POMP Modeling for Spot-futures Spreads by a Variation of CIR Model

Stats 531 Final Project

---

1 Introduction

In this project, we use a variation of CIR model\(^{[1]}\) which is a Markov process and the IF2 algorithm\(^{[2]}\) to find an appropriate model for spot-futures spread\(^{[3]}\). The POMP\(^{[4]}\) simulations are beneficial for exploration of the proper form of the model and the FLUX is conducive to the estimation of parameters. We model the spot-futures spread time series by POMP which was covered in class to give some statistical, mathematical and financial explanations for empirical phenomena. At the beginning I was going to continue the exploration of second order dependence of log returns, but after doing research, in order to avoid overlapping, I chose the spreads as the topic. The spreads have nice statistical properties because theoretically it should have expectation close to 0 given the maturity is small\(^{[5]}\). The spreads also have significant economic meaning because they can be constructed from two actively traded assets. Modeling the dynamics of spreads and evaluating the model are of great importance in the pairs trading strategy.

1.1 Raising Questions

The cointegration of S&P500Index and S&P500Futures Index&\(^{[6]}\) in my midterm project aroused my curiosity of finding the patterns of the spreads. Are they approximately Markovian? Will the spreads behave like individual stock or interest rate? Do they have heavier tails than normal distribution does? Should the coefficients of drift terms and diffusion terms be constants or deterministic functions or even progressively measurable random variables? Can we model spreads in different markets in one model? A series of questions about the spreads interest me a lot. Being haunted by these questions, I turned to papers first to see how predecessors deal with these problems.

1.2 Literature Reviews

Spreads are the difference between price of spot and futures. Financial stock indexes are observed to be fitted quite well by fractional Ornstein-Uhlenbeck process\(^{[7]}\). However, the fractional Ornstein-Uhlenbeck process is not Markovian\(^{[8]}\). Although the estimates can be obtained in other methods\(^{[9]}\), let us focus on spreads models that can be modeled by partially observed Markov process about which we have learned the theory and saw many cases in lecture notes\(^{[10]}\). Depending on market conditions, the spreads can be either positive or negative, and so different from interest rates, the choice of an Ornstein-Uhlenbeck process makes sense\(^{[11]}\). Wavelet analysis was also applied to analyze the dynamics between spot and futures \(^{[12]}\). The distribution of spreads are poorly modeled by normal distribution and have heavier tails and asymmetry\(^{[13]}\). Daily spreads and nonstorable commodities spreads are less normally distributed than weekly and storable commodities spreads are are. Hence, as traders search for the probability distributions of spreads, each spread is likely to have its own unique characteristics, making it difficult for trade\(^{[14]}\). In order to take care of the autocorrelation, asymetry and tails, we use a variation of Cox-Ingersoll-Ross model which is Markovian and has nice mathematical characteristics and interpretable parameters.

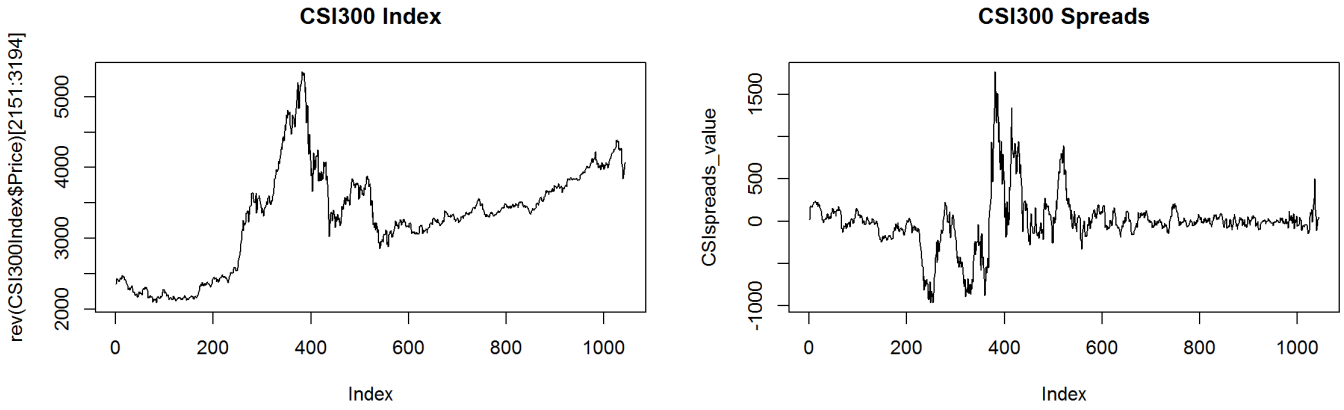
2 Introduction of Data

The CSI 300 Index and futures data I analyzed are the close price and last settlement price of CSI300 Index and futures from Jan 2st 2014 to Feb 23th, 2018 because the spreads in a less mature and less efficient market volatile more and revert to 0 more slowly\(^{[15]}\). The data is downloaded from Factset at Tozzi Financial Trading Center, Ross School of Business, University of Michigan. The HangSeng Index and futures, S&P500 Index and futures are from the same source and all end on Feb 23th, 2018.

First we load and take a look at the CSI300Index on the left. We are quite familiar with the exponential increasing graph of index in the class, although this graph has several spikes due to bullish and bearish periods.

```
CSI300Index=read.csv(file = "https://raw.githubusercontent.com/RickYuankangHung/Quant-
python/master/CSI300IndexPriceHistory.csv",header = T)
CSI300FuturesIndex=read.csv(file="https://raw.githubusercontent.com/RickYuankangHung/Quant-
python/master/CSI300ContinuousPriceHistory.csv",header = T)
CSIspreads_value=(array(rev(as.numeric(as.character(CSI300Index$Price))))[2151:3194])-
(array(rev(as.numeric(as.character(CSI300FuturesIndex$SettlementPrice)))))
```

```
par(mfrow=c(1,2))
plot(rev(CSI300Index$Price)[2151:3194],type='l',main="CSI300 Index")
plot(CSIspreads_value,type = 'l',main="CSI300 Spreads")
```



Observing the plot of spreads on the right, we may first find that the mean is almost 0. The volatility varies as time goes by. The mean reversion speed is proportional to its deviation from mean. The volatility is small when the magnitude of spread is small. The volatility becomes greater when the magnitude of spread is big. Importantly, the volatility is always dominated by the mean reversion. As what we can see, although during some periods of time the volatility is extremely big, spreads always come back to mean within a not long period of time. From economic perspective, the greater volatility of spreads clusters when the trend of the Index changes especially in the bearish market when there is sharp sell-off of futures by institutional investors. However, no matter how volatile the spread is, as long as there is no default in settlement, the spread will come back to the mean because arbitrage pushes prices of spot and futures to converge. Obtaining these features, we build the model for spreads.

---

3 Dynamics of Spread

The following model is built to capture what we observed in the previous graph. This is the model selected from several models. Other models have flaws such as explosion, lack of volatility, improper kurtosis, non-mean reversion. This model solves these flaws and is quite interpretable.

3.1 Continuous Model for Spread

\[dY_t=a (b-Y_{t})dt+(\sigma+ s\sqrt{|Y_{t}|})dW_t\]
\(W_t\) is Wiener processe.

Interpretations:

1.Parameter ‘a’ captures the speed of mean reversion. It can take value between 0 and 1.

2.Parameter 'b' depicts the mean of the spreads. Because the spreads can be any real value as long as the prices are possible, b can take any real value. Theoretically, it should be close to 0.

3.The drift coefficient is a linear form of Y_{t-1} because the mean reversion speed is proportional to its deviation from mean.

4.The diffusion term is to describe the volatility of spreads. Because the diffusion should be dominated by the mean reversion effect, otherwise it may diverge and never come back to the mean(this happens in some simulations of other models), the Y_{t-1} is raised to the power of $\frac{1}{2}$. To make it meaningful when Y_{t-1} takes negative value, we take the absolute value of Y_{t-1}.

5.Parameter 's' serves as scale of the volatility to make the model simulations volatile as much as the data do.

6.Parameter σ was introduced as the floor of volatility because we observed that even when the spreads are zero, they still volatile in a range.

3.2 Discrete Model for Spread

We write the discrete form for this model: $$Y_n-Y_{n-1}=a (b-Y_{n-1})+(\sigma+ s\sqrt{|Y_{n-1}|})\epsilon_n\\ \text{Model Representation:}\\Y_n=X_n+V_n\epsilon_n\\X_n=a (b-Y_{n-1})+Y_{n-1}\\V_n=(\sigma+s\sqrt{|Y_{n-1}|}) $$ where ${\epsilon_n}$ is an iid $N(0,1)$ sequence. Y denotes the spread. X denotes the latent process. V denotes the volatility latent process.

Parameters have constrained conditions:

$$ \sigma>0, s>0, 0<a<1, b \text{ no constraints} $$.

---

4 Building a POMP model

Based on the model mentioned above, we build a POMP model. The code is adapted from lecture notes $^{[10]}$.

```
require(pomp)

CSIspreads_statenames <- c("X","V","Y_state")
CSIspreads_rp_names <- c("sigma","a","b","s")
CSIspreads_ivp_names <- c("X_0","V_0")
CSIspreads_paramnames <- c(CSIspreads_rp_names,CSIspreads_ivp_names)
CSIspreads_covarnames <- "covaryt"

rproc1 <- "
  X=a*(b-Y_state)+Y_state;
  V = sigma+s*sqrt(abs(Y_state));
"
rproc2.sim <- "
  Y_state = rnorm( X,V );
 "
rproc2.filt <- "
  Y_state = covaryt;
 "
CSIspreads_rproc.sim <- paste(rproc1,rproc2.sim)
CSIspreads_rproc.filt <- paste(rproc1,rproc2.filt)
```

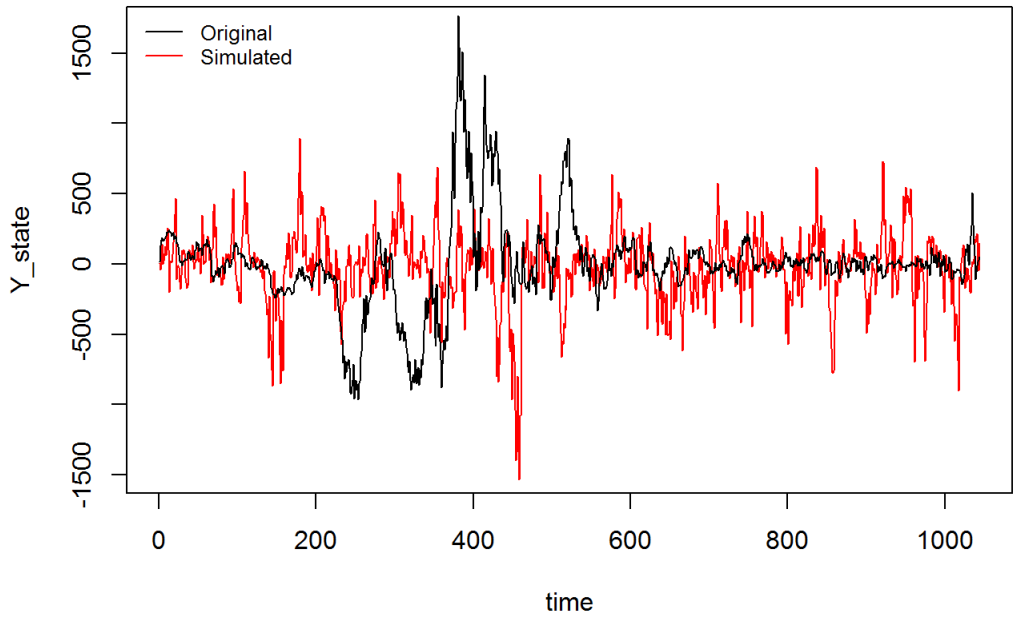
```
CSIspreads_initializer <- "
  V = V_0;
  X = X_0;
  Y_state = rnorm( X,V );
"
CSIspreads_rmeasure <- "
   y=Y_state;
"
CSIspreads_dmeasure <- "
   lik=dnorm(y,X,V,give_log);
"
```

It is better to convert parameters to the whole real value range, so we use log&exp to transform parameters constrained larger than 0, and use logit&expit to transform parameters within (0,1).

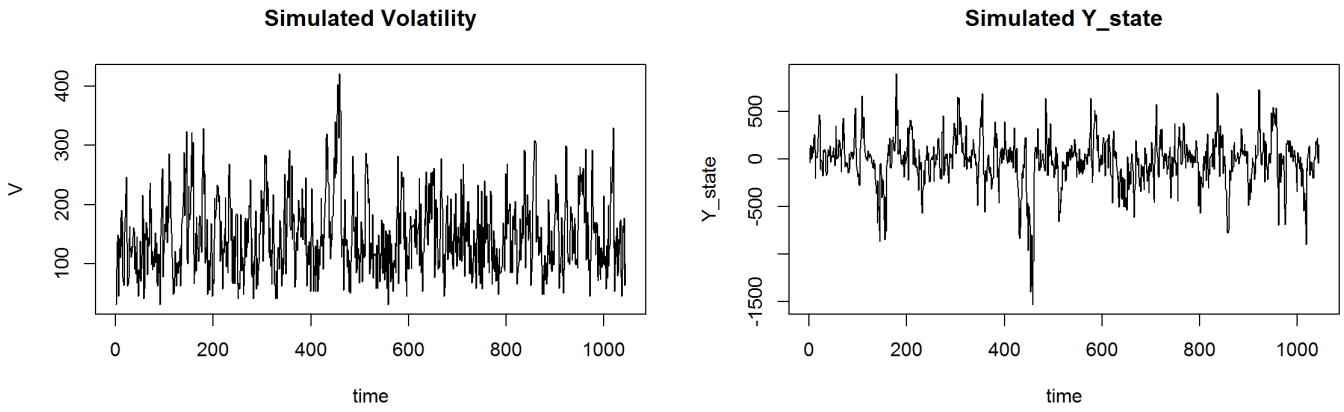
```
CSIspreads_toEstimationScale <- "
  Tsigma = log(sigma);
  Ts = log(s);
  Ta=logit(a);
"
CSIspreads_fromEstimationScale <- "
  Tsigma = exp(sigma);
  Ts = exp(s);
  Ta=expit(a);
"
```

To justify and test our model, we choose a set of parameters and simulate data from the model and then we compare the original data and simulated data.

```
##----------test for parameters--------------##
CSIspd.filt <- pomp(data=data.frame(y=CSIspreads_value,
                      time=1:length(CSIspreads_value)),
              statenames=CSIspreads_statenames,
              paramnames=CSIspreads_paramnames,
              covarnames=CSIspreads_covarnames,
              times="time",
              t0=0,
              covar=data.frame(covaryt=c(0,CSIspreads_value),
                      time=0:length(CSIspreads_value)),
              tcovar="time",
              rmeasure=Csnippet(CSIspreads_rmeasure),
              dmeasure=Csnippet(CSIspreads_dmeasure),
              rprocess=discrete.time.sim(step.fun=Csnippet(CSIspreads_rproc.filt),delta.t=1),
              initializer=Csnippet(CSIspreads_initializer),
              toEstimationScale=Csnippet(CSIspreads_toEstimationScale),
              fromEstimationScale=Csnippet(CSIspreads_fromEstimationScale)
)

expit<-function(real){1/(1+exp(-real))}
logit<-function(p.arg){log(p.arg/(1-p.arg))}

params_test <- c(
     sigma =30,
     a=0.15,
     b = 0,
     s=10,
     V_0= 1.1,
     X_0=1
  )
set.seed(700)
sim1.sim <- pomp(CSIspd.filt,
               statenames=CSIspreads_statenames,
               paramnames=CSIspreads_paramnames,
               covarnames=CSIspreads_covarnames,
               rprocess=discrete.time.sim(step.fun=Csnippet(CSIspreads_rproc.sim),delta.t=1)
)

sim1.sim <- simulate(sim1.sim,seed=1,params=params_test)
plot(Y_state~time,data=sim1.sim,type='l',col="red",ylim=c(-1500,1700))
lines(CSIspreads_value,type='l')
legend("topleft",legend=c("Original","Simulated"),col=c("black","red"),
       cex=0.8,lty=1,bty="n")
```




```
par(mfrow=c(1,2))
plot(V~time,data=sim1.sim,type='l',main="Simulated Volatility")
plot(Y_state~time,data=sim1.sim,type='l',main="Simulated Y_state")
```



From the plot, we can see that our model is reasonable. We have also tried other models and parameter values. Only some appropriate values such as the test values make the patterns similar to what we see in the historical data.

---

5 Filtering on Simulated Data

Then we check that whether we can filter on simulated data and try to gain more knowledge on the range of parameters. We set three different run levels.

```
##--------filtering on simulated data----------
run_level <- 3
CSIspreads_Np <-          c(100,1e3,5e3)
CSIspreads_Nmif <-        c(10, 100,300)
CSIspreads_Nreps_eval <-  c(4,  10,  30)
CSIspreads_Nreps_local <- c(10, 20, 30)
CSIspreads_Nreps_global <-c(10, 20, 50)
```

```
require(doParallel)
registerDoParallel(2)
```

```
sim1.filt <- pomp(sim1.sim,
  covar=data.frame(
    covaryt=c(obs(sim1.sim),NA),
    time=c(timezero(sim1.sim),time(sim1.sim))),
  tcovar="time",
  statenames=CSIspreads_statenames,
  paramnames=CSIspreads_paramnames,
  covarnames=CSIspreads_covarnames,
  rprocess=discrete.time.sim(step.fun=Csnippet(CSIspreads_rproc.filt),delta.t=1)
)

stew(file=sprintf("pf1.rda",run_level),{
  t.pf1 <- system.time(
    pf1 <- foreach(i=1:CSIspreads_Nreps_eval[run_level],.packages='pomp',
                   .options.multicore=list(set.seed=TRUE),.export = c('run_level', 'CSIspreads_Np', '
CSIspreads_Nmif', 'CSIspreads_Nreps_eval',
'CSIspreads_Nreps_local','CSIspreads_Nreps_global','sim1.filt')) %dopar% try(
                     pfilter(sim1.filt,Np=CSIspreads_Np[run_level])
                   )
  )
},seed=493536993,kind="L'Ecuyer")
(L.pf1 <- logmeanexp(sapply(pf1,logLik),se=TRUE))
```

```
##                           se
## -6.628405e+03  4.795887e-04
```

We get an log likelihood estimate of -7.494654e+03 with a standard error of 2.083143e-04. The standard error is quite small and so the result is valid.

---

6 Search for MLE

6.1 Local Search on Log Likelihood Surface on Model of the Spreads

We now start to filter on the spreads. We set run level as 3.

```
##---------Fitting the model to CSI300 Spread data------------
run_level <- 3
CSIspreads_Np <-          c(100,1e3,5e3)
CSIspreads_Nmif <-        c(10, 100,300)
CSIspreads_Nreps_eval <-  c(4,  10,  30)
CSIspreads_Nreps_local <- c(10, 20, 30)
CSIspreads_Nreps_global <-c(10, 20, 50)

CSIspreads_rw.sd_rp <- 0.001
CSIspreads_rw.sd_ivp <- 0.001
CSIspreads_cooling.fraction.50 <- 0.5

params_test <- c(
     sigma =70,
     a=0.15,
     b = 0,
     s=15,
     V_0= 2.1,
     X_0=4
```

```
  )

stew(file=sprintf("mif1-%d.rda",run_level),{
   t.if1 <- system.time({
   if1 <- foreach(i=1:CSIspreads_Nreps_global[run_level],
                  .packages='pomp', .combine=c,
                  .options.multicore=list(set.seed=TRUE),.export = c('CSIspreads_Np',
'CSIspreads_Nmif','CSIspreads_Nreps_eval','CSIspreads_Nreps_local','CSIspreads_Nreps_global','CSIspr
eads_rw.sd_rp','CSIspreads_rw.sd_ivp','CSIspreads_cooling.fraction.50','CSIspd.filt','run_level','pa
rams_test')) %dopar% try(
                    mif2(CSIspd.filt,
                         start=params_test,
                         Np=CSIspreads_Np[run_level],
                         Nmif=CSIspreads_Nmif[run_level],
                         cooling.type="geometric",
                         cooling.fraction.50=CSIspreads_cooling.fraction.50,
                         transform=TRUE,
                         rw.sd = rw.sd(
                           sigma=CSIspreads_rw.sd_rp,
                          a=CSIspreads_rw.sd_rp,
                           b=CSIspreads_rw.sd_rp,
                            s=CSIspreads_rw.sd_rp,
                             V_0=CSIspreads_rw.sd_ivp,
                             X_0=CSIspreads_rw.sd_ivp
                         )
                    )
                  )
   L.if1 <- foreach(i=1:CSIspreads_Nreps_global[run_level],.packages='pomp',
                    .combine=rbind,.options.multicore=list(set.seed=TRUE),.export =
c('CSIspreads_Np',
'CSIspreads_Nmif','CSIspreads_Nreps_eval','CSIspreads_Nreps_local','CSIspreads_Nreps_global','CSIspr
eads_rw.sd_rp','CSIspreads_rw.sd_ivp','CSIspreads_cooling.fraction.50','CSIspd.filt','run_level','pa
rams_test')) %dopar%
                      {
                        logmeanexp(
                          replicate(CSIspreads_Nreps_eval[run_level],

logLik(pfilter(CSIspd.filt,params=coef(if1[[i]]),Np=CSIspreads_Np[run_level]))
                          ),
                          se=TRUE)
                      }

  })
},seed=318817883,kind="L'Ecuyer")

r.if1 <- data.frame(logLik=L.if1[,1],logLik_se=L.if1[,2],t(sapply(if1,coef)))
if (run_level>=1)
  write.table(r.if1,file="CSIspreads_params.csv",append=TRUE,col.names=FALSE,row.names=FALSE)
summary(r.if1$logLik,digits=5)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -6234   -6234   -6234   -6234   -6234   -6234
```

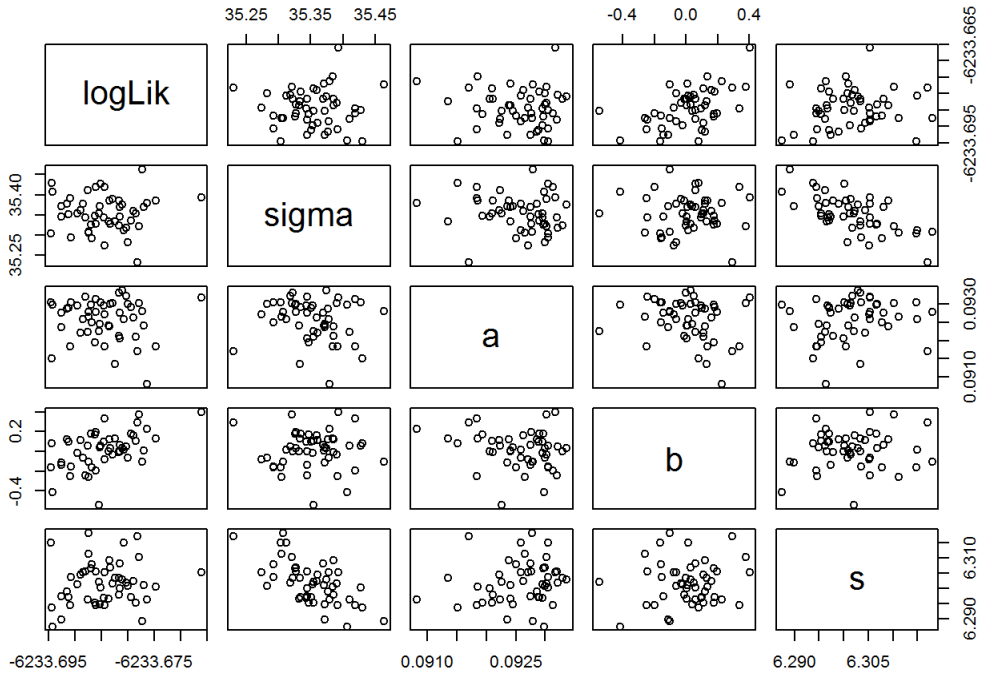
We see that the largest log likelihood is -6234 for this POMP model.

```
r.if1[which.max(r.if1$logLik),]
```

```
##              logLik    logLik_se   sigma          a         b        s
## result.19 -6233.666 0.0002330228 35.3932 0.09316657 0.4025641 6.305309
##               V_0      X_0
## result.19 2.11421 3.575516
```

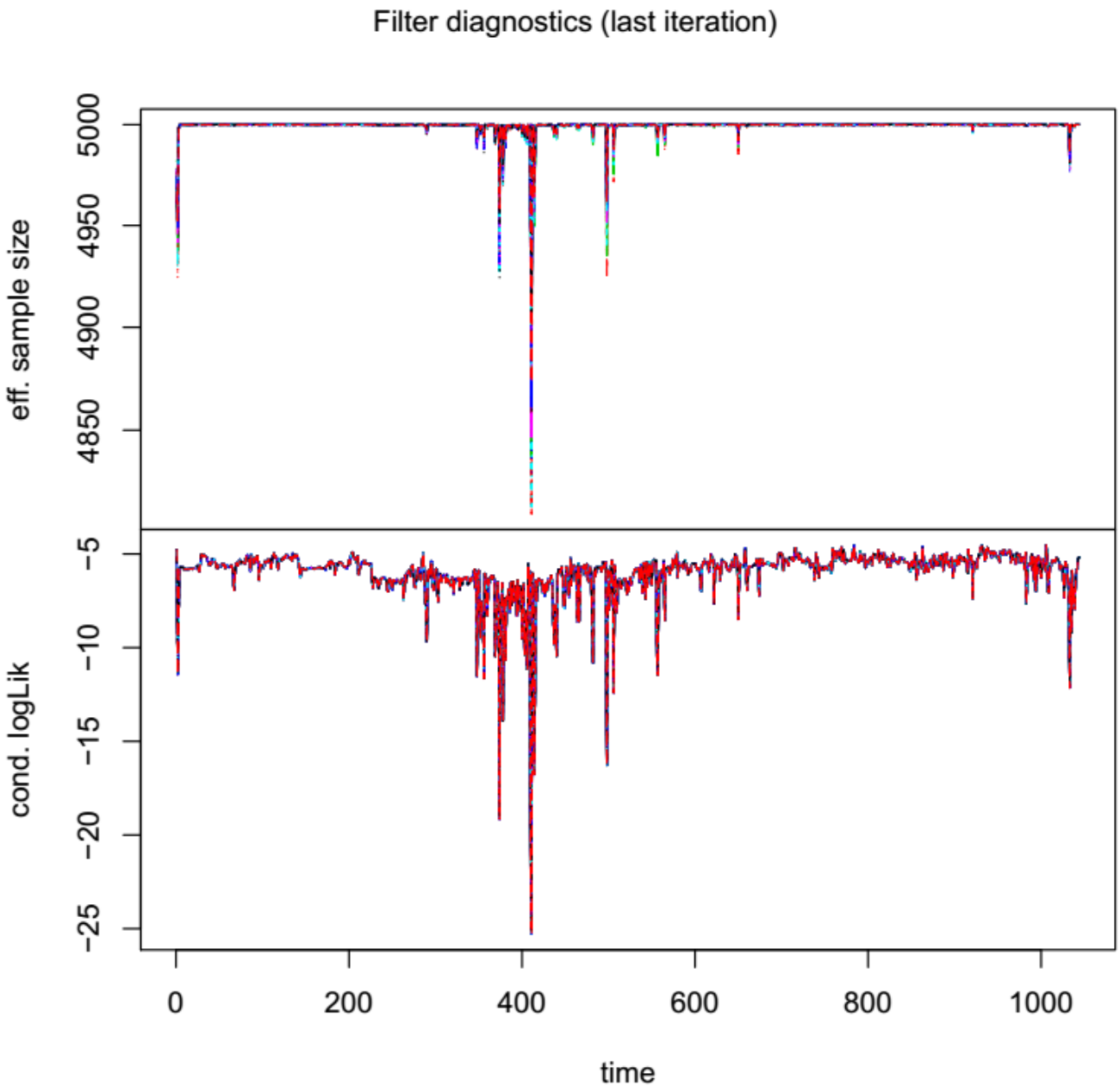
Compare the initial and last iteration value of parameters, \(\sigma\) has converged from 70 to 35.3932. Parameter a has converged from 0.15 to 0.09316657. Parameter s has converged from 15 to 6.305309 while b, V_0, X_0 do not change significantly.

```
pairs(~logLik+sigma+a+b+s,data=r.if1)
```

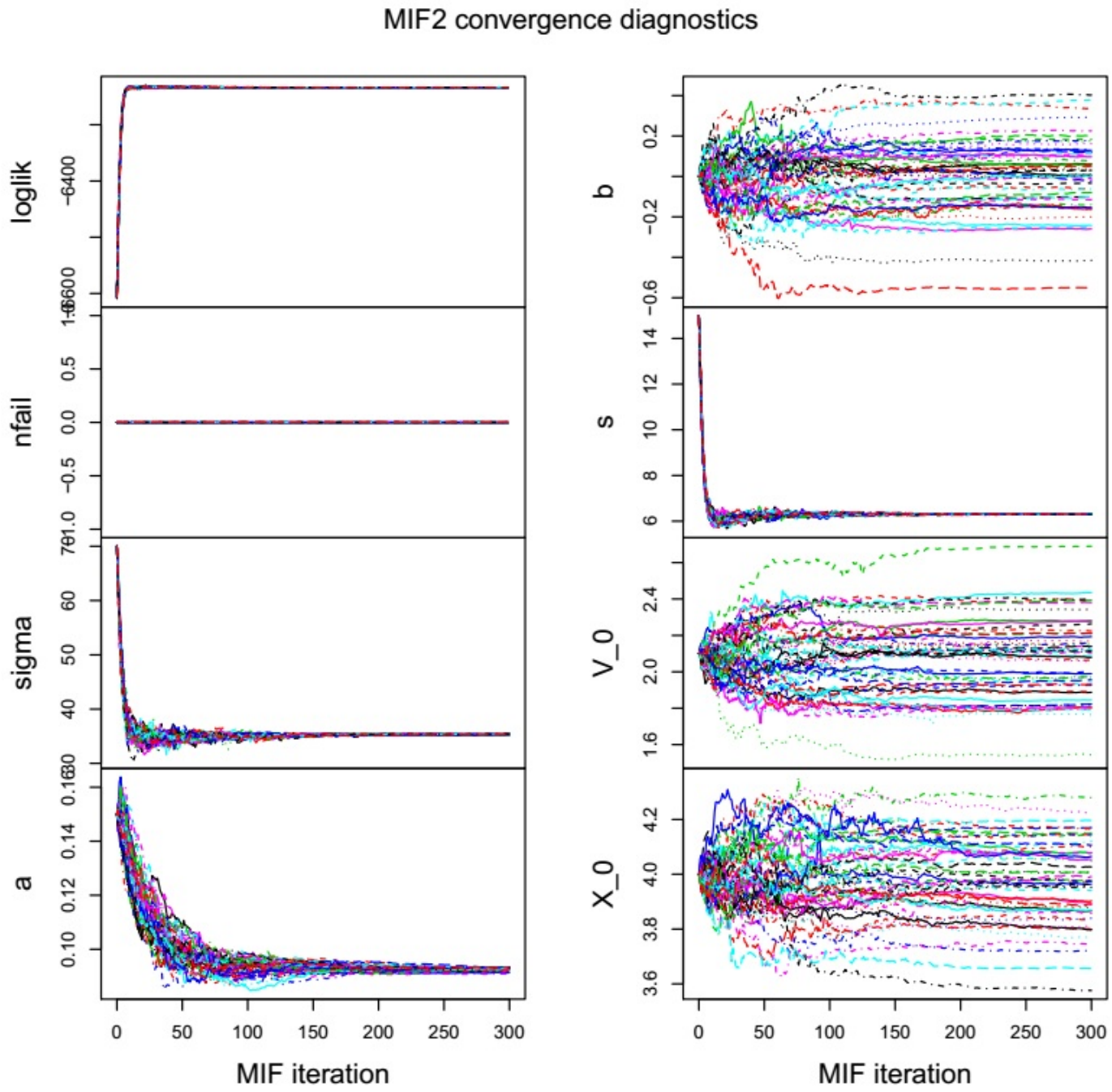


From the log likelihood surface with respect to different parameters, we see that peaks are significant for all parameters. S and \(\sigma\) have negative linear relationship, because the s and \(\sigma\) both serve to increase the volatility.

```
#plot(if1) The graph was derived from FLUX directly
```



From the filter diagnostics we can see that the effective sample size is always quite high except when time is around 400. The loglikelihood is also quite high except when time is 400 because there is a pattern change. In general the effective sample size is above 4800 which is quite good.



According to the filter and convergence diagnostic, the efficient sample size is large enough. The likelihood achieves the maximum at 20 and lasts until end of iteration. \(\sigma\), a and s have nice convergence and become flat within a very small interval.

From the MIF2 convergence diagnostics we can see that the sigma, s and a converge well. Sigma should be around 20. S should be around 6 while a is around 0.05. Parameter b does not converge well which is acceptable because the stationary mean model does not seem to fit the data, although we know that in the long run the spread process should have mean 0. It is okay that the \(X_0\) and \(V_0\) does not converge.

6.2 Likelihood Maximization Using Randomized Starting Values

For better estimating parameters, we need to try different initial values of parameters. We could set a value box in parameter space and then randomly choose initial values from this box. As for the specification for the box, I choose the bounds based on the local search results.

```
##--------Likelihood maximization using randomized starting values--------

CSIspreads_box <- rbind(
     sigma =c(10,100),
     a=c(0.001,0.999),
     b = c(-50,50),
     s=c(1,50),
     V_0= c(10,100),
     X_0= c(-50,50)
)

stew(file=sprintf("box_eval-%d.rda",run_level),{
  t.box <- system.time({
    if.box <- foreach(i=1:CSIspreads_Nreps_global[run_level],.packages='pomp',.combine=c,
                  .options.multicore=list(set.seed=TRUE),.export = c('CSIspreads_Np',
'CSIspreads_Nmif','CSIspreads_Nreps_eval','CSIspreads_Nreps_local','CSIspreads_Nreps_global','CSIspr
eads_rw.sd_rp','CSIspreads_rw.sd_ivp','CSIspreads_cooling.fraction.50','CSIspd.filt','run_level','pa
rams_test','if1','CSIspreads_box')) %dopar%
      mif2(
        if1[[1]],
        start=apply(CSIspreads_box,1,function(x)runif(1,x[1],x[2]))
      )

    L.box <- foreach(i=1:CSIspreads_Nreps_global[run_level],.packages='pomp',.combine=rbind,
                     .options.multicore=list(set.seed=TRUE),.export = c('CSIspreads_Np',
'CSIspreads_Nmif','CSIspreads_Nreps_eval','CSIspreads_Nreps_local','CSIspreads_Nreps_global','CSIspr
eads_rw.sd_rp','CSIspreads_rw.sd_ivp','CSIspreads_cooling.fraction.50','CSIspd.filt','run_level','pa
rams_test','if1','CSIspreads_box')) %dopar% {
                       set.seed(87932+i)
                       logmeanexp(
                         replicate(CSIspreads_Nreps_eval[run_level],

logLik(pfilter(CSIspd.filt,params=coef(if.box[[i]]),Np=CSIspreads_Np[run_level]))
                         ),
                         se=TRUE)
                     }
  })
},seed=453267856,kind="L'Ecuyer")

r.box <- data.frame(logLik=L.box[,1],logLik_se=L.box[,2],t(sapply(if.box,coef)))
if(run_level>1)
write.table(r.box,file="CSIspreads_params.csv",append=TRUE,col.names=FALSE,row.names=FALSE)
summary(r.box$logLik,digits=5)
```

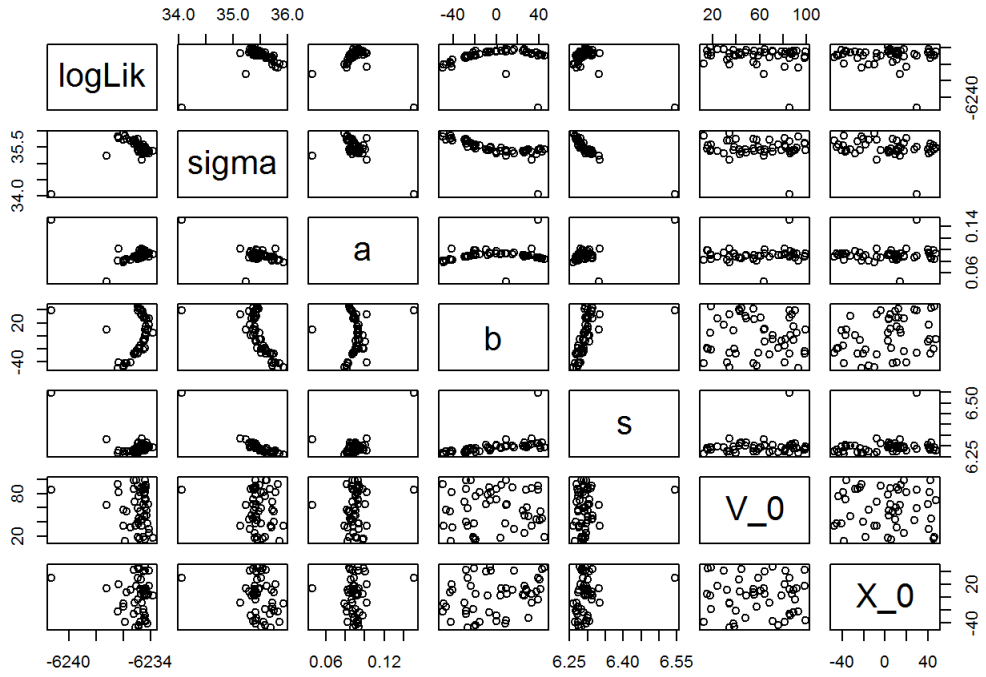
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -6241   -6235   -6235   -6235   -6234   -6234
```

```
r.box[which.max(r.box$logLik),]
```

```
##              logLik    logLik_se    sigma          a        b        s
## result.47 -6233.844 0.0005044258 35.39093 0.09253351 4.236419 6.293695
##               V_0      X_0
## result.47 17.8144 2.510589
```

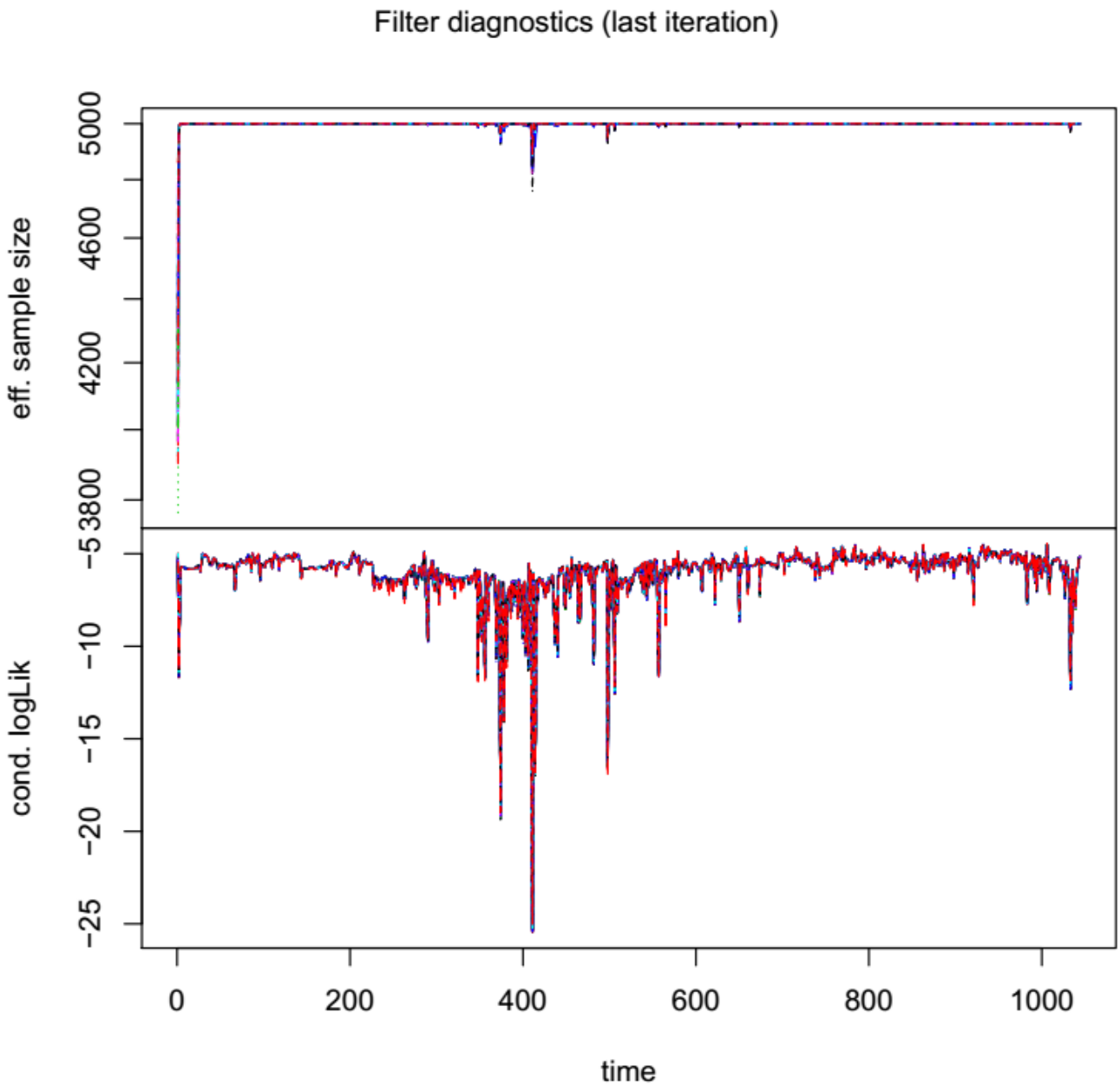
The maximum logLikelihood is -6233.844, and parameters which maximizes the likelihood are also shown.

```
pairs(~logLik+sigma+a+b+s+V_0+X_0,data=r.box)
```

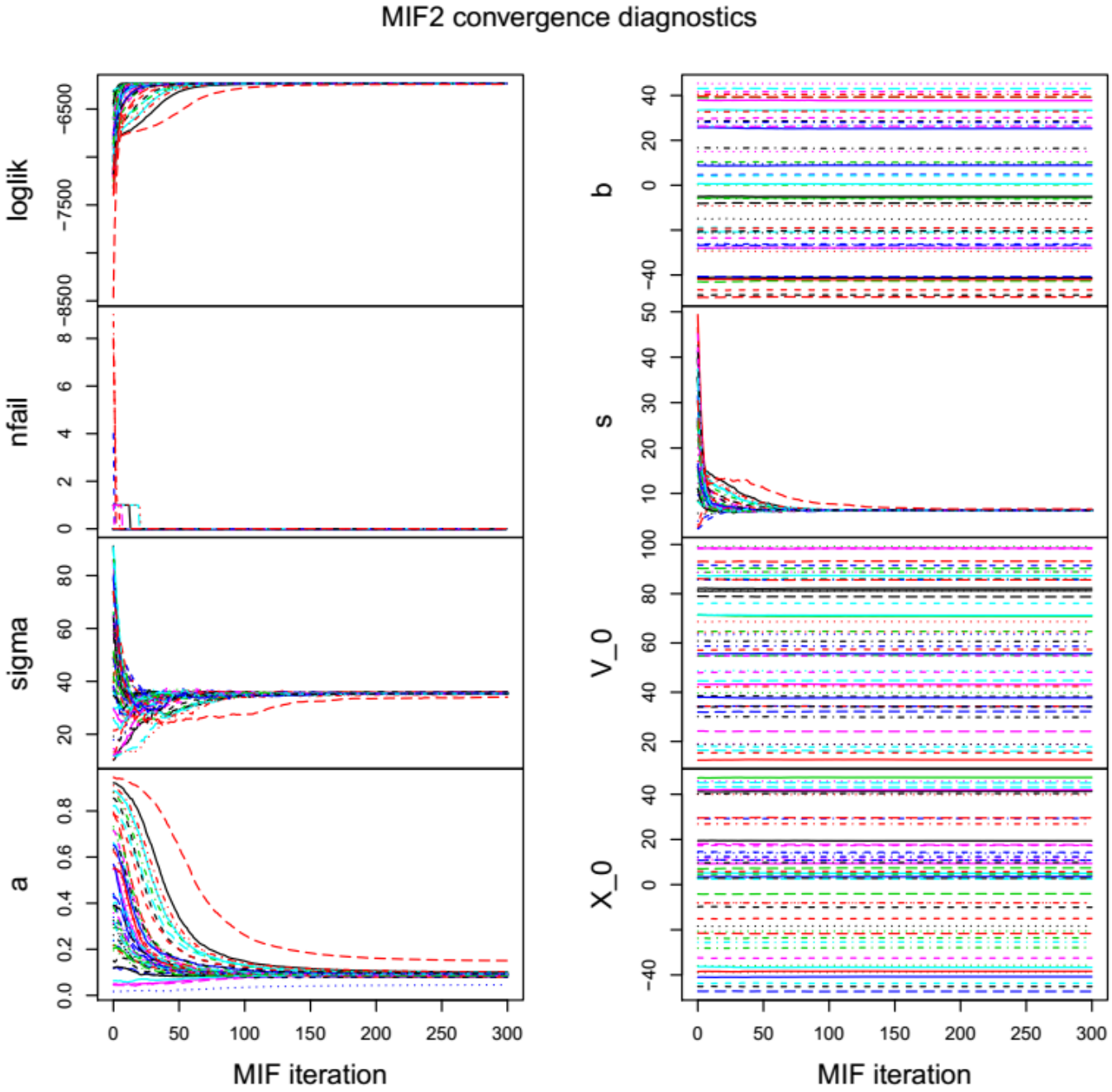


From the log likelihood surface with respect to different parameters, we see that peaks are significant for all parameters. S and \(\sigma\) have negative linear relationship, because the s and \(\sigma\) both serve to increase the volatility.

```
#plot(if.box) The graph was derived from FLUX directly
```



From the filter diagnostics we can see that the effective sample size is always quite high. The loglikelihood is also quite high except when time is around 390. In general the effective sample size is above 4800 which is quite good.



From the MIF2 convergence diagnostics we can see that the sigma converges well. Sigma should converge to 38. Parameter s converges to 6. Parameter a converge to 0.1. Parameter b does not converge well which is acceptable because the stationary mean model does not seem to fit the data although we know that in the long run the spread process should have mean 0. It is okay that the \(X_0\) and \(V_0\) does not converge.

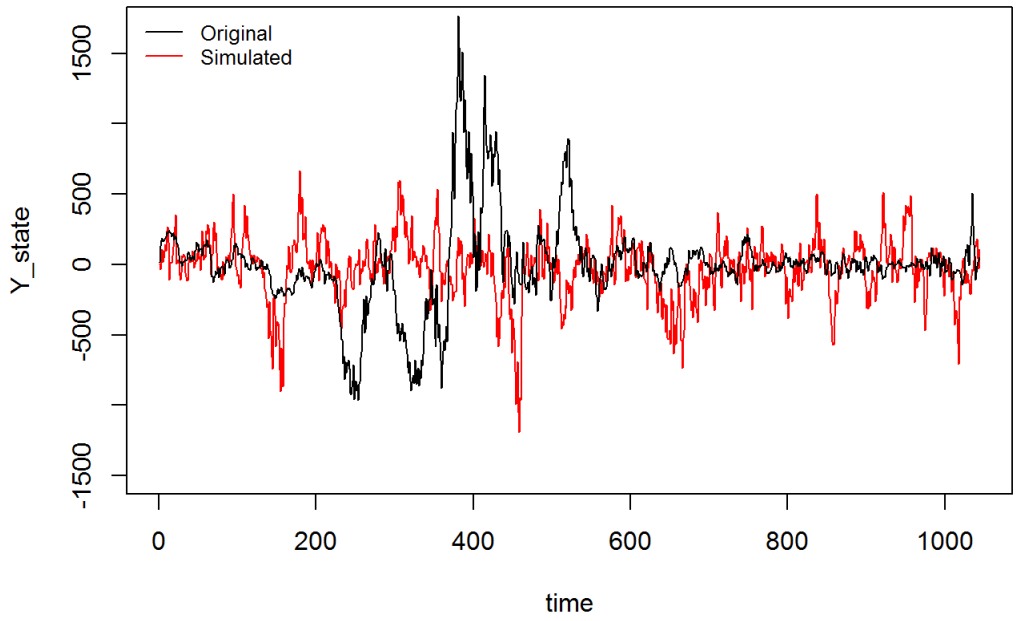
---

7 Model Evaluation

In order to evaluate this model, we compare the simulations with historical data, compare the log likelihoods with GARCH and fit the model to other markets spreads to see how well it works.

7.1 Patterns of Calibrated Model Compared with Historical Data

```
params_test <- c(
     sigma =35.39093,
     a=0.09253351,
     b = 4.236419,
     s=6.293695,
     V_0= 17.8144,
     X_0=2.510589
  )
set.seed(700)
sim1.sim <- pomp(CSIspd.filt,
                 statenames=CSIspreads_statenames,
                 paramnames=CSIspreads_paramnames,
                 covarnames=CSIspreads_covarnames,
                 rprocess=discrete.time.sim(step.fun=Csnippet(CSIspreads_rproc.sim),delta.t=1)
)

sim1.sim <- simulate(sim1.sim,seed=1,params=params_test)
plot(Y_state~time,data=sim1.sim,type='l',col="red",ylim=c(-1500,1700))
lines(CSIspreads_value,type='l')
legend("topleft",legend=c("Original","Simulated"),col=c("black","red"),
       cex=0.8,lty=1,bty="n")
```



The simulations show quite similar patterns of historical data. They have mean-reversion property and volatiles in similar way.

7.2 Benchmark Log Likelihoods for GARCH Models

We compare our model with GARCH model to evaluate the results of our model.

```
library(tseries)
L.garch.benchmark = logLik(garch(CSIspreads_value))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     8.804865e+04     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1  6.394e+03
##      1    3  6.309e+03  1.33e-02  2.30e-02  5.7e-07  1.5e+04  1.0e-01  1.68e+02
##      2    5  6.304e+03  8.17e-04  7.84e-04  5.0e-08  8.6e+00  1.0e-02  7.34e+00
##      3    7  6.295e+03  1.52e-03  1.51e-03  9.9e-08  3.4e+00  2.0e-02  7.47e+00
##      4    9  6.278e+03  2.69e-03  2.65e-03  1.9e-07  2.4e+00  4.0e-02  7.31e+00
##      5   12  6.277e+03  5.06e-05  5.06e-05  3.4e-09  4.0e+02  8.0e-04  6.74e+00
##      6   14  6.277e+03  1.01e-04  1.01e-04  6.8e-09  5.9e+01  1.6e-03  5.36e+00
##      7   16  6.275e+03  2.00e-04  2.00e-04  1.4e-08  3.1e+01  3.2e-03  5.32e+00
##      8   18  6.275e+03  3.99e-05  3.99e-05  2.7e-09  6.1e+02  6.4e-04  5.25e+00
##      9   20  6.275e+03  7.97e-06  7.97e-06  5.4e-10  3.1e+03  1.3e-04  5.15e+00
##     10   22  6.275e+03  1.59e-05  1.59e-05  1.1e-09  3.8e+02  2.6e-04  5.12e+00
##     11   24  6.275e+03  3.19e-06  3.19e-06  2.1e-10  7.7e+03  5.1e-05  5.12e+00
##     12   26  6.275e+03  6.37e-06  6.37e-06  4.3e-10  9.6e+02  1.0e-04  5.11e+00
##     13   28  6.275e+03  1.27e-06  1.27e-06  8.6e-11  1.9e+04  2.0e-05  5.11e+00
##     14   30  6.275e+03  2.55e-07  2.55e-07  1.7e-11  9.6e+04  4.1e-06  5.11e+00
##     15   32  6.275e+03  5.10e-07  5.10e-07  3.4e-11  1.2e+04  8.2e-06  5.10e+00
##     16   34  6.275e+03  1.02e-07  1.02e-07  6.9e-12  2.4e+05  1.6e-06  5.10e+00
##     17   36  6.275e+03  2.04e-08  2.04e-08  1.4e-12  1.2e+06  3.3e-07  5.10e+00
##     18   39  6.275e+03  1.63e-07  1.63e-07  1.1e-11  3.8e+04  2.6e-06  5.10e+00
##     19   42  6.275e+03  3.26e-09  3.26e-09  2.2e-13  7.5e+06  5.2e-08  5.10e+00
##     20   44  6.275e+03  6.52e-09  6.52e-09  4.4e-13  9.4e+05  1.0e-07  5.10e+00
##     21   46  6.275e+03  1.30e-09  1.30e-09  8.8e-14  1.9e+07  2.1e-08  5.10e+00
##     22   48  6.275e+03  2.61e-09  2.61e-09  1.8e-13  2.4e+06  4.2e-08  5.10e+00
##     23   50  6.275e+03  5.22e-10  5.22e-10  3.5e-14  4.7e+07  8.4e-09  5.10e+00
##     24   52  6.275e+03  1.04e-09  1.04e-09  7.0e-14  5.9e+06  1.7e-08  5.10e+00
##     25   54  6.275e+03  2.09e-10  2.09e-10  1.4e-14  1.2e+08  3.4e-09  5.10e+00
##     26   56  6.275e+03  4.17e-10  4.17e-10  2.8e-14  1.5e+07  6.7e-09  5.10e+00
##     27   58  6.275e+03  8.35e-10  8.35e-10  5.6e-14  7.3e+06  1.3e-08  5.10e+00
##     28   60  6.275e+03  1.67e-10  1.67e-10  1.1e-14  1.5e+08  2.7e-09  5.10e+00
##     29   62  6.275e+03  3.34e-11  3.34e-11  2.3e-15  2.0e+00  5.4e-10 -4.19e-02
##     30   64  6.275e+03  6.68e-11  6.68e-11  4.5e-15  2.0e+00  1.1e-09 -4.19e-02
##     31   65  6.275e+03 -1.59e+06  1.34e-10  9.0e-15  1.8e+08  2.1e-09  5.10e+00
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION     6.274915e+03   RELDX        9.005e-15
##  FUNC. EVALS      65         GRAD. EVALS      31
##  PRELDF       1.336e-10      NPRELDF      5.104e+00
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    8.804865e+04     1.000e+00     3.657e-03
##      2    2.141582e-01     1.000e+00    -2.882e+02
##      3    8.881373e-10     1.000e+00     2.632e+02
```

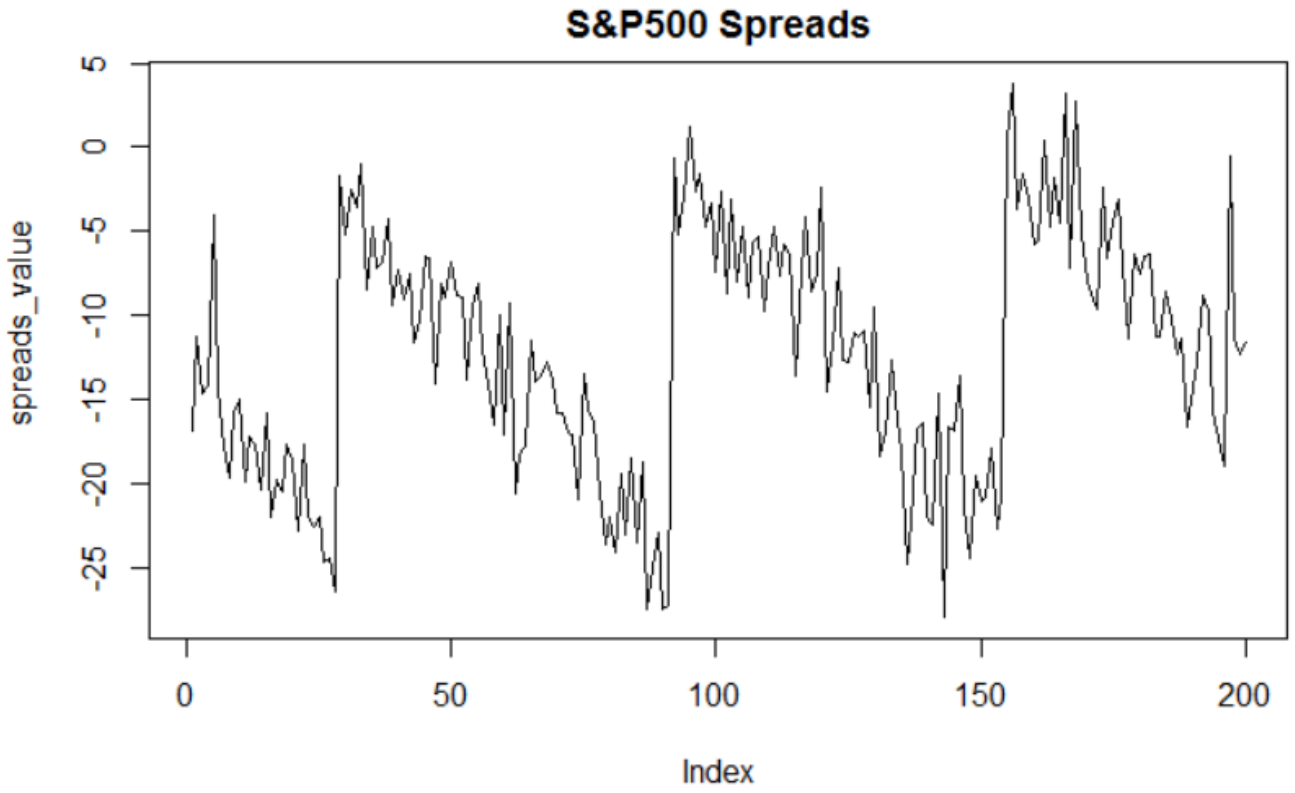
```
L.garch.benchmark
```

```
## 'log Lik.' -7233.367 (df=3)
```

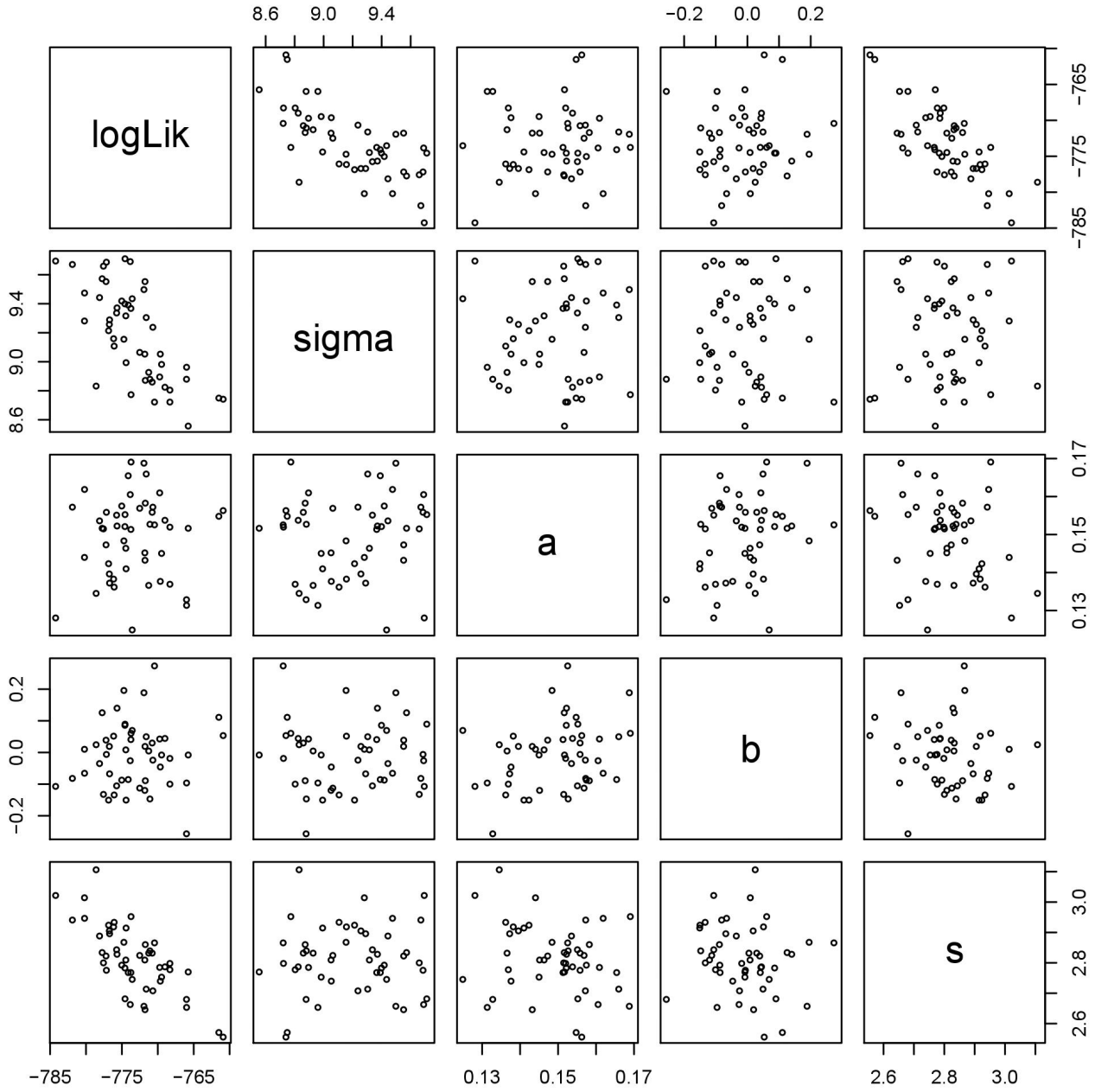
Comparing the log likelihood of this model with the GARCH model, we found the log likelihood of this model(-6233.844 (df=4)) is significantly greater than that of GARCH model(-7233.367 (df=3)). It only have one more parameter than GARCH.

7.3 Fitting to Spreads in Other Markets

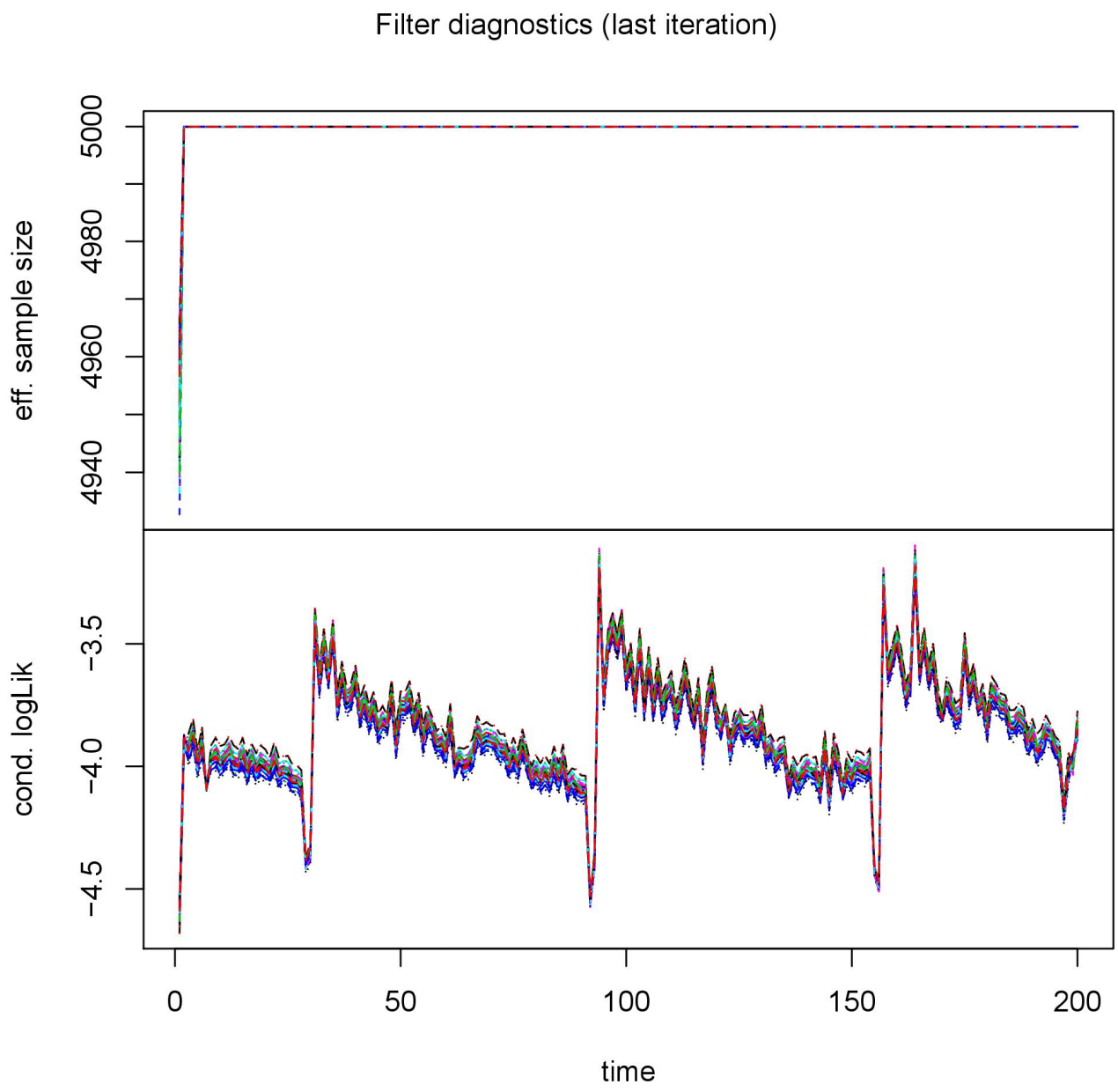
Does this model fit all markets well? Based on what we read in the literature before, each spread is likely to have its own unique characteristics\(^{[14]}\). Let us take US S&P500 as an example because it is a quite efficient market in the world.



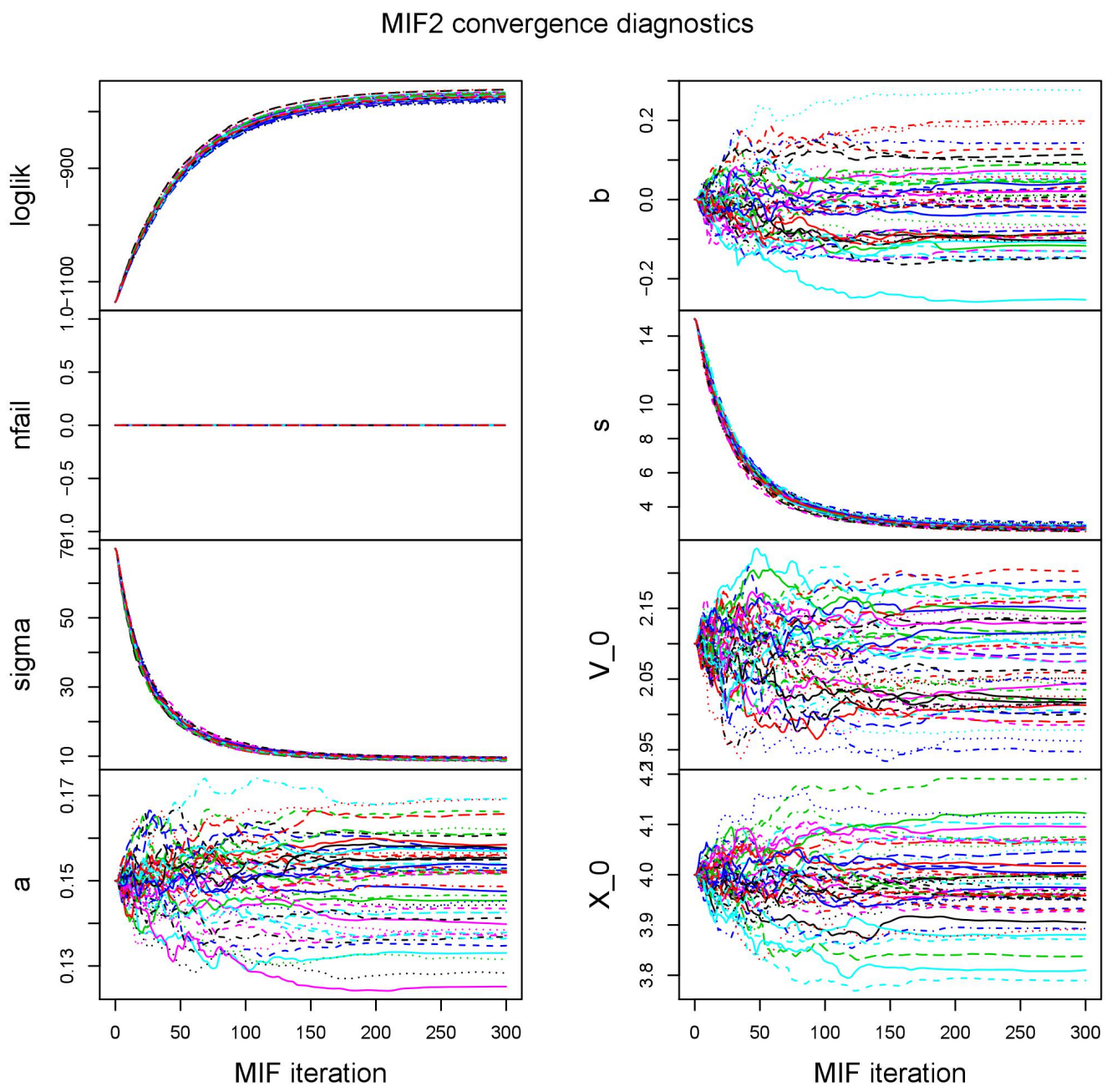
As we can see, the spreads are really small and have quite obvious cycles. This is due to the arbitrage activity has already made the relationship between spot and futures follow the theoretical relationship of no-arbitrage. We fit our model to the S&P500 spreads to see whether it works or not.



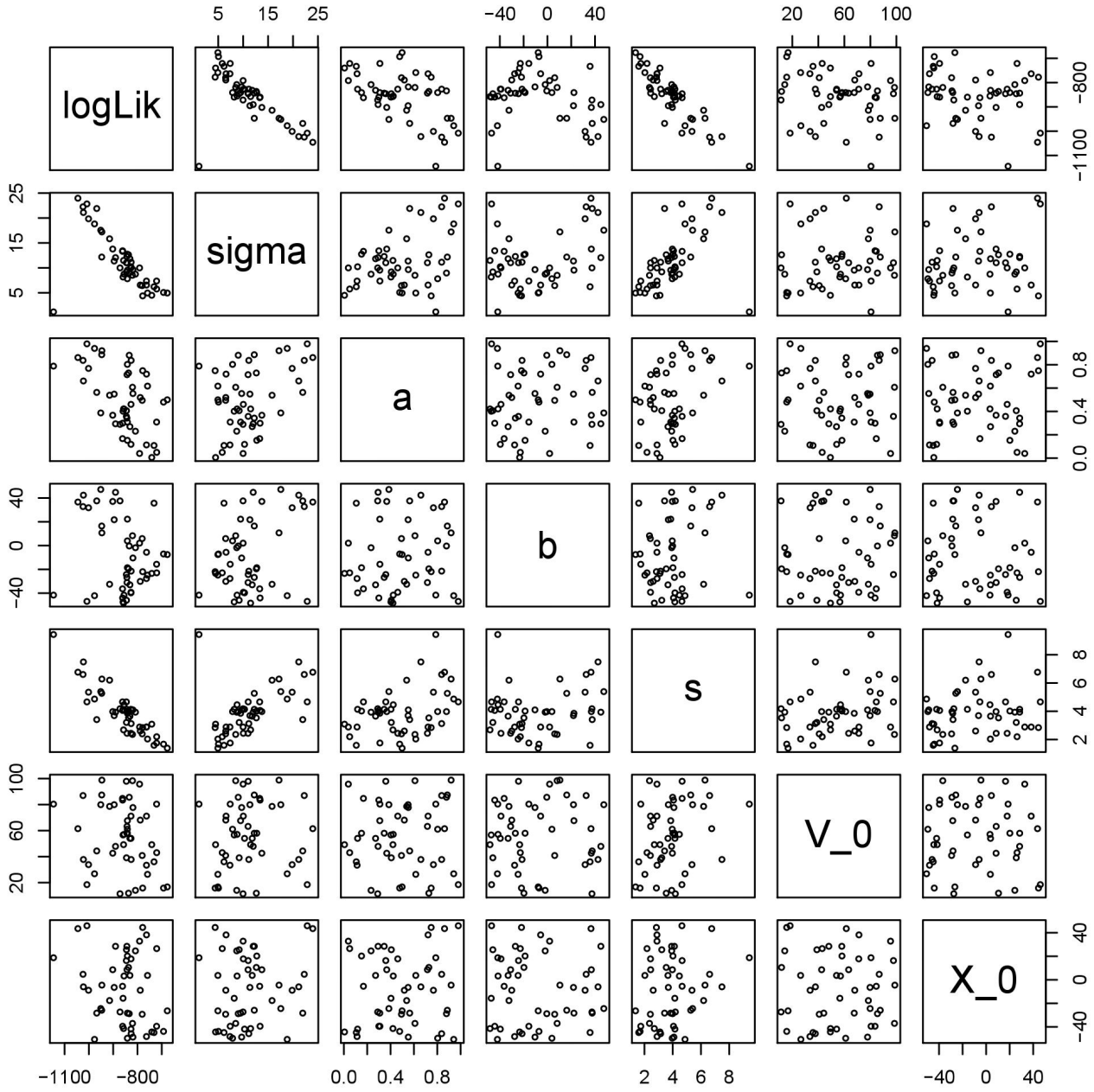
From the log likelihood surface with respect to different parameters, we see \(\sigma\) and s tend to be smaller. S and \(\sigma\) does not longer have negative linear relationship, different from previous result. When parameter b is around 0, log likelihood is maximized. Parameter a should be around 0.16.



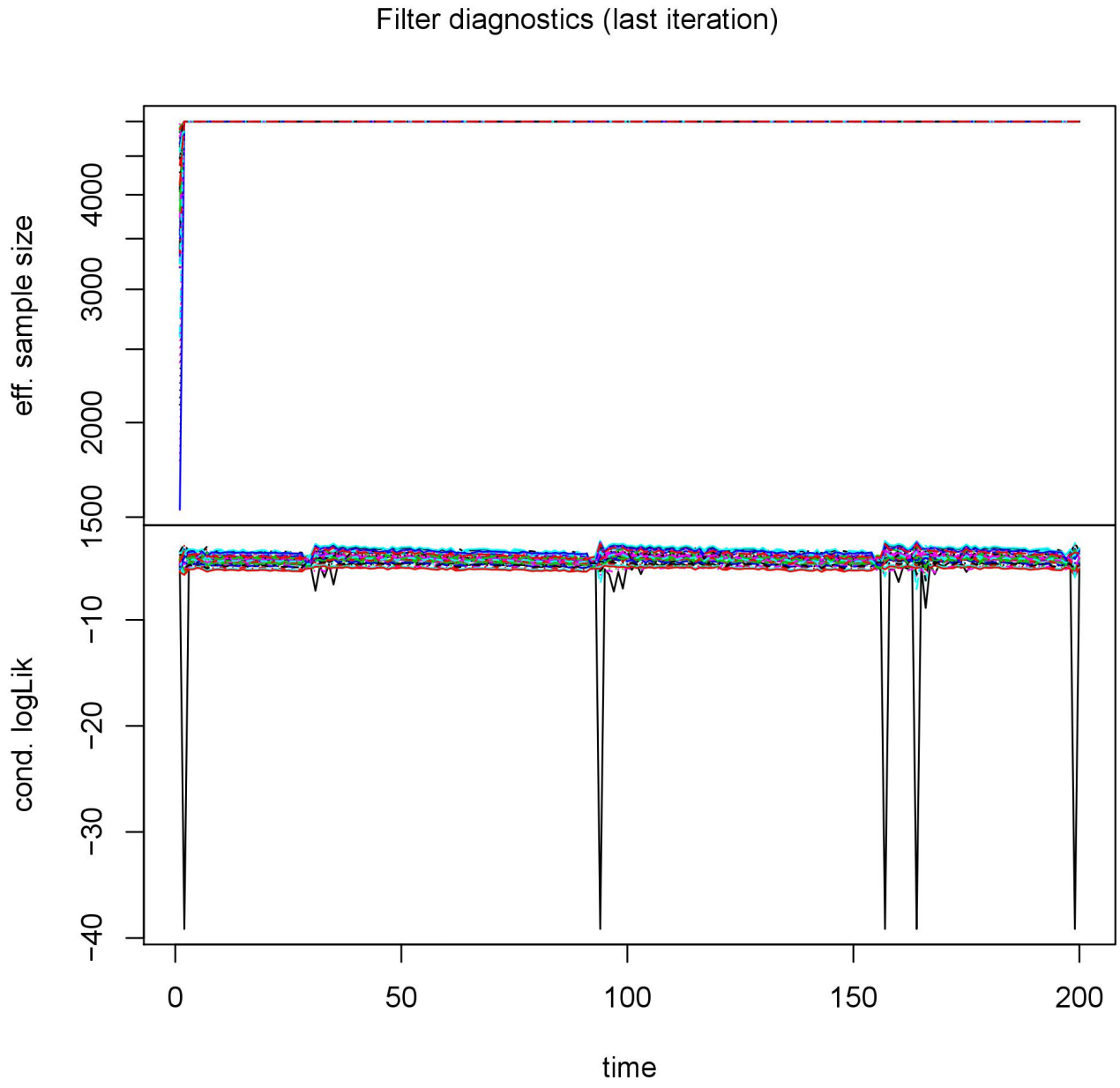
From the filter diagnostics we can see that the effective sample size is always quite high. My concern is that the log likelihoods behave like the original data. They decrease periodically, which is not ideal.



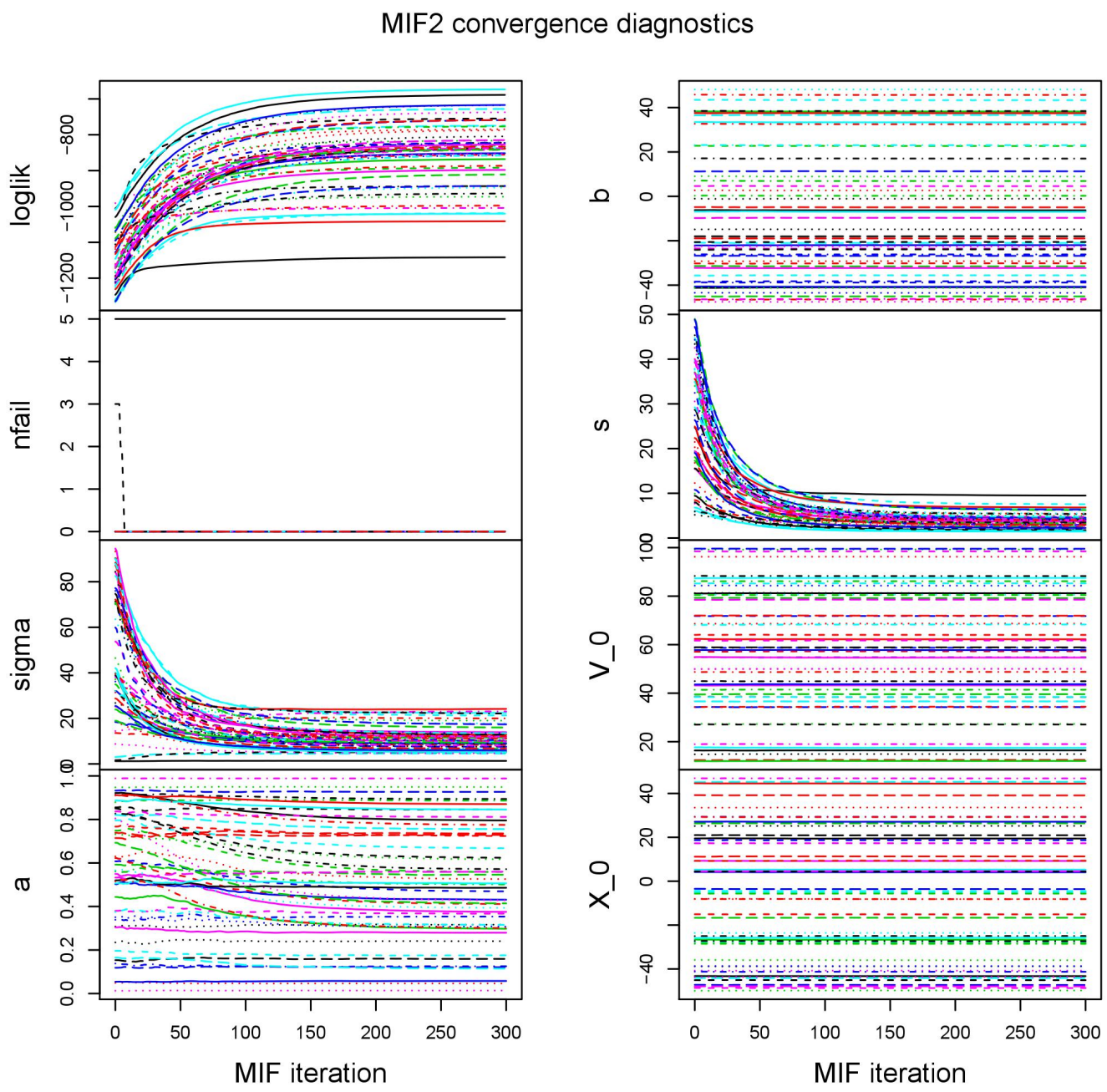
From the MIF2 convergence diagnostics we can see that the sigma and s converges well. Sigma should converge to 8. S converge to 2. Parameter a and b do not converge which is acceptable because there are cycles and the stationary mean model does not seem to fit the data although we know that in the long run the spread process should have mean 0. It is okay that the \(X_0\) and \(V_0\) does not converge.



From the log likelihood surface with respect to different parameters, we see \(\sigma\) and s tend to be smaller. S and \(\sigma\) show positive linear relationship, different from previous result. When parameter b is around 0, log likelihood is maximized. The surface of parameter a is quite flat. The confidence interval for it might be large.



From the filter diagnostics we can see that the effective sample size is always quite high. My concern is that the log likelihoods still decrease periodically, which is not ideal.



From the MIF2 convergence diagnostics we can see that the sigma and s converges not as well as before. Parameter a and b do not converge which is acceptable because there are cycles and the stationary mean model does not seem to fit the data although we know that in the long run the spread process should have mean 0. It is okay that the \(X_0\) and \(V_0\) does not converge. In general, this model does not fit S&P500 spread as well as it does to CSI300 because S&P500 is in a more efficient market and the spreads follow different patterns and dynamics. The model can fit CSI300 spreads well and even better than GARCH does.

---

8 Conclusion and Discussion on the Improvement

First, according to simulations, the mean reversion and self-exciting volatility are well depicted in this spread model. It also has nice mathematical characteristics. For example, it will not explode.

For CSI300 spreads data, given initial values and starting values, the sigma, s and a converges well. Sigma should be around 35.3932. S should be around 6.305309 while a is 0.09316657. Parameter b does not converge well which is acceptable because the stationary mean model does not seem to fit the data although we know that in the long run the spread process should have mean 0.

If the starting values are randomized, sigma convergences to 35.39093. Parameter a converges to 0.09253351. Parameter s

converges to 6.293695. Parameter b does not converge. Obviously the data has non-constant mean and non-constant variance. In genreral, we can determine sigma, a and s in this model.

From the filter diagnostics perspective, we believe this is a good model. The effective sample size and log likelihood are quite high and do not decay as time goes by.

After we get the estimates from global search, we plug them into the simulation and found that the patterns of simulations are close to the patterns of true data. The simulation results coincide with the historical data quite well.

Comparing the log likelihood of this model with the GARCH model, we found the log likelihood of this model(-6233.844 (df=4)) is significantly greater than that of GARCH model(-7233.367 (df=3)). It only have one more parameter than GARCH.

Fitting this model to a more efficient market such as S&P500 does not work well because the spreads follow different patterns and dynamics in different markets and products. Even sigma and s do not converge well in S&P500. There is no panacea in spreads modeling according to previous literature but I think they can be classified based on the characteristics of underlying such as storable or not.

In this model, I make the simulations heavier tails by changing the coefficient of the drift term. For future improvement, I might use t-distribution to serve as drift. Another method might be changing the power of the absolute value of Y_{state}.

Because the time horizon is so long, the patterns of this spread time series may have already changed. In practice, this model might be more useful in local forecasting and predicting. Moreover, the coefficient of the difussion term can also make it $|b-Y_{t-1}|$ to capture how much it deviates from the mean rather than the magnitude of $|Y_t|$.

To find an appropriate model and forecast the spreads, we can use more than the spread time series itself. More relative information can boost the accuracy of the prediction and modeling. Time series models with covariates are my improvement directions. For example, interest rate also plays a vital part in determing spread. More generally, the commodity futures spreads are even influenced by the production cost and storage cost$^{[16]}$.

After obtaining the model and parameters of this model, we gained better understanding of the dynamics of the spreads. We can even calculate the exit time, optimal stopping time, the distribution of running maximum and minimum to enhance our performance in trading decisions.

---

9 Reference

[1] Shreve, S. E. (2004). Stochastic calculus for finance II: Continuous-time models (Vol. 11). Springer Science & Business Media.

[2] Ionides, E. L., Nguyen, D., Atchad??, Y., Stoev, S., & King, A. A. (2015). Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. Proceedings of the National Academy of Sciences, 112(3), 719-724.

[3] Lien, D., & Yang, L. (2006). Spot?spread, time?correlation, and hedging with currency futures. Journal of Futures Markets, 26(10), 1019-1038.

[4] King, A. A., Nguyen, D., & Ionides, E. L. (2015). Statistical inference for partially observed Markov processes via the R package pomp. arXiv preprint arXiv:1509.00503.

[5] Pindyck, R. S. (2001). The dynamics of commodity spot and futures markets: a primer. The Energy Journal, 1-29.

[6] https://ionides.github.io/531w18/midterm_project/project20/MidtermProject.html

[7] Xiao, W., Zhang, W., & Xu, W. (2011). Parameter estimation for fractional Ornstein?CUhlenbeck processes at discrete observation. Applied Mathematical Modelling, 35(9), 4196-4207.

[8] El Onsy, B., Es-Sebaiy, K., & G. Viens, F. (2017). Parameter estimation for a partially observed Ornstein?CUhlenbeck process with long-memory noise. Stochastics, 89(2), 431-468.

[9] Xiao, W., Zhang, W., & Xu, W. (2011). Parameter estimation for fractional Ornstein?CUhlenbeck processes at discrete observation. Applied Mathematical Modelling, 35(9), 4196-4207.

[10] https://ionides.github.io/531w18/14/notes14.html

[11] Hofmann, K. F., & Schulz, T. (2016). A General Ornstein?CUhlenbeck Stochastic Volatility Model With L??vy Jumps. International Journal of Theoretical and Applied Finance, 19(08), 1650044.

[12] Kavitha, G., Udhayakumar, A., & Nagarajan, D. (2013). Stock market trend analysis using hidden markov models. arXiv preprint arXiv:1311.4771.

[13] Pirrong, S. C., Haddock, D., & Kormendi, R. C. (2012). Grain Futures Contracts: An Economic Appraisal. Springer Science & Business Media.

[14] Kim, M. K., & Leuthold, R. M. (2000). The Distributional Behavior of Futures Price Spreads. Journal of Agricultural and Applied Economics, 32(1), 73-87.

[15] Hazen, T. L. (1987). Volatility and Market Inefficiency: A Commentary on the Effects of Options, Futures, and Risk Arbitrage on the Stock Market. Wash. & Lee L. Rev., 44, 789.

[16] Carmona, R., & Ludkovski, M. (2004). Spot convenience yield models for the energy markets. Contemporary Mathematics, 351, 65-80.