Package ‘fracdiff’

January 24, 2020

Version 1.5-1

VersionNote Released 1.4-2 on 2012-12-01, 1.5-0 on 2019-12-09

Date 2020-01-20

Title Fractionally Differenced ARIMA aka ARFIMA(P,d,q) Models

Description Maximum likelihood estimation of the parameters of a fractionally differenced ARIMA(p,d,q) model (Haslett and Raftery, Appl.Statistics, 1989); including inference and basic methods. Some alternative algorithms to estimate "H".

Imports stats

Suggests longmemo, forecast, urca

License GPL (>= 2)

URL https://github.com/mmaechler/fracdiff

BugReports https://github.com/mmaechler/fracdiff/issues

Encoding UTF-8

NeedsCompilation yes

Author Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>),
Chris Fraley [ctb, cph] (S original; Fortran code),
Friedrich Leisch [ctb] (R port,
<https://orcid.org/0000-0001-7278-1983>),
Valderio Reisen [ctb] (fdGPH() & fdSperio()),
Artur Lemonte [ctb] (fdGPH() & fdSperio()),
Rob Hyndman [ctb] (residuals() & fitted(),
<https://orcid.org/0000-0002-2140-5352>)

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

Repository CRAN

Date/Publication 2020-01-24 17:30:05 UTC

R topics documented:

  confint.fracdiff ........................................................... 2
  diffseries .................................................................. 3
confint.fracdiff

Confidence Intervals for Fracdiff Model Parameters

Description
Computes (Wald) confidence intervals for one or more parameters in a fitted fracdiff model, see fracdiff.

Usage
## S3 method for class 'fracdiff'
confint(object, parm, level = 0.95, ...)

Arguments
object an object of class fracdiff, typically result of fracdiff(..).
parm a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
level the confidence level required.
... additional argument(s) for methods.

Value
A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

Warning
As these confidence intervals use the standard errors returned by fracdiff() (which are based on finite difference approximations to the Hessian) they may end up being much too narrow, see the example in fracdiff.var.

Author(s)
Spencer Graves posted the initial version to R-help.
**diffseries**  

Fractionally Differenciate Data

**Description**

Differenciates the time series data using the approximated binomial expression of the long-memory filter and an estimate of the memory parameter in the ARFIMA(p,d,q) model.

**Usage**

diffseries(x, d)

**Arguments**

- `x`: numeric vector or univariate time series.
- `d`: number specifying the fractional difference order.

**Details**

Since 2018, we are using (an important correction of) the fast algorithm based on the discrete Fourier transform (`fft`) by Jensen and Nielsen which is significantly faster for large n = length(x).

**Value**

the fractionally differenced series x.

**Author(s)**

Valderio A. Reisen <valderio@cce.ufes.br> and Artur J. Lemonte (first slow version), now hidden as diffseries.0().


**See Also**

the generic `confint::fracdiff` model fitting, notably `fracdiff::var()` for re-estimating the variance-covariance matrix on which `confint()` builds entirely.

**Examples**

```r
set.seed(101)
ts2 <- fracdiff.sim(5000, ar = .2, ma = -.4, d = .3)
mFD <- fracdiff::fracdiff(ts2$series, nar = length(ts2$ar), nma = length(ts2$ma))
coef(mFD)
confint(mFD)
```
References

See those in `fdSperio`; additionally


See Also

`fracdiff.sim`

Examples

```r
memory.long <- fracdiff.sim(80, d = 0.3)
str(mGPH <- fdGPH(memory.long$series))
r <- diffseries(memory.long$series, d = mGPH$d)
#acf(r) # shouldn't show structure - ideally
```

---

**fdGPH**

*Geweke and Porter-Hudak Estimator for ARFIMA(p,d,q)*

Description

Estimate the fractional (or “memory”) parameter \(d\) in the ARFIMA(p,d,q) model by the method of Geweke and Porter-Hudak (GPH). The GPH estimator is based on the regression equation using the periodogram function as an estimate of the spectral density.

Usage

```r
fdGPH(x, bandw.exp = 0.5)
```

Arguments

- **x**: univariate time series
- **bandw.exp**: the bandwidth used in the regression equation

Details

The function also provides the asymptotic standard deviation and the standard error deviation of the fractional estimator.

The bandwidth is \(bw = \text{trunc}(n \times \text{bandw.exp})\), where \(0 < \text{bandw.exp} < 1\) and \(n\) is the sample size. Default bandw.exp = 0.5.
fdSperio

Value

d        GPH estimate
sd.as    asymptotic standard deviation
sd.reg   standard error deviation

Author(s)
Valderio A. Reisen and Artur J. Lemonte

References
see those in fdSperio.

See Also
fdSperio, fracdiff

Examples
memory.long <- fracdiff.sim(1500, d = 0.3)
fdGPH(memory.long$series)

Description
This function makes use Reisen (1994) estimator to estimate the memory parameter d in the ARFIMA(p,d,q) model. It is based on the regression equation using the smoothed periodogram function as an estimate of the spectral density.

Usage
fdSperio(x, bandw.exp = 0.5, beta = 0.9)

Arguments
x        univariate time series data.
bandw.exp numeric: exponent of the bandwidth used in the regression equation.
beta     numeric: exponent of the bandwidth used in the lag Parzen window.

Details
The function also provides the asymptotic standard deviation and the standard error deviation of the fractional estimator.

The bandwidths are bw = trunc(n ^ bandw.exp), where 0 < bandw.exp < 1 and n is the sample size. Default bandw.exp= 0.5;
and bw2 = trunc(n ^ beta), where 0 < beta < 1 and n is the sample size. Default beta = 0.9.
Value

a list with components

- d: Sperio estimate
- sd.as: asymptotic standard deviation
- sd.reg: standard error deviation

Author(s)

Valderio A. Reisen <valderio@cce.ufes.br> and Artur J. Lemonte

References


See Also

fdGPH, fracdiff

Examples

```r
memory.long <- fracdiff.sim(1500, d = 0.3)
spm <- fdSperio(memory.long$series)
str(spm, digits=6)
```

Description

Calculates the maximum likelihood estimators of the parameters of a fractionally-differenced ARIMA (p,d,q) model, together (if possible) with their estimated covariance and correlation matrices and standard errors, as well as the value of the maximized likelihood. The likelihood is approximated using the fast and accurate method of Haslett and Raftery (1989).

Usage

```r
fracdiff(x, nar = 0, nma = 0,
         ar = rep(NA, max(nar, 1)), ma = rep(NA, max(nma, 1)),
         dtol = NULL, drange = c(0, 0.5), h, M = 100, trace = 0)
```
Arguments

- `x`: time series (numeric vector) for the ARIMA model
- `nar`: number of autoregressive parameters `p`.
- `nma`: number of moving average parameters `q`.
- `ar`: initial autoregressive parameters.
- `ma`: initial moving average parameters.
- `dtol`: interval of uncertainty for `d`. If `dtol` is negative or NULL, the fourth root of machine precision will be used. `dtol` will be altered if necessary by the program.
- `drange`: interval over which the likelihood function is to be maximized as a function of `d`.
- `h`: size of finite difference interval for numerical derivatives. By default (or if negative),
  \[ h = \min(0.1, \text{eps.}5 \times (1 + \text{abs(cllf)})) \]
  where `cllf` := log. max. likelihood (as returned) and `eps.5` := sqrt(.Machine$double.neg.eps) (typically 1.05e-8).

  This is used to compute a finite difference approximation to the Hessian, and hence only influences the cov, cor, and std.error computations; use `fracdiff.var()` to change this after the estimation process.
- `M`: number of terms in the likelihood approximation (see Haslett and Raftery 1989).
- `trace`: optional integer, specifying a trace level. If positive, currently the “outer loop” iterations produce one line of diagnostic output.

Details

The `fracdiff` package has — for historical reason, namely, S-plus `arima()` compatibility — used an unusual parametrization for the MA part, see also the ‘Details’ section in `arima` (in standard R’s `stats` package). The ARMA (i.e., `d = 0`) model in `fracdiff()` and `fracdiff.sim()` is

\[ X_t - a_1 X_{t-1} - \cdots - a_p X_{t-p} = e_t - b_1 e_{t-1} - \cdots - b_q e_{t-q}, \]

where `e_t` are mean zero i.i.d., for `fracdiff()`’s estimation, `e_t \sim \mathcal{N}(0, \sigma^2)`. This model indeed has the signs of the MA coefficients `b_j` inverted, compared to other parametrizations, including Wikipedia’s [http://en.wikipedia.org/wiki/Autoregressive_moving-average_model](http://en.wikipedia.org/wiki/Autoregressive_moving-average_model) and the one of `arima`.

Note that NA’s in the initial values for `ar` or `ma` are replaced by 0’s.

Value

an object of S3 class "fracdiff", which is a list with components:

- `log.likelihood`: logarithm of the maximum likelihood
- `d`: optimal fractional-differencing parameter
- `ar`: vector of optimal autoregressive parameters
- `ma`: vector of optimal moving average parameters
covariance.dpq  covariance matrix of the parameter estimates (order : d, ar, ma).
stderr.dpq    standard errors of the parameter estimates c(d,ar,ma).
correlation.dpq correlation matrix of the parameter estimates (order : d, ar, ma).
h           interval used for numerical derivatives, see h argument.
dtol       interval of uncertainty for d; possibly altered from input dtol.
M          as input.
hessian.dpq the approximate Hessian matrix $H$ of 2nd order partial derivatives of the likelihood with respect to the parameters; this is (internally) used to compute covariance.dpq, the approximate asymptotic covariance matrix as $C = (-H)^{-1}$.

Method

The optimization is carried out in two levels:
an outer univariate unimodal optimization in d over the interval drange (typically [0,.5]), using Brent’s fmin algorithm, and
an inner nonlinear least-squares optimization in the AR and MA parameters to minimize white noise variance (uses the MINPACK subroutine lmDER). written by Chris Fraley (March 1991).

Warning

The variance-covariance matrix and consequently the standard errors may be quite inaccurate, see the example in fracdiff.var.

Note

Ordinarily, nar and nma should not be too large (say < 10) to avoid degeneracy in the model. The function fracdiff.sim is available for generating test problems.

References


See Also

coeff.fracdiff and other methods for "fracdiff" objects; fracdiff.var() for re-estimation of variances or standard errors; fracdiff.sim
Examples

```r
ts.test <- fracdiff.sim(5000, ar = .2, ma = -.4, d = .3)
fd. <- fracdiff(ts.test$series, nar = length(ts.test$ar), nma = length(ts.test$ma))
fd.
## Confidence intervals
confint(fd.)
## with iteration output
fd2 <- fracdiff(ts.test$series, nar = 1, nma = 1, trace = 1)
all.equal(fd., fd2)
```

Description

Many “accessor” methods for `fracdiff` objects, notably `summary`, `coef`, `vcov`, and `logLik`; further `print()` methods were needed.

Usage

```r
## S3 method for class 'fracdiff'
coef(object, ...)
## S3 method for class 'fracdiff'
logLik(object, ...)
## S3 method for class 'fracdiff'
print(x, digits = getOption("digits"), ...)
## S3 method for class 'fracdiff'
summary(object, symbolic.cor = FALSE, ...)
## S3 method for class 'summary.fracdiff'
print(x, digits = max(3, getOption("digits") - 3),
      correlation = FALSE, symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)
## S3 method for class 'fracdiff'
fitted(object, ...)
## S3 method for class 'fracdiff'
residuals(object, ...)
## S3 method for class 'fracdiff'
vcov(object, ...)
```

Arguments

- `x, object` object of class `fracdiff`.
- `digits` the number of significant digits to use when printing.
- `...` further arguments passed from and to methods.
correlation logical; if TRUE, the correlation matrix of the estimated parameters is returned and printed.

symbolic.cor logical. If TRUE, print the correlations in a symbolic form (see symnum) rather than as numbers.

signif.stars logical. If TRUE, “significance stars” are printed for each coefficient.

Author(s)
Martin Maechler; Rob Hyndman contributed the residuals() and fitted() methods.

See Also
fracdiff to get “fracdiff” objects, confint.fracdiff for the confint method; further, fracdiff.var.

Examples
set.seed(7)
ts4 <- fracdiff.sim(10000, ar = c(0.6, -.05, -0.2), ma = -0.4, d = 0.2)
modFD <- fracdiff(ts4$series, nar = length(ts4$ar), nma = length(ts4$ma))
## -> warning (singular Hessian) % FIXME ????
coef(modFD) # the estimated parameters
vcov(modFD)
smFD <- summary(modFD)
smFD
coef(smFD) # gives the whole table
AIC(modFD) # AIC works because of the logLik() method
stopifnot(exprs = {
  
})
fracdiff.sim

Arguments

  n  length of the time series.
  ar  vector of autoregressive parameters; empty by default.
  ma  vector of moving average parameters; empty by default.
  d  fractional differencing parameter.
  rand.gen  a function to generate the innovations; the default, rnorm generates white N(0,1) noise.
  innov  an optional times series of innovations. If not provided, rand.gen() is used.
  n.start  length of “burn-in” period. If NA, the default, the same value as in arima.sim is computed.
  backComp  logical indicating if back compatibility with older versions of fracdiff.sim is desired. Otherwise, for d = 0, compatibility with R's arima.sim is achieved.
  allow.0.nstart  logical indicating if n.start = 0 should be allowed even when p + q > 0. This not recommended unless for producing the same series as with older versions of fracdiff.sim.
  start.innov  an optional vector of innovations to be used for the burn-in period. If supplied there must be at least n.start values.
  ...  additional arguments for rand.gen(). Most usefully, the standard deviation of the innovations generated by rnorm can be specified by sd.
  mu  time series mean (added at the end).

Value

  a list containing the following elements :

    series  time series
    ar, ma, d, mu, n.start  same as input

See Also

  fracdiff, also for references; arima.sim

Examples

  ## Pretty (too) short to "see" the long memory
  fracdiff.sim(100, ar = .2, ma = .4, d = .3)

  ## longer with "extreme" ar:
  r <- fracdiff.sim(n=1500, ar=-0.9, d= 0.3)
  plot(as.ts(r$series))

  ## Show that MA coefficients meaning is inverted
  ## compared to  stats :: arima :
  AR <- 0.7
MA <- -0.5
n.st <- 2

AR <- c(0.7, -0.1)
MA <- c(-0.5, 0.4)
n <- 512; sd <- 0.1
n.st <- 10

set.seed(101)
Y1 <- arima.sim(list(ar = AR, ma = MA), n = n, n.start = n.st, sd = sd)
plot(Y1)

# For our fracdiff, reverse the MA sign:
set.seed(101)
Y2 <- fracdiff.sim(n = n, ar = AR, ma = - MA, d = 0,
n.start = n.st, sd = sd)$series
lines(Y2, col=adjustcolor("red", 0.5))
### no, you don't need glasses ;-) Y2 is Y1 shifted slightly

r <- function(x, k) {
  stopifnot(k == round(k))
  n <- length(x)
  if(k <= k %% n) x[c((k+1):n, 1:k)] else x
}
k <- n.st - 2
Y2.s <- rot(Y2, k)
head.matrix(cbind(Y1, Y2.s))
plot(Y1, Y2.s); i <- (n-k+1):n
text(Y1[i], Y2.s[i], i, adj = c(0,0)-.1, col=2)

### With backComp = FALSE, get *the same* as arima.sim():
set.seed(101)
Y2. <- fracdiff.sim(n = n, ar = AR, ma = - MA, d = 0,
n.start = n.st, sd = sd, backComp = FALSE)$series
stopifnot( all.equal( c(Y1), Y2., tolerance=1e-15))

---

fracdiff.var  

Recompute Covariance Estimate for fracdiff

Description

Allows the finite-difference interval to be altered for recomputation of the covariance estimate for fracdiff.

Usage

fracdiff.var(x, fracdiff.out, h)
Arguments

- **x**: a univariate time series or a vector. Missing values (NAs) are not allowed.
- **fracdiff.out**: output from `fracdiff` for time series `x`.
- **h**: finite-difference interval for approximating partial derivatives with respect to the `d` parameter.

Value

An object of S3 class "fracdiff", i.e., basically a list with the same elements as the result from `fracdiff`, but with possibly different values for the hessian, covariance, and correlation matrices and for standard error, as well as for `h`.

See Also

`fracdiff`, also for references.

Examples

```r
## Generate a fractionally-differenced ARIMA(1,d,1) model :
 ts.test <- fracdiff.sim(10000, ar = .2, ma = .4, d = .3)
## estimate the parameters in an ARIMA(1,d,1) model for the simulated series
 fd.out <- fracdiff(ts.test$series, nar= 1, nma = 1)

## Modify the covariance estimate by changing the finite-difference interval
(fd.o2 <- fracdiff.var(ts.test$series, fd.out, h = .0001))
## looks identical as print(fd.out),
## however these (e.g.) differ :
 vcov(fd.out)
 vcov(fd.o2)

## A case, were the default variance is *clearly* way too small:
set.seed(1); fdc <- fracdiff(X <- fracdiff.sim(n=100,d=0.25)$series)
 fdc
# Confidence intervals just based on asymp.normal approx. and std.errors:
confint(fdc) # ridiculously too narrow
```
Index

*Topic models
  confint.fracdiff, 2
  fracdiff-methods, 9
*Topic print
  fracdiff-methods, 9
*Topic ts
  diffseries, 3
  fdGPH, 4
  fdsSperio, 5
  fracdiff, 6
  fracdiff.sim, 10
  fracdiff.var, 12
arima, 7
arima.sim, 11
class, 7, 13
coef, 9
coef.fracdiff, 8
coef.fracdiff(fracdiff-methods), 9
confint, 3, 10
confint.fracdiff, 2, 10
diffseries, 3
fdGPH, 4, 6
fdsSperio, 4, 5, 5
fft, 3
fitted, 10
fitted.fracdiff(fracdiff-methods), 9
fracdiff, 2, 3, 5, 6, 6, 9–11, 13
fracdiff-methods, 9
fracdiff.sim, 4, 7, 8, 10
fracdiff.var, 2, 3, 7, 8, 10, 12
logLik, 9
logLik.fracdiff(fracdiff-methods), 9
print, 9
print.fracdiff(fracdiff-methods), 9
print.summary.fracdiff
  (fracdiff-methods), 9
residuals, 10
residuals.fracdiff(fracdiff-methods), 9
rnorm, 11
summary, 9
summary.fracdiff(fracdiff-methods), 9
symnum, 10
vcov, 9
vcov.fracdiff(fracdiff-methods), 9