```
library(INLA)
library(FGN)
# An ARIMA simulation
ts.simar1 <- arima.sim(list(ar = 0.7), n = 500)
par(mfrow=c(2,2))
ts.plot(ts.simar1,main="AR(1) simulation")
acf(ts.simar1,main="ACF of AR(1)")

# An ARIMA simulation
ts.simma <- arima.sim(list(ma = 0.7), n = 500)

ts.plot(ts.simma,main="MA(1) simulation")
acf(ts.simma,lag.max = n,main="ACF of MA(1)")
#USE ARMAacf to compare.


# An ARIMA simulation
ts.simar1 <- arima.sim(list(ar = 0.7), n = 500)
par(mfrow=c(2,1))
ts.plot(ts.simar1,main="AR(1) simulation")
pacf(ts.simar1,main="PACF of AR(1)")

# An ARIMA simulation
ts.simma <- arima.sim(list(ma = 0.7), n = 500)

ts.plot(ts.simma,main="MA(1) simulation")
pacf(ts.simma,main="ACF of MA(1)")

library(FGN)
#simulate FGN and compare theoretical and sample autocovariances
H1<-0.6
H2<-0.8
H3<-0.9

n<-5000
#data
z<-SimulateFGN(n, H1)
z2<-SimulateFGN(n, H2)
z3<-SimulateFGN(n, H3)
data1<-read.table("brown72.txt",header = F,sep = ",",fill = T)
brown<-data1[,1]
###########
a<-acvfFGN(0.8,100)
b<-acf(z2,main="H=0.9",lag.max = 100)
xrange<-seq(0,100)
dim(a)
dim(xrange)
length(b)
# HurstK(z3)
# FitFGN(X)
```

```
# acfvFGN(X)
#par(mfrow=c(2,1))


n=500
H=0.8
y=scale(SimulateFGN(n,H))

time=1:n
data=list(y=y,time=time)

formula =y~-1+f(time,model="fgn",order=4)
result =inla(formula,data=data,verbose=F,control.family
             = list(hyper = list(prec = list(initial = 12, fixed=TRUE))))

result$summary.hyperpar[2,1]



plot(xrange,)
lines(a,col=2)

#xrange<-seq(0,100)
#plot(xrange,acf(z3,main="H=0.9",lag.max = 100))
#lines(acvfFGN(0.9,100),col=2)
# plot(z2,type="l",main="H=0.78")
# pacf(z2,main="H=0.78")
# plot(z3,type="l",main="H=0.9")
# acf(z3,main="H=0.9")

################
#Rescaled Range#
################
rescaledRange<-function(x){
  n<-length(x)
  m<-mean(x)
  Yt<-cumsum(x-m)
  Rt<-rep(0,n)
  for(i in 1:n) {Rt[i]<-max(Yt[1:i])-min(Yt[1:i])}
  n=length(x)
  x.mean=cumsum(x)/(1:n)
  s=rep(0,n)
  for (i in 1:n) {s[i]=sqrt(1/i*sum((x[1:i]-rep(x.mean[i],i))^2)) }
  RS.statistic<-Rt/s
  RS.statistic<-RS.statistic[-1]
  return(RS.statistic)
}
################
#hursttest
##############
```

```
hurstest<-function(X){
  n=length(X)
  n1=trunc(n/2)
  n2=trunc(n1/2)
  n3=trunc(n2/2)
  n4=trunc(n3/2)
  RSave0=mean(rescaledRange(X))
  RSave1=mean(rescaledRange(X[1:n1]))
  RSave2=mean(rescaledRange(X[1:n2]))
  RSave3=mean(rescaledRange(X[1:n3]))
  RSave4=mean(rescaledRange(X[1:n4]))
  region<-c(n,n1,n2,n3)
  RSave<-c(RSave0,RSave1,RSave2,RSave3)
  log.Reg<-log2(region)
  log.RSave<-log2(RSave)
  reghurst<-lm(log.RSave~log.Reg)
  return(reghurst$coefficients[2])
}
###############
#ILNA ESTimateof H
#############
in.hurst<-function(x){
  n=length(x)
  time=1:n
  data=list(y=x,time=time)
  formula =y~-1+f(time,model="fgn",order=4)
  result =inla(formula,data=data,verbose=F,control.family
             = list(hyper = list(prec = list(initial = 5, fixed=TRUE))))
  return(result$summary.hyperpar[2,1])
}
########
#simulation
########
in.hurst(y)
n.sims<-10
H=matrix(0,n.sims,4)
# Start the clock!
ptm <- proc.time()
for (i in 1:n.sims){
  H[i,1]<-hurstest(y)
  H[i,2]<-HurstK(y)
  H[i,3]<-FitFGN(y)$H
  H[i,4]<-in.hurst(y)
}
# Stop the clock
proc.time() - ptm

Hurst.mean<-apply(H,2,mean)
Hurst.sd<-apply(H,2,sd)
print(Hurst.mean)
```

```
print(Hurst.sd)
```