The Final Project

Federico Vicentini, Alice Pratesi, Riccardo Dal Cero, Xhesjana Shametaj

2022-07-17

"Are Central Bankers Inflation Nutters? An MCMC Estimator of the Long-Memory Parameter in a State Space Model"

Inflation targeting has become an increasingly popular monetary policy regime since the 90s. All inflation targeting central banks are faced with a dilemma: i) A strict focus on inflation targeting may increase volatility elsewhere in the economy. ii) On the other hand an approach to inflation targeting which is too flexible may undermine credibility in the target. Most central banks have opted for policy strategy somewhere in the middle between strict and flexible inflation targeting.

In the long run the bank focuses on inflation, while it takes other non-inflation considerations into account over the short-to medium run. So, in the long run only inflation matters, while in the short run shock in demand count as well. The paper proposes that the fractional integration order from an autoregressive fractionally integrated moving average (ARFIMA) model can serve as an estimate of the degree of flexibility. Note that the main difference between a simple ARIMA model and an ARFIMA is the fact that the integration order can take the value of any real number, not just integers.

The results from those studies suggest that inflation is mean-reverting, but a covariance non-stationary series, i.e. fractional integration with an integration order between 0.5 and 1. Note that a time series is said to be covariance non-stationary if mean and variance become constant over time. The higher (lower) the fractional integration order is, the longer (shorter) are the deviations from the mean (i.e. target) and the more flexible (strict) is the inflation target policy.

The authors in the first section of the paper demonstrate that, even if usually we can obtain good results with a Kalman filter, using an MCMC algorithm provides better results. In the end, we decided to use Kalman filter.

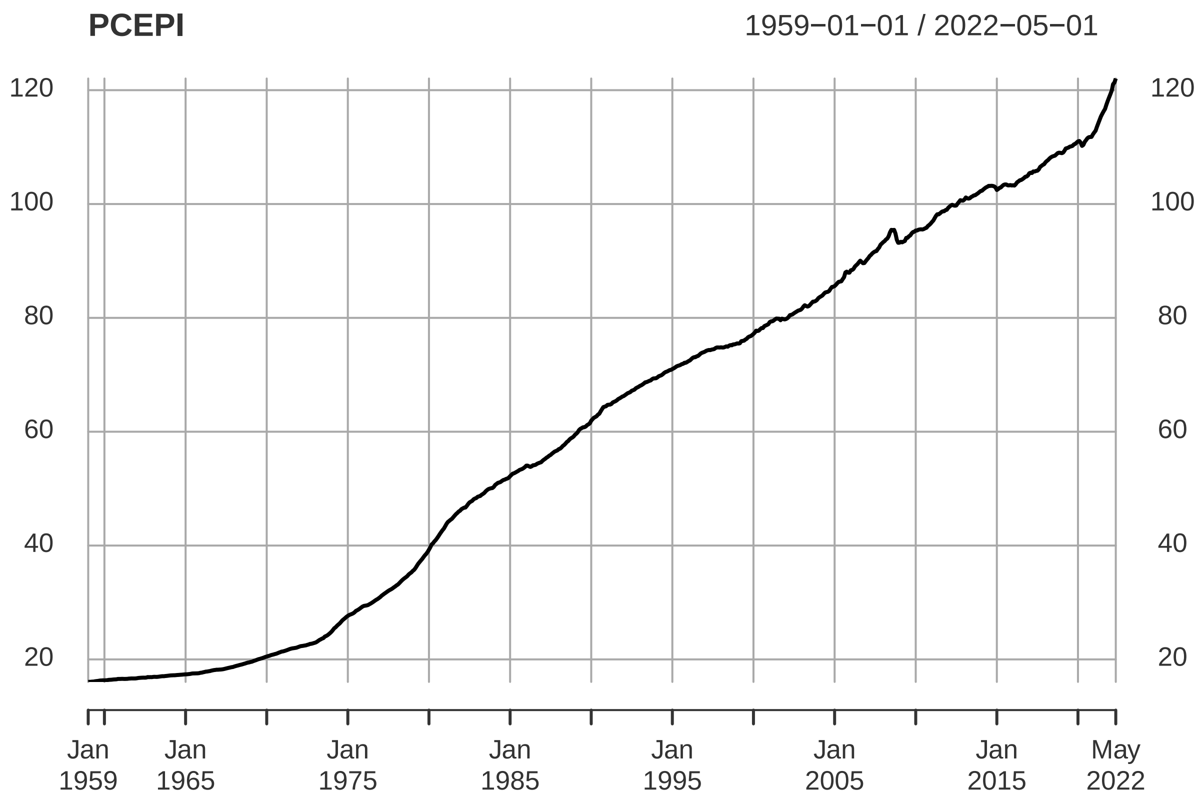
We then applied the algorithm to estimate the fractional integration order for seven economies (Canada, Euro area, Germany, Norway, Sweden, the United Kingdom and the United States) between 1993 and 2017 using monthly data. We divided the period in different chunks to account for the effect of the financial crisis in 2008.

```r
rm(list = ls())
chooseCRANmirror(graphics=FALSE, ind=87)
# Set the working directory to source file location
# setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# Install packages
packages <- c("tidyverse", "rsdmx", "eurostat", "tbl2xts",
              "tidyquant", "BCDating", "pwt10", "dplyr",
              "stargazer", "forecast", "tseries",
              "quantmod", "eurostat", "stargazer",
              "skedastic","Metrics","mFilter", "aTSA","lmtest","xts", "prediction")
new.packages <- packages[!(packages %in% installed.packages()[, "Package"])]
if (length(new.packages)) install.packages(new.packages)
invisible(lapply(packages, library, character.only = TRUE))
```

```r
# Load packages
library(quantmod)
library(eurostat)
library(FKF)
library(arfima)
```

First we imported the data from the Fred database checking for NA and NULL values

```r
# Import data
# Download inflation rates from Fred
freddata <- c("PCEPI")
for (i in 1:length(freddata)) {
  getSymbols(freddata[i], src = "FRED")
}
plot(PCEPI)
```



```r
#Check for NA values
nas=c(which(is.na(PCEPI)))
nas
```

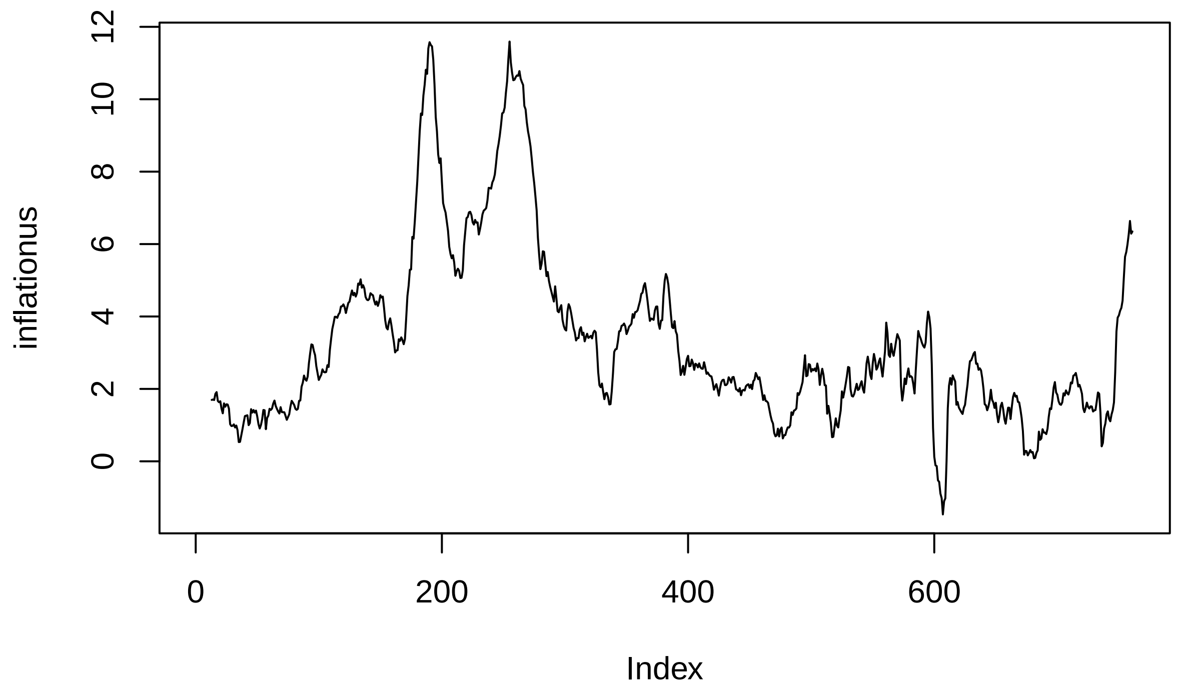
```
## integer(0)
```

```r
#Check for null values
zeroes=c(which(PCEPI==0))
zeroes
```

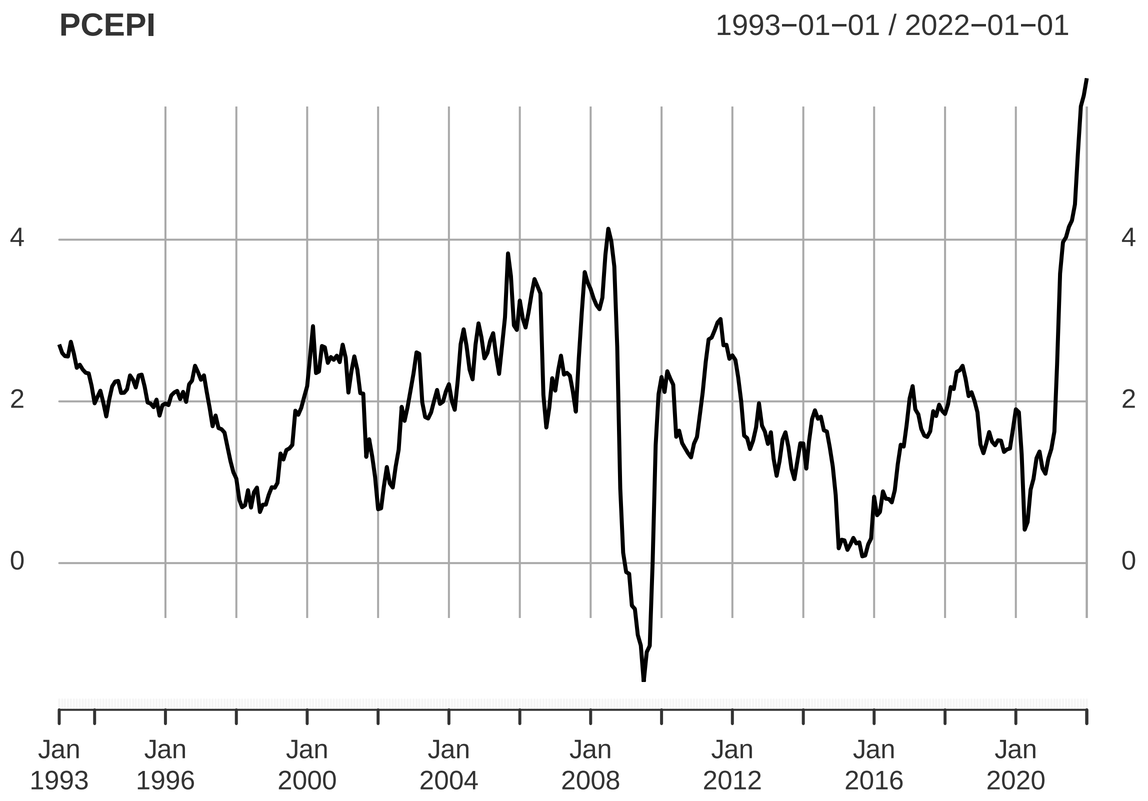
```
## integer(0)
```

We now use two for cycles to transform PCEPI in PCE inflation rates which is a time series with monthly year on year inflation rates. Obviously, by doing this we loose the first twelve observations thus, we remove them from the time series. Also, we cut the time series starting from January 1993.

```
inflationus=c()
for(i in 13:length(PCEPI)){
  inflationus[i]=((as.numeric(PCEPI[i])-as.numeric(PCEPI[i-12]))/(as.numeric(PCEPI[i-12])))*100
}
plot(inflationus, type="l")
```

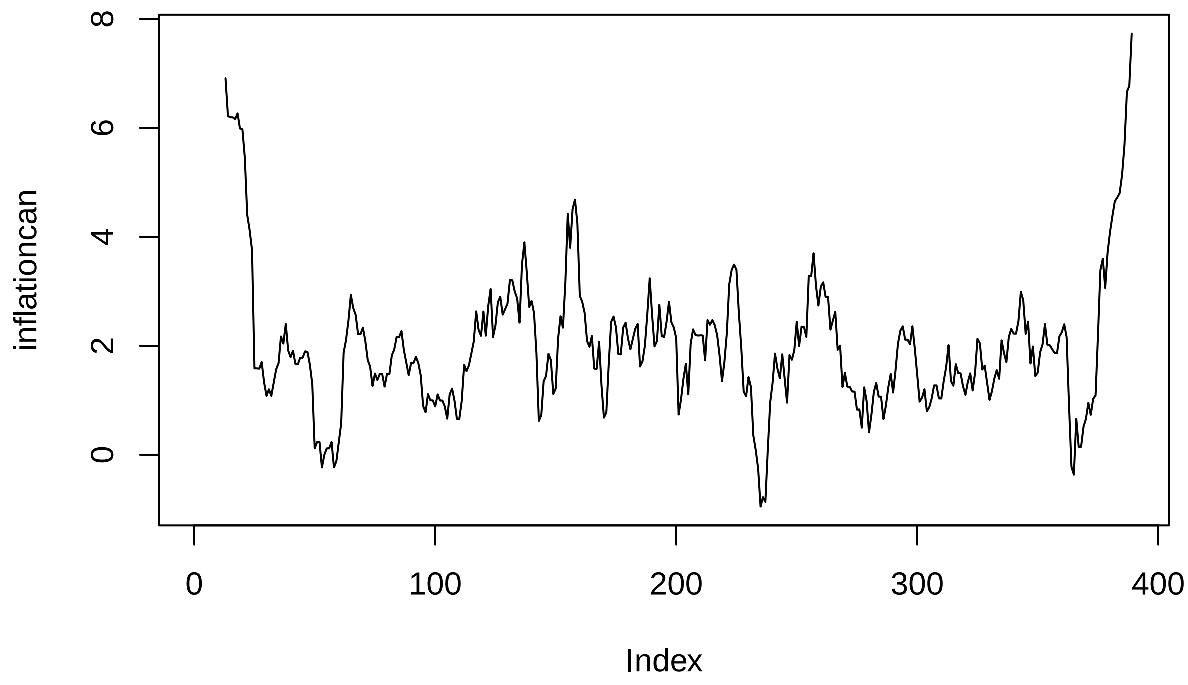


```
for(i in 13:length(PCEPI)){
  PCEPI[i]=inflationus[i]
}
PCEPI[1:12]=NA
PCEPI=na.omit(PCEPI)
PCEPI=PCEPI["1993/2022-01-01"]

plot(PCEPI, type="l")
```

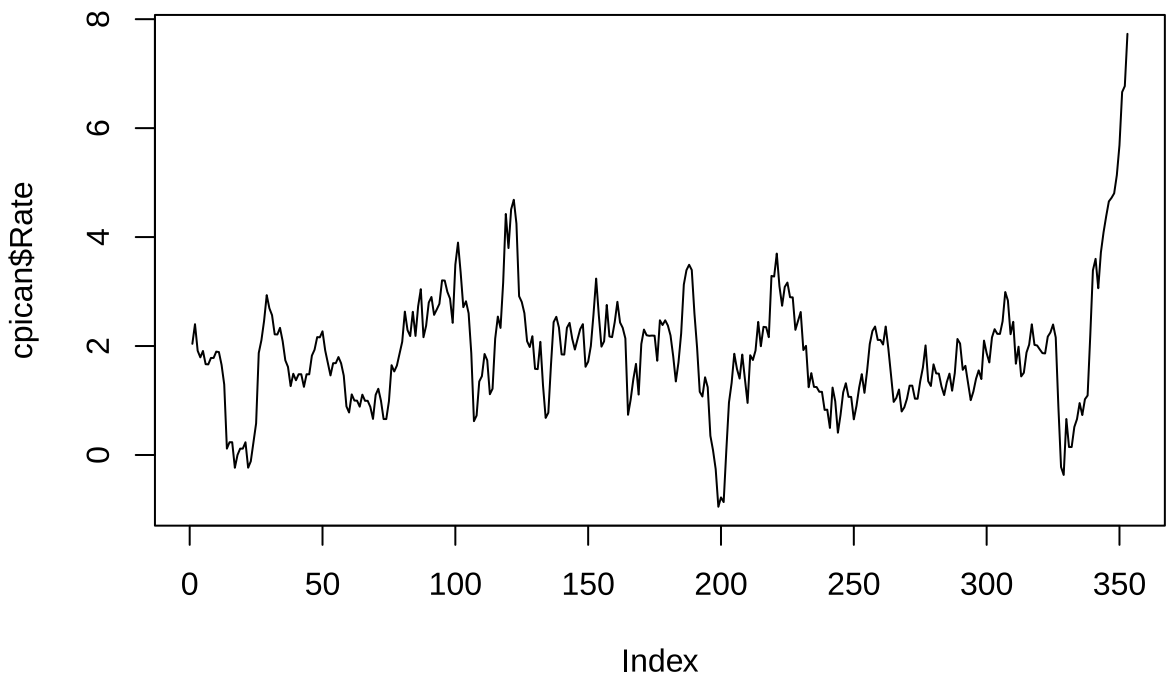


Inflation data for Canada was unfortunately available in unpleasant format thus, we needed to filter only "All-items" values and only for "Canada" as a whole. Then, we used the same for cycles as before to convert the time series to inflation rates.

```
#Canada
canadacpi=read.csv("CanadaCPI3.csv", sep=",")
canadacpi <- canadacpi %>%
  filter(Products.and.product.groups == "All-items") %>%
  filter(GEO == "Canada")
cutcanadacpi=as.data.frame(canadacpi[,11])
cutcanadacpi[,2]=canadacpi[,1]
cutcanadacpi[,3]=cutcanadacpi[,2]
cutcanadacpi[,2]=cutcanadacpi[,1]
cutcanadacpi[,1]=cutcanadacpi[,3]
cutcanadacpi[,3]=NULL
names(cutcanadacpi)=c("Date","Rate")
cpican=cutcanadacpi


inflationcan=c()
for(i in 13:length(cpican$Rate)){
  inflationcan[i]=((as.numeric(cpican$Rate[i])-as.numeric(cpican$Rate[(i-12)]))/(as.numeric(cpican$Rate
}
plot(inflationcan, type="l")
```

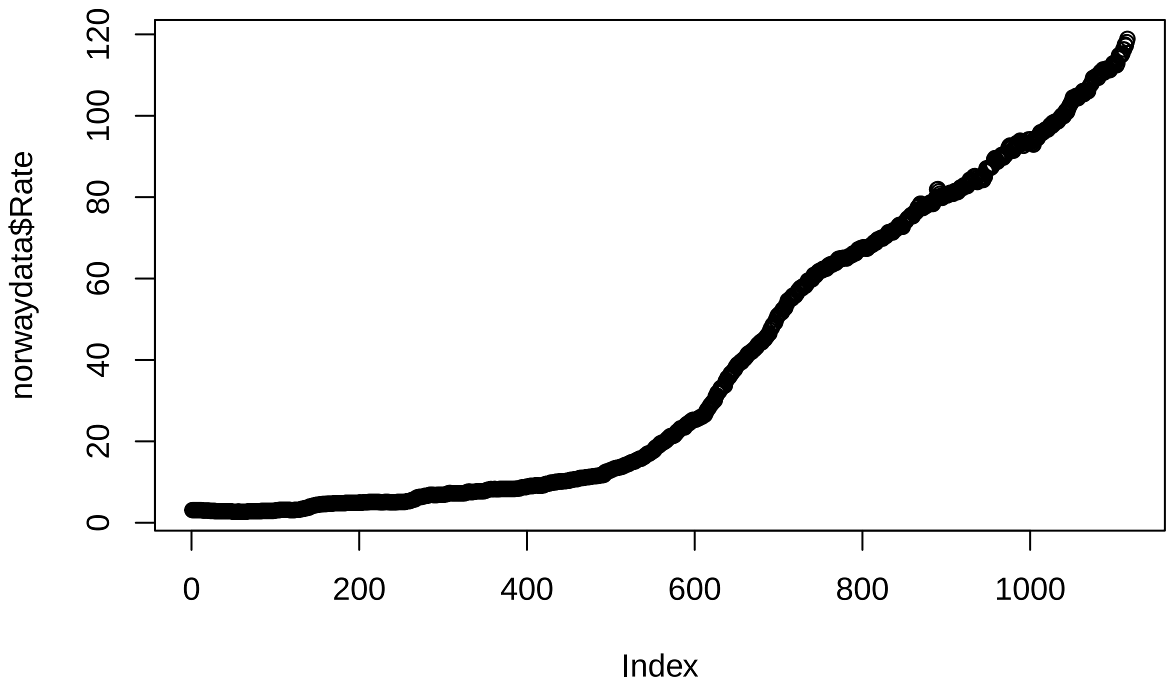


```
for(i in 13:length(cpican$Rate)){
  cpican$Rate[i]=inflationcan[i]
}
cpican[1:12,]=NA
cpican=na.omit(cpican)
cpican=cpican[25:length(cpican$Rate),]

plot(cpican$Rate, type="l")
```

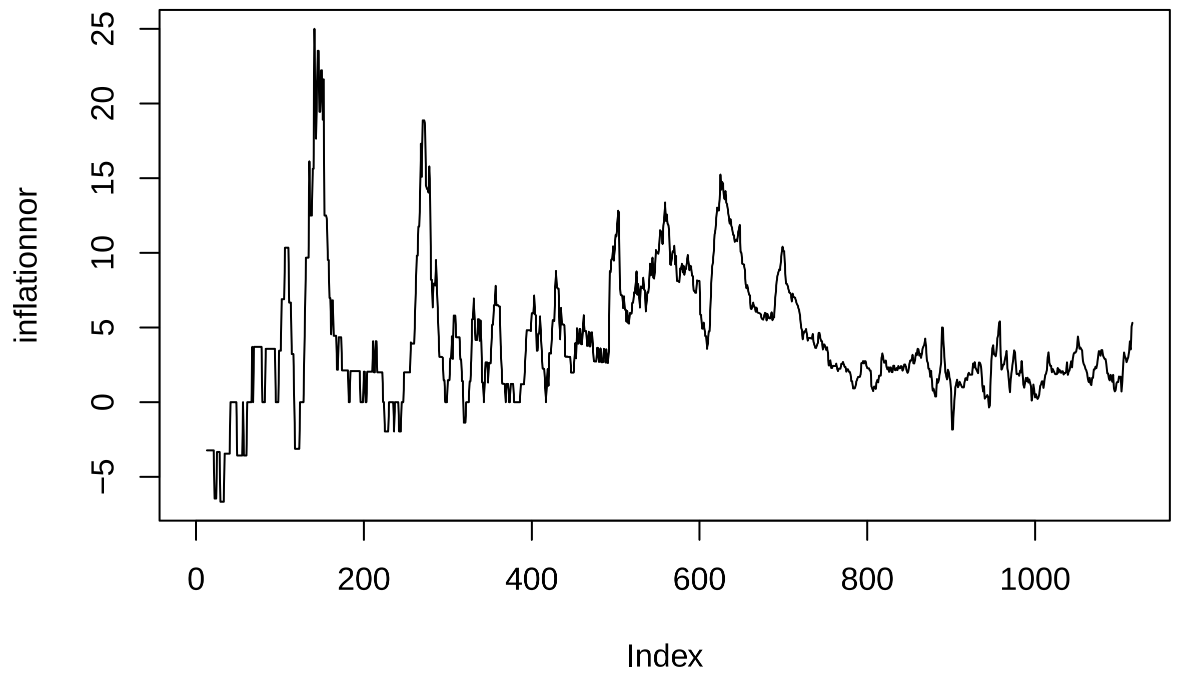


Data for Norway was presented in an even worse format than Canadian data. In fact, every row represented a different year and every column was filled with a different month. Thus, we had to design a for cycle in order to straighten up the data in an orderly manner. Then, we used the same for cycles as before to convert the time series to inflation rates.

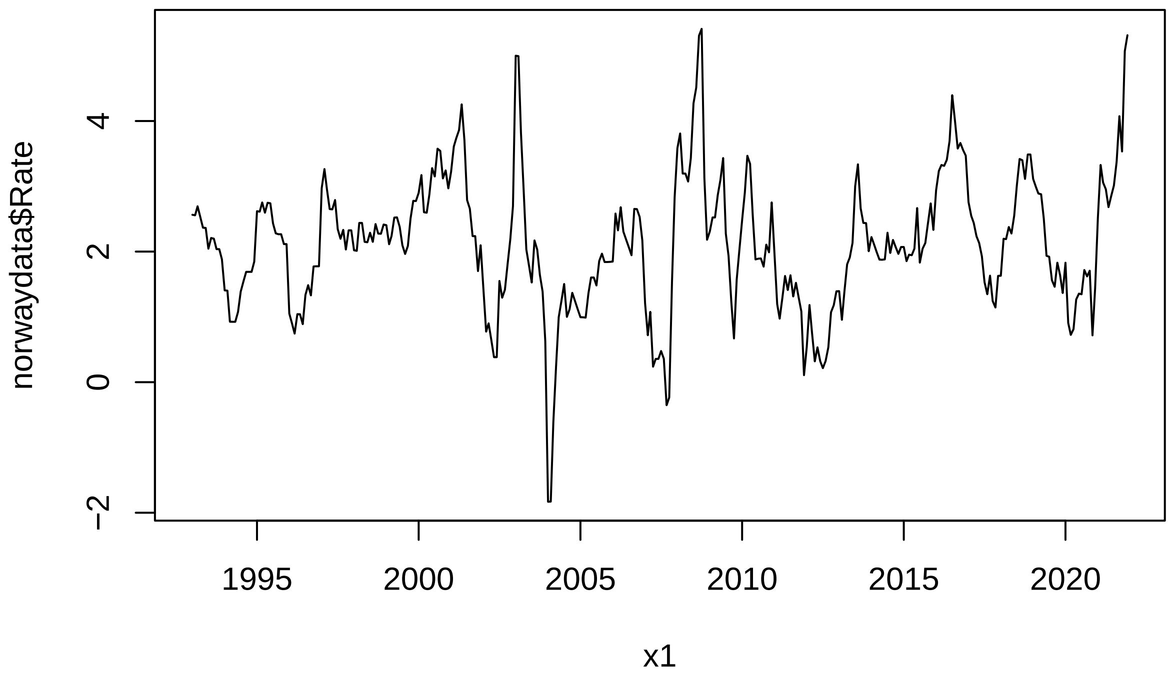
```
#Norway
norwaycpi=read.csv("Norway-CPI.csv", sep=";")
norwaydata=matrix(NA,1,2)
norwaydata=as.data.frame(norwaydata)
names(norwaydata)=c("Date","Rate")
count=1
for (i in 2:length(norwaycpi$X)){
  for (s in 12:1){
    norwaydata[count,2]=norwaycpi[i,2+s]
    count=count+1
  }
}
norwaydata$Rate=rev(norwaydata$Rate)
plot(norwaydata$Rate)
```



```r
norwaydata$Date <- data.frame(time = seq(as.Date('1929-01-01'), by = 'months', length = 1116))

inflationnor=c()
for(i in 13:length(norwaydata$Rate)){
  inflationnor[i]=((as.numeric(norwaydata$Rate[i])-as.numeric(norwaydata$Rate[(i-12)]))/(as.numeric(nor
}
plot(inflationnor, type="l")
```



```
for(i in 13:length(norwaydata$Rate)){
  norwaydata$Rate[i]=inflationnor[i]
}
norwaydata[1:12,]=NA
norwaydata=na.omit(norwaydata)
norwaydata=norwaydata[757:1116,]

plot(norwaydata$Date,norwaydata$Rate, type="l")
```



Data for Sweden was unexpectedly well presented and also it was already converted into inflation rates.

```
#Sweden
swedencpi=read.csv("Sweden-CPI.csv", sep=" ")
names(swedencpi)=c("Date","Rate")
swedencpi=swedencpi[73:length(swedencpi$Date),]
```

Data for UK was also well presented and already converted into inflation rates.

```
#UK
ukcpi=read.csv("UK-CPI.csv", sep=",")
ukcpi=ukcpi[222:length(ukcpi$Title),]
names(ukcpi)=c("Date","Rate")
```

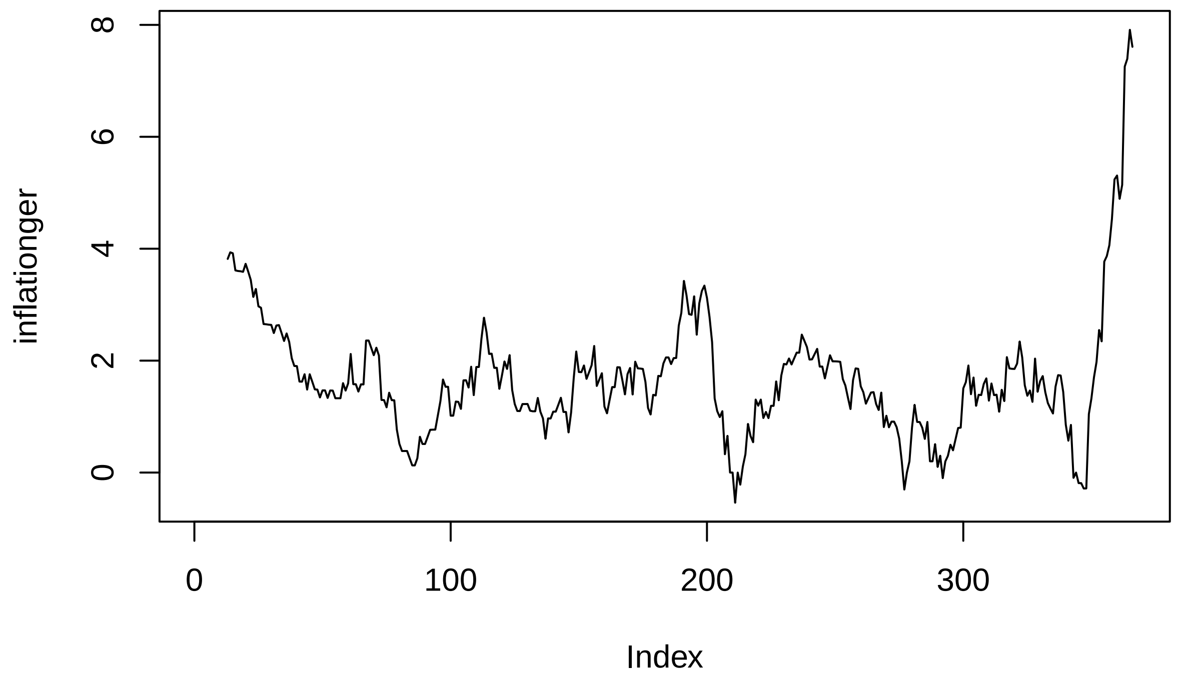
Data for EU was also well presented and already converted into inflation rates.

```
#EU (from 1999 onward)
eucpi=read.csv("eu-CPI.csv", sep=",")
eucpi=eucpi[4466:4770,7:8]
eucpi=eucpi[25:length(eucpi$TIME_PERIOD),]
names(eucpi)=c("Date","Rate")
```

Data for Germany was well presented but nonetheless we had to convert it into inflation rates using the usual two for cycles.

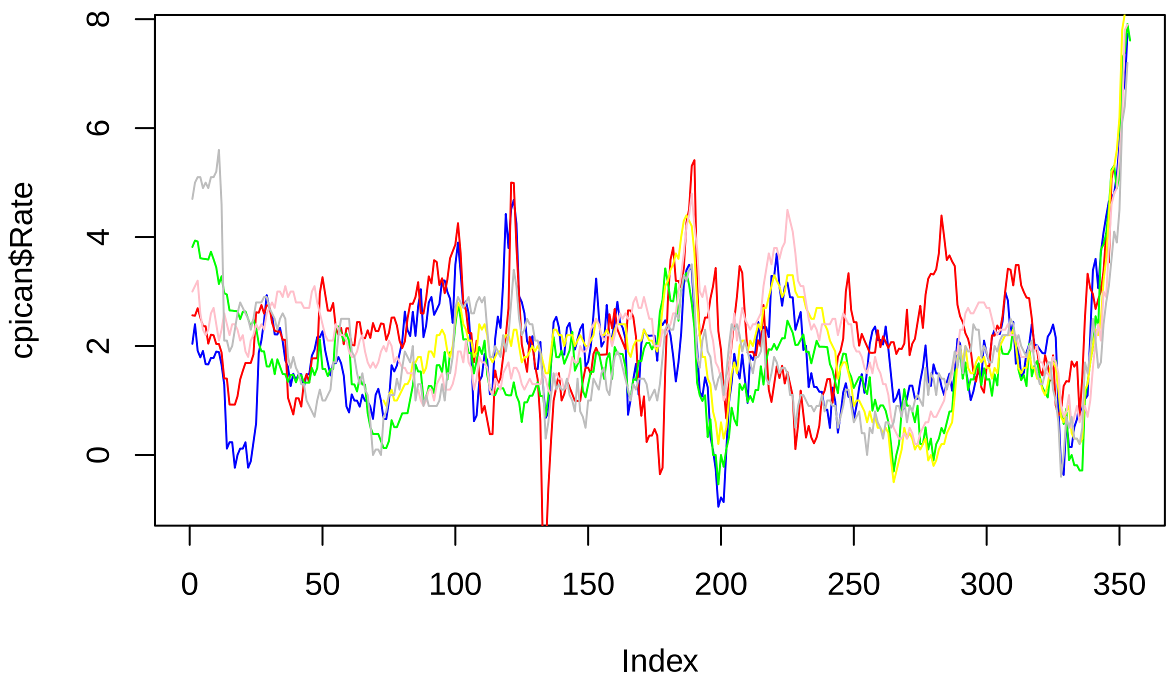
```
#Germany (from 1993 onward)
gercpi=read.csv("Germany-CPI.csv", sep=",")
gercpi=gercpi[,c(1,2)]
names(gercpi)=c("Date","Rate")
```

```
inflationger=c()
for(i in 13:length(gercpi$Rate)){
  inflationger[i]=((as.numeric(gercpi$Rate[i])-as.numeric(gercpi$Rate[(i-12)]))/(as.numeric(gercpi$Rate
}
plot(inflationger, type="l")
```

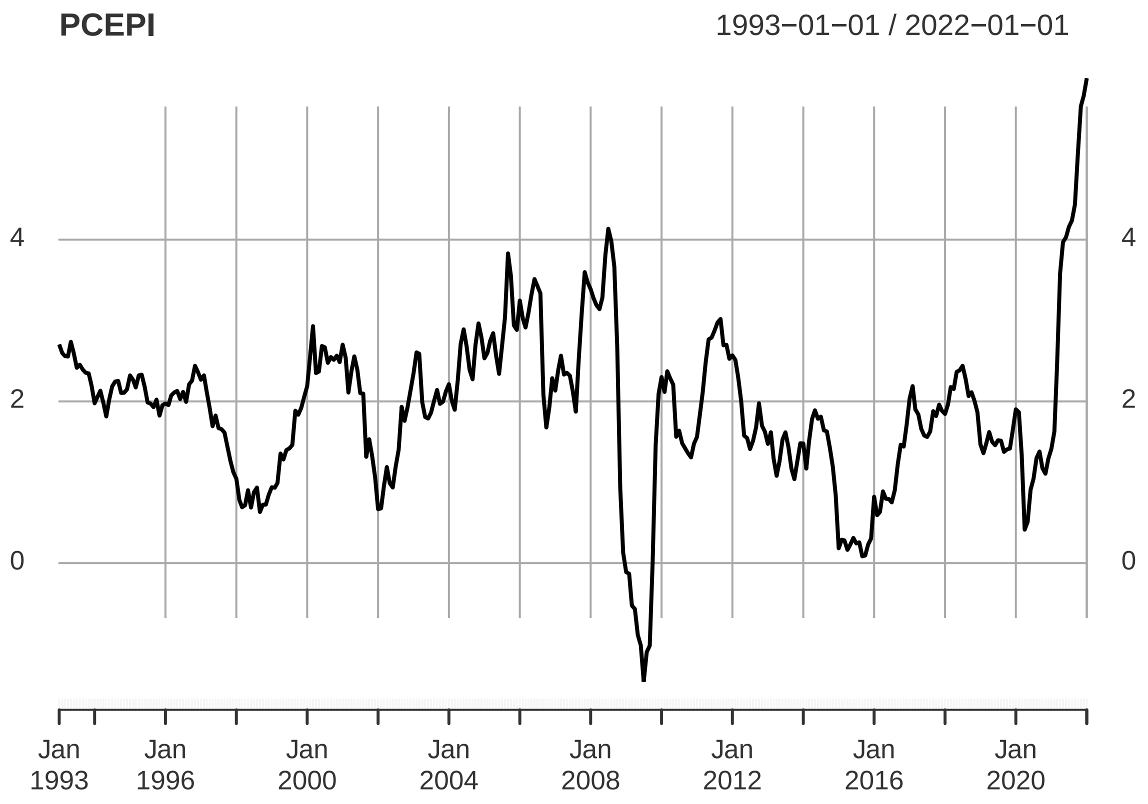


```
for(i in 13:length(gercpi$Rate)){
  gercpi$Rate[i]=inflationger[i]
}
gercpi[1:12,]=NA
gercpi=na.omit(gercpi)
```

```
#Graph with all the inflation rates

plot(cpican$Rate, type="l", col="blue", dev="svg")
lines(norwaydata$Rate, type="l", col="red")
lines(gercpi$Rate, type="l", col="green")
eucpigraph=eucpi
nas=matrix(NA,72,1)
eucpigraph=append(eucpigraph$Rate, nas,after=0)
lines(eucpigraph, type="l", col="yellow")
lines(ukcpi$Rate, type="l", col="pink")
lines(swedencpi$Rate, type="l", col="grey")
```



```
lines(PCEPI)
```



#Divide time series in chunks for analysis

```
#Euro area
eupart1=data.frame(eucpi[1:228,])
eupart2=data.frame(eucpi[1:96,])

#Germany
gerpart1=data.frame(gercpi[1:300,])
gerpart2=data.frame(gercpi[1:168,])
gerpart3=data.frame(gercpi[73:300,])

#UK
ukpart1=data.frame(ukcpi[1:300,])
ukpart2=data.frame(ukcpi[1:168,])
ukpart3=data.frame(ukcpi[73:300,])

#US
uscpi=PCEPI
uspart1=data.frame(uscpi[1:300,])
uspart2=data.frame(uscpi[1:168,])
uspart3=data.frame(uscpi[73:300,])

#Canada
cancpi=cpican
canpart1=data.frame(cancpi[1:300,])
canpart2=data.frame(cancpi[1:168,])
canpart3=data.frame(cancpi[73:300,])
```

```r
#Norway
norcpi=norwaydata
norpart1=data.frame(norcpi[1:300,])
norpart2=data.frame(norcpi[1:168,])
norpart3=data.frame(norcpi[73:300,])

#Sweden
swecpi=swedencpi
swepart1=data.frame(swecpi[1:300,])
swepart2=data.frame(swecpi[1:168,])
swepart3=data.frame(swecpi[73:300,])
```

#State Space Maximum Likelihood Estimator of the Fractional Difference Parameter

```r
#Functions definition

arma21ss <- function(ar1, ar2, ma1, sigma) {
    Tt <- matrix(c(ar1, ar2, 1, 0), ncol = 2)
    Zt <- matrix(c(1, 0), ncol = 2)
    ct <- matrix(0)
    dt <- matrix(0, nrow = 2)
    GGt <- matrix(0)
    H <- matrix(c(1, ma1), nrow = 2) * sigma
    HHt <- H %*% t(H)
    a0 <- c(0, 0)
    P0 <- matrix(1e6, nrow = 2, ncol = 2)
    return(list(a0 = a0, P0 = P0, ct = ct, dt = dt, Zt = Zt, Tt = Tt, GGt = GGt,
                HHt = HHt))
}

## The objective function passed to 'optim'
objective <- function(theta, yt) {
    sp <- arma21ss(theta["ar1"], theta["d"], theta["ma1"], theta["sigma"])
    ans <- fkf(a0 = sp$a0, P0 = sp$P0, dt = sp$dt, ct = sp$ct, Tt = sp$Tt,
               Zt = sp$Zt, HHt = sp$HHt, GGt = sp$GGt, yt = yt)
    return(-ans$logLik)
}

kalmanfilter <- function(y){

    theta <- c(ar1=0,d=0, ma1 = 0, sigma = 1)
  fit <- optim(theta, objective, yt = rbind(y), hessian = TRUE)
    ## Confidence intervals
    p <- cbind(estimate = fit$par,
          lowerCI = fit$par - qnorm(0.975) * sqrt(diag(solve(fit$hessian))),
          upperCI = fit$par + qnorm(0.975) * sqrt(diag(solve(fit$hessian))))## Filter the series with
    sp <- arma21ss(fit$par["ar1"], fit$par["d"], fit$par["ma1"], fit$par["sigma"])
    ans <- fkf(a0 = sp$a0, P0 = sp$P0, dt = sp$dt, ct = sp$ct, Tt = sp$Tt,
              Zt = sp$Zt, HHt = sp$HHt, GGt = sp$GGt, yt = rbind(y))
    plot(ans, type = "acf")
    sm <- fks(ans)
    #plot(sm)
    #lines(y,col="black", lty="dotted")
    return(p)
}
```

We use the Kalman filter: To estimate the parameters using numerical optimisation two functions are specified. The first creates a state space representation out of the four ARFMA parameters. The second is the objective function passed to optim which returns the negative log-likelihood Estimation is then performed using a numeric search by optim. The results are not the same as in the paper, because the state space representation is not the same, due to a lack of information about the form of the matrix. We have tried to implement the fractional state space model as close to the specification in the paper. We consider a fairly general state-space model specification. State space form: The following notation is closest to the one of Koopman et al. The state space model is represented by the transition equation and the measurement equation. Let m be the dimension of the state variable, d be the dimension of the observations, and n the number of observations. The transition equation and the measurement equation are given by

$$(1)\alpha_{t+1} = d_t + T_t\text{ů}\alpha_t + H_t \cdot \eta_t$$

$$(2)y_t = c_t + Z_t \cdot \alpha_t + G_t \cdot \epsilon_t$$

where $\eta_t \sim N(0, Qt)$ and $\epsilon_t \sim N(0, Ht)$. a_0 A vector giving the initial value/estimation of the state variable. P_0 A matrix giving the variance of a0. d_t A matrix giving the intercept of the transition equation. c_t A matrix giving the intercept of the measurement equation. T_t An array giving the factor of the transition equation. Z_t An array giving the factor of the measurement equation. HH_t An array giving the variance of the innovations of the transition equation. GG_t An array giving the variance of the disturbances of the measurement equation. y_t A matrix containing the observations. "NA"-values are allowed. The state equation (1) describes the dynamics of the state vector α_t, driven by deterministic (c_t) and stochastic (η_t) inputs. The observation (or measurement) equation links the observed response y_t with the unobserved state vector, with noise ϵ_t and (possibly) deterministic inputs d_t. The fkf package is a wrap envelope in R of a C routine implementing the filter. The state space model considered is as described by equations (1)-(2) with an added input matrix St in the measurement equation, which is thus

$$(3)y_t = d_t + Z_t \cdot \alpha_t + G_t \cdot \epsilon_t.$$

All system matrices ct, Tt, dt, Zt, Qt, St and Ht may be constant or time-varying, and we used a time constant specification as in the paper. Otherwise, we would have needed to define them as three-dimensional arrays, the last dimension being time.

```
likelihood = function(param){
    likelihoods=c()
    #param a vector contaning (d,sigma) if sigma or d is free set equal to 0
    d<- c(seq(0.1,0.5,by=0.1))
    sigma <- c(seq(1,10,by=1))

    likelihoods = append(likelihoods, kalmanfilter(param[1],param[2])$logLik)
    out<-likelihoods[which.max(likelihoods)]
    return(out)
}



posterior = function(param,d){
    options(digits=6)
    like <- likelihood(param)
    if (d==TRUE) {
       prior<- punif(param[1],0,0.5)
    } else {
        prior <- punif(param[2],0,10)
    }
    post <- exp(like) * prior
    return(post)
}
run_metropolis_MCMC = function(startvalue, iterations){
    chain = array(dim = c(iterations+1,2))
    chain[1,] = startvalue

    for (i in 2:iterations){
        chain[i,]<-cbind(runif(1,0,0.5),runif(1,0,10))
        probab = exp(posterior(cbind(chain[i,1],chain[i-1,2]),TRUE) - posterior(chain[i-1,],TRUE))
        probab =min(1,probab)
        if (runif(1) < probab){
            chain[i,1] <- d <-chain[i,1]
        }else{
            chain[i,1] <- d <- chain[i-1,1]
        }

        sigma <- chain[i,2]
        probab = exp(posterior(cbind(d,sigma),FALSE) - posterior(cbind(d,chain[i-1,2]),FALSE))
        probab =min(1,probab)
        if (runif(1) < probab){
            chain[i,2] <- sigma
        }else{
            chain[i,2] <- chain[i-1,1]
        }

        print(paste("Iteration:",i,"d=",chain[i,1], sep = " ", collapse = NULL))
    }
    return(chain)
}
```

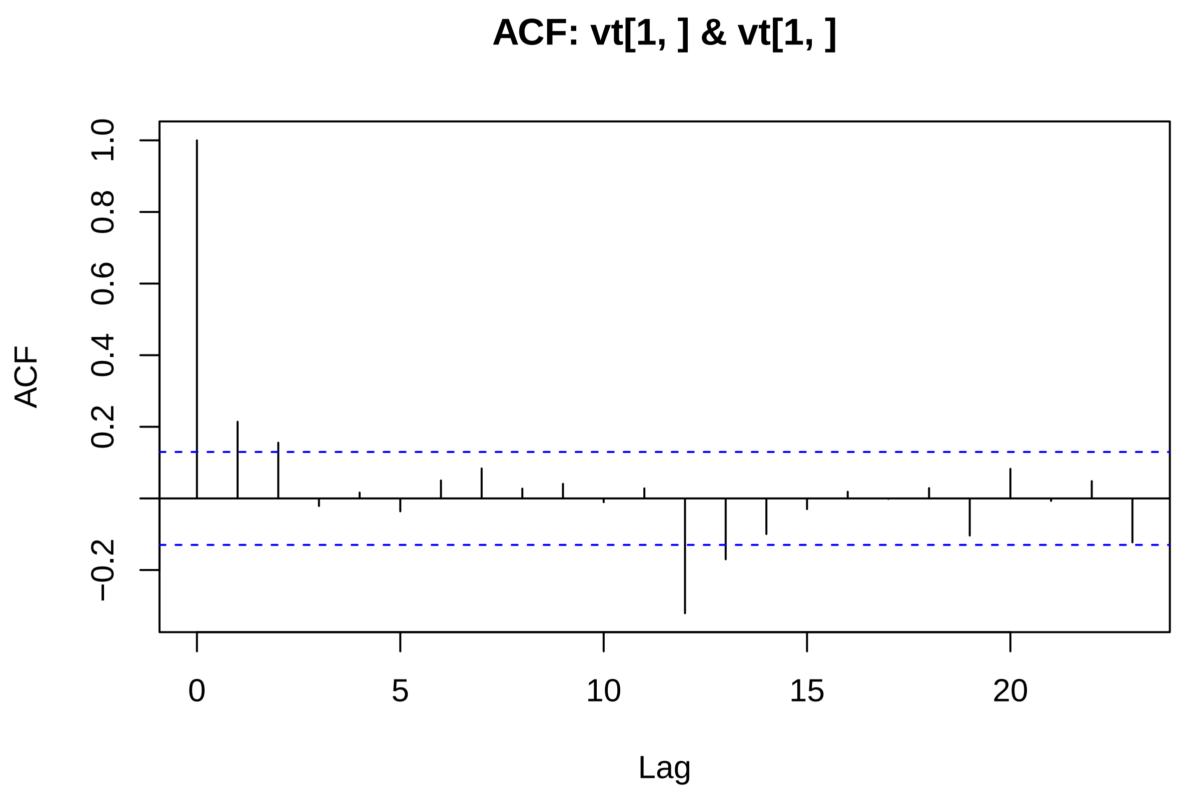
As we had implemented, we applied it to the state space model that we have specified in the arma21ss function. The results were very different from the results of the paper, that issue was the specification of the state space model, because of the matrix Tt (The array giving the factor of the transition equation). We had tried to implement the Tt matrix and all the other matrices, but without the specification of the GGt matrix (the array giving the variance of the disturbances of the measurement equation), it is not feasible. So we exclude the MCMC Bayesian method.

#Results

```
library(knitr)
print("Table1.1: EU 1999-2017")
```

```
## [1] "Table1.1: EU 1999-2017"
```

```
kable(kalmanfilter(eupart1$Rate))
```

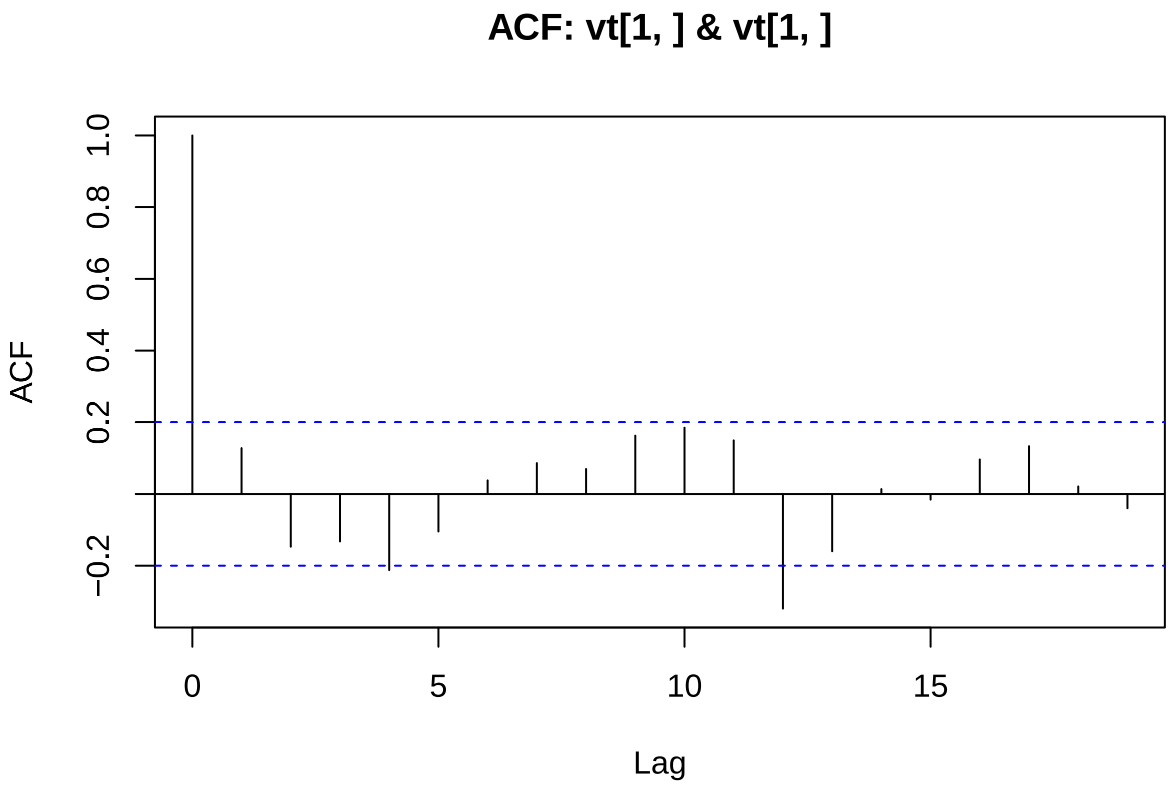


	estimate	lowerCI	upperCI
ar1	0.0417721	-0.0047236	0.0882679
d	0.9475423	0.9010337	0.9940508
ma1	1.0022554	0.9943674	1.0101433
sigma	0.2240787	0.2034706	0.2446868

```
print("Table1.2: EU 1999-2006")
```

```
## [1] "Table1.2: EU 1999-2006"
```

```
kable(kalmanfilter(eupart2$Rate))
```



	estimate	lowerCI	upperCI
ar1	-0.0018261	-0.0250908	0.0214386
d	1.0164474	0.9910613	1.0418334
ma1	1.1012450	1.0988962	1.1035939
sigma	0.1807183	0.1407781	0.2206586