Package ‘longmemo’

July 30, 2024

Version 1.1-3

Date 2024-07-30

Title Statistics for Long-Memory Processes (Book Jan Beran), and Related Functionality

Description Datasets and Functionality from 'Jan Beran' (1994). Statistics for Long-Memory Processes; Chapman & Hall. Estimation of Hurst (and more) parameters for fractional Gaussian noise, 'fARIMA' and 'FEXP' models.

Imports graphics, utils, stats

Enhances fracdiff

Suggests sfsmisc

BuildResaveData no

License GPL (>= 2)

NeedsCompilation no

Author Jan Beran [aut] (original S functions and scripts),
Martin Maechler [cre, aut] (Toplevel R functions and much more, <https://orcid.org/0000-0002-8685-9910>),
Brandon Whitcher [ctb] (Datasets)

Maintainer Martin Maechler <maechler@stat.math.ethz.ch>

Repository CRAN

Date/Publication 2024-07-30 09:30:02 UTC

Contents

`CetaARIMA` *Covariance for fractional ARIMA*

Description

Compute the covariance matrix of $\hat{eta}$ for a fractional ARIMA process.

Usage

```
CetaARIMA(eta, p, q, m = 10000, delta = 1e-9)
```

Arguments

`eta`	parameter vector `eta` = `c(H, phi, psi)`.
`p, q`	integer scalars giving the AR and MA order respectively.
`m`	integer specifying the length of the Riemann sum, with step size `2 * pi/m`.
`delta`	step size for numerical derivative computation.

Details

builds on calling `specARIMA(eta,p,q,m)`

Value

the (square) matrix containg covariances up to ...

Author(s)

Jan Beran (principal) and Martin Maechler (fine tuning)

References

Beran(1984), listing on p.224–225.

Examples

```
(C.7  <- CetaARIMA(0.7, m = 256, p = 0, q = 0))
(C.5  <- CetaARIMA(eta = c(H = 0.5, phi=c(-.06, 0.42, -0.36), psi=0.776),
                   m = 256, p = 3, q = 1))
```

CetaFGN — *Covariance Matrix of Eta for Fractional Gaussian Noise*

Description

Covariance matrix of $\hat{\eta}$ for fractional Gaussian noise (fGn).

Usage

```
CetaFGN(eta, m = 10000, delta = 1e-9)
```

Arguments

`eta`	parameter vector `eta = c(H, *)`.
`m`	integer specifying the length of the Riemann sum, with step size `2 * pi/m`. The default (10000) is realistic.
`delta`	step size for numerical derivative computation.

Details

Currently, the step size for numerical derivative is the same in all coordinate directions of `eta`. In principle, this can be far from optimal.

Value

Variance-covariance matrix of the estimated parameter vector $\hat{\eta}$.

Author(s)

Jan Beran (principal) and Martin Maechler (speedup, fine tuning)

See Also

`specFGN`

Examples

```
(C.7  <- CetaFGN(0.7, m = 256))
(C.5  <- CetaFGN(eta = c(H = 0.5), m = 256))
(C.5. <- CetaFGN(eta = c(H = 0.5), m = 1024))
```

`ckARMA0` *Covariances of a Fractional ARIMA(0,d,0) Process*

Description

Compute the Autocovariances of a fractional ARIMA(0,d,0) process (d = H - 1/2).

Usage

```
ckARMA0(n, H)
```

Arguments

`n`	sample size (length of time series).
`H`	self-similarity ('Hurst') parameter.

Details

The theoretical formula,

$$C(k) = (-1)^k \Gamma(1-2d)/(\Gamma(k+1-d)\Gamma(1-k-d)),$$

where $d = H - 1/2$, leads to over-/underflow for larger lags k; hence use the asymptotical formula there.

Value

numeric vector of length `n` of covariances $C(0) \ldots C(n-1)$.

Author(s)

Jan Beran (principal) and Martin Maechler (speedup, fine tuning)

References

Jan Beran (1994), p.63, (2.35) and (2.39).

See Also

`ckFGN0` which does the same for fractional Gaussian noise.

Examples

```
str(C.8 <- ckARMA0(50, H = 0.8))
yl <- c(0,max(C.8))
plot(0:49, C.8, type = "h", ylim = yl)
plot(0:49, C.8, type = "h", log = "xy",
     main = "Log-Log  ACF for ARIMA(0,d,0)")
```

---

ckFGN0 *Covariances of a Fractional Gaussian Process*

Description

Compute the Autocovariances of a fractional Gaussian process

Usage

```
ckFGN0(n, H)
```

Arguments

n	sample size (length of time series).
H	self-similarity ('Hurst') parameter.

Value

numeric vector of covariances upto lag n-1.

Author(s)

Jan Beran (principal) and Martin Maechler (fine tuning)

See Also

ckARMA0 which does the same for a fractional ARIMA process.

Examples

```
str(C.8 <- ckFGN0(50, H = 0.8))
plot(0:49, C.8, type = "h", ylim = 0:1)
plot(0:49, C.8, type = "h", log = "xy",
     main = "Log-Log  ACF for frac.GaussNoise(H = 0.8)")
```

---

ethernetTraffic *Ethernet Traffic Data Set*

Description

Ethernet traffic data from a LAN at Bellcore, Morristown (Leland et al. 1993, Leland and Wilson 1991). The data are listed in chronological sequence by row.

Usage

```
data(ethernetTraffic)
```

Format

A times series of length 4000.

Source

Jan Beran and Brandon Whitcher by E-mail in fall 1995.

Examples

```
data(ethernetTraffic)
str(ethernetTraffic)
plot(ethernetTraffic)## definitely special
```

FEXPest *Fractional EXP (FEXP) Model Estimator*

Description

`FEXPest(x, *)` computes Beran's Fractional EXP or 'FEXP' model estimator.

`.ffreq(n)` returns the Fourier frequencies $\frac{2\pi j}{n}$ (of a time series of length `n`).

Usage

```
FEXPest(x, order.poly, pvalmax, verbose = FALSE)
## S3 method for class 'FEXP'
print(x, digits = getOption("digits"), ...)

.ffreq(n, full = FALSE)
```

Arguments

`x`	numeric vector representing a time series.
`order.poly`	integer specifying the maximal polynomial order that should be taken into account. `order.poly = 0` is equivalent to a FARIMA(0,d,0) model.
`pvalmax`	maximal P-value – the other iteration stopping criterion and "model selection tuning parameter". Setting this to `1`, will use `order.poly` alone, and hence the final model order will be = `order.poly`.
`verbose`	logical indicating if iteration output should be printed.
`digits, ...`	optional arguments for `print` method, see `print.default`.
`n`	a positive integer, typically the length of a time series.
`full`	logical indicating if `n %/% 2` or by default "only" `(n-1) %/% 2` values should be returned.

Value

`FEXPest(x,..)` returns an object of class `FEXP`, basically a list with components

- `call` — the function `call`.
- `n` — time series length `length(x)`.
- `H` — the "Hurst" parameter which is simply `(1-theta[2])/2`.
- `coefficients` — numeric 4-column matrix as returned from `summary.glm()`, with estimate of the full parameter vector θ, its standard error estimates, t- and P-values, as from the `glm(*, family = Gamma)` fit.
- `order.poly` — the effective polynomial order used.
- `max.order.poly` — the original `order.poly` (argument).
- `early.stop` — logical indicating if `order.poly` is less than `max.order.poly`, i.e., the highest order polynomial terms were dropped because of a non-significant P-value.
- `spec` — the spectral estimate $f(\omega_j)$, at the Fourier frequencies ω_j. Note that `.ffreq(x$n)` recomputes the Fourier frequencies vector (from a fitted FEXP or WhittleEst model `x`).
- `yper` — raw periodogram of (centered and scaled `x`) at Fourier frequencies $I(\omega_j)$.

There currently are methods for `print()`, `plot` and `lines` (see `plot.FEXP`) for objects of class `"FEXP"`.

Author(s)

Martin Maechler, using Beran's "main program" in Beran(1994), p.234 ff

References

Beran, Jan (1993) Fitting long-memory models by generalized linear regression. *Biometrika* **80**, 817–822.

Beran, Jan (1994). *Statistics for Long-Memory Processes*; Chapman & Hall.

See Also

`WhittleEst`; the plot method, `plot.FEXP`.

Examples

```
data(videoVBR)
(fE  <- FEXPest(videoVBR, order = 3, pvalmax = .5))
(fE3 <- FEXPest(videoVBR, order = 3, pvalmax = 1 ))

(fE7 <- FEXPest(videoVBR, order = 3, pvalmax = 0.10))
##--> this also choses order 2, as "FE" :
all.equal(fE $coef,
          fE7$coef) # -> TRUE

confint(fE)
confint(fE7, level = 0.99)
```

```
.ffreq(8)
.ffreq(8, TRUE)
stopifnot(all.equal((1:3)/4,
                    .ffreq(8) / pi))
```

---

llplot *Log-Log and Log-X Plot of Spectrum*

Description

Log-Log and "Log-X" plot of spectrum. Very simple utilities, kept here mainly for back compatibility, as they appear in the book scripts.

Usage

```
llplot(yper, spec)
lxplot(yper, spec)
```

Arguments

yper	periodogram values
spec	spectrum values

Author(s)

Jan Beran (principal) and Martin Maechler (speedup, fine tuning)

See Also

`spectrum()` from standard R (package **stats**).

NBSdiff1kg *NBS measurement deviations from 1 kg*

Description

NBS weight measurements - deviation from 1 kg in micrograms, see the references. The data are listed in chronological sequence by row.

Usage

```
data(NBSdiff1kg)
```

Format

A time series of length 289.

Source

Jan Beran and Brandon Whitcher by E-mail in fall 1995.

References

H.P. Graf, F.R. Hampel, and J.Tacier (1984). The problem of unsuspected serial correlations. In J. Franke, W. Härdle, and R.D. Martin, editors, *Robust and Nonlinear Time Series Analysis*, Lecture Notes in Statistics **26**, 127–145; Springer.

Pollak, M., Croakin, C., and Hagwood, C. (1993). *Surveillance schemes with applications to mass calibration.* NIST report 5158; Gaithersburg, MD.

Examples

```
data(NBSdiff1kg)
plot(NBSdiff1kg)
```

`NhemiTemp` *Northern Hemisphere Temperature*

Description

Monthly temperature for the northern hemisphere for the years 1854-1989, from the data base held at the Climate Research Unit of the University of East Anglia, Norwich, England. The numbers consist of the temperature (degrees C) difference from the monthly average over the period 1950-1979.

Usage

```
data(NhemiTemp)
```

Format

Time-Series (`ts`) of length 1632, frequency 12, starting 1854, ending 1990.

Source

Jan Beran and Brandon Whitcher by E-mail in fall 1995.

References

Jones, P.D. and Briffa, K.R. (1992) Global surface air temperature variations during the twentieth century, part 1. *The Holocene* **2**, 165–179.

Jan Beran (1994). Dataset no. 5, p.29–31.

Examples

```
data(NhemiTemp)
plot(NhemiTemp)
mean(window(NhemiTemp, 1950,1979))# (about) 0 ``by definition''
```

NileMin *Nile River Minima, yearly 622–1284*

Description

Yearly minimal water levels of the Nile river for the years 622 to 1281, measured at the Roda gauge near Cairo, (Tousson, p. 366–385).

Usage

```
data(NileMin)
```

Format

Time-Series (ts) of length 663.

Source

The original Nile river data supplied by Beran only contained only 500 observations (622 to 1121). However, the book claimed to have 660 observations (622 to 1281). First added the remaining observations from the book by hand, and still came up short with only 653 observations (622 to 1264). Finally have 663 observations : years 622–1284 (as in orig. source)

References

Tousson, O. (1925). Mémoire sur l'Histoire du Nil; *Mémoire de l'Institut d'Egypte*.

Jan Beran (1994). Dataset no.1, p.20–22.

Examples

```
data(NileMin)
plot(NileMin, main = "Nile River Minima 622 - 1284")
```

per *Simple Periodogram Estimate*

Description

Simply estimate the periodogram via the Fast Fourier Transform.

Usage

```
per(z)
```

Arguments

z	numeric vector with the series to compute the periodogram from.

Details

This is basically the same as
`spec.pgram(z, fast = FALSE, detrend = FALSE, taper = 0) $ spec`, and not really recommended to use — exactly for the reason that `spec.pgram` has the defaults differently, `fast = TRUE, detrend = TRUE, taper = 0.1`, see that help page.

Value

numeric vector of length $1 + floor(n/2)$ where `n = length(z)`.

Author(s)

Jan Beran (principal) and Martin Maechler (fine tuning)

See Also

a more versatile periodogram estimate by `spec.pgram`.

Examples

```
data(NileMin)
plot(10*log10(per(NileMin)), type='l')
```

---

plot.FEXP *Plot Method for FEXP and WhittleEst Model Fits*

Description

(S3) methods for the generic functions `plot` (and `lines`) applied to **f**ractional EXP (FEXP) and `"WhittleEst"` (`FEXPest`, models. `plot()` plots the data periodogram and the 'FEXP" model estimated spectrum, where `lines()` and does the latter.

Usage

```
## S3 method for class 'FEXP'
plot(x, log = "xy", type = "l",
      col.spec = 4, lwd.spec = 2, xlab = NULL, ylab = expression(hat(f)(nu)),
      main = paste(deparse(x$call)[1]), sub = NULL, ...)

## (With identical argument list:)
## S3 method for class 'WhittleEst'
plot(x, log = "xy", type = "l",
      col.spec = 4, lwd.spec = 2, xlab = NULL, ylab = expression(hat(f)(nu)),
      main = paste(deparse(x$call)[1]), sub = NULL, ...)

## S3 method for class 'FEXP'
      lines(x, type = "l", col = 4, lwd = 2, ...)
## S3 method for class 'WhittleEst'
lines(x, type = "l", col = 4, lwd = 2, ...)
```

Arguments

- `x` — an R object of class "FEXP", as from `FEXPest()`.
- `log` — character specifying log scale should be used, see `plot.default`. Note that the default log-log scale is particularly sensible for long-range dependence.
- `type` — plot type for the periodogram, see `plot.default`.
- `col.spec, lwd.spec, col, lwd` — graphical parameters used for drawing the estimated spectrum, see `lines`.
- `xlab, ylab, main, sub` — labels for annotating the plot, see `title`, each with a sensible default.
- `...` — further arguments passed to `plot.default`.

Author(s)

Martin Maechler

See Also

`FEXPest`, `WhittleEst`; `plot.default` and `spectrum`.

Examples

```
data(videoVBR)
fE <- FEXPest(videoVBR, order = 3, pvalmax = .5)
plot(fE)
fE3 <- FEXPest(videoVBR, order = 3, pvalmax = 1)#-> order 3
lines(fE3, col = "red3", lty=2)

f.GN    <- WhittleEst(videoVBR)
f.am21  <- WhittleEst(videoVBR, model = "fARIMA",
                      start= list(H= .5, AR = c(.5,0), MA= .5))
lines(f.GN,   col = "blue4")
lines(f.am21, col = "goldenrod")

##--- Using a tapered periodogram ---------
spvVBR <- spec.pgram(videoVBR, fast=FALSE, plot=FALSE)
fam21 <- WhittleEst(periodogr.x = head(spvVBR$spec, -1),
                    n = length(videoVBR), model = "fARIMA",
                    start= list(H= .5, AR = c(.5,0), MA= .5))
fam21
f.am21 # similar but slightly different

plot(fam21)

## Now, comparing to traditional ("log-X", not "log-log") spectral plot:
plot(fam21, log="y")

## compared to the standard R spectral plot : s
if(dev.interactive(TRUE)) getOption("device")()# a new graphics window
plot(spvVBR, log = "yes", col="gray50")
all.equal(.ffreq(fE$n) / (2*pi) -> ffr.,
```

```
          head(spvVBR$freq, -1))# TRUE
lines(ffr., fam21$spec, col=4, lwd=2)
## need to adjust for different 'theta1':
lines(ffr., f.am21$spec * fam21$theta1 / f.am21$theta1,
      col = adjustcolor("tomato", 0.6), lwd=2)
```

Qeta *Approximate Log Likelihood for Fractional Gaussian Noise / Fractional ARIMA*

Description

`Qeta()` ($= \tilde{Q}(\eta)$ of Beran(1994), p.117) is up to scaling the negative log likelihood function of the specified model, i.e., fractional Gaussian noise or fractional ARIMA.

Usage

```
Qeta(eta, model = c("fGn","fARIMA"), n, yper, pq.ARIMA, give.B.only = FALSE)
```

Arguments

`eta`	parameter vector = (H, phi[1:p], psi[1:q]).
`model`	character specifying the kind model class.
`n`	data length
`yper`	numeric vector of length `(n-1)%/% 2`, the periodogram of the (scaled) data, see `per`.
`pq.ARIMA`	integer, = c(p,q) specifying models orders of AR and MA parts — only used when `model = "fARIMA"`.
`give.B.only`	logical, indicating if only the `B` component (of the `Values` list below) should be returned. Is set to `TRUE` for the Whittle estimator minimization.

Details

Calculation of A, B and $T_n = A/B^2$ where $A = 2\pi/n \sum_j 2 * [I(\lambda_j)/f(\lambda_j)]$, $B = 2\pi/n \sum_j 2 * [I(\lambda_j)/f(\lambda_j)]^2$ and the sum is taken over all Fourier frequencies $\lambda_j = 2\pi * j/n$, $(j = 1, \ldots, (n-1)/2)$.

f is the spectral density of fractional Gaussian noise or fractional ARIMA(p,d,q) with self-similarity parameter H (and p AR and q MA parameters in the latter case), and is computed either by `specFGN` or `specARIMA`.

$$cov(X(t), X(t+k)) = \int \exp(iuk) f(u) du$$

Value

a list with components

`n`	= input
`H`	(*input*) Hurst parameter, = `eta[1]`.
`eta`	= input
`A, B`	defined as above.
`Tn`	the goodness of fit test statistic $Tn = A/B^2$ defined in Beran (1992)
`z`	the standardized test statistic
`pval`	the corresponding p-value P(W > z)
`theta1`	the scale parameter $$\hat{\theta_1} = \frac{\hat{\sigma}_\epsilon^2}{2\pi}$$ such that $f() = \theta_1 f_1()$ and $integral(\log[f_1(.)]) = 0$.
`spec`	scaled spectral density f_1 at the Fourier frequencies ω_j, see `FEXPest`; a numeric vector.

Note

yper[1] must be the periodogram $I(\lambda_1)$ at the frequency $2\pi/n$, i.e., not the frequency zero !

Author(s)

Jan Beran (principal) and Martin Maechler (fine tuning)

References

Jan Beran (1992). A Goodness-of-Fit Test for Time Series with Long Range Dependence. *JRSS B* **54**, 749–760.

Beran, Jan (1994). *Statistics for Long-Memory Processes*; Chapman & Hall. (Section 6.1, p.116–119; 12.1.3, p.223 ff)

See Also

`WhittleEst` computes an approximate MLE for fractional Gaussian noise / fractional ARIMA, by minimizing `Qeta`.

Examples

```
data(NileMin)
y <- NileMin
n <- length(y)
yper <- per(scale(y))[2:(1+ (n-1) %/% 2)]
eta <- c(H = 0.3)
q.res <- Qeta(eta, n=n, yper=yper)
str(q.res)
```

simGauss — *Simulate (Fractional) Gaussian Processes*

Description

Simulation of a Gaussian series $X(1), \ldots, X(n)$. Whereas `simGauss` works from autocovariances, where `simFGN0` and `simARMA0` call it, for simulating a fractional ARIMA(0,d,0) process ($d = H - 1/2$), or fractional Gaussian noise, respectively.

Usage

```
simARMA0  (n, H)
simFGN0   (n, H)
simFGN.fft(n, H, ...)
simGauss(autocov)
```

Arguments

`n`	length of time series
`H`	self-similarity parameter
`...`	optional arguments passed to `B.specFGN()`.
`autocov`	numeric vector of auto covariances $\gamma(0), \ldots, \gamma(n-1)$.

Details

`simGauss` implements the method by Davies and Harte which is relatively fast using the FFT (`fft`) twice.

To simulate ARIMA(p, d, q), (for d in (-1/2, 1,2), you can use `arima.sim(n, model = list(ar= .., ma = ..), innov= simARMA0(n,H=d+1/2) , n.start = 0)`.

`simFGN.fft()` is about twice as fast as `simFGN0()` and uses Paxson's proposal, by default via `B.specFGN(*, k.approx=3, adjust=TRUE)`.

Value

The simulated series $X(1), \ldots, X(n)$, an R object of class `"ts"`, constructed from `ts()`.

Author(s)

Jan Beran (original) and Martin Maechler (`simGauss`, speedup, simplication). `simFGN.fft`: Vern Paxson.

References

Beran (1994), 11.3.3, p.216 f, referring to

Davis, R.B. and Harte, D.S. (1987). Tests for Hurst effect, *Biometrika* **74**, 95–102.

Vern Paxson (1997). Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic; *Computer Communications Review* **27** 5, 5–18.

See Also

`ckARMA0` on which `simARMA0` relies, and `ckFGN0` on which `simFGN0` relies.

Examples

```
x1 <- simFGN0(100, 0.7)
x2 <- simARMA0(100, 0.7)
plot(simFGN0(1000, 0.8)) #- time series plot
```

---

specARIMA — *Spectral Density of Fractional ARMA Process*

Description

Calculate the spectral density of a fractional ARMA process with standard normal innovations and self-similarity parameter H.

Usage

```
specARIMA(eta, p, q, m)
```

Arguments

`eta`	parameter vector `eta = c(H, phi, psi)`.
`p, q`	integers giving AR and MA order respectively.
`m`	sample size determining Fourier frequencies.

Details

at the Fourier frequencies $2*\pi*j/n$, $(j = 1, \ldots, (n-1))$, cov(X(t),X(t+k)) = (sigma/(2*pi))*integral(exp(iuk)g(u)du).

— or rather – FIXME –

1. cov(X(t),X(t+k)) = integral[exp(iuk)f(u)du]
2. f() = theta1 * f*() ; spec = f*(), and integral[log(f*())] = 0

Value

an object of class `"spec"` (see also `spectrum`) with components

`freq`	the Fourier frequencies (in $(0, \pi)$) at which the spectrum is computed, see `freq` in `specFGN`.
`spec`	the *scaled* values spectral density $f(\lambda)$ values at the `freq` values of λ. $f^*(\lambda) = f(\lambda)/\theta_1$ adjusted such $\int \log(f^*(\lambda))d\lambda = 0$.
`theta1`	the scale factor θ_1.
`pq`	a vector of length two, `= c(p,q)`.
`eta`	a named vector `c(H=H, phi=phi, psi=psi)` from input.
`method`	a character indicating the kind of model used.

Author(s)

Jan Beran (principal) and Martin Maechler (fine tuning)

References

Beran (1994) and more, see ....

See Also

The spectral estimate for fractional Gaussian noise, `specFGN`. In general, `spectrum` and `spec.ar`.

Examples

```
str(r.7  <- specARIMA(0.7, m = 256, p = 0, q = 0))
str(r.5  <- specARIMA(eta = c(H = 0.5, phi=c(-.06, 0.42, -0.36), psi=0.776),
                      m = 256, p = 3, q = 1))
plot(r.7)
plot(r.5)
```

`specFGN` *Spectral Density of Fractional Gaussian Noise*

Description

Calculation of the spectral density f of normalized fractional Gaussian noise with self-similarity parameter H at the Fourier frequencies 2*pi*j/m (j=1,...,(m-1)).

`B.specFGN` computes (approximations of) the $B(\lambda, H)$ component of the spectrum $f_H(\lambda)$.

Usage

```
  specFGN(eta, m, ...)
B.specFGN(lambd, H, k.approx=3, adjust = (k.approx == 3), nsum = 200)
```

Arguments

`eta`	parameter vector `eta = c(H, *)`.
`m`	sample size determining Fourier frequencies.
`...`	optional arguments for `B.specFGN()`: `k.approx`, etc
`lambd`	numeric vector of frequencies in [0, pi]
`H`	Hurst parameter in $(\frac{1}{2}, 1)$, (can be outside, here).
`k.approx`	either integer (the order of the Paxson approximation), or `NULL`, `NA` for choosing to use the slow direct sum (of `nsum` terms.)
`adjust`	logical indicating (only for `k.approx == 3`, the default) that Paxson's empirical adjustment should also be used.
`nsum`	if the slow sum is used (e.g. for k.approx = NA), the number of terms.

Details

Note that

1. cov(X(t),X(t+k)) = integral[exp(iuk)f(u)du]
2. f=theta1*spec and integral[log(spec)]=0.

Since **longmemo** version 1.1-0, a fast approximation is available (and default), using `k.approx` terms and an adjustment (`adjust=TRUE` in the default case of `k.approx=3`), which is due to the analysis and S code from Paxson (1997).

Value

`specFGN()` returns an object of class `"spec"` (see also `spectrum`) with components

`freq`	the Fourier frequencies $\omega_j \in (0, \pi)$) at which the spectrum is computed. Note that $omega_j = 2\pi j/m$ for $j = 1, ..., m-1$, and $m = \left\lfloor \frac{n-1}{2} \right\rfloor$.
`spec`	the *scaled* values spectral density $f(\lambda)$ values at the `freq` values of λ. $f^*(\lambda) = f(\lambda)/\theta_1$ adjusted such $\int \log(f^*(\lambda))d\lambda = 0$.
`theta1`	the scale factor θ_1.
`H`	the self-similarity parameter from input.
`method`	a character indicating the kind of model used.

`B.specFGN()` returns a vector of (approximate) values $B(\lambda, H)$.

Author(s)

Jan Beran originally (using the slow sum); Martin Maechler, based on Vern Paxson (1997)'s code.

References

Jan Beran (1994). *Statistics for Long-Memory Processes*; Chapman & Hall, NY.

Vern Paxson (1997). Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic; *Computer Communications Review* **27** 5, 5–18.

See Also

The spectral estimate for fractional ARIMA, `specARIMA`; more generally, `spectrum`.

Examples

```
 str(rg.7  <- specFGN(0.7, m = 100))
 str(rg.5  <- specFGN(0.5, m = 100))# { H = 0.5 <--> white noise ! }

 plot(rg.7) ## work around plot.spec() `bug' in R < 1.6.0
 plot(rg.5, add = TRUE, col = "blue")
 text(2, mean(rg.5$spec), "H = 0.5 [white noise]", col = "blue", adj = c(0,-1/4))
## This was the original method in longmemo, upto version 1.0-0 (incl):
 rg.7.o <- specFGN(0.7, m = 100, k.approx=NA, nsum = 200)
 ## quite accurate (but slightly slower):
```

```
rg.7f  <- specFGN(0.7, m = 100, k.approx=NA, nsum = 10000)
## comparing old and new default :
all.equal(rg.7, rg.7.o)# different in about 5th digit
all.equal(rg.7, rg.7f )# ==> new default is *more* accurate: 1.42 e-6
## takes about  7 sec {in 2011}:
rg.7ff <- specFGN(0.7, m = 100, k.approx=NA, nsum = 500000)
all.equal(rg.7f, rg.7ff)# ~ 10 ^ -7
all.equal(rg.7  $spec, rg.7ff$spec)# ~ 1.33e-6 -- Paxson is accurate!
all.equal(rg.7.o$spec, rg.7ff$spec)# ~ 2.40e-5 -- old default is less so
```

---

videoVBR *Video VBR data*

Description

Amount of coded information (**v**ariable **b**it **r**ate) per frame for a certain video sequence. There were about 25 frames per second.

Usage

```
data(videoVBR)
```

Format

a time-series of length 1000.

References

Heeke, H. (1991) Statistical multiplexing gain for variable bit rate codecs in ATM networks. *Int. J. Digit. Analog. Commun. Syst.* **4**, 261–268.

Heyman, D., Tabatabai, A., and Lakshman, T.V. (1991) Statistical analysis and simulation of video teleconferencing in ATM networks. *IEEE Trans. Circuits. Syst. Video Technol.* **2**, 49–59.

Jan Beran (1994). Dataset no. 2, p.22–23.

Examples

```
data(videoVBR)
plot(log(videoVBR), main="VBR Data (log)")
```

WhittleEst *Whittle Estimator for Fractional Gaussian Noise / Fractional ARIMA*

Description

Computes Whittle's approximate MLE for fractional Gaussian noise or fractional ARIMA (=: fARIMA) models, according to Beran's prescript.

Relies on minmizing Qeta() (= $\tilde{Q}(\eta)$), which itself is based on the "true" spectrum of the corresponding process; for the spectrum, either specFGN or specARIMA is used.

Usage

```
WhittleEst(x,
           periodogr.x = per(if(scale) x/sd(x) else x)[2:((n+1) %/% 2)],
           n = length(x), scale = FALSE,
           model = c("fGn", "fARIMA"),
           p, q,
           start = list(H= 0.5, AR= numeric(), MA=numeric()),
           verbose = getOption("verbose"))

## S3 method for class 'WhittleEst'
print(x, digits = getOption("digits"), ...)
```

Arguments

x	numeric vector representing a time series. Maybe omitted if periodogr.x and n are specified instead.
periodogr.x	the (raw) periodogram of x; the default, as by Beran, uses per, but tapering etc may be an alternative, see also spec.pgram.
n	length of the time series, length(x).
scale	logical indicating if x should be standardized to (sd) scale 1; originally, scale = TRUE used to be built-in; for compatibility with other methods, notably plotting spectra, scale = FALSE seems a more natural default.
model	numeric vector representing a time series.
p, q	optional integers specifying the AR and MA orders of the fARIMA model, i.e., only applicable when model is "fARIMA".
start	list of starting values; currently necessary for model = "fARIMA" and with a reasonable default for model = "fGn".
verbose	logical indicating if iteration output should be printed.
digits, ...	optional arguments for print method, see print.default.

Value

An object of class `WhittleEst` which is basically a list with components

`call`	the function `call`.
`model`	= input
`n`	time series length `length(x)`.
`p, q`	for "fARIMA": order of AR and MA parts, respectively.
`coefficients`	numeric 4-column matrix of coefficients with estimate of the full parameter vector η, its standard error estimates, z- and P-values. This includes the Hurst parameter H.
`theta1`	the scale parameter $\hat{\theta}_1$, see `Qeta`.
`vcov`	the variance-covariance matrix for η.
`periodogr.x`	= input (with default).
`spec`	the spectral estimate $\hat{f}(\omega_j)$.

There is a `print` method, and `coef`, `confint` or `vcov` methods work as well for objects of class `"WhittleEst"`.

Author(s)

Martin Maechler, based on Beran's "main program" in Beran(1994).

References

Beran, Jan (1994). *Statistics for Long-Memory Processes*; Chapman & Hall. (Section 6.1, p.116–119; 12.1.3, p.223 ff)

See Also

`Qeta` is the function minimized by these Whittle estimators.

`FEXPest` for an alternative model with Hurst parameter, also estimated by a "Whittle" approximate MLE, i.e., a Whittle's estimator in the more general sense.

The plot method, `plot.WhittleEst`.

Examples

```
data(NileMin)
(f.Gn.N  <- WhittleEst(NileMin))                             # H = 0.837
(f.A00.N <- WhittleEst(NileMin, model = "fARIMA", p=0, q=0)) # H = 0.899
confint(f.Gn.N)
confint(f.A00.N)

data(videoVBR)
(f.GN    <- WhittleEst(videoVBR))

## similar {but faster !}:
(f.am00  <- WhittleEst(videoVBR, model = "fARIMA", p=0, q=0))
```

```
rbind(f.am00$coef,
      f.GN  $coef)# really similar

f.am11  <- WhittleEst(videoVBR, model = "fARIMA",
                      start= list(H= .5, AR = .5, MA= .5))
f.am11
vcov(f.am11)

op <- if(require("sfsmisc"))
  mult.fig(3, main = "Whittle Estimators for videoVBR data")$old.par  else
  par(mar = c(3,1), mgp = c(1.5, 0.6, 0), mar = c(4,4,2,1)+.1)
plot(f.GN)
plot(f.am00)
plot(f.am11)

et <- as.list(coef(f.am11))
et$AR <- c(et$AR, 0, 0) # two more AR coefficients ..
f.am31  <- WhittleEst(videoVBR, model = "fARIMA", start = et)
## ... warning non nonconvergence,  but "kind of okay":
lines(f.am31, col = "red3") ## drawing on top of ARMA(1,1) above - *small* diff

f.am31 # not all three are "significant"
round(cov2cor(vcov(f.am31)), 3) # and they are highly correlated

et <- as.list(coef(f.am31))
et$AR <- unname(unlist(et[c("AR1", "AR2")]))
f.am21  <- WhittleEst(videoVBR, model = "fARIMA",
                      start = c(et[c("H","AR", "MA")]))
f.am21
lines(f.am21, col = adjustcolor("gold", .4), lwd=4)

par(op)## (reset graphic layout)

##--> ?plot.WhittleEst  for an example using  'periodogr.x'
```

Index