Universite Paris Dauphine

Master Statistic and Big Data

BERGUIGA Oussama

Money Market modeling with a random-coefficient linear model

#### Introduction

In that problem, we are asked to find the time varying coefficients that better fit the following equation thanks to the Kalman filter algorithm :

where :

- is the log-ratio of US Money Supply

- is the increment of short term rates

- is the log ratio of american consumer price index

- is the surplus or deficit of the US federal government budget

#### 1. Data Importation

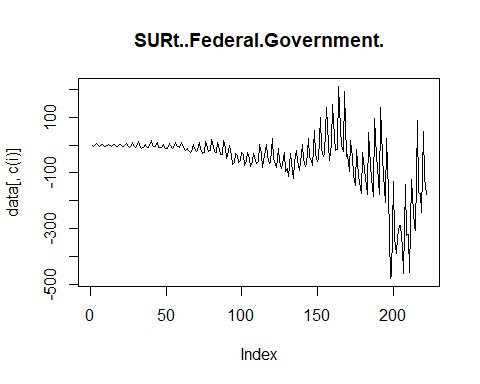
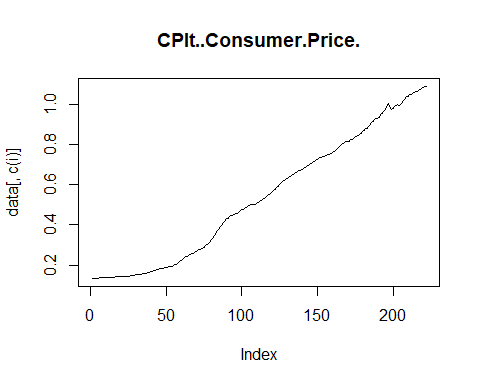
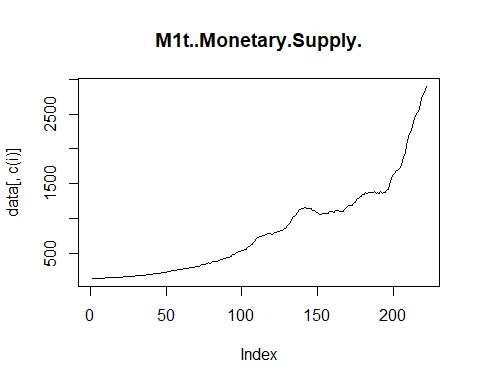
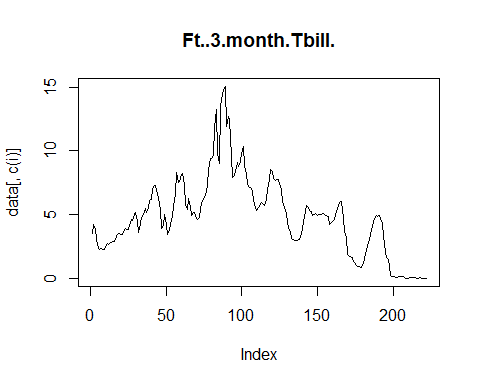
setwd("C:\\Users\\oussa\\Downloads\\Data Science\\Master Statistique Big Data Dauphine\\Module 2\\Séries temporelles")  
  
data=read.csv("data\_DM2.csv")  
data=data[-c(1),]#there is a missing value in the first row  
  
head(data)

## observation\_date Ft..3.month.Tbill. M1t..Monetary.Supply.  
## 2 1959-07-01 3.50 140.2  
## 3 1959-10-01 4.22 142.0  
## 4 1960-01-01 3.95 140.5  
## 5 1960-04-01 3.03 138.4  
## 6 1960-07-01 2.35 139.6  
## 7 1960-10-01 2.31 142.7  
## CPIt..Consumer.Price. SURt..Federal.Government.  
## 2 0.1338803 -3.0  
## 3 0.1346905 -4.5  
## 4 0.1348128 3.8  
## 5 0.1356230 4.4  
## 6 0.1356994 -0.8  
## 7 0.1365707 -3.9

colnames(data)

## [1] "observation\_date" "Ft..3.month.Tbill."   
## [3] "M1t..Monetary.Supply." "CPIt..Consumer.Price."   
## [5] "SURt..Federal.Government."

for (i in 2:ncol(data)){  
plot(data[,c(i)],main=colnames(data)[i],type='l')  
}



We can see an exponential tendancy on the Money Supply and the Consumer Price plots, which justifies the log-ratio operation

#### 2. Density of the vectors

Let the variance matrix of (as we have homoscedasticity is constant)

, and the determinant

Then, density of the is

#### 3. Steps of the Kalman prediction algorithm

In this problem, we are in a state-space model with random coefficients, and under the normal condition. According to the Kalman theorem, if we chose well, we can compute the following algorithm recursively : ,

the quadratic prediction error,

,, a strong white noise,

First we are asked to chose a random ,then we can compute the following recursion (simplified since ):

(This is a stochastic gradient algorithm starting from ) .Then , and .At last :

#### 4. Likelyhood expression

For calibrating the hyperparameter , we can compute the likelihood contrast . First we initialize which follows a random (the choice of the initialization is important for the algorithm convergence, it should correspond to the most likely fit on the observations), and ,then we compute the innovation

. Then we update de QLIK loss

. Finally, we compute the next linear prediction and the associated risk . Moreover we can estimate with

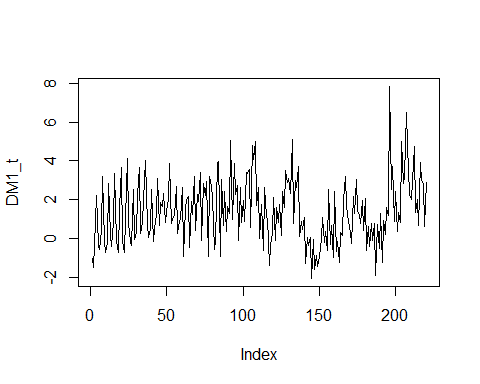
#### 5. Implementation of the state-space model

From the basic data we have to generate new features before implementing the state-space model :

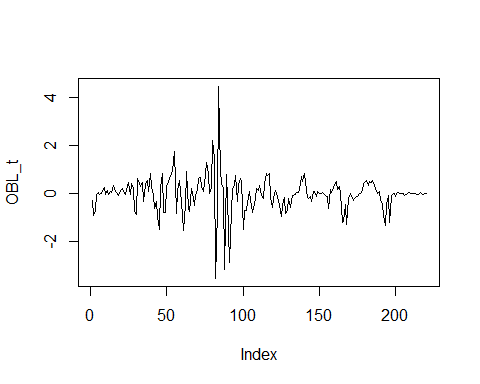
colnames(data)

## [1] "observation\_date" "Ft..3.month.Tbill."   
## [3] "M1t..Monetary.Supply." "CPIt..Consumer.Price."   
## [5] "SURt..Federal.Government."

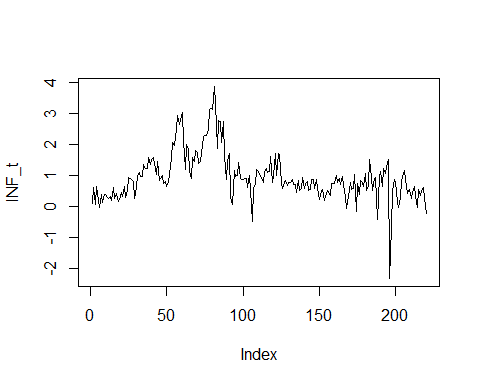
DM1\_t=100\*diff(log(data$M1t..Monetary.Supply.))[-1]  
DM1\_t\_1=100\*diff(log(data$M1t..Monetary.Supply.))[-(length(DM1\_t)+1)]  
  
   
  
OBL\_t=diff(data$Ft..3.month.Tbill.)[-1]  
OBL\_t\_1=diff(data$Ft..3.month.Tbill.)[-(length(OBL\_t)+1)]  
  
  
INF\_t=100\*diff(log(data$CPIt..Consumer.Price.))[-1]  
INF\_t\_1=100\*diff(log(data$CPIt..Consumer.Price.))[-(length(INF\_t)+1)]  
  
SUR\_t=diff(data$SURt..Federal.Government.)[-1]  
SUR\_t\_1=diff(data$SURt..Federal.Government.)[-(length(SUR\_t)+1)]  
  
  
Intercept=rep(1,length(DM1\_t))#for Beta\_o,t  
  
  
plot(DM1\_t,type='l')



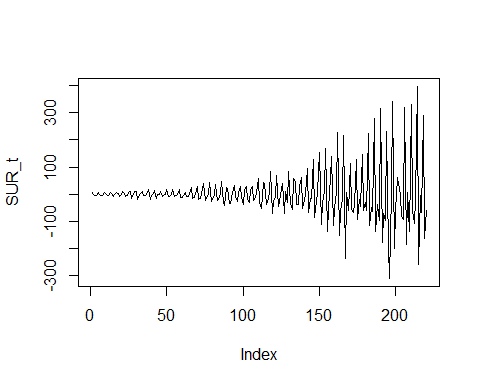
plot(OBL\_t,type='l')



plot(INF\_t,type='l')



plot(SUR\_t,type='l')



library(KFAS)

## Warning: package 'KFAS' was built under R version 3.3.3

?fitSSM

## starting httpd help server ...

## done

model=SSModel(DM1\_t~SSMregression(~Intercept+OBL\_t\_1+INF\_t\_1+DM1\_t\_1+SUR\_t\_1,Q=diag(NA,5),a1 = c(0,0,0,0,0),P1=diag(50^2,5))-1,H=NA)#a1 and P1 for initialization of beta\_o with N(0,50I)

#### 6. Computation of the maximum Likelyhood estimator

#calibration of the hyperparameter  
  
fit=fitSSM(model,inits=c(0.5,0.1,0.1,0.1,0.1,0.1),method="BFGS")  
  
model=fit$model

#### 7. Computation of beta and sigma

The coefficients of the Kalman recursion are given by :

#to get the Beta\_hat:  
out=KFS(model)  
model$a1

## [,1]  
## Intercept 0  
## OBL\_t\_1 0  
## INF\_t\_1 0  
## DM1\_t\_1 0  
## SUR\_t\_1 0

model$P1

## Intercept OBL\_t\_1 INF\_t\_1 DM1\_t\_1 SUR\_t\_1  
## Intercept 2500 0 0 0 0  
## OBL\_t\_1 0 2500 0 0 0  
## INF\_t\_1 0 0 2500 0 0  
## DM1\_t\_1 0 0 0 2500 0  
## SUR\_t\_1 0 0 0 0 2500

summary(out$alphahat)#summary of the Beta t/t-1

## Intercept OBL\_t\_1 INF\_t\_1 DM1\_t\_1   
## Min. :-0.3059 Min. :-1.0582 Min. :-0.40208 Min. :-0.13261   
## 1st Qu.: 0.9650 1st Qu.:-0.8805 1st Qu.:-0.38956 1st Qu.:-0.12843   
## Median : 1.9612 Median :-0.4685 Median :-0.31821 Median :-0.11977   
## Mean : 1.7661 Mean :-0.4801 Mean :-0.22858 Mean :-0.11456   
## 3rd Qu.: 2.5510 3rd Qu.:-0.1737 3rd Qu.:-0.09273 3rd Qu.:-0.10028   
## Max. : 3.4417 Max. : 0.2150 Max. : 0.13986 Max. :-0.08781   
## SUR\_t\_1   
## Min. :-0.00316   
## 1st Qu.:-0.00316   
## Median :-0.00316   
## Mean :-0.00316   
## 3rd Qu.:-0.00316   
## Max. :-0.00316

print(out$alphahat[1:5,])

## Intercept OBL\_t\_1 INF\_t\_1 DM1\_t\_1 SUR\_t\_1  
## [1,] 0.1504809 -0.3394001 -0.4014659 -0.1251954 -0.003159826  
## [2,] 0.1840718 -0.3355977 -0.4010254 -0.1251746 -0.003159826  
## [3,] 0.3265539 -0.3364211 -0.4003699 -0.1252099 -0.003159826  
## [4,] 0.4525352 -0.3348559 -0.3999295 -0.1252332 -0.003159826  
## [5,] 0.4826046 -0.3230294 -0.3996067 -0.1252967 -0.003159826

out

## Smoothed values of states and standard errors at time n = 220:  
## Estimate Std. Error  
## Intercept 2.5828090 0.6573011  
## OBL\_t\_1 -0.4682340 0.8368277  
## INF\_t\_1 0.1298639 0.3616862  
## DM1\_t\_1 -0.0889131 0.0857485  
## SUR\_t\_1 -0.0031598 0.0009837

#diagonal terms on "out$P"" give us Sigma\_t\_t\_1

The of the Kalman recursion is given by :

print(model$H)

## , , 1  
##   
## [,1]  
## [1,] 1.695996

The variances of the coefficients, which are the diagonal terms of matrix is given by

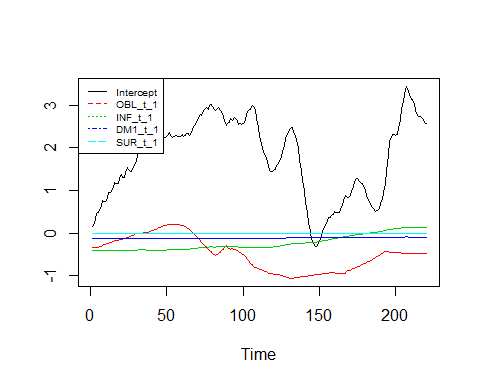
print(model$Q)

## , , 1  
##   
## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0.09977607 0.0000000 0.000000000 0.000000e+00 0.000000e+00  
## [2,] 0.00000000 0.0156983 0.000000000 0.000000e+00 0.000000e+00  
## [3,] 0.00000000 0.0000000 0.002170704 0.000000e+00 0.000000e+00  
## [4,] 0.00000000 0.0000000 0.000000000 4.842559e-05 0.000000e+00  
## [5,] 0.00000000 0.0000000 0.000000000 0.000000e+00 2.615101e-40

We can see that the variance of the coefficients on the diagonal terms is very low,except perhaps for the "intercept".

#### 8. Curves and prediction

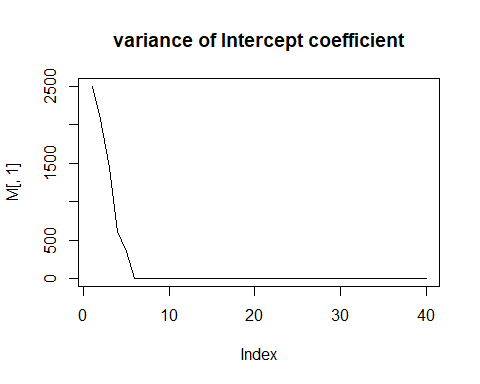
#plot of the Beta\_t\_t-1  
ts.plot(out$alphahat,col=1:5)  
  
legend("topleft", c("Intercept", "OBL\_t\_1", "INF\_t\_1","DM1\_t\_1","SUR\_t\_1"), col = 1:5, lty = 1:5,cex=0.65)



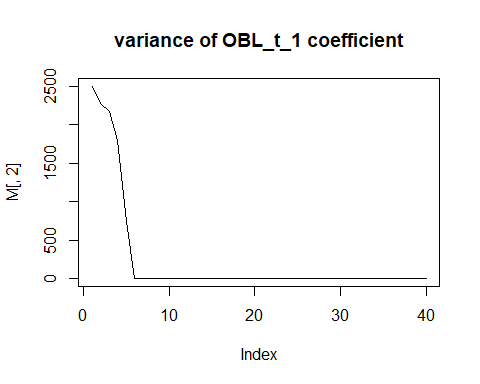
#plot of the Sigma\_t\_t-1 diagonal coefficients  
  
M=matrix(0,40,5)  
#M  
#nrow(M)  
for (i in 1:40)  
{M[i,]=diag(out$P[,,i])}  
M

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 2500.0000000 2500.0000000 2500.0000000 2.500000e+03 2.500000e+03  
## [2,] 2066.1116226 2275.0362395 2342.0236094 1.793714e+03 1.523527e+03  
## [3,] 1438.7891419 2166.9391742 2217.5165483 1.599913e+03 7.760363e+01  
## [4,] 606.8127183 1794.3746957 1883.8074101 7.047840e+02 1.149445e+01  
## [5,] 349.1852223 736.2512267 1065.0902681 3.423985e+02 1.143410e+01  
## [6,] 2.8543279 3.5545860 6.5626362 9.316699e-01 5.863289e-02  
## [7,] 2.3578577 3.3255163 6.1560137 9.175482e-01 5.803877e-02  
## [8,] 0.8836018 2.0363010 4.0653158 5.421620e-01 3.229688e-02  
## [9,] 0.8445361 1.8662217 4.0277539 4.749432e-01 2.526632e-02  
## [10,] 0.8591730 1.6951000 3.6358981 2.217951e-01 2.107880e-02  
## [11,] 0.9094343 1.6768693 3.4385867 2.218399e-01 1.816462e-02  
## [12,] 0.9058916 1.6764873 3.3954545 2.071158e-01 1.808676e-02  
## [13,] 0.7650451 1.4485600 3.3249263 1.522121e-01 1.138262e-02  
## [14,] 0.7862177 1.3803417 3.3140173 1.168379e-01 1.121187e-02  
## [15,] 0.8337389 1.3881694 3.2855408 1.114980e-01 8.367179e-03  
## [16,] 0.7484038 1.3905670 3.2247890 1.066572e-01 8.286768e-03  
## [17,] 0.8096653 1.3238198 2.9933887 1.005054e-01 6.944619e-03  
## [18,] 0.8431997 1.3307093 2.9861023 7.960714e-02 6.749895e-03  
## [19,] 0.8976636 1.3347287 2.8874910 7.943841e-02 5.849004e-03  
## [20,] 0.7642775 1.3499374 2.7713070 7.468911e-02 5.749612e-03  
## [21,] 0.7126478 1.3655732 2.7271675 7.468090e-02 5.204889e-03  
## [22,] 0.7966162 1.3652025 2.6353173 6.111896e-02 4.987383e-03  
## [23,] 0.7972873 1.3635667 2.6373462 6.105155e-02 4.567846e-03  
## [24,] 0.8673534 1.3690075 2.3297916 5.875796e-02 4.564836e-03  
## [25,] 0.7881548 1.3844262 2.2958896 5.716384e-02 3.853544e-03  
## [26,] 0.8646468 1.4001021 2.2567633 4.920533e-02 3.794052e-03  
## [27,] 0.9640882 1.4017052 1.8457167 4.925351e-02 3.587647e-03  
## [28,] 1.0597329 1.3264137 1.6386453 4.889091e-02 3.241847e-03  
## [29,] 1.1055867 1.2432957 1.5717644 3.663486e-02 1.817113e-03  
## [30,] 1.1762787 1.2478421 1.5347622 3.443481e-02 1.799641e-03  
## [31,] 1.0412646 1.0211260 1.4977465 3.421266e-02 1.718162e-03  
## [32,] 1.1314603 0.8099019 1.4383315 3.358101e-02 1.679967e-03  
## [33,] 1.1737413 0.7981304 1.4059282 3.337617e-02 1.398287e-03  
## [34,] 1.2687921 0.8062869 1.3344538 3.087295e-02 1.341647e-03  
## [35,] 1.3076222 0.8112249 1.3199840 3.027944e-02 1.329778e-03  
## [36,] 1.3438776 0.7791198 1.3210534 3.032785e-02 1.185497e-03  
## [37,] 1.4348598 0.6371370 1.0605895 3.010662e-02 1.142673e-03  
## [38,] 1.5231379 0.6524795 1.0416844 2.836954e-02 1.139376e-03  
## [39,] 1.5695951 0.6382079 1.0348348 2.792490e-02 1.122170e-03  
## [40,] 1.6626744 0.6484836 0.9030054 2.794475e-02 1.026399e-03

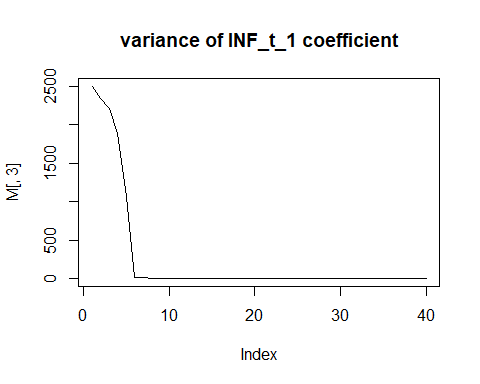
plot(M[,1],main='variance of Intercept coefficient',type='l')



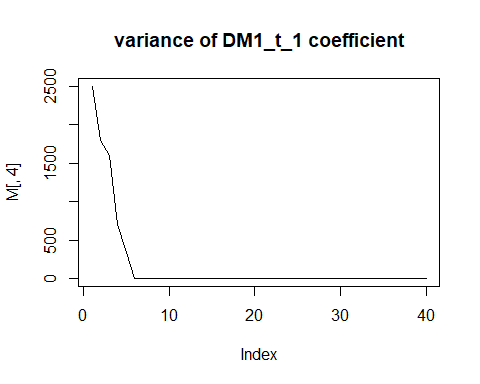
plot(M[,2],main='variance of OBL\_t\_1 coefficient',type='l')



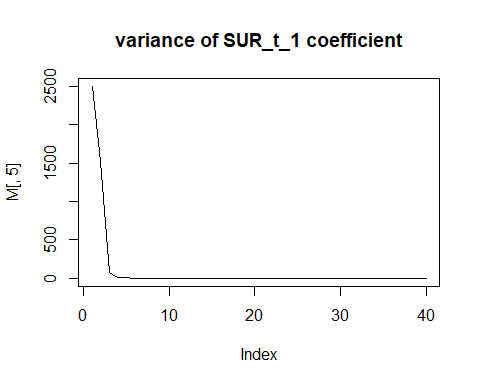
plot(M[,3],main='variance of INF\_t\_1 coefficient',type='l')



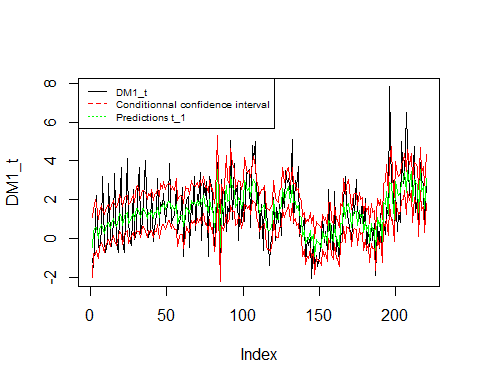
plot(M[,4],main='variance of DM1\_t\_1 coefficient',type='l')



plot(M[,5],main='variance of SUR\_t\_1 coefficient',type='l')



pred = predict(fit$model, interval = "conf", level = 0.95)  
plot(DM1\_t,type='l')  
lines(pred[,1],col="green")  
lines(pred[,2],col="red")  
lines(pred[,3],col="red")  
legend("topleft", c("DM1\_t","Conditionnal confidence interval","Predictions t\_1"), col = c("black","red","green"), lty = 1:3,cex=0.65)



summary(pred)

## fit lwr upr   
## Min. :-0.9451 Min. :-2.2448 Min. :-0.01285   
## 1st Qu.: 0.7442 1st Qu.:-0.2695 1st Qu.: 1.75265   
## Median : 1.3625 Median : 0.3653 Median : 2.35389   
## Mean : 1.3713 Mean : 0.3492 Mean : 2.39332   
## 3rd Qu.: 1.9066 3rd Qu.: 0.9259 3rd Qu.: 2.91951   
## Max. : 3.5820 Max. : 2.4890 Max. : 5.30803

In the first plot, we can see that the coefficients of SUR\_t\_1 and DM1\_t\_1 are nearly equal to 0, the coefficient of INF\_t\_1 is slightly rising around 0.The coefficient of OBL\_T\_1 is varying between 0 and -1. At last, the Intercept coefficient (which can be interpreted as a deterministic trend ) varies a lot between 0 and 3. Moreover, we can see that the variances ( the diagonal terms of ) converge very quickly, before 10 steps of the algorithm.

*Remark : the values given by model$Q are not strictly the same as the last values of out$P, I cannot understand why.*

To conclude, in view of the last graph, we can say that the prediction and the interval prediction fit very well to the DM1\_t.