Econ 7218 Problem Set 2

Due by Monday, April 26, 2021

Consider a spatial autoregressive (SAR) model for studying social interactions,

$$y_{i,g} = \lambda \sum_{j=1}^{n_g} W_{ij,g} y_{j,g} + x_{i,g}\beta_1 + \sum_{j=1}^{n_g} W_{ij,g} x_{j,g}\beta_2 + \alpha_g + \epsilon_{i,g} \tag{1}$$

where $g = 1, \cdots, G$ denote the groups (networks). $W_{ij,g} = 1$ if individual i links to individual j and zero otherwise. The vector form of this SAR model is

$$y_g = \lambda W_g y_g + x_g\beta_1 + W_g x_g \beta_2 + \alpha_g \ell_g + \epsilon_g, \tag{2}$$

where ℓ_g is a $m_g \times 1$ vector of ones. Assume $\epsilon_g | W_g, x_g \sim N(0, \sigma^2 I_{n_g})$, denote the common model parameters by $\theta = (\lambda, \beta', \sigma^2)'$ and group fixed effects by $\alpha = \{\alpha_1, \cdots, \alpha_G\}$, the log-likelihood of the SAR model is given by

$$\begin{aligned}\mathcal{L}(\theta; \alpha) = &- \sum_g \frac{n_g}{2} \ln(2\pi) - \sum_g \frac{n_g}{2} \ln \sigma^2 + \sum_g \ln\left(det(I_{n_g} - \lambda W_g)\right) \\ &- \frac{1}{2\sigma^2} \sum_g \epsilon_g(\lambda, \beta, \alpha)' \epsilon_g(\lambda, \beta, \alpha),\end{aligned} \tag{3}$$

where $det(A)$ is the determinant of matrix A and $\epsilon_g(\lambda, \beta, \alpha) = (I_{n_g} - \lambda W_g) y_g - x_g\beta_1 - W_g x_g \beta_2 - \alpha_g \ell_g$. Notice that for the SAR model to be stationary, we need the coefficient λ to be properly bounded. One sufficient bound is $|\lambda| \leq 1/||W_g||_\infty$, for $g = 1, \cdots, G$, where $||W_g|| = \max_i \sum_{j=1}^{n_g} W_{ij,g}$ is the maximum row sum of W_g. You are given a sample of student networks and school outcome (GPA). The sample consists of 76 networks. Individual characteristics and GPA are available in the file *group*. The list of variables in each column is summarized in Table 1. The network adjacency matrix of each group is provided in the file *network*.

Questions:

1. Define the dependent variable y as student's GPA and the independent variables x using age, male, black, Asian, hisp, race other, less hs, more hs, momedu miss, welfare, momjob miss, prof, and job other. Estimate the common parameters (θ) of the SAR model in Equation (1) by the maximum likelihood method to study the peer effect and determination of student's GPA using the sample of 76 student networks. Report your parameter estimates and standard errors.

   Hint: In this data set the maximum number of friends for each individual is 10, so you can set the constraint of λ to be $|\lambda| \leq 1/10$.

2. Consider the Bayesian estimation of the SAR model in Equation (1):

   2-1 $\beta = (\beta_1', \beta_2')'$ is a $2k \times 1$ parameter vector, where k denotes the dimension of x. Assume the prior distribution of β, $\beta \sim \mathcal{N}(\beta_0, B_0)$, where β_0 is a $2k \times 1$ vector and B_0 is a $2k \times 2k$ matrix denoting fixed hyperparameters. Derive the conditional posterior distribution of β, i.e., $p(\beta|\{y_g\}, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\})$.

   2-2 Assume the prior distribution of α_g, $\alpha_g \sim \mathcal{N}(\alpha_0, A_0)$, where α_0 and A_0 are fixed hyperparameters specified by a user. Derive the conditional posterior distribution of α_g, i.e., $p(\alpha_g|y_g, W_g, x_g, \lambda, \beta)$.

   2-3 Using the mixed MCMC strategy (M-H within Gibbs) to estimate the common parameters (θ) of the SAR model in Equation (1) to study the peer effect and determination of student's GPA. Report the posterior mean and posterior standard deviation for each parameter as a point estimate.

   2-4 Using the MCMC output to compute the Savage-Dickey density ratio for testing the hypothesis: $H_0 : \lambda = 0$ vs $H_1 : \lambda \neq 0$.

Turn in your code and a short write-up which includes the results and algorithms on NTU COOL.

Table 1: Variable definitions

Variable	Definition
age	student's age
male	male dummy
black	black dummy
Asian	asian dummy
hisp	hispanic dummy
race other	other race dummy
both par	living with both parents
less hs	mom's education less than high school
more hs	mom's education higher than high school
momedu miss	mom's education is missing
welfare	mom participate in welfare programs
momjob miss	mom's job is missing
prof	mom's job is professional
job other	mom's job is other types
sport	participate in sport clubs
white	white dummy
yr school	number of years stayed in this school
gpa	GPA
overage	dummy of older than the average age in the grade

1. Define the dependent variable y as student's GPA and the independent variables x using age, male, black, Asian, hisp, race other, less hs, more hs, momedu miss, welfare, momjob miss, prof, and job other. Estimate the common parameters (θ) of the SAR model in Equation (1) by the maximum likelihood method to study the peer effect and determination of student's GPA using the sample of 76 student networks. Report your parameter estimates and standard errors.

   Hint: In this data set the maximum number of friends for each individual is 10, so you can set the constraint of λ to be $|\lambda| \leq 1/10$.

```
[1] import pandas as pd
    import numpy as np
    import scipy.optimize as optimize
    import matplotlib.pyplot as plt
    from tqdm import tqdm
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
[3] path = '/content/drive/MyDrive/problem_set_2_sample/'
```

problem 1

```
[4] def logfun(X, y, network, sigma, lambda_, beta1, beta2, alpha):
        mg = len(X)
        I = np.eye(mg)
        epsilon = np.dot((I - lambda_*network),y) - np.dot(X, beta1).reshape(-1,1) - np.dot(network,np.dot(X, beta2)).reshape(-1,1) - alpha
        return (mg/2*np.log(2*np.pi)) + (mg/2*np.log(sigma**2)) - np.log(np.linalg.det(I-lambda_*network)) + (1/(2*sigma**2)*np.dot(epsilon.T,epsilon))

    def data_loadding(i):
        group = np.loadtxt(path + 'group/group%s.dat'%i)
        network = np.loadtxt(path + 'network/network%s.dat'%i)
        data = pd.DataFrame(group)
        X = data[[0,1,2,3,4,5,7,8,9,10,11,12,13]].values
        y = data[[17]].values
        return X, y, network

    def total_logfun(para):
        for i in range(1,77):
            sigma = para[0]
            lambda_ = para[1]
            beta1 = para[2:15]
            beta2 = para[15:28]
            alpha = para[i+27]
            value = logfun(load_X['group%s'%i], load_y['group%s'%i], load_network['group%s'%i], sigma, lambda_, beta1, beta2, alpha)
            value += value
        return value[0][0]
```

```
[5] load_X = {}
    load_y = {}
    load_network = {}
    for i in range(1,77):
        X, y, network = data_loadding(i)
        load_X['group%s'%i] = X
        load_y['group%s'%i] = y
        load_network['group%s'%i] = network
```

```
[6] def estimator(sig, lamb, beta1, beta2):
            alpha = np.zeros(76)+5
            beta1 = np.zeros(13)+beta1
            beta2 = np.zeros(13)+beta2
            sig_lamb = np.array([sig, lamb])
            value = 0
            para = np.concatenate((sig_lamb, beta1, beta2, alpha), axis=0)

            buds = [(None,None), (-0.1, 0.1)]
            for i in range(len(para)-2):
                buds.append((None,None))

            result = optimize.minimize(total_logfun, para, bounds=buds, method='SLSQP',options={'disp': True, 'maxiter' : 200})
            #result = optimize.minimize(total_logfun, para, bounds=buds,options={'disp': True, 'maxiter' : 1000})
            return result
```

```
[9] from datetime import datetime
    start=datetime.now()
    value = estimator(0.706, 0.068, 0.499, 0.491) #teacher table1 SAR modle
    print(datetime.now()-start)
```

```
para = value.x
std = value.jac
print('sigma :', para[0])
print('sigma standard error:', (1/(std[0]*std[0]))**0.5)
print('-------------------------------------------------------------------------------')
print('Lambda :', para[1])
print('Lambda standard error:', (1/(std[1]*std[1]))**0.5)
print('-------------------------------------------------------------------------------')
print('Beta1 :', para[2:16])
print('Beta1 standard error:', (1/(std[2:16]*std[2:16]))**0.5)
print('-------------------------------------------------------------------------------')
print('Beta2 :', para[16:30])
print('Beta2 standard error:', (1/(std[16:30]*std[16:30]))**0.5)
print('-------------------------------------------------------------------------------')
print('Alpha :', para[30:])
print('Alpha standard error:', (1/(std[30:]*std[30:]))**0.5)
```

```
sigma : -0.18569969400174893
sigma standard error: 54.87915423666719
-------------------------------------------------------------------------------
Lambda : -0.06069261736745064
Lambda standard error: 11.242462011863502
-------------------------------------------------------------------------------
Beta1 : [ 0.74843038  0.18470747  0.499      -3.85017135 -0.75946049 -0.71177146
  0.499      -1.89822857 -3.93403551  0.499      -2.30698297 -1.06460285
 -3.0346119  -0.23784592]
Beta1 standard error: [1.53162873e+01 2.66966075e+02            inf 1.73032343e+03
 4.98135867e+02 1.09340563e+03            inf 1.76751117e+03
 4.45018992e+02            inf 5.64509287e+03 2.81213812e+02
 7.69879589e+02 2.31596559e+00]
-------------------------------------------------------------------------------
Beta2 : [ 1.71677677  0.491       0.491       2.09419424  1.17650296  0.491
  0.42710457 -0.8447962   0.491       1.03779278  2.14813572  1.50757798
  5.          5.        ]
Beta2 standard error: [ 227.50618355           inf           inf  473.18411552 1319.37842089
           inf   46.77958087 1603.32721713           inf  221.88562662
   38.4153577   217.84065649           inf           inf]
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
Alpha : [5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         5.         5.         5.         5.         5.
 5.         0.46819449]
Alpha standard error: [        inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf          inf          inf
           inf          inf          inf 212.73605194]
```

2. Consider the Bayesian estimation of the SAR model in Equation (1):

2-1 $\beta = (\beta_1', \beta_2')'$ is a $2k \times 1$ parameter vector, where k denotes the dimension of x. Assume the prior distribution of β, $\beta \sim \mathcal{N}(\beta_0, B_0)$, where β_0 is a $2k \times 1$ vector and B_0 is a $2k \times 2k$ matrix denoting fixed hyperparameters. Derive the conditional posterior distribution of β, i.e., $p(\beta|\{y_g\}, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\})$.

we know $p(\beta) = (2\pi)^{-\frac{2k}{2}} (\sigma^2)^{-\frac{1}{2}} |B_0|^{-\frac{1}{2}} e^{\left(-\frac{1}{2}(\beta-\beta_0)' B_0^{-1} (\beta-\beta_0)\right)}$

$p(y_g | \beta, W_g, x_g, \lambda, \alpha_g) = (2\pi)^{-\frac{n_g}{2}} (\sigma^2)^{-\frac{n_g}{2}} |I_{n_g} - \lambda W_g| e^{\left(-\frac{1}{2\sigma^2}\varepsilon_g'\varepsilon_g\right)}$

Then

$$p(\beta | \{y_g\}, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\}) = \frac{p(\beta)\, p(\{y_g\} | \beta, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\})}{p(\{y_g\}, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\})}$$

$$\propto p(\beta)\, p(\{y_g\} | \beta, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\})$$

$$= p(\beta) \prod_{g=1}^{G} p(y_g | \beta, W_g, x_g, \lambda, \alpha_g)$$

we get

$$\ln p(\beta | \{y_g\}, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\}) \propto -\frac{G}{2} \ln |B| - \frac{1}{2} \sum_{i=1}^{G} (\beta - \hat{\beta})' B^{-1} (\beta - \hat{\beta})$$

and $p(\beta | \{y_g\} | \beta, \{W_g\}, \{x_g\}, \lambda, \{\alpha_g\}) \propto \mathcal{N}(\hat{\beta}, B)$

2-2 Assume the prior distribution of α_g, $\alpha_g \sim \mathcal{N}(\alpha_0, A_0)$, where α_0 and A_0 are fixed hyperparameters specified by a user. Derive the conditional posterior distribution of α_g, i.e., $p(\alpha_g | y_g, W_g, x_g, \lambda, \beta)$.

$p(\alpha_g) = (2\pi)^{-\frac{k_g}{2}} (\sigma^2)^{-\frac{1}{2}} |A_0|^{-\frac{1}{2}} e^{\left(-\frac{1}{2}(\alpha-\alpha_0)' A_0^{-1} (\alpha-\alpha_0)\right)}$

$p(y_g | \alpha_g, W_g, x_g, \lambda, \beta) = (2\pi)^{-\frac{n_g}{2}} (\sigma^2)^{-\frac{n_g}{2}} |I_{n_g} - \lambda W_g| e^{\left(-\frac{1}{2\sigma^2}\varepsilon_g'\varepsilon_g\right)}$

Then

$$p(\alpha_g | y_g, W_g, x_g, \lambda, \beta) = \frac{p(\alpha)\, p(y_g | \alpha_g, W_g, x_g, \lambda, \beta)}{p(y_g \alpha_g, W_g, x_g, \lambda, \beta)}$$

$$\propto p(\alpha)\, p(y_g | \alpha_g, W_g, x_g, \lambda, \beta)$$

$$\ln p(\alpha_g | y_g, W_g, x_g, \lambda, \beta) \propto -\frac{1}{2} \ln |R_g| - \frac{1}{2} (\alpha_g - \hat{\alpha}_g)' R_g^{-1} (\alpha_g - \hat{\alpha}_g)$$

and $p(\alpha_g | y_g, W_g, x_g, \lambda, \beta) \propto \mathcal{N}(\hat{\alpha}_g, R_g)$

2-3 Using the mixed MCMC strategy (M-H within Gibbs) to estimate the common parameters (θ) of the SAR model in Equation (1) to study the peer effect and determination of student's GPA. Report the posterior mean and posterior standard deviation for each parameter as a point estimate.

problem 2-3

```
[25] def logfun(X, y, network, sigma, lambda_, beta1, beta2, alpha):
        mg = len(X)
        I = np.eye(mg)
        epsilon = np.dot((I - lambda_*network),y) - np.dot(X, beta1).reshape(-1,1) - np.dot(network,np.dot(X, beta2)).reshape(-1,1) - alpha
        return -(mg/2*np.log(2*np.pi)) - (mg/2*np.log(sigma**2)) + np.log(np.linalg.det(I-lambda_*network)) - (1/(2*sigma**2)*np.dot(epsilon.T,epsilon))

    def data_loadding(i):
        group = np.loadtxt(path + 'group/group%s.dat'%i)
        network = np.loadtxt(path + 'network/network%s.dat'%i)
        data = pd.DataFrame(group)
        X = data[[0,1,2,3,4,5,7,8,9,10,11,12,13]].values
        y = data[[17]].values
        return X, y, network

    def total_logfun(sigma, lambda_, beta1, beta2, alpha):
        sigma = sigma
        lambda_ = lambda_
        beta1 = beta1
        beta2 = beta2
        for i in range(1,77):
            alpha_ = alpha[i-1]
            value = logfun(load_X['group%s'%i], load_y['group%s'%i], load_network['group%s'%i], sigma, lambda_, beta1, beta2, alpha_)
            value += value
        return value[0][0]
```

```
[ ] load_X = {}
    load_y = {}
    load_network = {}
    for i in range(1,77):
        X, y, network = data_loadding(i)
        load_X['group%s'%i] = X
        load_y['group%s'%i] = y
        load_network['group%s'%i] = network
```

```
[26] # Lambda

    def lambda_(t):
        lambda_t = np.random.multivariate_normal([lambda_T[t-1]], np.eye(1)*0.1**2)
        like_2 = total_logfun(sigma2_T[t-1], lambda_T[t-1], beta1_T[t-1], beta2_T[t-1], alpha_T[t-1])
        like_1 = total_logfun(sigma2_T[t-1], lambda_t, beta1_T[t-1], beta2_T[t-1], alpha_T[t-1])
        pp_1 = like_1 - like_2
        pp_1 = min([np.exp(pp_1), 1])
        if np.random.rand(1)[0] <= pp_1:
            lambda_T[t] = lambda_t
        else:
            lambda_T[t] = lambda_T[t-1]
        return lambda_T
```

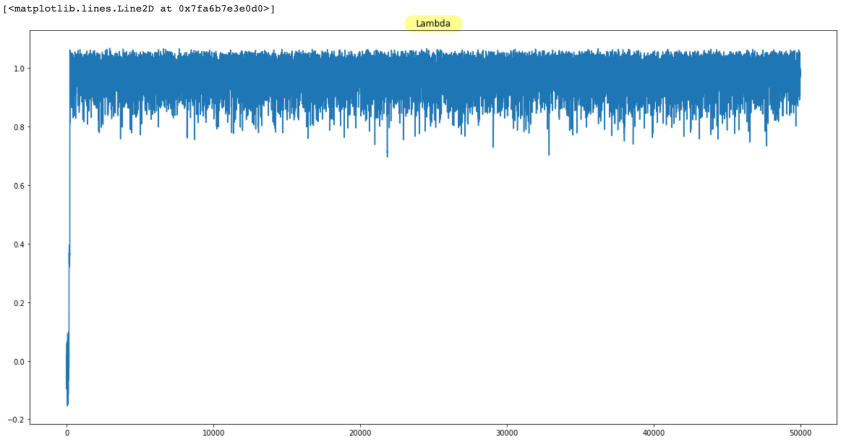
```
[27] # Beta

    def beta_(t):
        zvx1, zvx2 = 0, 0
        zvy1, zvy2 = 0, 0
        for i in range(1,77):
            mg = len(load_X['group%s'%i])
            I = np.eye(mg)
            ss = I - lambda_T[t]*load_network['group%s'%i] # mg*mg
            yy = np.dot(ss,load_y['group%s'%i]) - alpha_T[t-1][i-1]
            zz_x1 = load_X['group%s'%i] # mg*13
            zz_x2 = np.dot(load_network['group%s'%i], load_X['group%s'%i]) # mg*13

            zvx1 = zvx1 + np.dot(zz_x1.T, zz_x1)/(sigma2_T[t-1]**2) # 13*13
            zvx2 = zvx2 + np.dot(zz_x2.T, zz_x2)/(sigma2_T[t-1]**2)

            zvy1 = zvy1 + np.dot(zz_x1.T, yy)/(sigma2_T[t-1]**2) # mg*1
            zvy2 = zvy2 + np.dot(zz_x2.T, yy)/(sigma2_T[t-1]**2)

        B1 = np.linalg.inv((np.linalg.inv(B_0)+zvx1))
        B2 = np.linalg.inv((np.linalg.inv(B_0)+zvx2))
        mean_beta1 = np.ravel((np.dot(B1, (np.dot(np.linalg.inv(B_0), beta1_0).reshape(-1, 1)+zvy1))))
        mean_beta2 = np.ravel((np.dot(B2, (np.dot(np.linalg.inv(B_0), beta2_0).reshape(-1, 1)+zvy2))))
        beta1_hat = np.random.multivariate_normal(mean_beta1, B1, 1)
        beta2_hat = np.random.multivariate_normal(mean_beta2, B2, 1)

        beta1_T[t], beta2_T[t]= beta1_hat, beta2_hat
        return beta1_T, beta2_T
```

[28]
```
# Sigma^2

def sigma2_(t):
    ep_ = np.zeros(1)
    for i in range(1, 77):
        y = load_y['group%s'%i]
        X = load_X['group%s'%i]
        network = load_network['group%s'%i]
        mg = len(load_X['group%s'%i])
        I = np.eye(mg)
        epsilon = np.dot((I - lambda_T[t]*network),y) - np.dot(X, beta1_T[t]).reshape(-1,1)
        - np.dot(network,np.dot(X, beta2_T[t])).reshape(-1,1) - alpha_T[t-1][i-1]
        ep_ = np.append(ep_, epsilon)
    ep_ = ep_[1:]
    rho_1 = rho_0 + len(ep_)
    chi = np.random.chisquare(rho_1)
    sigma2_T[t] = (np.dot(ep_.reshape(1,-1), ep_.reshape(-1,1))[0][0] + eta_0)/chi
    return sigma2_T
```

[29]
```
# Alpha

def alpha_(t):
    dd = 1/(1/alpha_0 + 1/sigma2_T[t])
    for i in range(1,77):
        mg = len(load_X['group%s'%i])
        I = np.eye(mg)
        ss = I - lambda_T[t]*load_network['group%s'%i] # mg*mg
        yy = np.dot(ss,load_y['group%s'%i]) # mg*1
        zz_x1 = load_X['group%s'%i] # mg*13
        zz_x2 = np.dot(load_network['group%s'%i], load_X['group%s'%i]) # mg*13
        alpha_T[t][i-1] = (dd/sigma2_T[t]*sum(yy - np.dot(load_X['group%s'%i], beta1_T[t]).reshape(-1,1)
            - np.dot(load_network['group%s'%i],np.dot(load_X['group%s'%i], beta2_T[t])).reshape(-1,1))[0]) + np.random.normal(0, dd**0.5, 1)
    return alpha_T
```

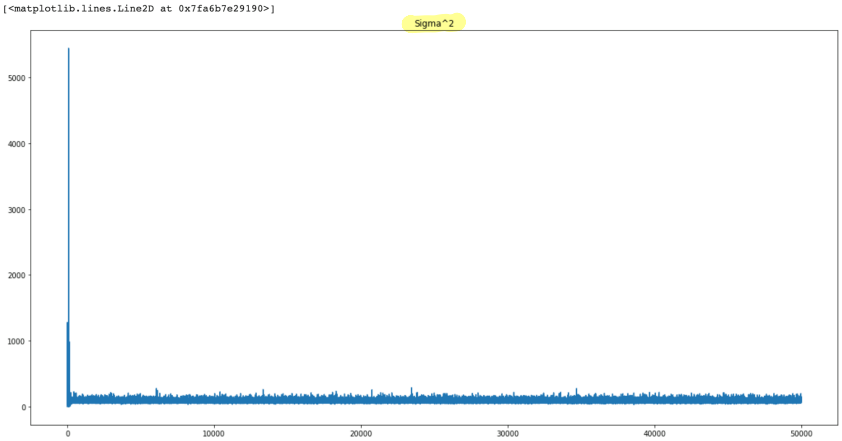
[40]
```
plt.figure(figsize=(20,10))
plt.title('Lambda')
plt.plot(lambda_T)
```

```
[<matplotlib.lines.Line2D at 0x7fa6b7e3e0d0>]
```



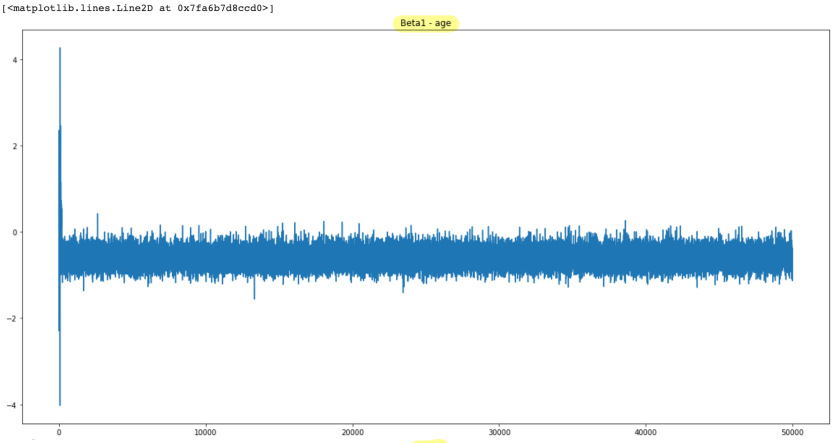

```
plt.figure(figsize=(20,10))
plt.title('Sigma^2')
plt.plot(sigma2_T)
```

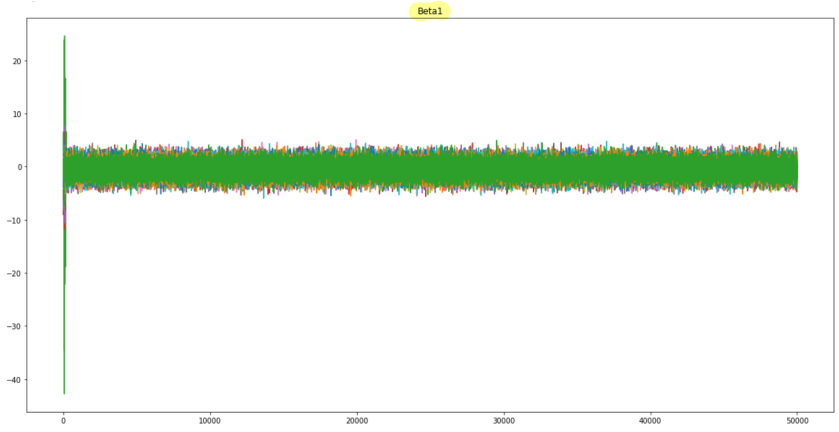
```
[<matplotlib.lines.Line2D at 0x7fa6b7e29190>]
```

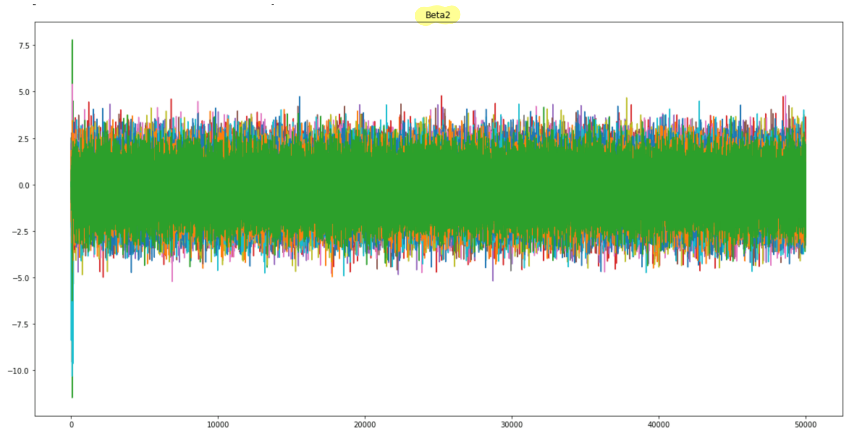


```
plt.figure(figsize=(20,10))
plt.title('Beta1 - age')
plt.plot(beta1_T.T[0])
```

```
[<matplotlib.lines.Line2D at 0x7fa6b7d8ccd0>]
```








```
lambda : 0.9720813793519459
--------------------------------------------------------------------------------
sigma^2 : 88.48471277769215
--------------------------------------------------------------------------------
Beta1 : [-0.6063103  -0.39091051 -0.46814114 -0.42674791 -0.37835096 -0.38468831
 -0.4176231  -0.46593212 -0.41663859 -0.43427467 -0.40690225 -0.51134077
 -0.45988717]
--------------------------------------------------------------------------------
Beta2 : [-0.19519107 -0.2104816  -0.31992145 -0.23944977 -0.21719103 -0.20579426
 -0.20259032 -0.31773385 -0.22804107 -0.24283528 -0.20916937 -0.30806492
 -0.26702373]
--------------------------------------------------------------------------------
Alpha : [ 1.57890615  1.68079515  2.50922633  2.12698521  3.08323495  2.75502553
  2.4626334   1.89247806  2.2559129   4.17624147  7.95038423  9.07376831
  9.35445763 10.17505256  2.74386849  2.96984785  4.86054894  3.36784171
  4.09532461  5.61870841  2.06300502  2.806082    2.22142073  2.94780559
  6.22269181  4.8744939   4.94284234  2.89469462  3.87890477  2.21977895
  3.84712833  7.55866052  8.39011718  6.09067573  6.82763514  8.12288504
  7.7212274   1.53639357  2.26668008  2.96838384  2.63445673  5.6579683
  5.9625881   6.52630558  5.98600725  9.28508772  1.94078279  2.08942913
  2.99817174  3.43752808  2.18827087  7.91890063  1.48821142  2.06300765
  2.03735142  1.09052     2.0178097   2.16006561  1.65916157  6.03055211
  6.46709061  2.51458011  7.57653248  1.70646429  6.19196408  4.14630294
  5.96497532  6.67078438  6.29562531  5.71684     4.13767157  4.94760933
  2.58781675  2.94376085  3.99924565  4.37516847]
```