Package ‘FGN’

October 18, 2007

Version 1.0

Date 2007-09-17

Title Fractional Gaussian Noise Model Fitting

Author A.I. McLeod

Maintainer A.I. McLeod <aimcleod@uwo.ca>

Depends R (>= 2.1.0), ltsa

Description MLE for H parameter in FGN; MLE for regression with FGN error; simulation of FGN

License GPL (version 2 or later)

URL http://www.stats.uwo.ca/faculty/aim

R topics documented:

- Boot.FitFGN
- Boot
- FGNacf
- FGNLL
- FitFGN
- FitRegressionFGN
- GetFitFGN
- HurstK
- NileFlowCMS
- NileMin
- SimulateFGN
- coef.FitFGN
- plot.FitFGN
- predict.FitFGN
- print.FitFGN
- residuals.FitFGN
- summary.FitFGN

Index 20
Simulate Fitted FGN Model

Description
Simulate a realization from a fitted AR model. This is useful in the parametric bootstrap. Generic function for "Boot" method.

Usage
Boot.FitFGN(obj, R = 1, ...)

Arguments
obj the output from FitAR
R number of bootstrap replications
... optional arguments

Details
The method of Davies and Harte (1987) is used if it is applicable, otherwise the Durbin-Levinson recursion is used.

Value
If R=1, a simulated time series with the same length as the original fitted time series is produced. Otherwise if R>1, a matrix with R columns and number of rows equal to the length of the series containing R replications of the bootstrap.

Author(s)
A.I. McLeod

References

See Also
SimulateFGN, DHSimulate, DLSimulate

Examples
#Example 1
#Fit a FGN model and determine the bootstrap sd of H
#Measure cpu time. With R=250, it takes about 23 sec
#on 3.6 GHz Pentium IV.
data(NileMin)
outNileMin<-FitFGN(NileMin)
start<-proc.time()[1]
R<-25
Hs <- numeric(R)
Z <- Boot(outNileMin, R=R)
for (i in 1:R)
  Hs[i] <- GetFitFGN(Z[, i])$H
BootSD <- sd(Hs)  # this is the bootstrap sd
end <- proc.time()[1]
totTim <- end - start

---

**Boot**

**Generic Bootstrap Function**

**Description**

Generic function to bootstrap a fitted model.

**Usage**

Boot(obj, R=1, ...)

**Arguments**

- **obj**  
  fitted object

- **R**  
  number of bootstrap replicates

- **...**  
  optional arguments

**Value**

Parametric bootstrap simulation

**Author(s)**

A.I. McLeod

**References**


**See Also**

Boot.FitFGN

**Examples**

data(NileMin)
out <- FitFGN(NileMin)
Boot(out, R=3)
FGNAcf

Auxocorrelation of FGN

Description
The FGN time series is an example of a time series exhibiting long-range dependence and charac-
terized by the fact that its autocorrelation function exhibits hyperbolic decay rather than exponential
decay found in stationary ARMA time series. The FGN and other alternatives are discussed in Hipel
and McLeod (2005).

Usage
FGNAcf(k, H)

Arguments
k lag or lags - may be vector
H Hurst parameter

Value
value of the autocorrelation at lag(s) k

Note
The parameter H should be in (0,1). An error message is given if it is not.

Author(s)
A.I. McLeod

References
stats.uwo.ca/faculty/aim/1994Book/.

Journal of Statistical Software.

See Also
FGNLL, acf

Examples
#compute the acf at lags 0,1,...,10 when H=0.7
FGNAcf(0:10, 0.7)
FGNLL

Concentrated Loglikelihood Function for H

Description

The concentrated loglikelihood, that is, the loglikelihood function maximized over the innovation variance parameter, is computed.

Usage

FGNLL(H, z)

Arguments

H
parameter

z
data vector, assumed to be mean corrected

Value

the value of the loglikelihood

Author(s)

A.I. McLeod

References


See Also

FitFGN, DLLoglikelihood

Examples

#compute loglikelihood for NileFlowCMS with H=0.9
data(NileFlowCMS)
z<-NileFlowCMS
z<-z-mean(z)
FGNLL(0.9, z)

#simulate Gaussian white noise and tabulate the loglikelihood for H=0.40, 0.45, 0.50, 0.55, 0.60
set.seed(4321)
h<-c(0.40, 0.45, 0.50, 0.55, 0.60)
z<-rnorm(500, 100, 50)
z<-z-mean(z)
LL<-numeric(length(h))
for (i in 1:length(h))
    LL[i]<-FGNLL(h[i],z)
matrix(c(h,LL),ncol=2)
**FitFGN**

**MLE estimation for FGN**

**Description**

Exact MLE estimation for FGN

**Usage**

```r
FitFGN(z, demean = TRUE, MeanMLEQ = FALSE, lag.max = "default")
```

**Arguments**

- `z` : time series, vector or ts object.
- `demean` : if True, subtract mean. Otherwise assume it is zero.
- `MeanMLEQ` : if True, an iterative algorithm is used for exact simultaneous MLE estimation of the mean and other parameters.
- `lag.max` : the residual autocorrelations are tabulated for lags 1, ..., lag.max. Also lag.max is used for the Ljung-Box portmanteau test.

**Details**

The exact loglikelihood function is maximized numerically using `optimize`. The standard error for the H parameter is estimated (McLeod, Yu and Krougly, 2007).

**Value**

A list with class name "FitAR" and components:

- `loglikelihood` : value of the loglikelihood
- `H` : estimate of H parameter
- `SEH` : SE of H estimate
- `sigsqHat` : innovation variance estimate
- `muHat` : estimate of the mean
- `SEmu` : SE of mean
- `Rsq` : R-squared, coefficient of forecastability
- `LjungBox` : table of Ljung-Box portmanteau test statistics
- `res` : normalized residuals, same length as z
- `demean` : TRUE if mean estimated otherwise assumed zero
- `IterationCount` : number of iterations in mean mle estimation
- `MLEMeanQ` : TRUE if mle for mean algorithm used
- `tsp` : tsp(z)
- `call` : result from match.call() showing how the function was called
- `DataTitle` : returns attr(z,"title")
FitRegressionFGN

Author(s)
A.I. McLeod

References

See Also
GetFitFGN, FitRegressionFGN, Boot.FitFGN, coef.FitFGN, plot.FitFGN, print.FitFGN, summary.FitFGN, HurstK

Examples

data(NileMin)
out<-FitFGN(NileMin)
summary(out)
plot(out)
coef(out)

Description
Regression with FGN Errors

Usage
FitRegressionFGN(X, y)

Arguments
X design matrix, must include column of 1’s if constant term is present
y the response variable, a time series

Details
An iterative algorithm is used to compute the exact MLE.

Value
a list with 3 elements:
Loglikelihood
value of the maximized loglikelihood
H MLE for H
alpha MLE for regression coefficients corresponding to columns of X

Note
It is assumed that X is not collinear.
GetFitFGN

Description

Exact maximum likelihood estimation of the parameter H in fractional Gaussian noise (FGN). This is a utility function used by \texttt{FitFGN} but it is also useful in simulation experiments since it is faster than using \texttt{FitFGN}. See example below.

Usage

\texttt{GetFitFGN(z, MeanZeroQ = FALSE)}

Arguments

- \texttt{z} \quad \text{time series data vector}
- \texttt{MeanZeroQ} \quad \text{optional argument, default is MeanZeroQ=FALSE. Set to TRUE if the mean is known to be zero}

Details

The function \texttt{optimize} is used. It is very rare but it has been observed that \texttt{optimize} can incorrectly choose an endpoint. If this happens a warning is given and \texttt{optim} is used.

Examples

```r
#simulate FGN with mean zero and H=0.2 and fit exact mle for H and mean
H<-0.2
z<-SimulateFGN(512, H)
mean(z)
X<-matrix(rep(1,length(z)), ncol=1)
an<-FitRegressionFGN(X,z)
an

#fit a step intervention model to the Nile annual riverflow data
data(NileFlowCMS)
n<-length(NileFlowCMS)
X<-matrix(c(rep(1,n),rep(0,32),rep(1,n-32)),ncol=2)
an<-FitRegressionFGN(X,NileFlowCMS)
an
```
**GetFitFGN**

**Value**

a list with two elements:

- **Loglikelihood**
  value of the maximized loglikelihood

- **H**
  MLE for H

**Author(s)**

A.I. McLeod

**References**


**See Also**

optimize, optim, Boot.FitFGN, FitFGN, FitRegressionFGN

**Examples**

```r
#Example 1
#fit Gaussian White Noise, H=0.5
z<-rnorm(500, 100, 10)
GetFitFGN(z)

#Example 2
#estimate H for NileMin series
data(NileMin)
GetFitFGN(NileMin)

#Example 3
#Timing comparison for GetFitFGN and FitFGN
ns<-c(500,1000) #may extend this to other n's
H<-0.8
nR<-10
tim1<-tim2<-numeric(length(ns))
for (i in 1:length(ns)){
  n <- ns[i]
t1<-t2<-0
  sl1<-proc.time()[1]
  for (iR in 1:nR){
    z<-SimulateFGN(n, H)
    H1<-GetFitFGN(z)
  }
e1<-proc.time()[1]
t1<-t1+(e1-sl1)
  s2<-proc.time()[1]
  for (iR in 1:nR){
    z<-SimulateFGN(n, H)
    H2<-FitFGN(z)
  }
e2<-proc.time()[1]
t2<-t2+(e2-s2)
tim1[i]<-t1
```
```r
tim2[i]<-t2
}
tb<-matrix(c(tim1,tim2),ncol=2)
dimnames(tb)<-list(ns,c("GetFitFGN","FitFGN"))
```

### HurstK

**Hurst K Coefficient**

**Description**

The Hurst K provides a non-parametric estimate for the Hurst H coefficient

**Usage**

```r
HurstK(z)
```

**Arguments**

- `z`: time series vector

**Details**

There are many alternative non-parametric estimators for H. Some of the popular ones are discussed in Hipel and McLeod (2005).

**Value**

an estimate of H

**Author(s)**

A.I. McLeod

**References**


**See Also**

`FitFGN`

**Examples**

```r
# the Hurst coefficient for NID series is 0.5
z<-rnorm(1000)
HurstK(z)
#Hurst K for Nile Minima
data(NileMin)
HurstK(NileMin)
```

**NileFlowCMS**  
*Annual flow of Nile River at Aswan, 1871-1945*

---

**Description**

This is average annual flow of the Nile River below the Aswan Dam. The units are CMS (cubic meters per second).

**Usage**

```r
data(NileFlowCMS)
```

**Format**

The format is: Time-Series [1:75] from 1870 to 1944: 3958 3370 3485 3438 3702 ...

**Source**


**Examples**

```r
#Plot the time series
data(NileFlowCMS)
ts.plot(NileFlowCMS)

#Hurst K estimate
HurstK(NileFlowCMS)
```

---

**NileMin**  
*Nile Annual Minima, 622 AD to 1284 AD*

---

**Description**

Annual Minimum flow of Nile River. See below for details.

**Usage**

```r
data(NileMin)
```

**Format**

The format is: Time-Series [1:663] from 622 to 1284: 11.57 10.88 11.69 11.69 9.84 ... - attr(*, "title")= "Nile River minima series"
Details

The minimum annual level of the Nile has been recorded over many centuries and was given by Toussoun (1925). The data over the period 622 AD to 1284 AD is considered more homogenous and reliable than the full dataset and has been analyzed by Beran (1994) and Percival and Walden (2000). The full dataset is available StatLib Datasets hipel-mcleod archive – file: Minimum.

Source


References


Examples

```r
# Compute Hurst's K estimate of H
data(NileMin)
HurstK(NileMin)

# Script for comparing FGN/ARMA forecast performance
#
data(NileMin)
outNileMin<-FitFGN(NileMin)
set.seed(12177)
z<-Boot(outNileMin)
n<-length(z)
K<-100 # number of out-of-sample data values
z1<-z[1:(length(z)-K)] # training data
z2<-z[-(1:(length(z)-K))] # testing data
#
# FGN fit to z1 and forecast using z2
maxLead<-3
n1<-length(z1)
outz1<-FitFGN(z1)
H<-outz1$H
mu<-outz1$muHat
rFGN<-var(z1)*FGNacf(0:(n + maxLead -1), H)
F<-TrenchForecast(c(z1,z2), rFGN, mu, n1, maxLead=maxLead)$Forecasts
nF<-nrow(F)
err1<-z2-F[1:nF]
err2<-z2[-1]-F[2,nF-1])
err3<-z2[-c(1,2)]=F[3,-c(nF, (nF-2))]
rmse1<-sqrt(mean(err1^2))
rmse2<-sqrt(mean(err2^2))
rmse3<-sqrt(mean(err3^2))
FGNrmse<-c(rmse1, rmse2, rmse3)
#
# ARMA(p,q) fit to z1 and forecast using z2
p<-2
q<-1
ansz1<-arima(z1, c(p,0,q))
```
phi<-theta<-numeric(0)
if (p>0) phi<-coef(ansz1)[1:p]
if (q>0) theta<-coef(ansz1)[(p+1):(p+q)]
zm<-coef(ansz1)[p+q+1]
sigma2<-ansz1$sigma2
vz<-tacvfARMA(phi=phi, theta=theta, sigma2=sigma2, maxLag=0)
r<-vz*ARMAacf(ar=phi, ma=theta, lag.max=n + maxLead -1)
F<-TrenchForecast(c(z1,z2), r, zm, n1, maxLead=3)$Forecasts
err1<-z2-F[,1][-nF]
err2<-z2[-1]-F[,2][-c(nF,(nF-1))]
err3<-z2[-c(1,2)]-F[,3][-c(nF,(nF-1),(nF-2))]
rmse1<-sqrt(mean(err1^2))
rmse2<-sqrt(mean(err2^2))
rmse3<-sqrt(mean(err3^2))
ARMArmse<-c(rmse1,rmse2,rmse3)

# tabulate result
tb<-matrix(c(FGNrmse,ARMArmse),ncol=2)
dimnames(tb)<-list(c("lead1","lead2","lead3"),c("FGN","ARMA"))

SimulateFGN

Simulates FGN

Description
A fractional Gaussian noise time series is simulated.

Usage
SimulateFGN(n, H)

Arguments
n
depth of time series
H
Hurst coefficient

Details
The FFT is used so it is most efficient if you select n to be a power of 2.

Value
vector of length containing the simulated time series

Author(s)
A.I. McLeod

References
See Also

DLSimulate

Examples

#Example 1
#simulate a process with H=0.2 and plot it
z<-SimulateFGN(100, 0.2)
ts.plot(z)
#
#Example 2
#simulate FGN and compare theoretical and sample autocovariances
H<-0.7
n<-8192
z<-SimulateFGN(n, H)
#autocovariances
sacvf<-acf(z, plot=FALSE,type="covariance")$acf
tacf<-FGNAcf(0:(n-1), H)
tb<-matrix(c(tacf[1:10],sacvf[1:10]),ncol=2)
dimnames(tb)<-list(0:9, c("Tacvf","Sacvf"))
tb

coef.FitFGN

Display estimated parameters from FitFGN

Description

Method function to display fitted parameters, their standard errors and Z-ratio for FGN models fit with FitFGN.

Usage

## S3 method for class 'FitFGN':
coef(object, ...)

Arguments

object obj the output from FitFGN
...
optional parameters

Value

A matrix is returned. The columns of the matrix are labeled MLE, sd and Z-ratio. The rows labels indicate the AR coefficients which were estimated followed by mu, the estimate of mean.

Author(s)

A.I. McLeod

References

Examples

```r
data(NileMin)
out<-FitFGN(NileMin)
coef(out)
```

---

**plot.FitFGN**  
*Plot Method for "FitFGN" Object*

**Description**
Diagnostic plots of the residual autocorrelations and Ljung-Box test.

**Usage**

```r
plot.FitFGN(x, maxLag=30, ...)
```

**Arguments**

- `x`  
  object of class "FitFGN"
- `maxLag`  
  maximum lag in residual acf plot
- `...`  
  optional arguments

**Details**

The top plot shows the residual autocorrelations and their 5% significance limits. The bottom plot shows the p-values of the Ljung-Box test for various lags.

**Value**

No value is returned. A plots are produced as side-effect. The plot is a two-panel display showing the residual autocorrelations and the p-values for the Ljung-Box test.

**Author(s)**

A.I. McLeod

**References**


**See Also**

`summary.FitFGN`, `FitFGN`

**Examples**

```r
data(NileMin)
obj<-FitFGN(NileMin, c(1,2,6,7))
plot(obj)
```
Forecasts from a fitted FGN model

Description

The exact finite-sample minimum mean square error forecasts are computed using the Trench algorithm.

Usage

predict.FitFGN(object, n.ahead = 1, ...)

Arguments

- **object**: "FitFGN" object produced by FitFGN
- **n.ahead**: forecasts are done for lead times 1,...,n.ahead
- **...**: optional arguments, are ignored

Value

A list with components

- **Forecasts**: matrix with m+1 rows and maxLead columns with the forecasts
- **SDForecasts**: matrix with m+1 rows and maxLead columns with the sd of the forecasts

Author(s)

A.I. McLeod

References


See Also

FitFGN, TrenchForecast, PredictionVariance, predict.Arima

Examples

data(NileMin)
out<-FitFGN(NileMin)
predict(out, n.ahead=15)
Description

A terse summary is given.

Usage

```r
## S3 method for class 'FitFGN':
print(x, ...)
```

Arguments

- `x` object of class "FitFGN"
- `...` optional arguments

Value

A terse summary is displayed

Author(s)

A.I. McLeod

References


See Also

`summary.FitFGN`, `FitFGN`

Examples

```r
data(NileMin)
FitFGN(NileMin)
```
residuals.FitFGN  Extract Residuals from "FitFGN" Object

Description
Method function.

Usage
## S3 method for class 'FitFGN':
residuals(object, ...)

Arguments
object  object of class "FitFGN"
...  optional arguments

Value
Vector of standardized prediction residuals

Author(s)
A.I. McLeod

See Also
FitFGN

Examples
data(NileMin)
out<-FitFGN(NileMin)
qqnorm(resid(out))

summary.FitFGN  Summary Method for "FitFGN" Object

Description
summary for "FitFGN" object.

Usage
## S3 method for class 'FitFGN':
summary(object, ...)

Arguments
object  "FitFGN" object
...  optional arguments
Value

A printed summary is given

Author(s)

A.I. McLeod

See Also

print.FitFGN, FitFGN

Examples

data(NileMin)
out<-FitFGN(NileMin)
summary(out)
## Index

### Topic datagen
- SimulateFGN, 13

### Topic datasets
- NileFlowCMS, 11
- NileMin, 11

### Topic nonlinear
- FitRegressionFGN, 7

### Topic nonparametric
- HurstK, 9

### Topic regression
- FitRegressionFGN, 7

### Topic ts
- Boot, 2
- Boot.FitFGN, 1
- coef.FitFGN, 14
- FGNLcf, 3
- FGNLL, 4
- FitFGN, 5
- FitRegressionFGN, 7
- GetFitFGN, 8
- HurstK, 9
- plot.FitFGN, 15
- predict.FitFGN, 16
- print.FitFGN, 17
- residuals.FitFGN, 18
- SimulateFGN, 13
- summary.FitFGN, 18

### HurstK
- 6, 9

### lsfit
- 7

### NileFlowCMS
- 11

### NileMin
- 11

### optim
- 9

### optimize
- 9

### plot.FitFGN
- 6, 15

### predict.Arima
- 16

### predict.FitFGN
- 16

### PredictionVariance
- 16

### print.FitFGN
- 6, 17, 19

### residuals.FitFGN
- 18

### SimulateFGN
- 2, 13

### summary.FitFGN
- 6, 15, 17, 18

### TrenchForecast
- 16

### acf
- 4

### Boot
- 2

### Boot.FitFGN
- 1, 3, 6, 9

### coef.FitFGN
- 6, 14

### DHSimulate
- 2

### DLLLoglikelihood
- 5

### DLSimulate
- 2, 14

### FGNLcf
- 3

### FGNLL
- 4, 4

### FitFGN
- 5, 5, 7–10, 15–19

### FitRegressionFGN
- 6, 7, 9

### GetFitFGN
- 6, 8