_bar = 1/T * sum(y_lag.^2)
    y_bar = 1/T * sum(y)
    y_lag_bar = 1/T * sum(y_lag)
    z_bar = 1/T * sum(y .* y_lag)
end

mutable struct MCMC
    N::Int64                    # length of mcmc
    burn_in::Int64              # number of burn-in periods
    mu::Vector{Float64}         # vector of draws for m
    rho::Vector{Float64}        # vector of draws for rho
    sigma::Vector{Float64}      # vector of draws for omega
end

function Initialize()
    P = Priors()

    N = 1000
    burn_in = 100

    mu    = zeros(N+burn_in)
    rho   = zeros(N+burn_in)
    sigma = zeros(N+burn_in)

    mu[1]    = P.m
    rho[1]   = P.rho_tilde
    sigma[1] = P.beta/(P.alpha - 1)

    return MCMC(N, burn_in, mu, rho, sigma)
end

function Simulate_MCMC!(M::MCMC; truncate_rho::Bool = false)
    P = Priors()
    D = Data_Moments()

    for i = 2:(M.N+M.burn_in)
        # get previous values
        mu    = M.mu[i-1]
        rho   = M.rho[i-1]
        sigma = M.sigma[i-1]

        # draw mu
        nu_mu   = (sigma^2)/(P.s^2)
        m_tilde = P.m * (nu_mu/(nu_mu + D.T)) + (rho * D.y_lag_bar-D.y_bar)*(D.T/(nu_mu + D.T))
        s_tilde = sqrt(sigma^2/(nu_mu + D.T))
        M.mu[i] = rand(Normal(m_tilde, s_tilde))

        # draw rho
        nu_rho          = (sigma^2)/(P.omega^2)
        rho_tilde_tilde = (nu_rho/(nu_rho + D.T*D.y_lag_2_bar)) * P.rho_tilde +
```

```
                        (D.T/(nu_rho + D.T * D.y_lag_2_bar)*(D.z_bar - mu*D.y_lag_bar))
        omega_tilde     = sqrt((M.sigma[i-1]^2)/(nu_rho + D.T*D.y_lag_2_bar))
        rho_distribution = Normal(rho_tilde_tilde, omega_tilde)
        if truncate_rho
            rho_distribution = truncated(rho_distribution, -1.0, 1.0)
        end
        M.rho[i] = rand(rho_distribution)

        # draw sigma
        alpha_tilde = P.alpha + D.T
        beta_tilde = P.beta + D.T * (D.y_2_bar + mu^2 + rho^2*D.y_lag_2_bar -
                            2 * mu *D.y_bar - 2 * rho *D.z_bar + 2*rho*mu*D.y_lag_bar)
        M.sigma[i] = rand(InverseGamma(alpha_tilde/2, beta_tilde/2))
    end

    # drop burn-in
    M.mu = M.mu[(M.burn_in+1):(M.N + M.burn_in)]
    M.rho = M.rho[(M.burn_in+1):(M.N + M.burn_in)]
    M.sigma = M.sigma[(M.burn_in+1):(M.N + M.burn_in)]

    return M
end
```

The code that produces and plots a MCMC:

```
# Alex von Hafften
# FIN 970: Asset Pricing
# HW 1 Problem 2
# Professor Ivan Shaliastovich

# This code runs Gibbs sampling to estimate AR(1) process from ./gibbs.jl

cd("/Users/alexandervonhafften/Documents/UW Madison/problem_sets/fin_970/ps1")

include("gibbs.jl")

using Plots, StatsBase

M_1 = Initialize()
Simulate_MCMC!(M_1)

# question 3

# line plots

plot(M_1.mu, legend = false, title = "MCMC of ")
ylabel!("")
savefig("p2_q3_mu.png")

plot(M_1.rho, legend = false, title = "MCMC of ")
ylabel!("")
savefig("p2_q3_rho.png")
```

```
plot(M_1.sigma, legend = false, title = "MCMC of ")
ylabel!("")
savefig("p2_q3_sigma.png")

# ar(1) persistance

# mu
y = M_1.mu[2:end]
x = [ones(M_1.N-1) M_1.mu[1:end-1]]
inv(x'*x)*x'*y

# rho
y = M_1.rho[2:end]
x = [ones(M_1.N-1) M_1.rho[1:end-1]]
inv(x'*x)*x'*y

# sigma
y = M_1.sigma[2:end]
x = [ones(M_1.N-1) M_1.sigma[1:end-1]]
inv(x'*x)*x'*y

# autocorrelation plots

lags = 0:10

acf_mu = autocor(M_1.mu, lags)
plot(lags, acf_mu, line=:stems, marker=:circle, legend = false)
xlabel!("Lag")
ylabel!("Autocorrelation")
title!("ACF ")
savefig("p2_q3_mu_acf.png")

acf_rho = autocor(M_1.rho, lags)
plot(lags, acf_rho, line=:stems, marker=:circle, legend = false)
xlabel!("Lag")
ylabel!("Autocorrelation")
title!("ACF ")
savefig("p2_q3_rho_acf.png")

acf_sigma = autocor(M_1.sigma, lags)
plot(lags, acf_sigma, line=:stems, marker=:circle, legend = false)
xlabel!("Lag")
ylabel!("Autocorrelation")
title!("ACF ")
savefig("p2_q3_sigma_acf.png")



# scatterplots

scatter(M_1.mu, M_1.rho, legend = false, title = "Scatterplot of  and ", smooth=true)
xlabel!("")
ylabel!("")
savefig("p2_q3_mu_rho.png")
```

```
scatter(M_1.mu, M_1.sigma, legend = false, title = "Scatterplot of  and ", smooth=true)
xlabel!("")
ylabel!("")
savefig("p2_q3_mu_sigma.png")

scatter(M_1.rho, M_1.sigma, legend = false, title = "Scatterplot of   and ", smooth=true)
xlabel!("")
ylabel!("")
savefig("p2_q3_rho_sigma.png")

# scatterplots with lags

scatter(M_1.mu[1:end-1], M_1.rho[2:end], legend = false, title = "Scatterplot of  and  lag", smooth=tru
xlabel!("")
ylabel!("_lag")
savefig("p2_q3_mu_rho_lag.png")


# question 4 - posterior mean and standard deviations

mean(M_1.mu)
std(M_1.mu)

mean(M_1.rho)
std(M_1.rho)

mean(M_1.sigma)
std(M_1.sigma)


# question 5 - truncated rho Distributions

M_2 = Initialize()
Simulate_MCMC!(M_2; truncate_rho = true)

plot(M_2.rho, legend = false, title = "MCMC of ")
ylabel!("")
savefig("p2_q5_rho.png")
```