The Final Project

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

06-07-2022

"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:

- A strict focus on inflation targeting may increase volatility elsewhere in the economy.
- 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.

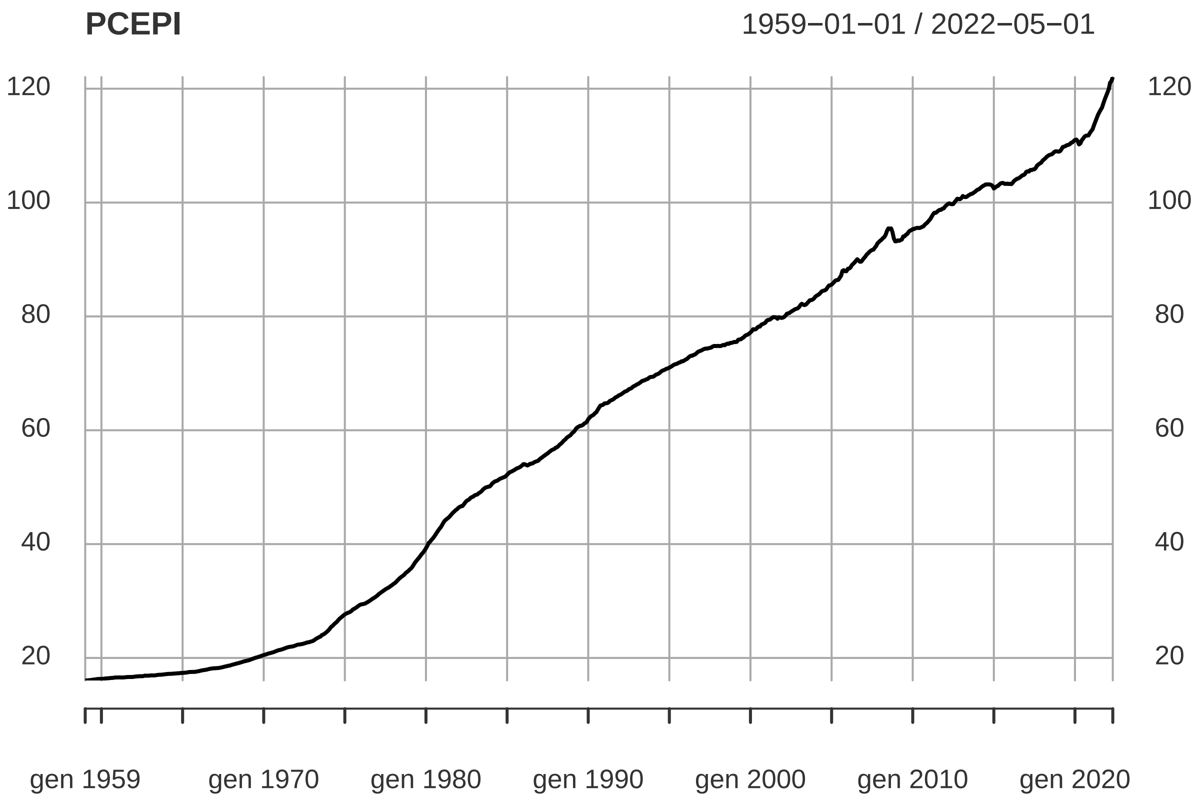
We then applied the algorithm to estimate the fractional integration order for seven economies (Canada, Euro area, Germany, Norway, Sweden, the United Kingdom and the United States) between 1993 and 2017 using monthly data. We divided the period in different chunks to account for the effect of the financial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

Data

USA

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

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



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

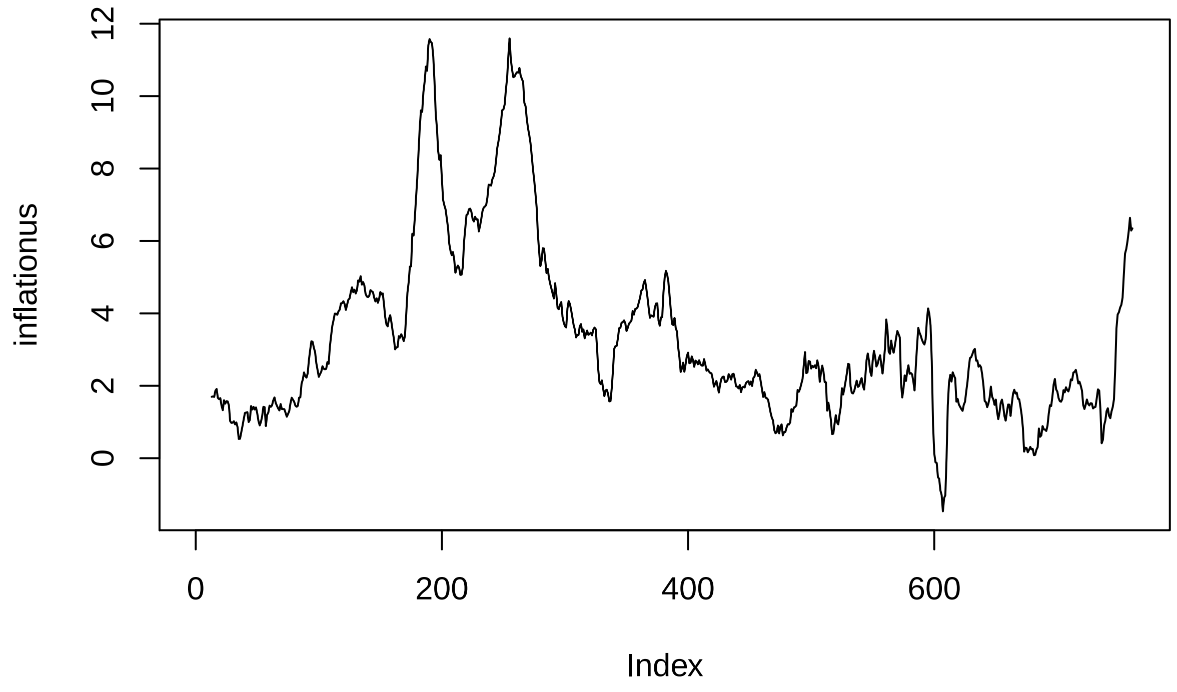
```
## integer(0)
```

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

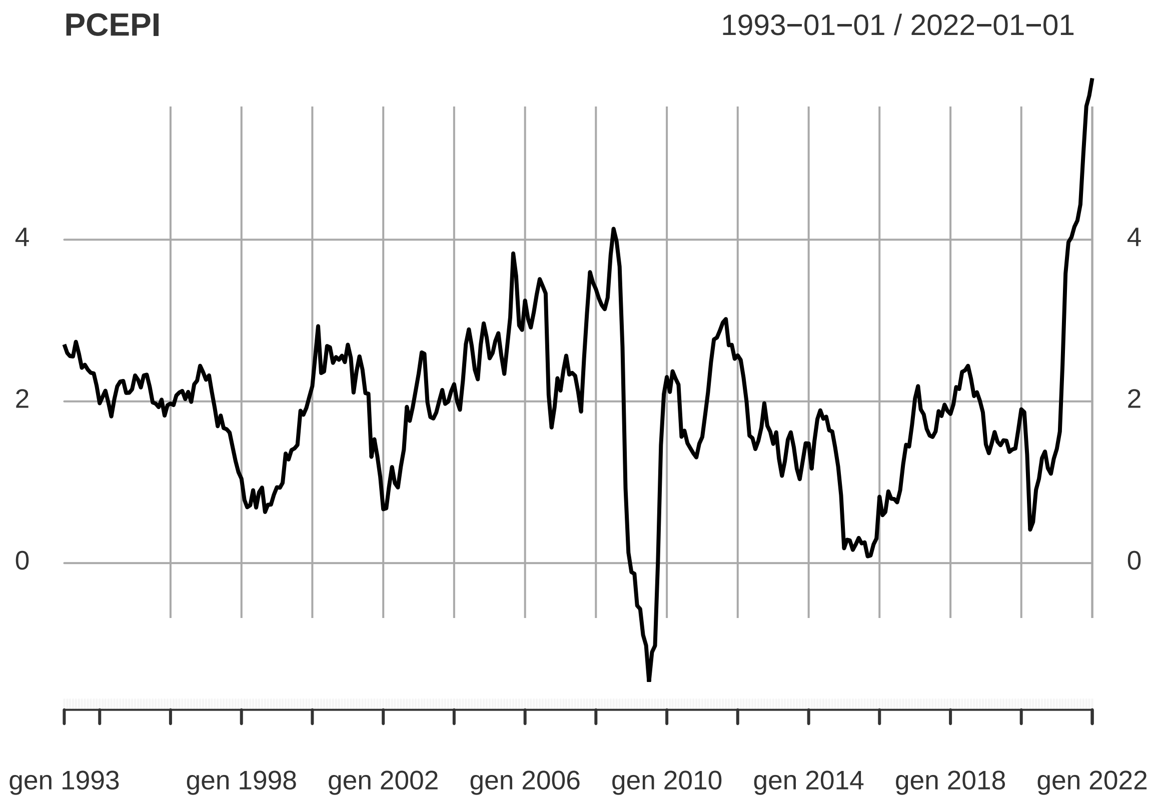
```
## integer(0)
```

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.

```r
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")
```



```r
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")
```

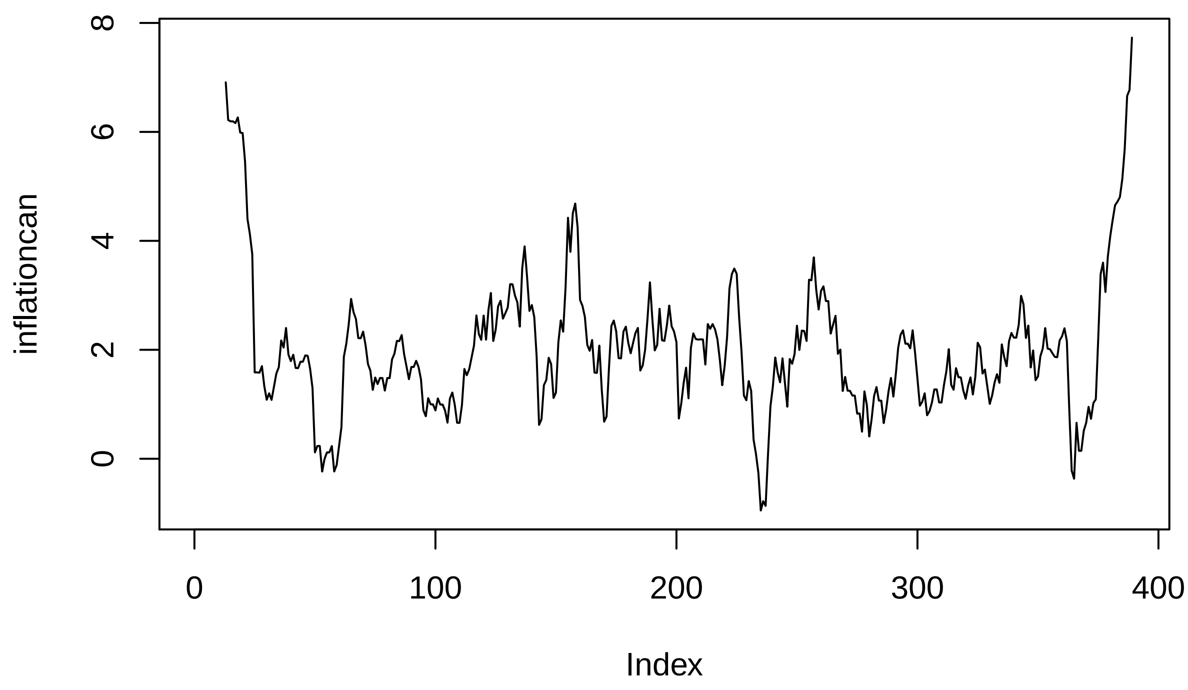


Canada

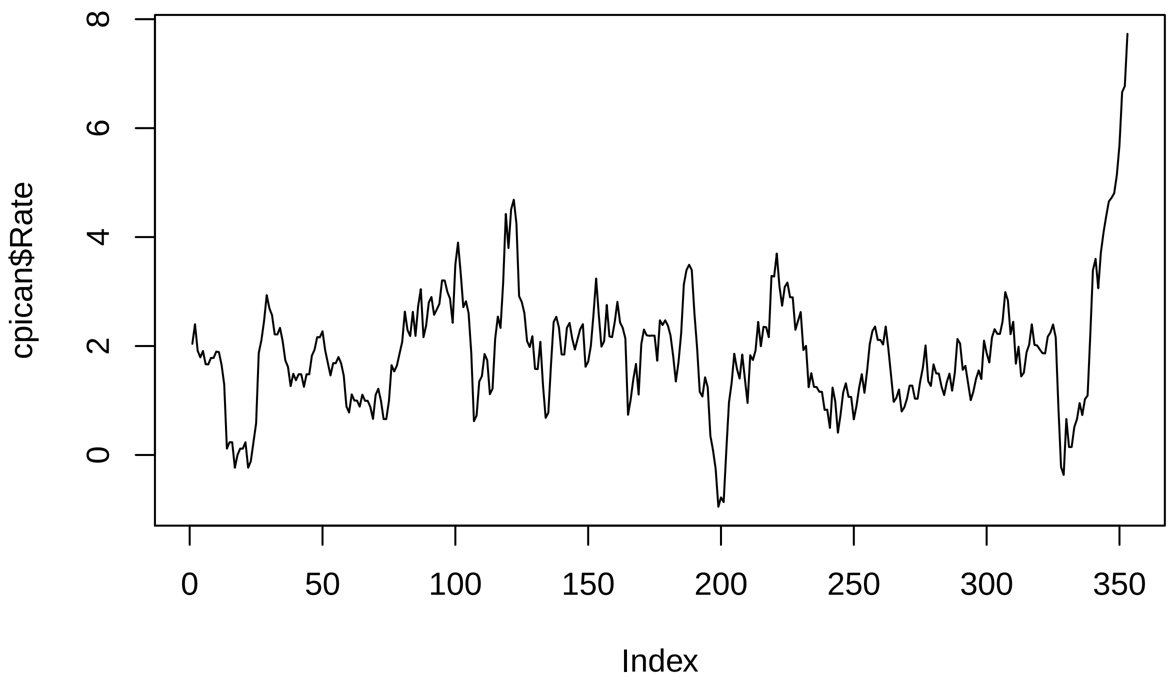
Inflation data for Canada was unfortunately available in unpleasant format :(. Thus, we needed to filter only "All-items" values and only for "Canada" as a whole. Then, we used the same for cycles as before to convert the time series to inflation rates.

```r
#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[(i-12)])))*100
}
```

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



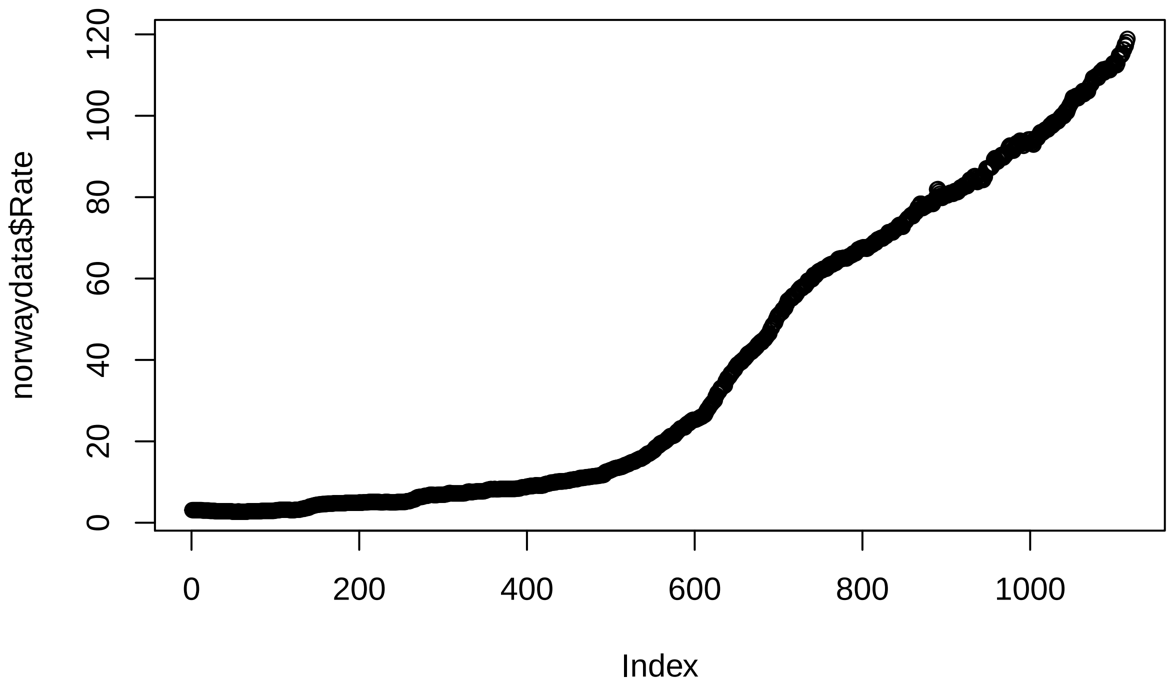
```
for(i in 13:length(cpican$Rate)){
  cpican$Rate[i]=inflationcan[i]
}
cpican[1:12,]=NA
cpican=na.omit(cpican)
cpican=cpican[25:length(cpican$Rate),]

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



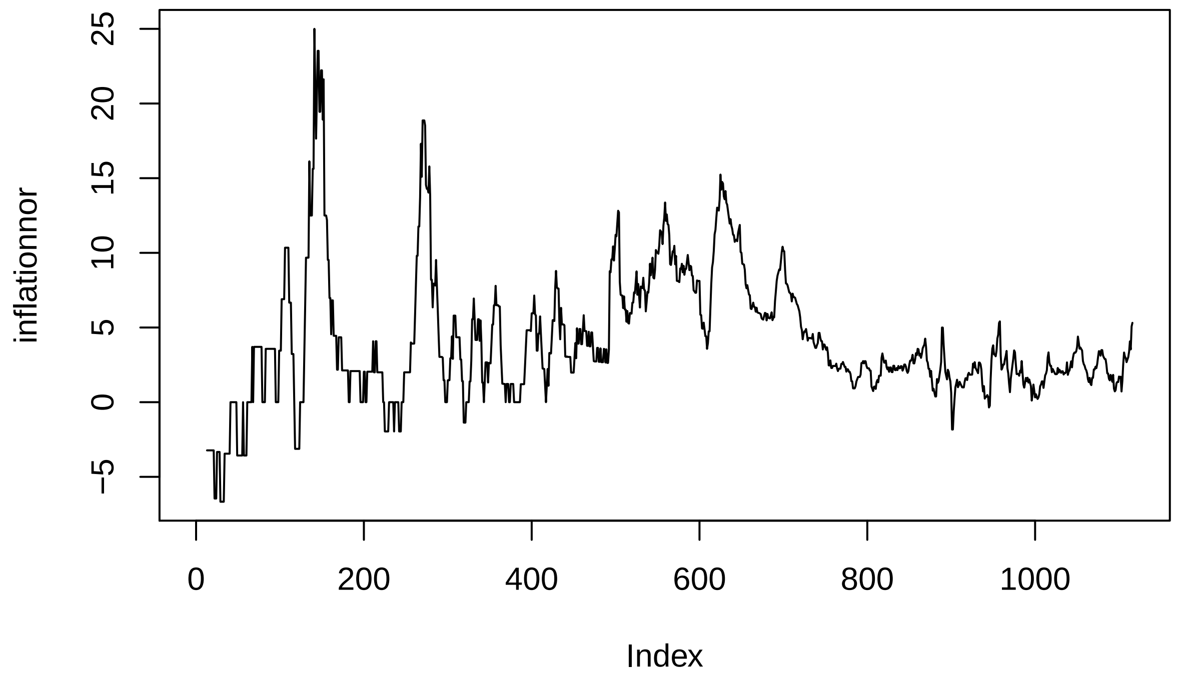
Norway

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 to straighten up the data in an orderly manner. Then, we used the same for cycles as before to convert the time series from CPI to inflation rates.

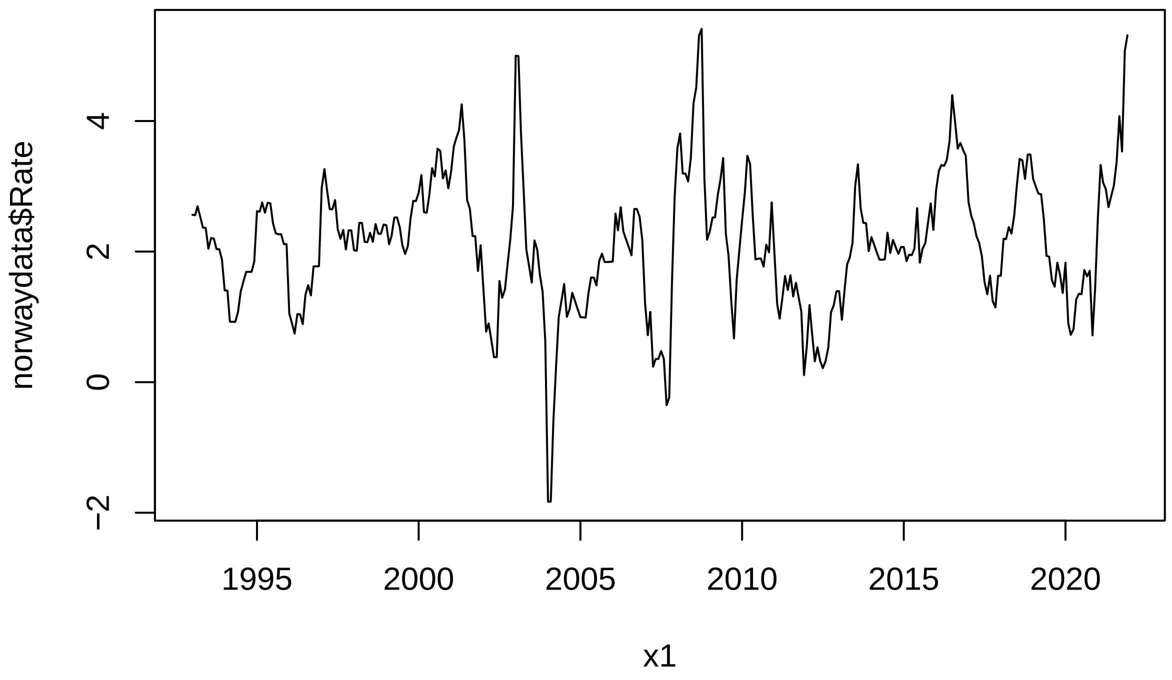
```r
#Norway
norwaycpi=read.csv("Norway-CPI.csv", sep=";")
norwaydata=matrix(NA,1,2)
norwaydata=as.data.frame(norwaydata)
names(norwaydata)=c("Date","Rate")
count=1
for (i in 2:length(norwaycpi$X)){
  for (s in 12:1){
    norwaydata[count,2]=norwaycpi[i,2+s]
    count=count+1
  }
}
norwaydata$Rate=rev(norwaydata$Rate)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



```
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")
```



```r
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")
```



Sweden

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

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

UK

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

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

EU

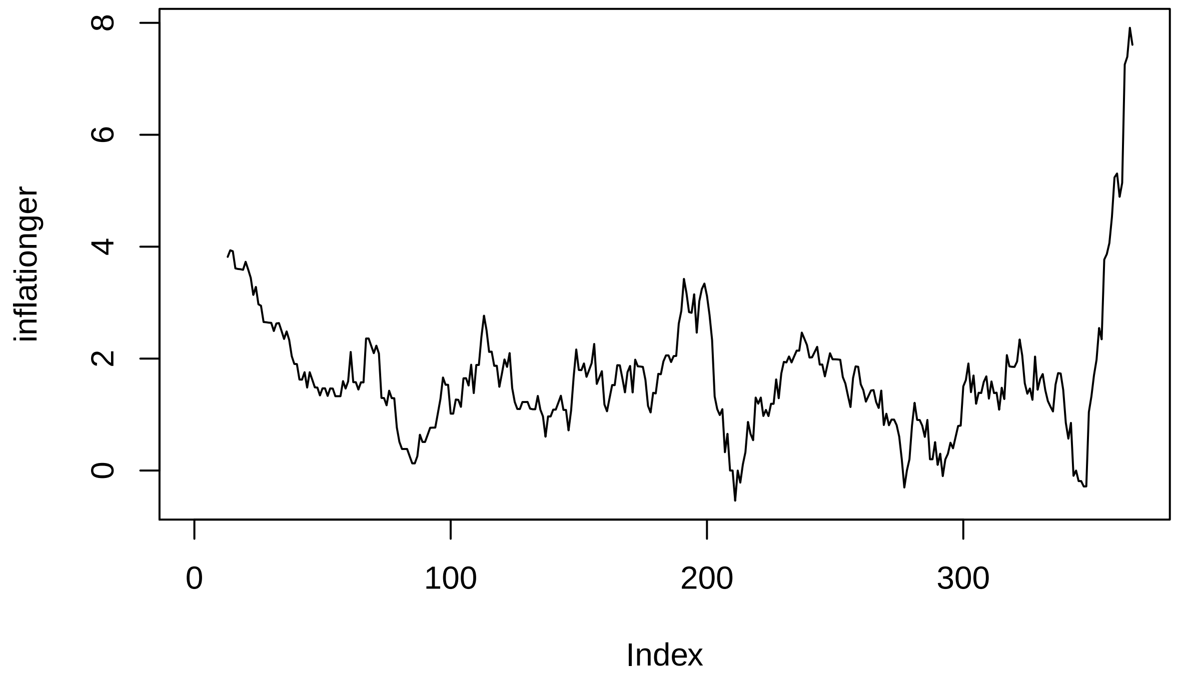
Data for EU was also well presented and already converted into inflation rates.

```
#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")
```

Germany

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

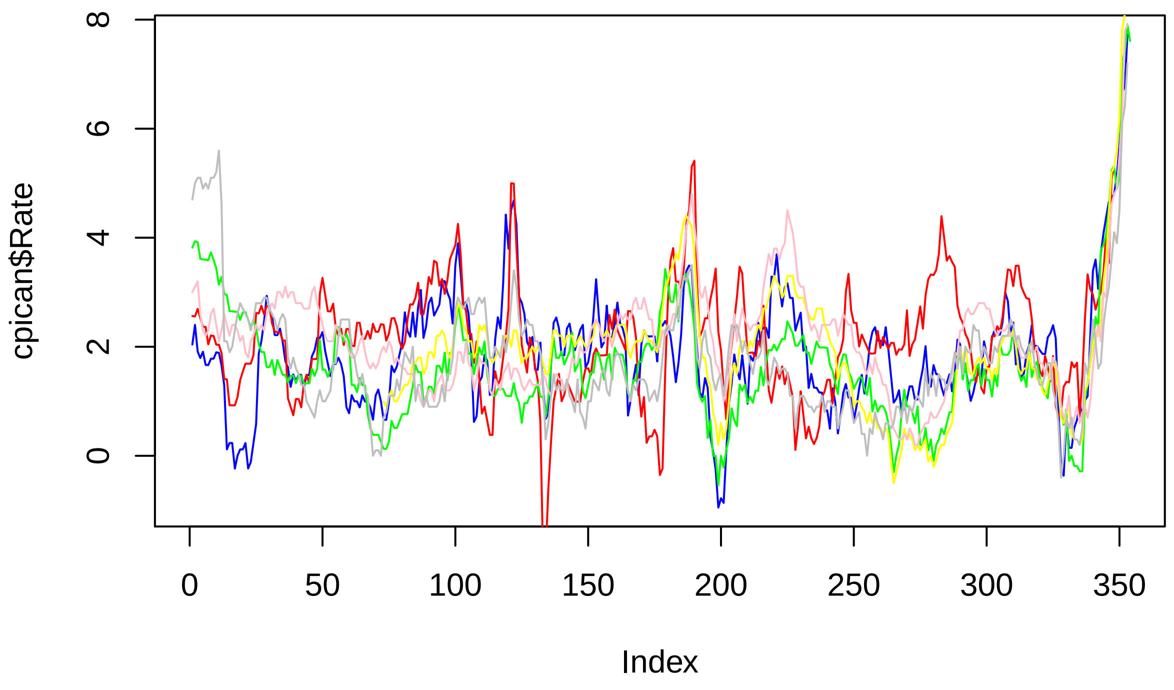
```
#Germany (from 1993 onward)
gercpi=read.csv("Germany-CPI.csv", sep=",")
gercpi=gercpi[,c(1,2)]
names(gercpi)=c("Date","Rate")

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[(i-12)])))*100
}
plot(inflationger, type="l")
```



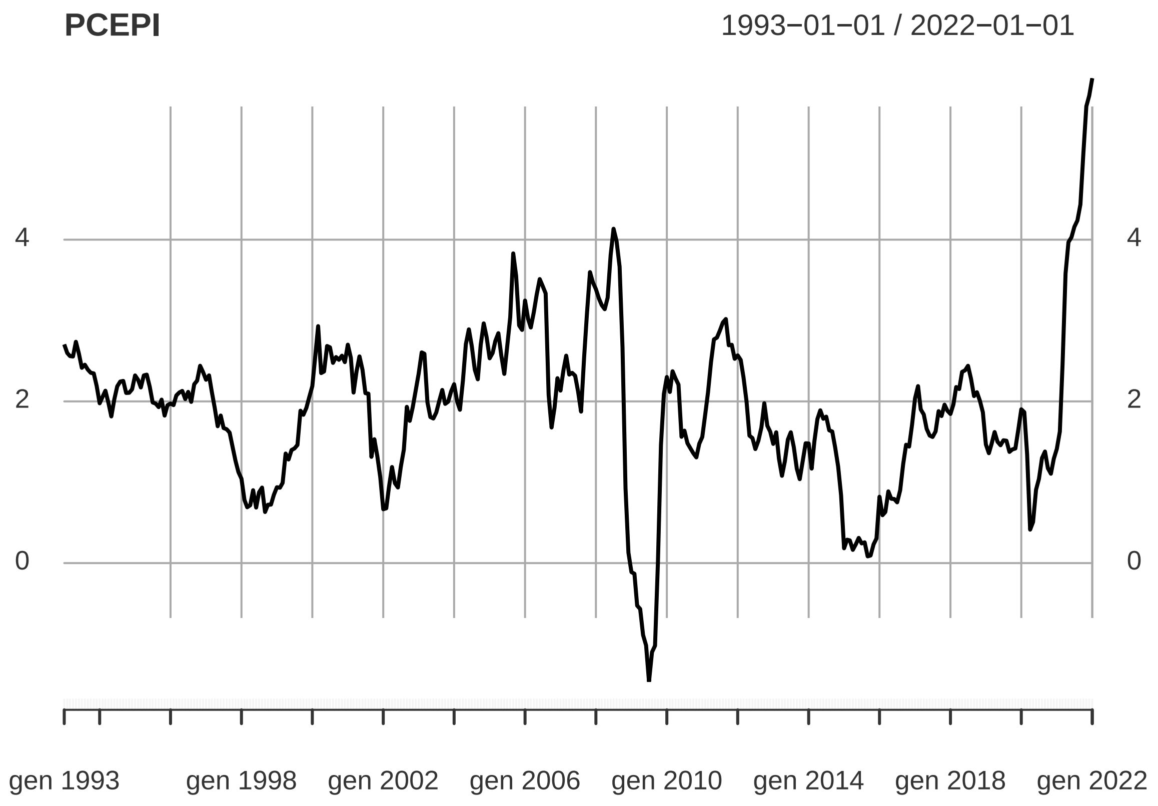
```
for(i in 13:length(gercpi$Rate)){
  gercpi$Rate[i]=inflationger[i]
}
gercpi[1:12,]=NA
gercpi=na.omit(gercpi)
```

Data plotting

```
#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,])

#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

```
#Function 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 estimated parameter values
    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 through numerical optimization two functions are specified. The first creates a state space representation out of the four ARFIMA parameters. The second is the objective function passed to `optim` which returns the negative log-likelihood estimation. This is 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 tried to contact both authors of the paper by mail but we got a reply only on the very last day. So we implemented the fractional state space model as close as possible 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

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

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

where $\eta_t \sim N(0, Qt)$ and $\epsilon_t \sim N(0, Ht)$.

1. a_0 A vector giving the initial value/estimation of the state variable.
2. P_0 A matrix giving the variance of a0.
3. d_t A matrix giving the intercept of the transition equation.
4. c_t A matrix giving the intercept of the measurement equation.
5. T_t An array giving the factor of the transition equation.
6. Z_t An array giving the factor of the measurement equation.
7. HH_t An array giving the variance of the innovations of the transition equation.
8. GG_t An array giving the variance of the disturbances of the measurement equation.
9. 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

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

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

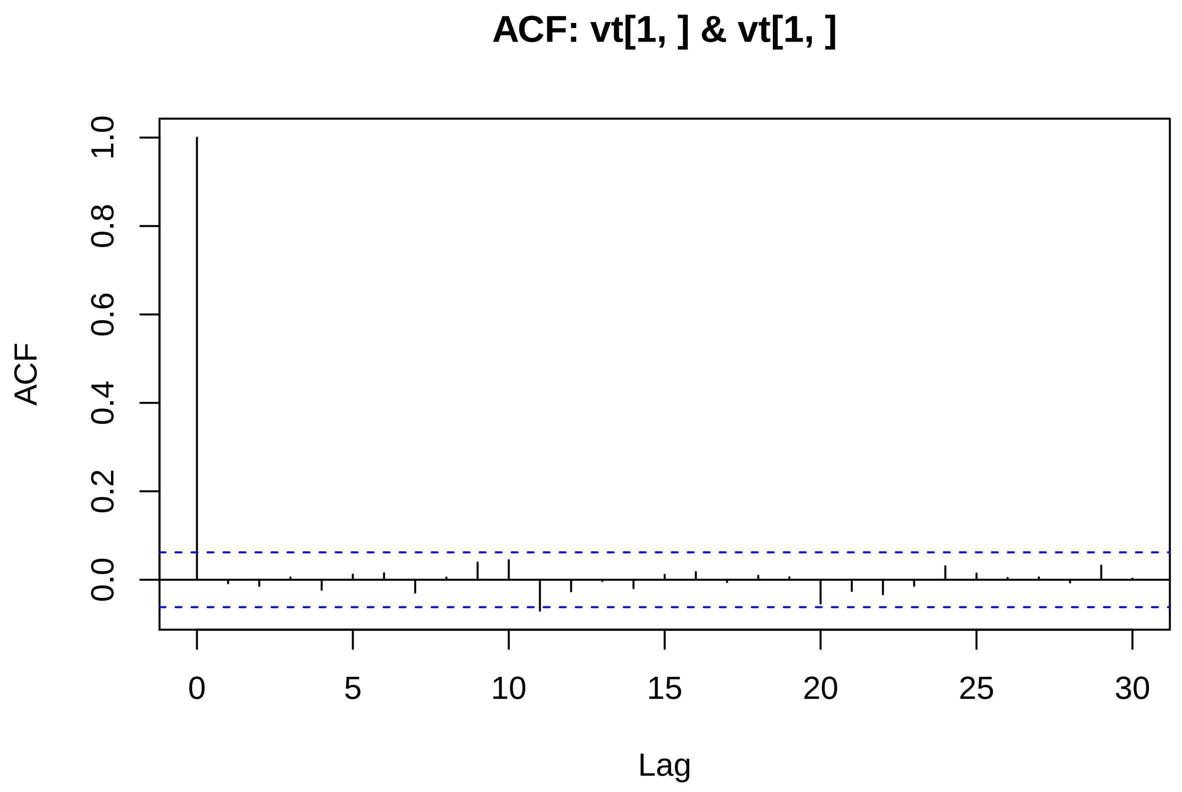
MCMC Bayesian estimation

```r
likelihood = function(param){
    likelihoods=c()
    #param a vector contianing (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)
}
```

We applied this Bayesian MCMC estimator to the state space model that we have specified in the arma21ss function. The results were very different from the ones in the paper. The issue being the specification of the state space model, because of the matrix Tt (the array giving the factor of the transition equation). We 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 decided not to use the MCMC Bayesian method, sticking to the Kalman filter.

Efficiency proof

We simulate an ARFIMA process with $d = 0.9, MA(1) = 0.4, AR(1) = 0.03, \sigma = \sqrt{2}$, and we use our state space model to estimate this coefficient.



	estimate	lowerCI	upperCI
ar1	0.0602035	0.0210902	0.0993169
d	0.8654641	0.8304743	0.9004540
ma1	0.3249525	0.2487821	0.4011228
sigma	1.4449716	1.3815738	1.5083694

As we can se the kalmanfilter estimation is very close to the true parameters of the arfima process.

Difference with the paper

This is one of the functions used in the paper to estimate the fractional difference with an MCMC algorithm:

```
dbayes2=function (Tsim, M,dsim,sigsim, dmcmc1,N,N0, rep) {

  dbayesest=numeric(0)
  dbayessd=numeric(0)

  sigbayesest=numeric(0)
  sigbayessd=numeric(0)
  acceptratio=numeric(0)
  acceptratiosig=numeric(0)

  for (k in 1:rep) {
    acc=0
    accsig=0

    dmcmc=numeric(0)
    dmcmc[1]=dmcmc1

    y=fracdiff.sim(n=Tsim,d=(dsim),innov = rnorm(Tsim,0,sigsim))$series

    sig=numeric()
    sig[1]=sd(y)

    for (n in 2:N)
    {

      dcandi=runif(1,0,0.5)


      K=matrix(0,Tsim,M)
      L=array(0,dim=c(Tsim,M,M))


      T=matrix(0,M,M)
      for (j in 1:M) {
        T[1,j]=-gamma(j-dcandi)/(gamma(j+1)*gamma(-dcandi))
      }


      Z=matrix(0,1,M)
      Z[,1]=1

      H=matrix(0,M,1)
      H[1]=1
      Q=matrix(0,M,M)
      Q[1]=sig[n-1]^2


      a1=matrix(0,Tsim,M)
      P1=array(0,dim=c(Tsim,M,M))
```

```
autocorr=as.vector(acf(y,lag.max=(M))$acf)

P1[1,,]=toeplitz(autocorr[-1])

F=numeric(0)
v=numeric(0)


for (i in 2:M)
{T[i,(i-1)]=1}

for (t in 1:(Tsim-1))
{
  F[t]=Z%*%P1[t,,]%*%t(Z)

  v[t]=y[t]-Z%*%a1[t,]
  K[t,]=T%*%P1[t,,]%*%t(Z)%*%F[t]^(-1)
  L[t,,]=T-K[t,]%*%Z


  a1[t+1,]=T%*%a1[t,] +K[t,]*v[t]
  P1[t+1,,]=T%*%P1[t,,]%*%t(L[t,,])+Q
}

value=sum(log(abs(F)))+sum(v^2/F)


K0=matrix(0,Tsim,M)
L0=array(0,dim=c(Tsim,M,M))


T0=matrix(0,M,M)
for (j in 1:M)
{T0[1,j]=-gamma(j-dmcmc[n-1])/(gamma(j+1)*gamma(-dmcmc[n-1]))
}


Z0=matrix(0,1,M)
Z0[,1]=1

H0=matrix(0,M,1)
H0[1]=1
Q0=matrix(0,M,M)

Q0[1,1]=sig[n-1]^2

a10=matrix(0,Tsim,M)
P10=array(0,dim=c(Tsim,M,M))

P10[1,,]=toeplitz(autocorr[-1])

F0=numeric(0)
v0=numeric(0)
```

```
for (i in 2:M)
{T0[i,(i-1)]=1}

for (t in 1:(Tsim-1))
{
  F0[t]=Z0%*%P10[t,,]%*%t(Z0)

  v0[t]=y[t]-Z0%*%a10[t,]
  K0[t,]=T0%*%P10[t,,]%*%t(Z0)%*%F0[t]^(-1)
  L0[t,,]=T0-K0[t,]%*%Z0


  a10[t+1,]=T0%*%a10[t,] +K0[t,]*v0[t]
  P10[t+1,,]=T0%*%P10[t,,]%*%t(L0[t,,])+Q0
}

value0=sum(log(abs(F0)))+sum(v0^2/F0)

a=min(1, exp(-0.5*value+0.5*value0))
if (runif(1,0,1)<a) {
  dmcmc[n]=dcandi
  acc=acc+1

}
else {
  dmcmc[n]=dmcmc[n-1]
}

sigcandi=runif(1,0,10)

Ksig=matrix(0,Tsim,M)
Lsig=array(0,dim=c(Tsim,M,M))


Tsig=matrix(0,M,M)
for (j in 1:M)
{Tsig[1,j]=-gamma(j-dmcmc[n])/(gamma(j+1)*gamma(-dmcmc[n]))
}


Zsig=matrix(0,1,M)
Zsig[,1]=1

Hsig=matrix(0,M,1)
Hsig[1]=1
Qsig=matrix(0,M,M)

Qsig[1,1]=sigcandi^2

a1sig=matrix(0,Tsim,M)
P1sig=array(0,dim=c(Tsim,M,M))
```

```
P1sig[1,,]=toeplitz(autocorr[-1])

Fsig=numeric(0)
vsig=numeric(0)


for (i in 2:M)
{Tsig[i,(i-1)]=1}

for (t in 1:(Tsim-1))
{
  Fsig[t]=Zsig%*%P1sig[t,,]%*%t(Zsig)

  vsig[t]=y[t]-Zsig%*%a1sig[t,]
  Ksig[t,]=Tsig%*%P1sig[t,,]%*%t(Zsig)%*%Fsig[t]^(-1)
  Lsig[t,,]=Tsig-Ksig[t,]%*%Zsig


  a1sig[t+1,]=Tsig%*%a1sig[t,] +Ksig[t,]*vsig[t]
  P1sig[t+1,,]=Tsig%*%P1sig[t,,]%*%t(Lsig[t,,])+Qsig
}

valuesig=sum(log(abs(Fsig)))+sum(vsig^2/Fsig)

Ksig0=matrix(0,Tsim,M)
Lsig0=array(0,dim=c(Tsim,M,M))


Tsig0=matrix(0,M,M)
for (j in 1:M)
{Tsig0[1,j]=-gamma(j-dmcmc[n])/(gamma(j+1)*gamma(-dmcmc[n]))
}

Zsig0=matrix(0,1,M)
Zsig0[,1]=1

Hsig0=matrix(0,M,1)
Hsig0[1]=1
Qsig0=matrix(0,M,M)

Qsig0[1,1]=sig[n-1]^2

a1sig0=matrix(0,Tsim,M)
P1sig0=array(0,dim=c(Tsim,M,M))

P1sig0[1,,]=toeplitz(autocorr[-1])

Fsig0=numeric(0)
vsig0=numeric(0)

for (i in 2:M)
{Tsig0[i,(i-1)]=1}
```

```
      for (t in 1:(Tsim-1))
      {
        Fsig0[t]=Zsig0%*%P1sig0[t,,]%*%t(Zsig0)

        vsig0[t]=y[t]-Zsig0%*%a1sig0[t,]
        Ksig0[t,]=Tsig0%*%P1sig0[t,,]%*%t(Zsig0)%*%Fsig0[t]^(-1)
        Lsig0[t,,]=Tsig0-Ksig0[t,]%*%Zsig0

        a1sig0[t+1,]=Tsig0%*%a1sig0[t,] +Ksig0[t,]*vsig0[t]
        P1sig0[t+1,,]=Tsig0%*%P1sig0[t,,]%*%t(Lsig0[t,,])+Qsig0
      }

      valuesig0=sum(log(abs(Fsig0)))+sum(vsig0^2/Fsig0)

      asig=min(1, exp(-0.5*valuesig+0.5*valuesig0))
      if (runif(1,0,1)<asig) {
        sig[n]=sigcandi
        accsig=accsig+1

      }
      else {
        sig[n]=sig[n-1]
      }
    }
    dbayesest[k]=(mean(dmcmc[N0:N]))
    dbayessd[k]=(sd(dmcmc[N0:N]))
    sigbayesest[k]=(mean(sig[N0:N]))
    sigbayessd[k]=(sd(sig[N0:N]))

    acceptratio[k]=(acc/n)
    acceptratiosig[k]=(accsig/n)

  }
  print("Bayes MCMC estimation stationary d sigmaunknown")
  print(mean(dbayesest)-dsim)
  print("sd of  d")
  print(sd(dbayessd))
  print("RMSE of d")
  print((mean((dbayesest-dsim)^2))^0.5)

  print("Bayes MCMC estimation stationary sigma")
  print(mean(sigbayesest)-sigsim)
  print("sd of  sigma")
  print(sd(sigbayessd))
  print("RMSE of sigma")
  print((mean((sigbayesest-sigsim)^2))^0.5)

  print("accept ratio")
  print(mean(acceptratio))
  print("accept ratio sig")
  print(mean(acceptratiosig))
}
```

To estimate the fractional difference we used the `fkf` package, but the authors instead adopted a customized method, computing each matrix operation without the aid of pre-made packages. Thus, their method differs significantly from the `fkf` package used in our replication attempt. So this is the reason why the empirical results we found are different from those in the paper.

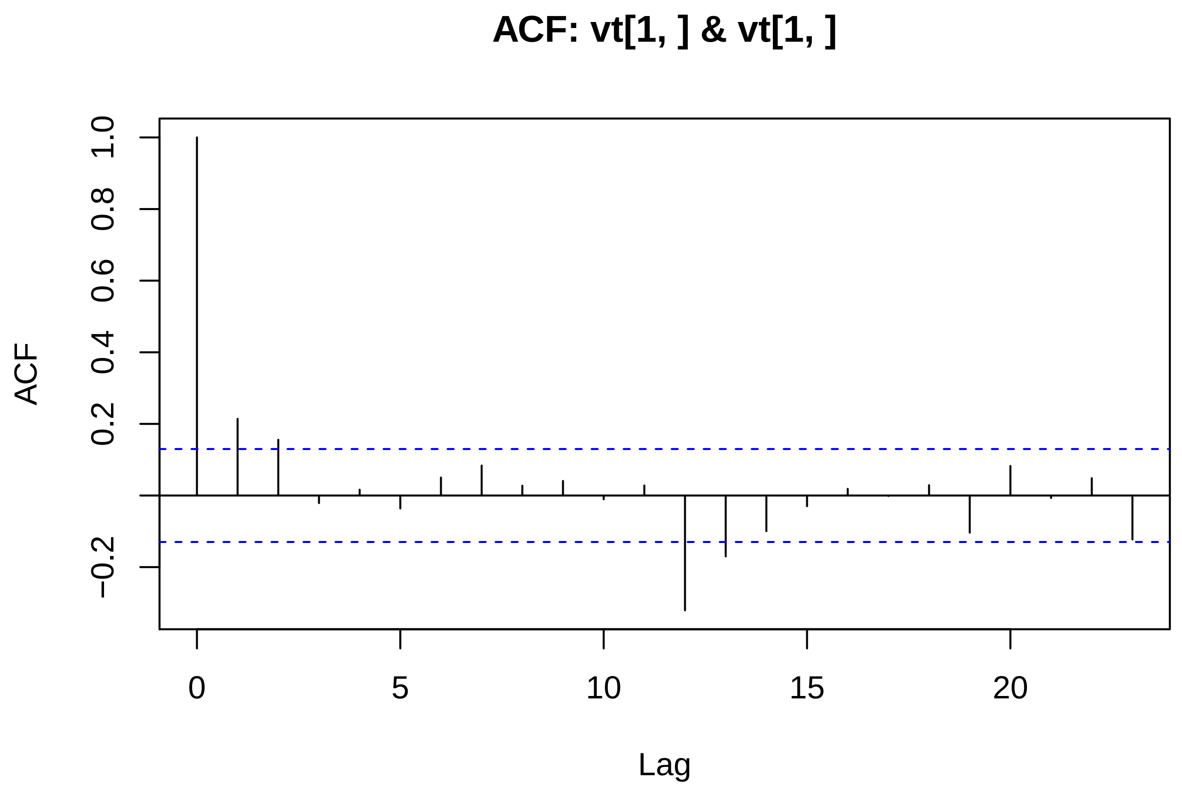
NOTE: The authors of the paper sent us the code to replicate the process on the last day available. So we decided to use our Kalman filter algorithm, even if we had the code to replicate the exact results. We did this because our algorithm can predict an ARFIMA process with a quite high degree of precision (as we saw in the "Efficiency Proof" section), even if it does not replicate the results of the paper in a reliable way. In addition to this, the algorithm designed by the authors is computationally quite demanding, so we didn't have time to try to compute the results (as a reference, it took more than 3 hours and it didn't even finish the first time series).

Empirical Results

We estimate the parameter d. Here d is the fractional integration order which indicates if the CB policy is more flexible with respect to its inflation target. If $d > 0.5$ the CB has a flexible target. If not there is evidence that the CB has sticked to the inflation target despite the business cycle or other economic determinants.

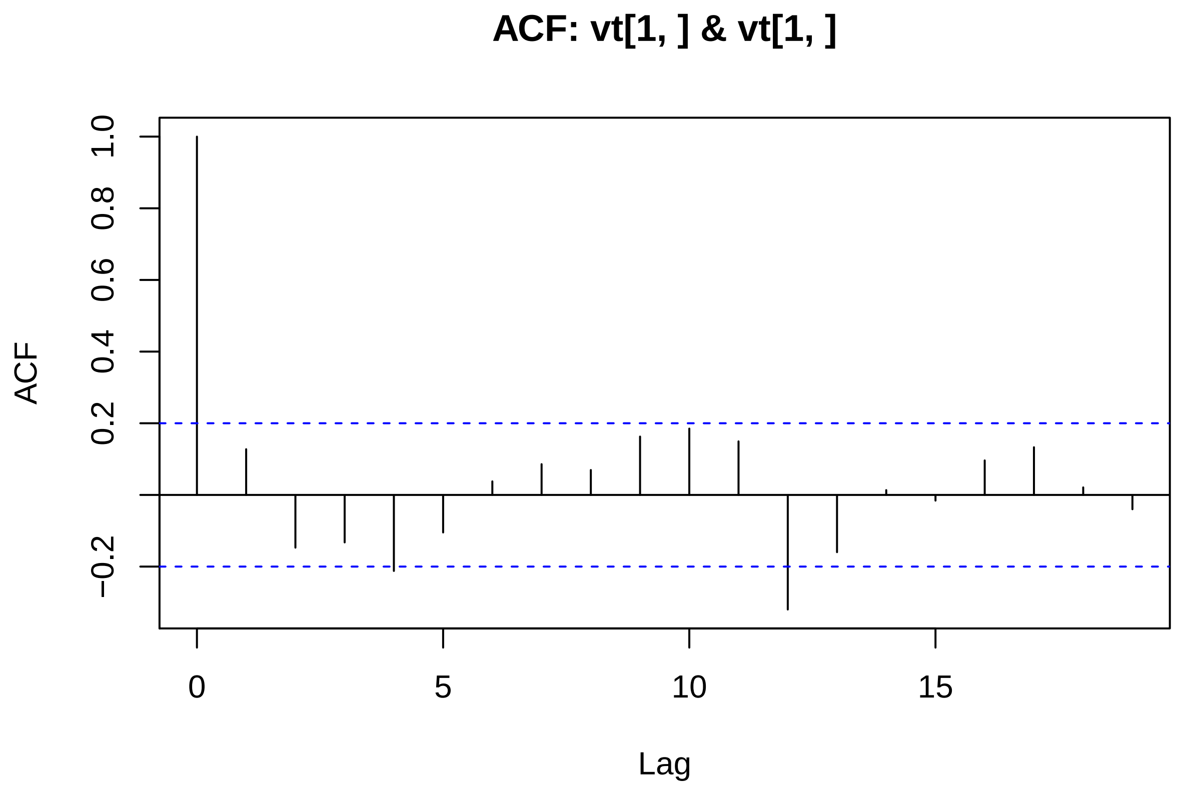
EU

```
## [1] "Table 1.1: EU 1999-2017"
```



	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

```
## [1] "Table 1.2: EU 1999-2006"
```

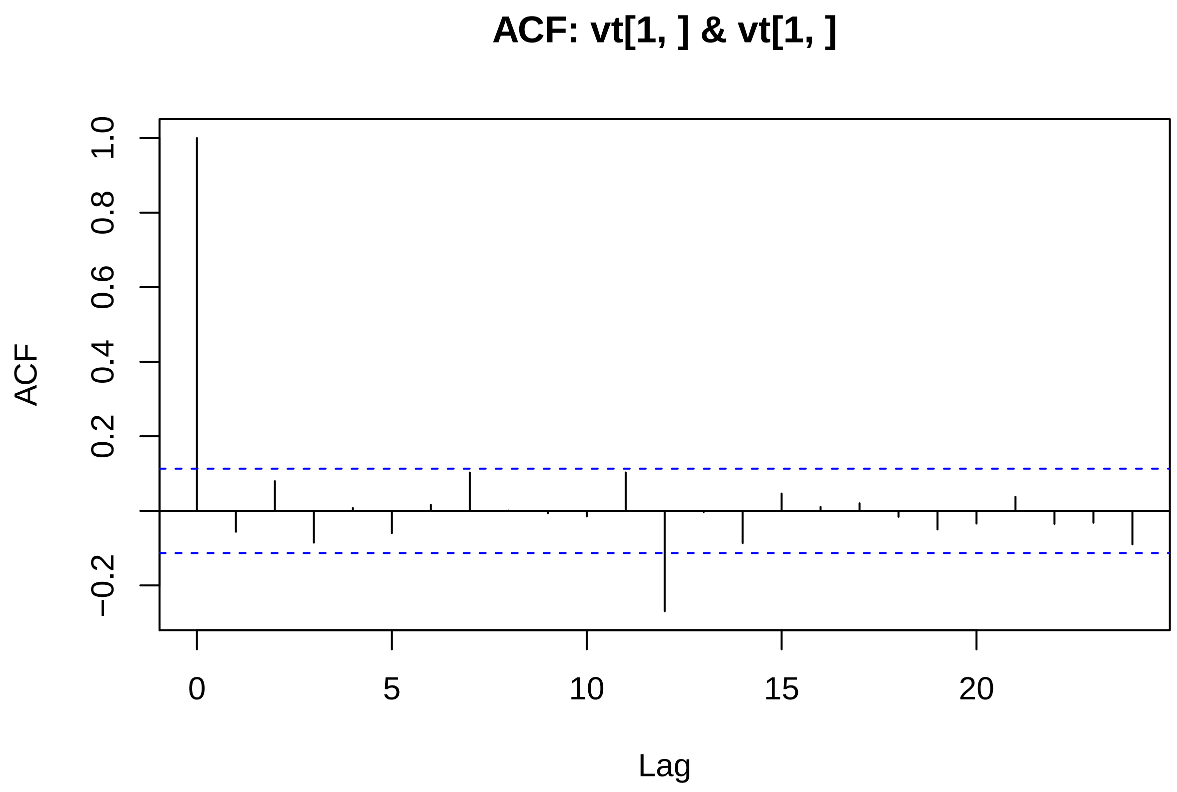


	estimate	lowerCI	upperCI
ar1	-0.0018261	-0.0250908	0.0214386
d	1.0164474	0.9910613	1.0418334
ma1	1.1012450	1.0988962	1.1035939
sigma	0.1807183	0.1407781	0.2206586

In the EU, the estimation of the fractional integration order d is quite high > 0.9, thus signaling that inflation is a persistent process characterized by relatively long swings away from the mean. In addition, these results indicate that inflation is a variance-covariance non-stationary, yet still mean-reverting process. This is consistent with the results we found in the paper.

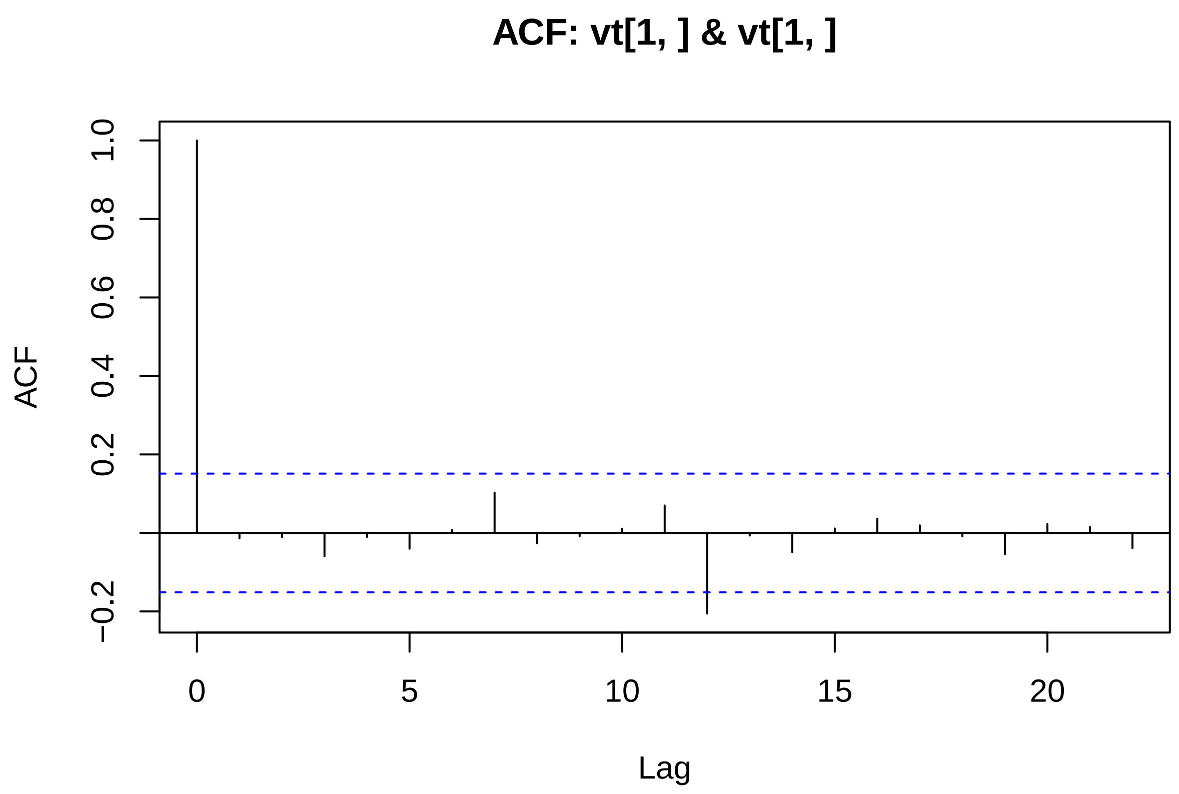
Germany

```
## [1] "Table 2.1: Germany 1993-2017"
```



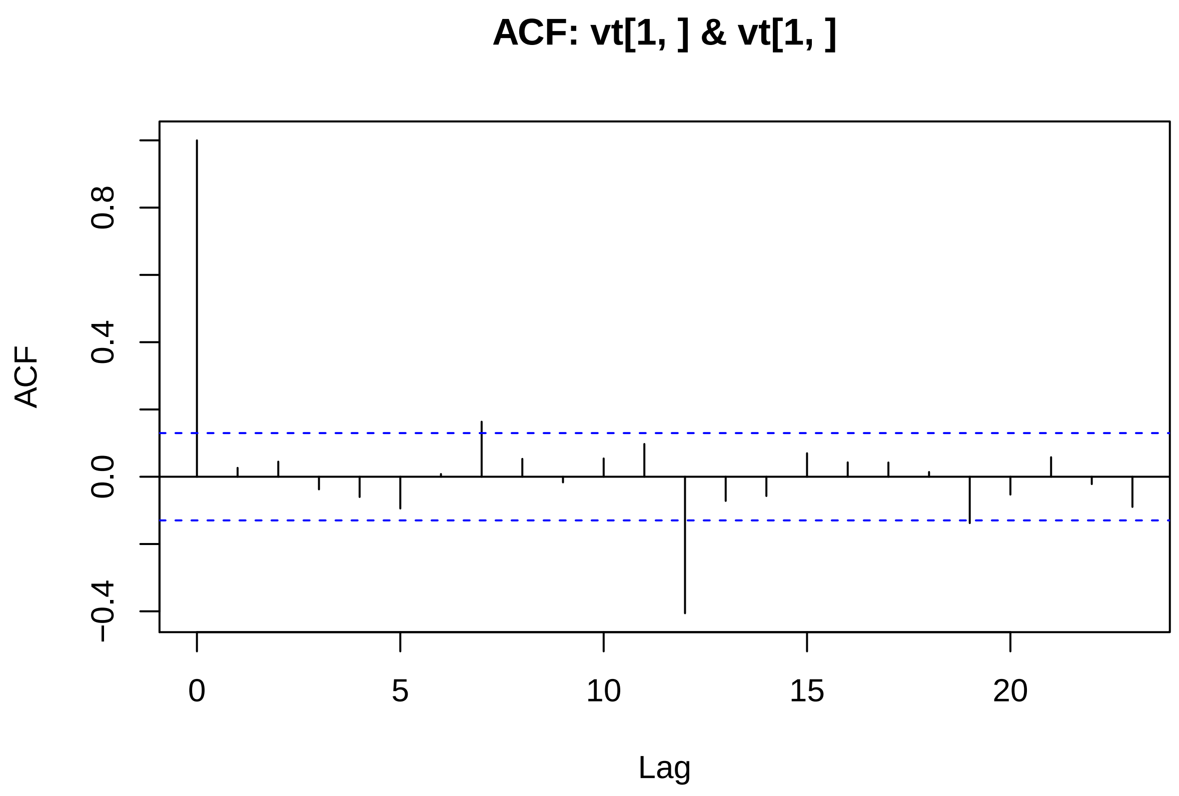
	estimate	lowerCI	upperCI
ar1	-0.0045566	-0.0166401	0.0075269
d	0.9788573	0.9663500	0.9913646
ma1	1.0349997	NaN	NaN
sigma	0.2746451	0.2597008	0.2895895

```
## [1] "Table 2.2: Germany 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	-0.0066673	-0.0159546	0.0026201
d	0.9721804	0.9623149	0.9820459
ma1	1.0612033	NaN	NaN
sigma	0.2562976	0.2306034	0.2819917

```
## [1] "Table 2.3: Germany 1999-2017"
```

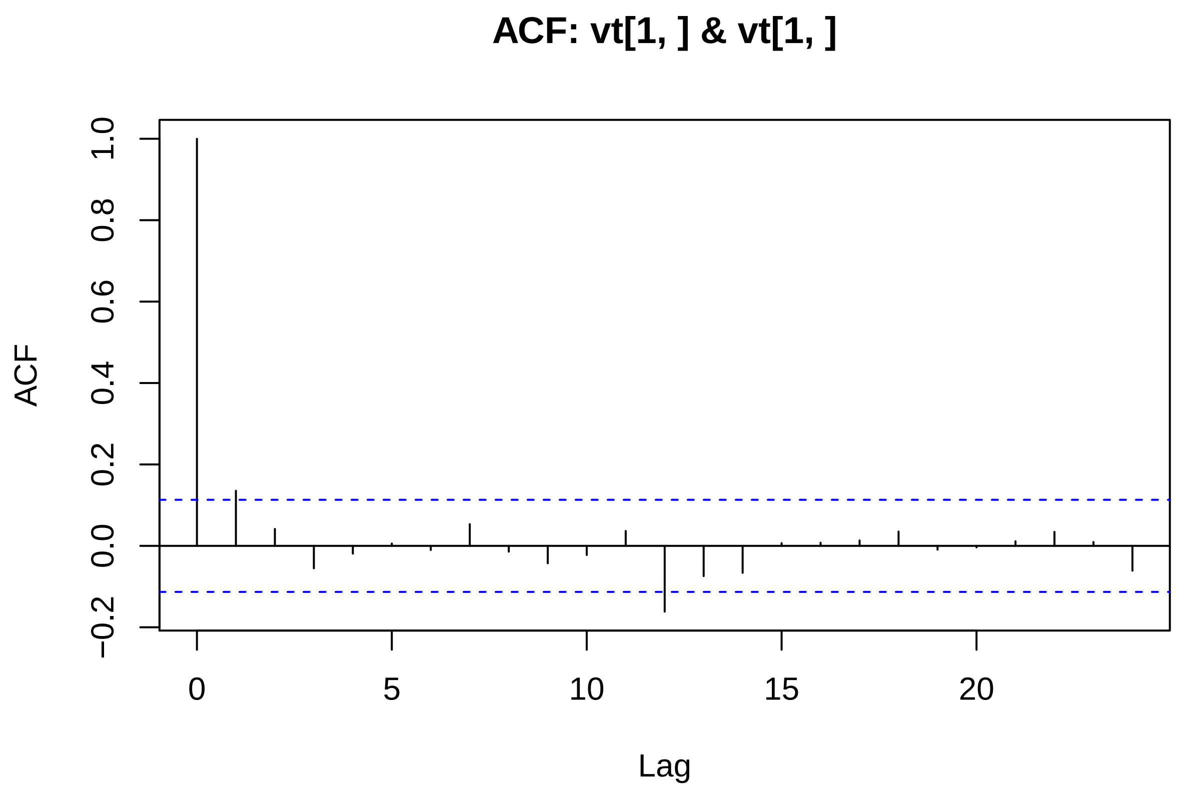


	estimate	lowerCI	upperCI
ar1	0.2858208	0.0073181	0.5643235
d	0.6919728	0.4186931	0.9652526
ma1	0.5685674	0.2589182	0.8782167
sigma	0.2939628	0.2669255	0.3210000

With Germany, the general level of d is lower than in the overall EU, signifying a stickier inflation than in the rest of the Union. Also, we found that by restricting the timeseries only to the period of the Monetary Union (1999-2017), the value of d decreases, indicating that the ECB policy is significantly stricter than the policy which was implemented by the Bundesbank. However, this contrasts with the results found by Andersson and Li, who found out the opposite result.

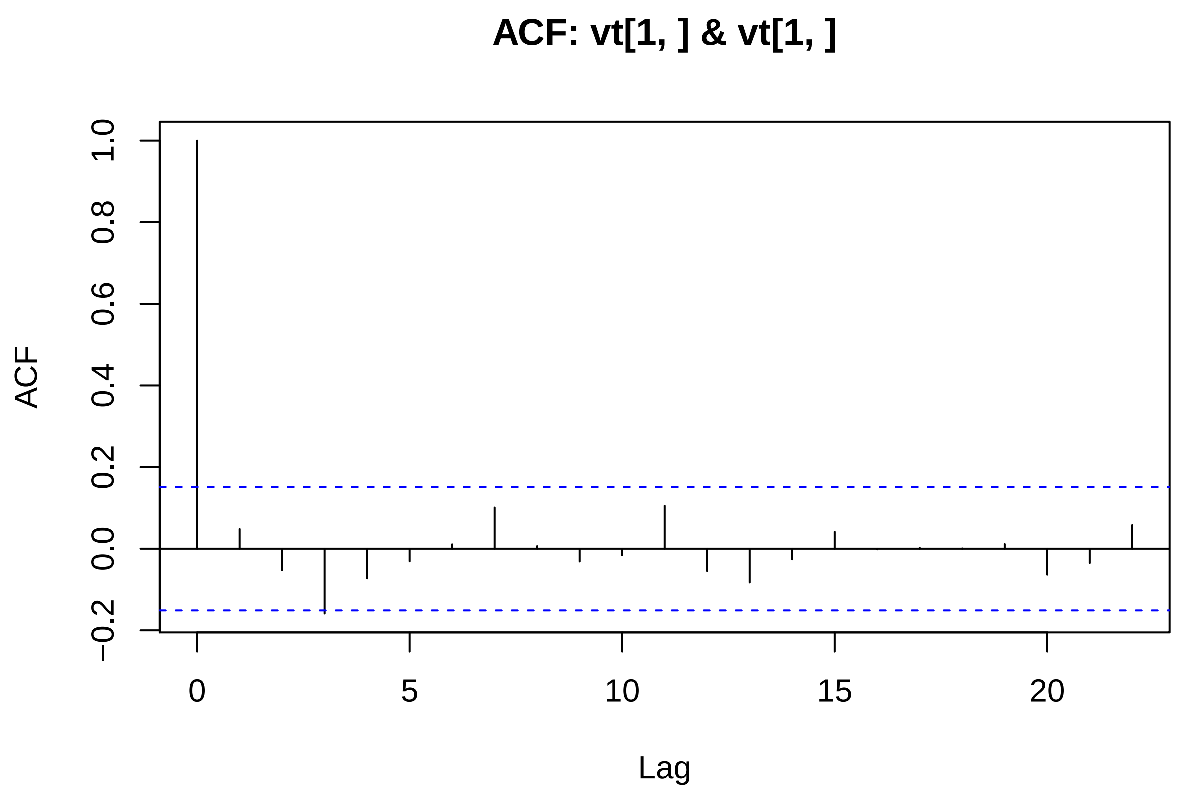
UK

```
## [1] "Table 3.1: UK 1993-2017"
```



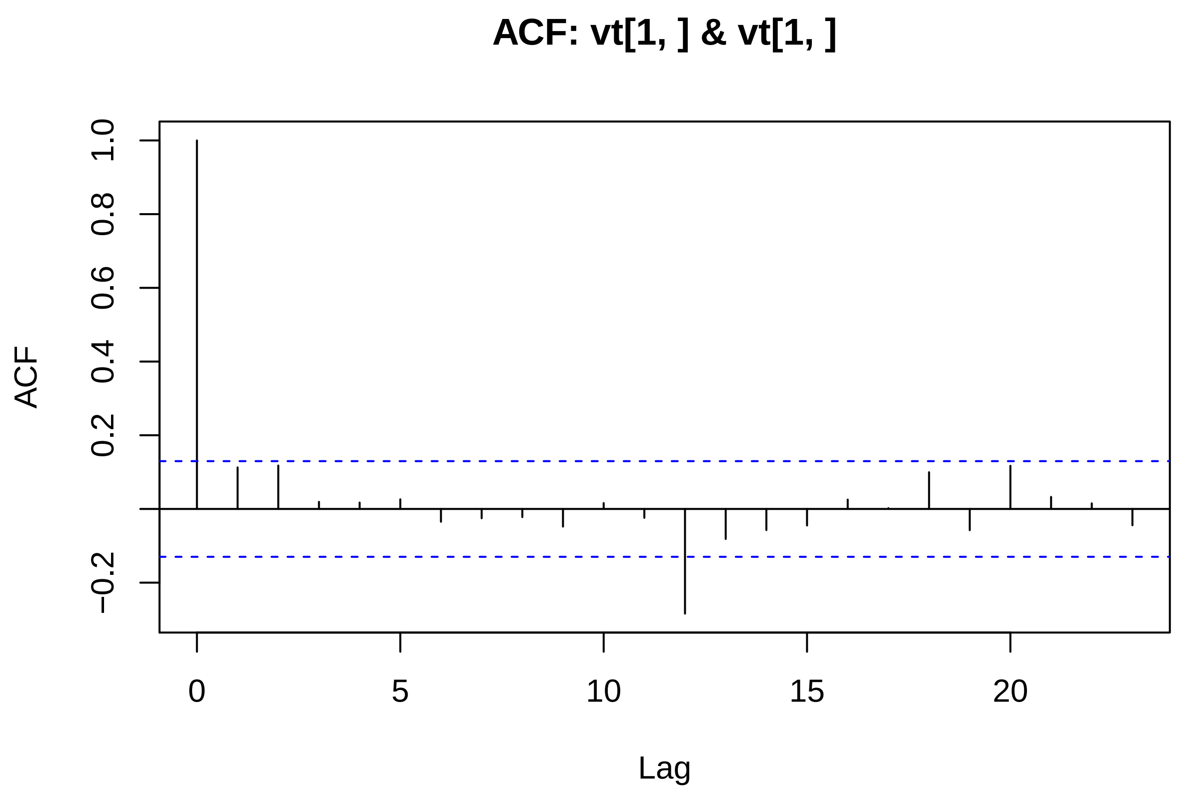
	estimate	lowerCI	upperCI
ar1	-0.0196829	-0.0274070	-0.0119588
d	0.9891613	0.9811929	0.9971298
ma1	1.0307199	NaN	NaN
sigma	0.2253546	0.2155173	0.2351919

```
## [1] "Table 3.2: UK 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	-0.0080975	-0.0215943	0.0053992
d	1.0024546	0.9884682	1.0164410
ma1	1.0393868	1.0366518	1.0421218
sigma	0.1806803	0.1609685	0.2003921

```
## [1] "Table 3.3: UK 1999-2017"
```



	estimate	lowerCI	upperCI
ar1	0.0316853	-0.0153867	0.0787573
d	0.9606100	0.9134295	1.0077905
ma1	0.9943890	0.9772156	1.0115625
sigma	0.2224719	0.2020112	0.2429325

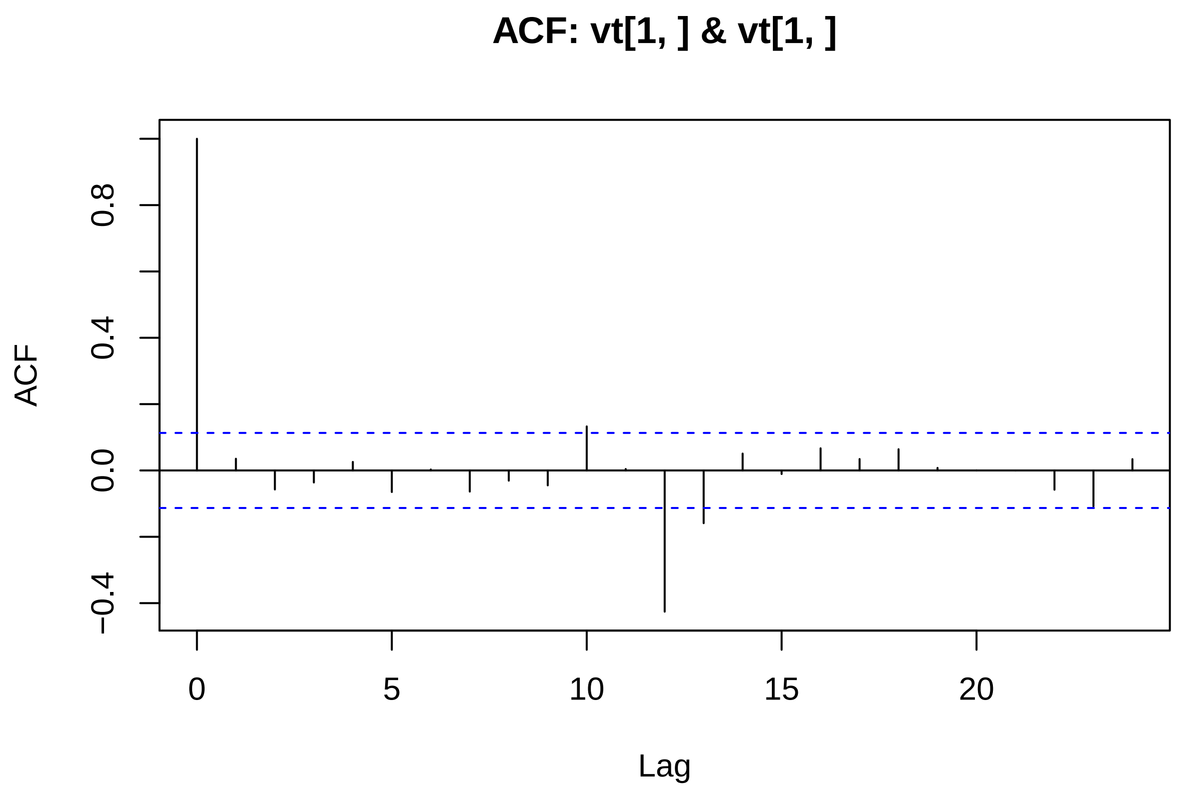
The United Kingdom's integration order of more than 1 suggests that British inflation is a non-stationary process, i.e., the Bank of England has not stabilized inflation around a stationary mean. This result is possibly explained by 1 of 3 factors:

1. the swings away from the mean are very long, and the sample period too short to capture the mean-reversion of the inflation series;
2. the financial crisis in 2008-09 caused the bank to deviate from its inflation target to focusing on stabilizing the economy;
3. there is a break in the mean, which may bias the estimate of the fractional integration order.

Following the crisis, inflation fluctuated substantially, falling to 1% in 2009 before again reaching 5% in 2012, then falling to 0% in 2016 before increasing to 3% in 2017. The financial crisis has clearly affected inflation, indicating a change of focus from central banks. However, average United Kingdom inflation is stable over time, which makes the third explanation less likely.

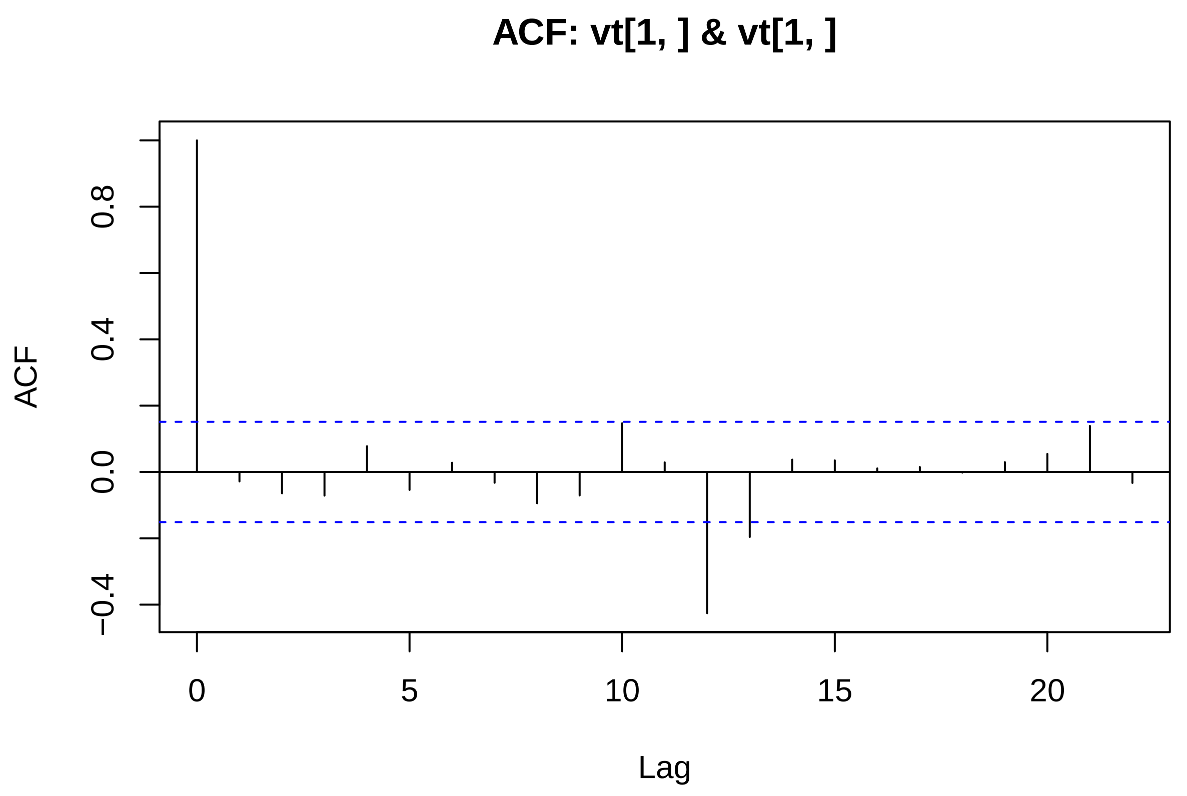
Canada

```
## [1] "Table 2.1: Canada 1993-2017"
```



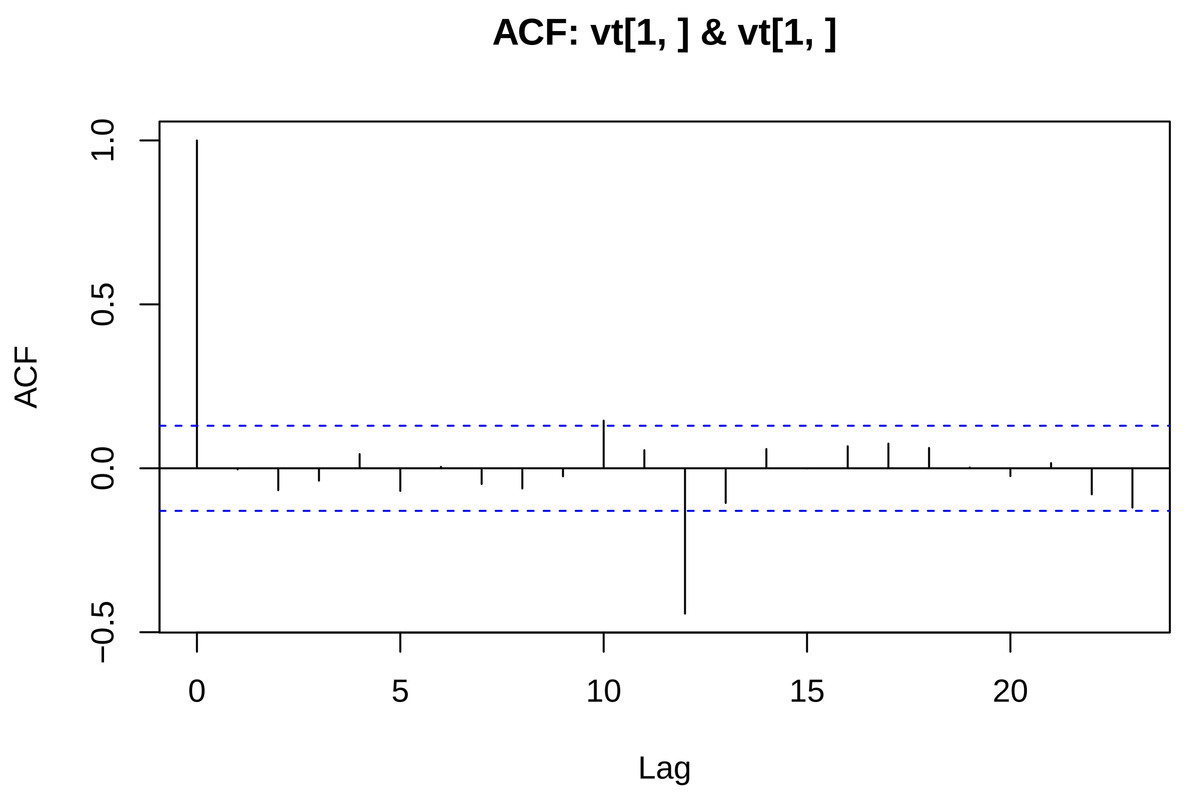
	estimate	lowerCI	upperCI
ar1	0.0929073	0.0104680	0.1753467
d	0.8611414	0.7786385	0.9436443
ma1	0.9757531	0.9399425	1.0115637
sigma	0.4122219	0.3791796	0.4452641

```
## [1] "Table 2.2: Canada 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	0.3628315	-0.0183419	0.7440049
d	0.5938094	0.2161094	0.9715094
ma1	0.7720086	0.4595661	1.0844511
sigma	0.4251940	0.3795889	0.4707992

```
## [1] "Table 2.3: Canada 1999-2017"
```

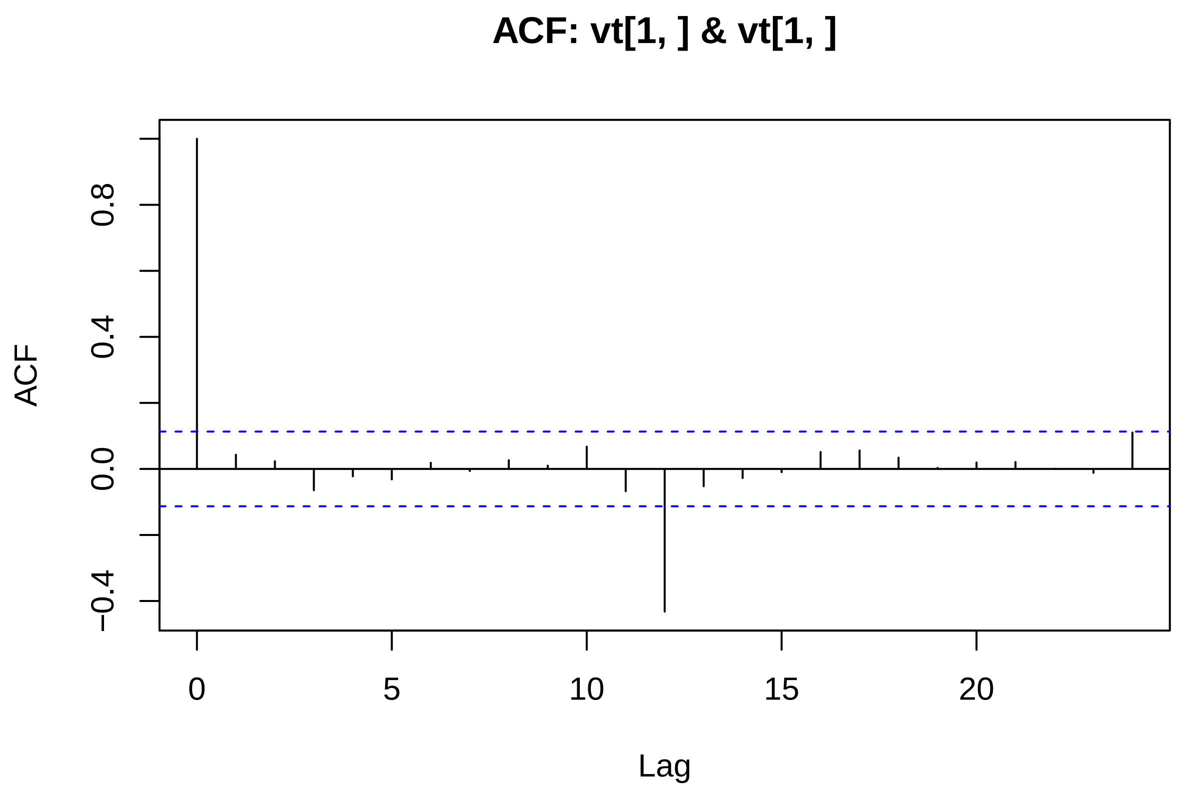


	estimate	lowerCI	upperCI
ar1	0.1359757	0.0405633	0.2313882
d	0.8209243	0.7253534	0.9164952
ma1	0.9708596	0.9336341	1.0080851
sigma	0.4368849	0.3967169	0.4770529

Canadian results indicate an high level of d both in the period 1993-2017 and 1999-2017, but (contrary to the results of the paper) d is lower in 1993-2006. We could hypothesize that this is due to a stricter inflation policy being implemented before the crisis, which was then softened to include other economic parameters in the CB response.

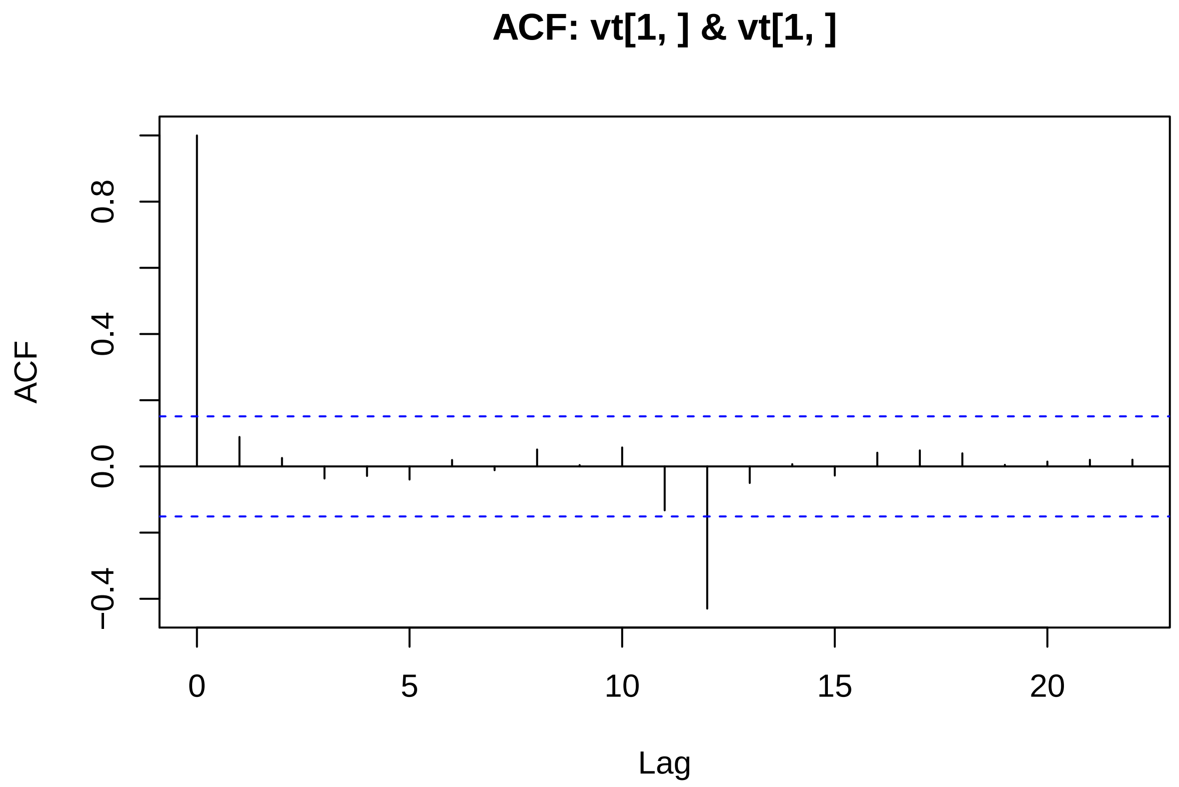
Sweden

```
## [1] "Table 2.1: Sweden 1993-2017"
```



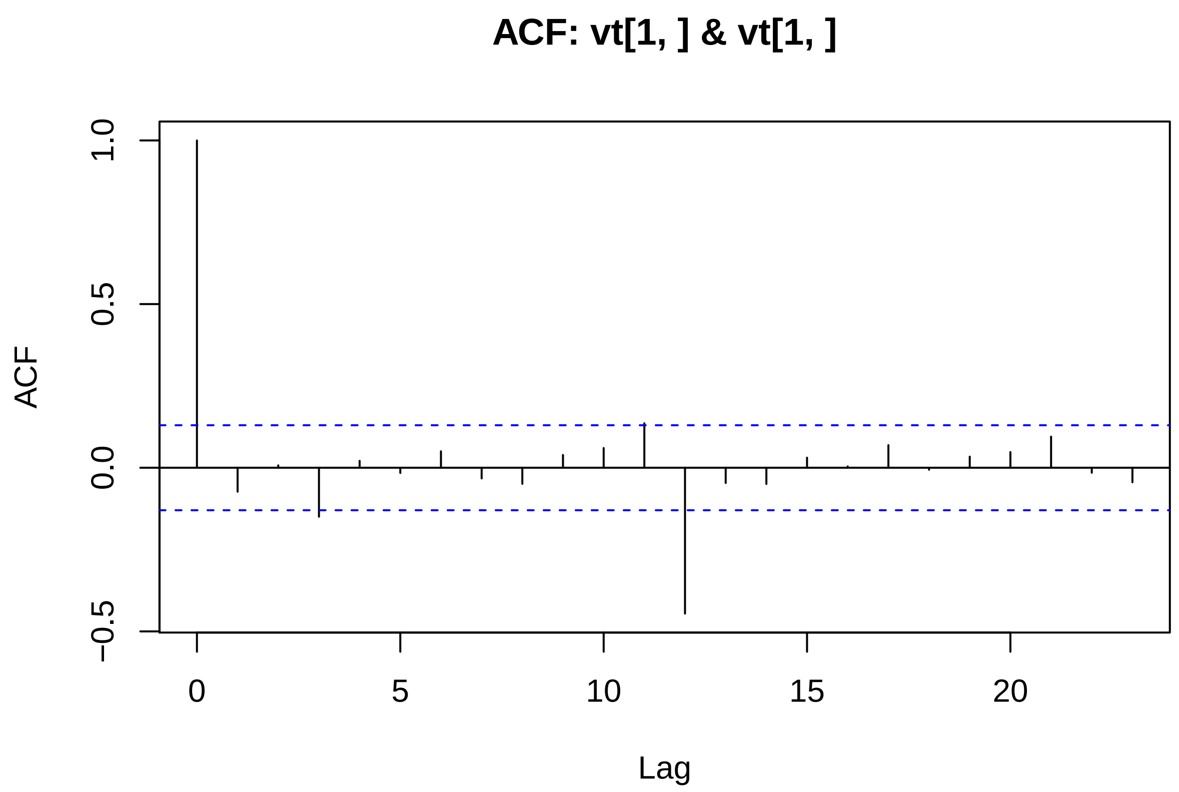
	estimate	lowerCI	upperCI
ar1	0.0204032	-0.0321244	0.0729307
d	0.9354845	0.8829160	0.9880529
ma1	0.9917332	0.9654208	1.0180456
sigma	0.3266496	0.3004681	0.3528310

```
## [1] "Table 2.2: Sweden 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	0.0184617	-0.0432887	0.0802122
d	0.9311269	0.8693790	0.9928747
ma1	0.9882917	0.9562743	1.0203091
sigma	0.3514912	0.3137946	0.3891877

```
## [1] "Table 2.3: Sweden 1999-2017"
```

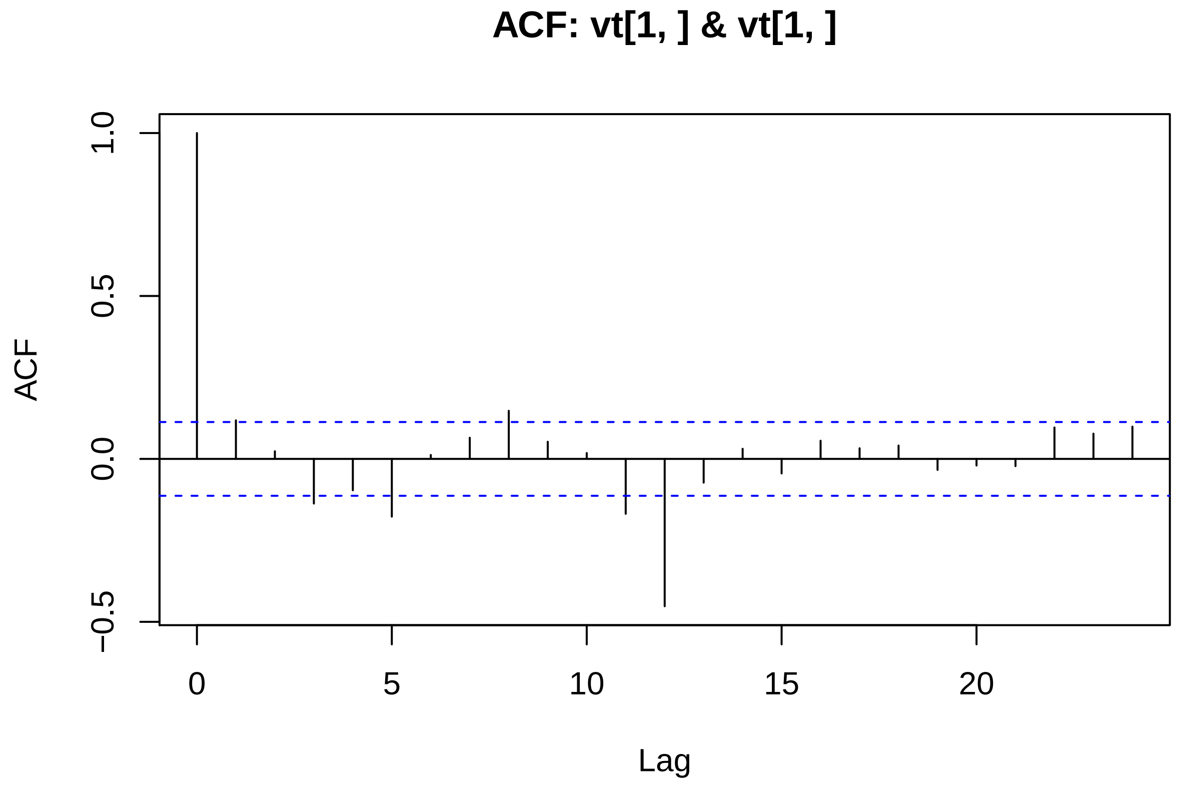


	estimate	lowerCI	upperCI
ar1	0.0542374	-0.0133878	0.1218626
d	0.9138320	0.8461413	0.9815227
ma1	0.9866394	0.9569576	1.0163213
sigma	0.3034719	0.2756149	0.3313288

Sweden presents controversial results. The issue here is that looking at our values for d it would appear that the Sveriges Riksbank has a relaxed approach on inflation, consistent across time. On the other hand, the results from the paper clearly indicates that Swedish policy on inflation targeting was stricter in 1993-2017 and 1993-2006, while it appears more flexible after 1999. Maybe this problem is linked to the different procedure applied in the paper to get the parameter d.

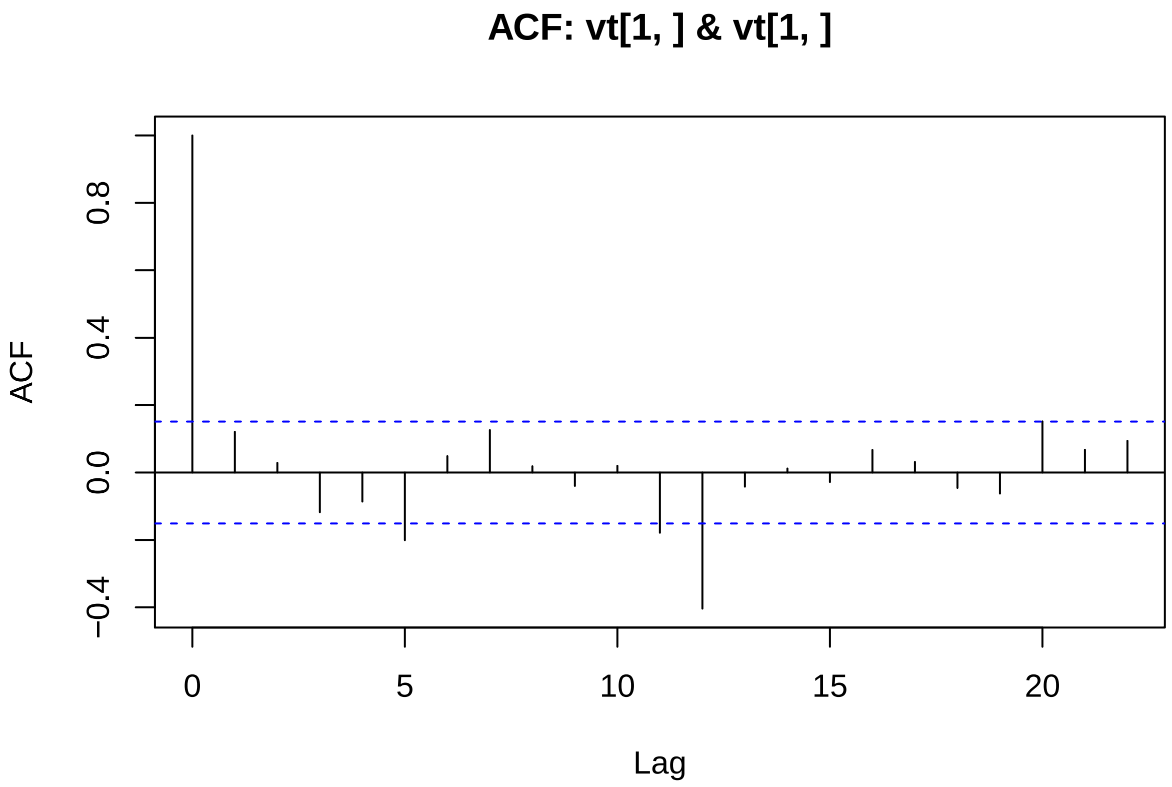
Norway

```
## [1] "Table 2.1: Norway 1993-2017"
```



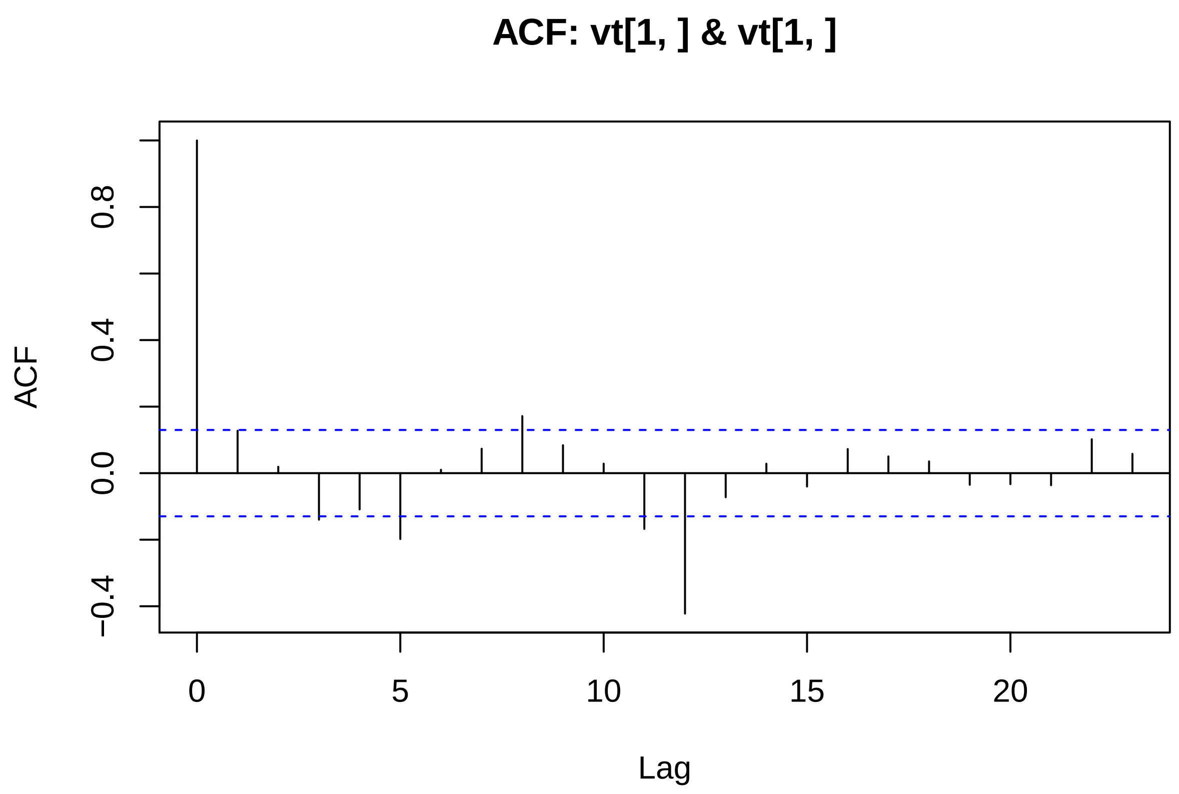
	estimate	lowerCI	upperCI
ar1	0.1414223	0.0242655	0.2585792
d	0.8138532	0.6970516	0.9306549
ma1	0.9481263	0.8842862	1.0119664
sigma	0.4615872	0.4245914	0.4985830

```
## [1] "Table 2.2: Norway 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	0.0943579	-0.0229592	0.2116750
d	0.8636945	0.7462946	0.9810943
ma1	0.9651802	0.9079644	1.0223959
sigma	0.4413327	0.3940004	0.4886650

```
## [1] "Table 2.3: Norway 1999-2017"
```

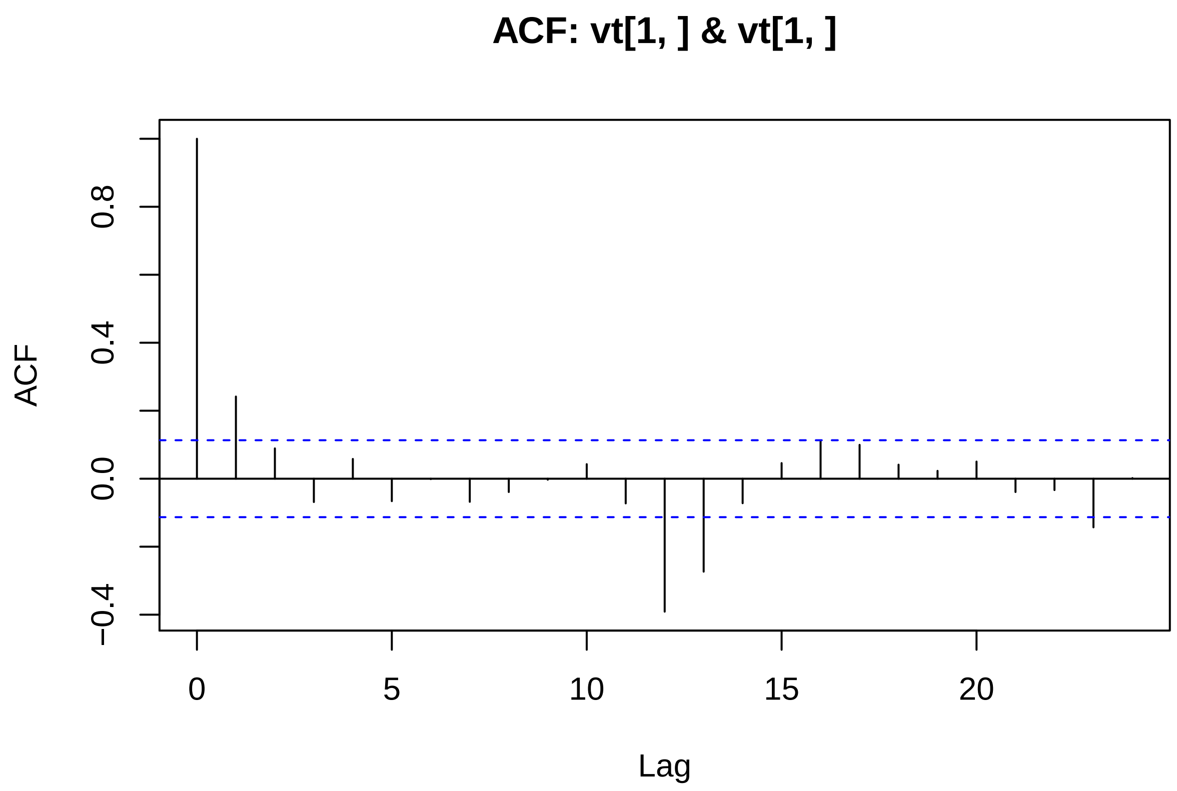


	estimate	lowerCI	upperCI
ar1	0.1309055	-0.0201412	0.2819521
d	0.8182120	0.6676869	0.9687372
ma1	0.9535101	0.8702048	1.0368153
sigma	0.5047029	0.4582771	0.5511287

Results for Norway indicate an high degree of flexibility across all time frames from the Norges Bank. In fact, the Norges Bank official policy is to keep inflation "approximately" at 2.5%, indicating a flexible approach to inflation. This finding is consistent with those from Andersson and Li.

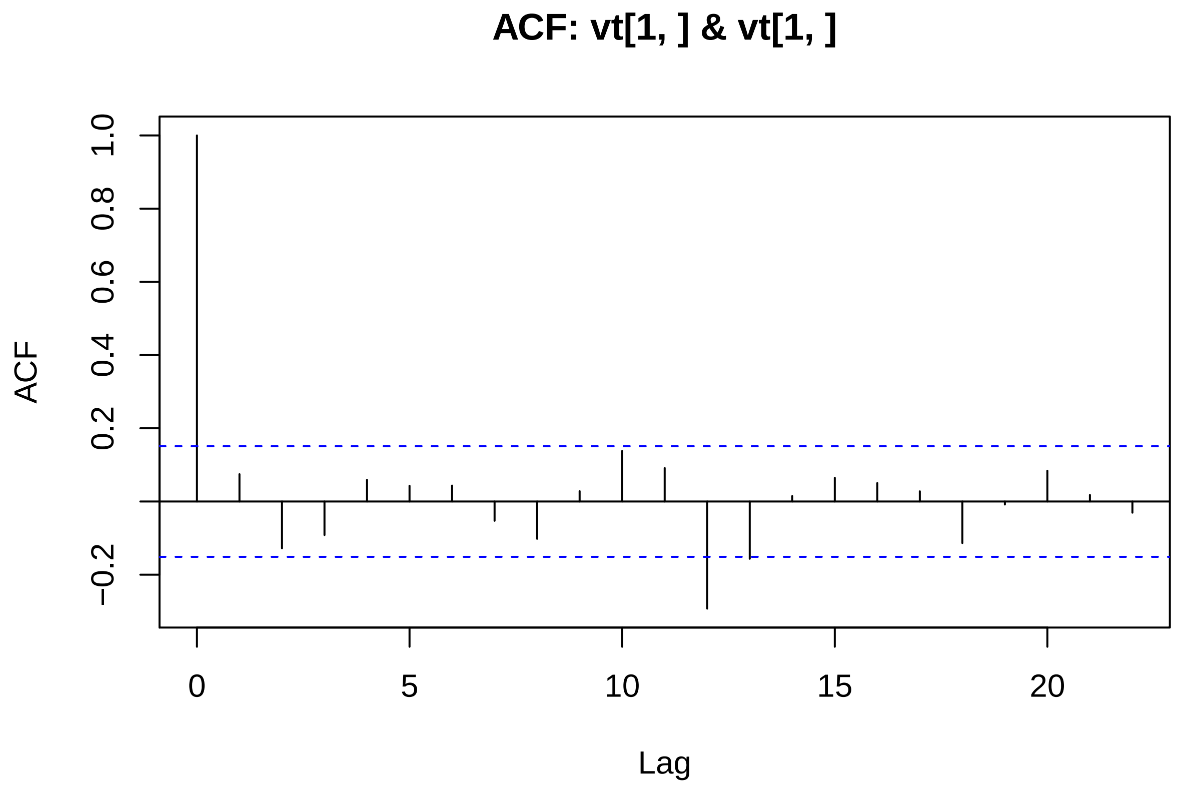
United States

```
## [1] "Table 2.1: United States 1993-2017"
```



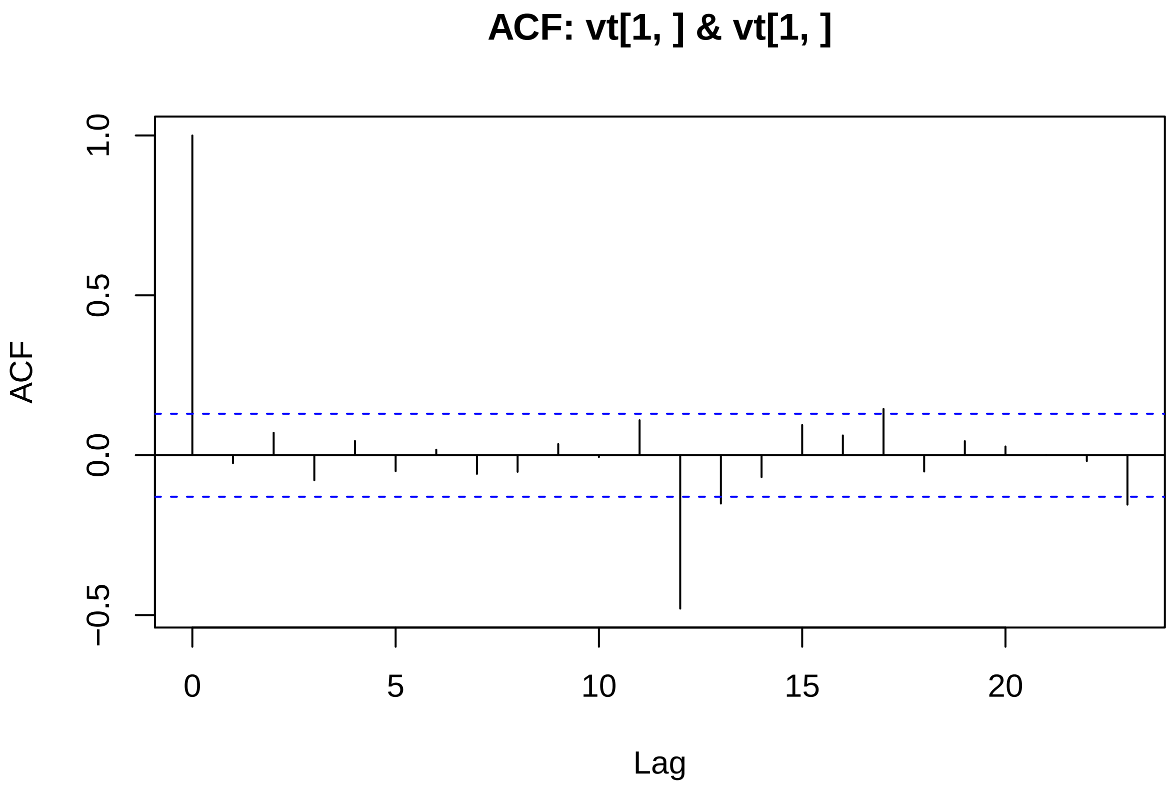
	estimate	lowerCI	upperCI
ar1	0.0467645	-0.0392655	0.1327946
d	0.9307252	0.8446434	1.0168069
ma1	0.9750798	0.9324983	1.0176612
sigma	0.2866401	0.2636663	0.3096139

```
## [1] "Table 2.2: United States 1993-2006"
```



	estimate	lowerCI	upperCI
ar1	0.0110205	0.0000216	0.0220194
d	1.0173267	1.0055623	1.0290911
ma1	1.0695476	NaN	NaN
sigma	0.2238076	0.2034313	0.2441840

```
## [1] "Table 2.3: United States 1999-2017"
```



	estimate	lowerCI	upperCI
ar1	1.0735099	0.7840712	1.3629485
d	-0.0923620	-0.3805188	0.1957948
ma1	0.2953611	0.0465175	0.5442048
sigma	0.3029743	0.2751104	0.3308381

Results from the US are not satisfactory, since the first two time frames are consistent with results found by Andersson and Li, and indicate an high degree of flexibility of the Federal Reserve policy on inflation targeting, but the third time frame deals meaningless results. In fact, as we can see in table 3, the third time frame produces a value of d equal to -0.09, which clearly makes no sense. Probably the cause can be traced back to the different procedure we used, but we were unable to track down the problem and resolve it.