Package 'midasr'

July 2, 2014

Title Mixed Data Sampling Regression

Description Econometric methods for mixed frequency time series data analysis

URL http://mpiktas.github.io/midasr/

Version 0.3

Maintainer Vaidotas Zemlys <zemlys@gmail.com>

Author Virmantas Kvedaras <virmantas.kvedaras@mif.vu.lt>, Vaidotas Zemlys <vaidotas.zemlys@mif.vu.lt>

Depends R (>= 2.11.0), sandwich, optimx

Imports MASS, numDeriv, Matrix

License GPL-2 | MIT + file LICENCE

BugReports https://github.com/mpiktas/midasr/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-30 11:03:11

R topics documented:

`midasr-package` *Estimating and testing MIDAS regression*

Description

Package for estimating and testing MIDAS regression.

Details

The main feature of this package is function `hAh.test` which performs test whether coefficients of MIDAS regression have certain functional form

Author(s)

Virmantas Kvedaras <virmantas.kvedaras@mif.vu.lt>, Vaidotas Zemlys (maintainer) <zemlys@gmail.com>

`+.lws_table` *Combine* `lws_table` *objects*

Description

Combines `lws_table` objects

Usage

```
## S3 method for class 'lws_table'
... + check = TRUE
```

Arguments

`...`	`lws_table` object
`check`	logical, if TRUE checks that the each `lws_table` object is named a list with names `c("weights","lags","starts")`

Details

The `lws_table` objects have similar structure to table, i.e. it is a list with 3 elements which are the lists with the same number of elements. The base function `c` would `cbind` such tables. This function `rbind`s them.

Value

`lws_table` object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
nlmn <- expand_weights_lags("nealmon",0,c(4,8),1,start=list(nealmon=rep(0,3)))
nbt <- expand_weights_lags("nbeta",0,c(4,8),1,start=list(nbeta=rep(0,4)))

nlmn+nbt
```

---

agk.test *Andreou, Ghysels, Kourtellos LM test*

Description

Perform the test whether hyperparameters of normalized exponential Almon lag weights are zero

Usage

```
agk.test(x)
```

Arguments

`x`	MIDAS regression object of class `midas_r`

Value

a `htest` object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Andreou E., Ghysels E., Kourtellos A. *Regression models with mixed sampling frequencies* Journal of Econometrics 158 (2010) 246-261

Examples

```
##' ##Load data
data("USunempr")
data("USrealgdp")

y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
t <- 1:length(y)

mr <- midas_r(y~t+fmls(x,11,12,nealmon),start=list(x=c(0,0,0)))

agk.test(mr)
```

`AICc` *Compute AICc*

Description

Function to compute AICc information criteria for a given model

Usage

```
AICc(mod, ...)
```

Arguments

`mod`	`midas_r` model
`...`	additional parameters

Details

A generic function for calculating AICc. It is implemented for midas_r package. For more examples see package AICcmodavg.

Value

a computed AICc value, a number.

Author(s)

Vaidotas Zemlys

almonp *Almon polynomial MIDAS weights specification*

Description

Calculate Almon polynomial MIDAS weights

Usage

```
almonp(p, d, m)
```

Arguments

p	parameters for Almon polynomial weights
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

almonp.gradient *Gradient function for Almon polynomial MIDAS weights*

Description

Calculate gradient for Almon polynomial MIDAS weights specification

Usage

```
almonp.gradient(p, d, m)
```

Arguments

p	vector of parameters for Almon polynomial specification
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Vaidotas Zemlys

`amidas_table` *Weight and lag selection table for aggregates based MIDAS regression model*

Description

Create weight and lag selection table for the aggregates based MIDAS regression model

Usage

```
amidas_table(formula, data, weights, wstart, type, start = NULL, from, to,
  IC = c("AIC", "BIC"), test = c("hAh.test"), Ofunction = "optim",
  user.gradient = FALSE, ...)
```

Arguments

`formula`	the formula for MIDAS regression, the lag selection is performed for the last MIDAS lag term in the formula
`data`	a list containing data with mixed frequencies
`weights`	the names of weights used in Ghysels schema
`wstart`	the starting values for the weights of the firs low frequency lag
`type`	the type of Ghysels schema see amweights, can be a vector of types
`start`	the starting values for optimisation excluding the starting values for the last term
`from`	a named list, or named vector with high frequency (NB!) lag numbers which are the beginnings of MIDAS lag structures. The names should correspond to the MIDAS lag terms in the formula for which to do the lag selection. Value NA indicates lag start at zero
`to`	to a named list where each element is a vector with two elements. The first element is the low frequency lag number from which the lag selection starts, the second is the low frequency lag number at which the lag selection ends. NA indicates lowest (highest) lag numbers possible.
`IC`	the names of information criteria which should be calculated
`test`	the names of statistical tests to perform on restricted model, p-values are reported in the columns of model selection table
`Ofunction`	see midasr
`user.gradient`	see midas_r
`...`	additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentialy increasing the midas lag from kmin to kmax and varying the weights of the last term of the given formula

This function estimates models sequentially increasing the midas lag from kmin to kmax and varying the weights of the last term of the given formula

Value

a midas_r_ic_table object which is the list with the following elements:

table	the table where each row contains calculated information criteria for both restricted and unrestricted MIDAS regression model with given lag structure
candlist	the list containing fitted models
IC	the argument IC
test	the argument test
weights	the names of weight functions
lags	the lags used in models

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)

tb <- amidas_table(y~trend+fmls(x,12,12,nealmon),
                   data=list(y=y,x=x,trend=trend),
                   weights=c("nealmon"),wstart=list(nealmon=c(0,0,0)),
                   start=list(trend=1),type=c("A"),
                   from=0,to=c(1,2))
```

---

amweights *Weights for aggregates based MIDAS regressions*

Description

Produces weights for aggregates based MIDAS regression

Usage

```
amweights(p, d, m, weight = nealmon, type = c("A", "B", "C"))
```

Arguments

p	parameters for weight functions, see details.
d	number of lags
m	the frequency
weight	the weight function
type	type of structure, a string, one of A, B or C.

Details

Given a weight function $w(\beta, \theta)$ which has a property of being defined as $\beta g(\theta)$ the following combinations are defined, corresponding to structure types A, B and C respectively:

$$(w(\beta_1, \theta_1), ..., w(\beta_k, \theta_k))$$

$$(w(\beta_1, \theta), ..., w(\beta_k, \theta))$$

$$\beta(w(1, \theta_1), ..., w(1, \theta_k))$$

The starting values p should be supplied then as follows:

$$(\beta_1, \theta_1, ..., \beta_k, \theta_k)$$

$$(\beta_1, ..., \beta_k, \theta)$$

$$(\beta, \theta_1, ..., \theta_k)$$

Value

a vector of weights

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

---

average_forecast *Average forecasts of MIDAS models*

Description

Average MIDAS model forecasts using specified weighting scheme. Produce in-sample and out-of-sample accuracy measures.

Usage

```
average_forecast(modlist, data, insample, outsample, type = c("fixed",
  "recursive", "rolling"), fweights = c("EW", "BICW", "MSFE", "DMSFE"),
  measures = c("MSE", "MAPE", "MASE"), showprogress = TRUE)
```

Arguments

modlist	a list of midas_r objects
data	a list with mixed frequency data
insample	the low frequency indexes for in-sample data
outsample	the low frequency indexes for out-of-sample data
type	a string indicating which type of forecast to use.
fweights	names of weighting schemes
measures	names of accuracy measures
showprogress	logical, TRUE to show progress bar, FALSE for silent evaluation

Details

Given the data, split it to in-sample and out-of-sample data. Then given the list of models, reestimate each model with in-sample data and produce out-of-sample forecast. Given the forecasts average them with the specified weighting scheme. Then calculate the accuracy measures for individual and average forecasts.

The forecasts can be produced in 3 ways. The "fixed" forecast uses model estimated with in-sample data. The "rolling" forecast reestimates model each time by increasing the in-sample by one low frequency observation and dropping the first low frequency observation. These reestimated models then are used to produce out-of-sample forecasts. The "recursive" forecast differs from "rolling" that it does not drop observations from the beginning of data.

Value

a list containing forecasts and tables of accuracy measures

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
set.seed(1001)
## Number of low-frequency observations
n<-250
## Linear trend and higher-frequency explanatory variables (e.g. quarterly and monthly)
trend<-c(1:n)
x<-rnorm(4*n)
z<-rnorm(12*n)
## Exponential Almon polynomial constraint-consistent coefficients
fn.x <- nealmon(p=c(1,-0.5),d=8)
fn.z <- nealmon(p=c(2,0.5,-0.1),d=17)
## Simulated low-frequency series (e.g. yearly)
y<-2+0.1*trend+mls(x,0:7,4)%*%fn.x+mls(z,0:16,12)%*%fn.z+rnorm(n)
mod1 <- midas_r(y ~ trend + mls(x, 4:14, 4, nealmon) + mls(z, 12:22, 12, nealmon),
                start=list(x=c(10,1,-0.1),z=c(2,-0.1)))
mod2 <- midas_r(y ~ trend + mls(x, 4:20, 4, nealmon) + mls(z, 12:25, 12, nealmon),
                start=list(x=c(10,1,-0.1),z=c(2,-0.1)))
```

```
##Calculate average forecasts
avgf <- average_forecast(list(mod1,mod2),
                         data=list(y=y,x=x,z=z,trend=trend),
                         insample=1:200,outsample=201:250,
                         type="fixed",
                         measures=c("MSE","MAPE","MASE"),
                         fweights=c("EW","BICW","MSFE","DMSFE"))
```

---

checkARstar *Check whether the MIDAS model is MIDAS-AR* model*

Description

Checks whether the MIDAS model is MIDAS-AR* model and provides necessary modifications

Usage

```
checkARstar(trms)
```

Arguments

trms	terms of the model formula

Author(s)

Julius Vainora

---

check_mixfreq *Check data for MIDAS regression*

Description

Given mixed frequency data check whether higher frequency data can be converted to the lowest frequency.

Usage

```
check_mixfreq(data)
```

Arguments

data	a list containing mixed frequency data

Details

The number of observations in higher frequency data elements should have a common divisor with the number of observations in response variable. It is always assumed that the response variable is of the lowest frequency.

Value

a boolean TRUE, if mixed frequency data is conformable, FALSE if it is not.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

---

deriv_tests *Check whether non-linear least squares restricted MIDAS regression problem has converged*

Description

Computes the gradient and hessian of the optimisation function of restricted MIDAS regression and checks whether the conditions of local optimum are met. Numerical estimates are used.

Usage

```
deriv_tests(x, tol = 1e-06)

## S3 method for class 'midas_r'
deriv_tests(x, tol = 1e-06)
```

Arguments

x	midas_r object
tol	a tolerance, values below the tolerance are considered zero

Value

a list with gradient, hessian of optimisation function and convergence message

Author(s)

Vaidotas Zemlys

See Also

midas_r

deviance.midas_r *MIDAS regression model deviance*

Description

Returns the deviance of a fitted MIDAS regression object

Usage

```
## S3 method for class 'midas_r'
deviance(object, ...)
```

Arguments

object	a midas_r object
...	currently nothing

Value

The sum of squared residuals

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

dmls *MIDAS lag structure for unit root processes*

Description

Prepares MIDAS lag structure for unit root processes

Usage

```
dmls(x, k, m, ...)
```

Arguments

x	a vector
k	maximal lag order
m	frequency ratio
...	further arguments used in fitting MIDAS regression

Value

a matrix containing the first differences and the lag k+1.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

`expand_amidas` *Create table of weights, lags and starting values for Ghysels weight schema*

Description

Create table of weights, lags and starting values for Ghysels weight schema, see amweights

Usage

```
expand_amidas(weight, type = c("A", "B", "C"), from = 0, to, m, start)
```

Arguments

`weight`	the names of weight functions
`type`	the type of Ghysels schema, `"A"`, `"B"` or `"C"`
`from`	the high frequency lags from which to start the fitting
`to`	to a vector of length two, containing minimum and maximum lags, high frequency if `m=1`, low frequency otherwise.
`m`	the frequency ratio
`start`	the starting values for the weights of the one low frequency lag

Details

Given weight function creates lags starting from `kmin` to `kmax` and replicates starting values for each low frequency lag.

Value

a `lws_table` object, a list with elements `weights`, `lags` and `starts`

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
expand_amidas("nealmon","A",0,c(1,2),12,c(0,0,0))
```

`expand_weights_lags` *Create table of weights, lags and starting values*

Description

Creates table of weights, lags and starting values

Usage

```
expand_weights_lags(weights, from = 0, to, m = 1, start)
```

Arguments

`weights`	either a vector with names of the weight functions or a named list of weight functions
`from`	the high frequency lags from which to start the fitting
`to`	a vector of length two, containing minimum and maximum lags, high frequency if `m=1`, low frequency otherwise.
`m`	the frequency ratio
`start`	a named list with the starting values for weight functions

Details

For each weight function creates lags starting from `kmin` to `kmax`. This is a convenience function for easier work with the function midas_r_ic_table.

Value

a `lws_table` object, a list with elements `weights`, `lags` and `starts`.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
expand_weights_lags(c("nealmon","nbeta"),0,c(4,8),1,start=list(nealmon=rep(0,3),nbeta=rep(0,4)))
nlmn <- expand_weights_lags("nealmon",0,c(4,8),1,start=list(nealmon=rep(0,3)))
nbt <- expand_weights_lags("nbeta",0,c(4,8),1,start=list(nbeta=rep(0,4)))

nlmn+nbt
```

fmls *Full MIDAS lag structure*

Description

Create a matrix of MIDAS lags, including contemporaneous lag up to selected order.

Usage

```
fmls(x, k, m, ...)
```

Arguments

x	a vector
k	maximum lag order
m	frequency ratio
...	further arguments

Details

This is a convenience function, it calls `link{msl}` to perform actual calculations.

Value

a matrix containing the lags

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

See Also

mls

forecast *Forecast MIDAS regression*

Description

Forecasts MIDAS regression given the future values of regressors. For dynamic models (with lagged response variable) there is an option to calculate dynamic forecast, when forecasted values of response variable are substituted into the lags of response variable.

Usage

```
forecast(object, ...)

## S3 method for class 'midas_r'
forecast(object, newdata = NULL, method = c("static",
  "dynamic"), insample = get_estimation_sample(object), ...)
```

Arguments

object	midas_r object
newdata	newdata
method	the forecasting method, either "static" or "dynamic"
insample	a list containing the historic mixed frequency data
...	additional arguments, not used

Details

Given future values of regressors this function combines the historical values used in the fitting the MIDAS regression model and calculates the forecasts.

Value

a vector of forecasts

Author(s)

Vaidotas Zemlys

Examples

```
data("USrealgdp")
data("USunempr")

y <- diff(log(USrealgdp))
x <- window(diff(USunempr), start = 1949)
trend <- 1:length(y)

##24 high frequency lags of x included
mr <- midas_r(y ~ trend + fmls(x, 23, 12, nealmon), start = list(x = rep(0, 3)))

##Forecast horizon
h <- 3
##Declining unemployment
xn <- rep(-0.1, 12*3)
##New trend values
trendn <- length(y) + 1:h

##Static forecasts combining historic and new high frequency data
forecast(mr, list(trend = trendn, x = xn), method = "static")
```

```
##Dynamic AR* model
mr.dyn <- midas_r(y ~ trend + mls(y, 1:2, 1, "*")
                  + fmls(x, 11, 12, nealmon),
                 start = list(x = rep(0, 3)))

forecast(mr.dyn, list(trend = trendn, x = xn), method = "dynamic")
```

`get_estimation_sample` *Get the data which was used to etimate MIDAS regression*

Description

Gets the data which was used to estimate MIDAS regression

Usage

```
get_estimation_sample(object)
```

Arguments

`object`	`midas_r` object

Details

A helper function.

Value

a named list with mixed frequency data

Author(s)

Vaidotas Zemlys

`gompertzp` *Normalized Gompertz probability density function MIDAS weights specification Calculate MIDAS weights according to normalized Gompertz probability density function specification*

Description

Normalized Gompertz probability density function MIDAS weights specification Calculate MIDAS weights according to normalized Gompertz probability density function specification

Usage

```
gompertzp(p, d, m)
```

Arguments

p	parameters for normalized Gompertz probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

---

gompertzp.gradient *Gradient function for normalized Gompertz probability density function MIDAS weights specification Calculate gradient function for normalized Gompertz probability density function specification of MIDAS weights.*

Description

Gradient function for normalized Gompertz probability density function MIDAS weights specification Calculate gradient function for normalized Gompertz probability density function specification of MIDAS weights.

Usage

```
gompertzp.gradient(p, d, m)
```

Arguments

p	parameters for normalized Gompertz probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

hAh.test *Test restrictions on coefficients of MIDAS regression*

Description

Perform a test whether the restriction on MIDAS regression coefficients holds.

Usage

```
hAh.test(x)
```

Arguments

x	MIDAS regression model with restricted coefficients, estimated with midas_r

Details

Given MIDAS regression:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + u_t$$

test the null hypothesis that the following restriction holds:

$$\theta_h = g(h, \lambda),$$

where $h = 0, ..., (k+1)m$.

Value

a htest object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Kvedaras V., Zemlys, V. *Testing the functional constraints on parameters in regressions with variables of different frequency* Economics Letters 116 (2012) 250-254

See Also

hAhr.test

Examples

```
##The parameter function
theta.h0 <- function(p, dk, ...) {
   i <- (1:dk-1)
   (p[1] + p[2]*i)*exp(p[3]*i + p[4]*i^2)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,0.1,-0.1,-0.001),4*12)

##Plot the coefficients
plot(theta0)

##Generate the predictor variable
set.seed(13)
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)

##Simulate the response variable
y <- midas.sim(500,theta0,x,1)

##Remove unnecessary history of x
x <- window(x,start=start(y))

##Fit restricted model
mr <- midas_r(y~fmls(x,4*12-1,12,theta.h0)-1,list(y=y,x=x),
              start=list(x=c(-0.1,0.1,-0.1,-0.001)))

##Perform test (the expected result should be the acceptance of null)

hAh.test(mr)

##Fit using gradient function

##The gradient function
theta.h0.gradient<-function(p, dk,...) {
   i <- (1:dk-1)
   a <- exp(p[3]*i + p[4]*i^2)
   cbind(a, a*i, a*i*(p[1]+p[2]*i), a*i^2*(p[1]+p[2]*i))
}

mr <- midas_r(y~fmls(x,4*12-1,12,theta.h0)-1,list(y=y,x=x),
              start=list(x=c(-0.1,0.1,-0.1,-0.001)),
              user.gradient=TRUE)

##The test will use an user supplied gradient of weight function. See the
##help of midas_r on how to supply the gradient.

hAh.test(mr)
```

`hAhr.test` *Test restrictions on coefficients of MIDAS regression using robust version of the test*

Description

Perform a test whether the restriction on MIDAS regression coefficients holds.

Usage

```
hAhr.test(x, PHI = vcovHAC(x$unrestricted, sandwich = FALSE))
```

Arguments

`x`	MIDAS regression model with restricted coefficients, estimated with `midas_r`
`PHI`	the "meat" covariance matrix, defaults to `vcovHAC(x$unrestricted, sandwich=FALSE)`

Details

Given MIDAS regression:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + u_t$$

test the null hypothesis that the following restriction holds:

$$\theta_h = g(h, \lambda),$$

where $h = 0, ..., (k+1)m$.

Value

a `htest` object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Kvedaras V., Zemlys, V. *The statistical content and empirical testing of the MIDAS restrictions*

See Also

hAh.test

Examples

```
##The parameter function
theta.h0 <- function(p, dk, ...) {
   i <- (1:dk-1)
   (p[1] + p[2]*i)*exp(p[3]*i + p[4]*i^2)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,0.1,-0.1,-0.001),4*12)

##Plot the coefficients
plot(theta0)

##Generate the predictor variable
set.seed(13)
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)

##Simulate the response variable
y <- midas.sim(500,theta0,x,1)

##Remove unnecessary history of x
x <- window(x,start=start(y))

##Fit restricted model
mr <- midas_r(y~fmls(x,4*12-1,12,theta.h0)-1,
              list(y=y,x=x),
              start=list(x=c(-0.1,0.1,-0.1,-0.001)))

##The gradient function
theta.h0.gradient <-function(p, dk,...) {
   i <- (1:dk-1)
   a <- exp(p[3]*i + p[4]*i^2)
   cbind(a, a*i, a*i*(p[1]+p[2]*i), a*i^2*(p[1]+p[2]*i))
}

##Perform test (the expected result should be the acceptance of null)

hAhr.test(mr)

mr <- midas_r(y~fmls(x,4*12-1,12,theta.h0)-1,
              list(y=y,x=x),
              start=list(x=c(-0.1,0.1,-0.1,-0.001)),
              user.gradient=TRUE)

##Use exact gradient. Note the
hAhr.test(mr)
```

---

harstep *HAR(3)-RV model MIDAS weights specification*

Description

HAR(3)-RV model MIDAS weights specification

Usage

```
harstep(p, d, m)
```

Arguments

p	parameters for Almon lag
d	number of the coefficients
m	the frequency, currently ignored.

Details

MIDAS weights for Heterogeneous Autoregressive model of Realized Volatilty (HAR-RV). It is assumed that month has 20 days.

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Corsi, F., *A Simple Approximate Long-Memory Model of Realized Volatility*, Journal of Financial Econometrics Vol. 7 No. 2 (2009) 174-196

---

harstep.gradient *Gradient function for HAR(3)-RV model MIDAS weights specification*

Description

Gradient function for HAR(3)-RV model MIDAS weights specification

Usage

```
harstep.gradient(p, d, m)
```

Arguments

p	parameters for Almon lag
d	number of the coefficients
m	the frequency, currently ignored.

Details

MIDAS weights for Heterogeneous Autoregressive model of Realized Volatilty (HAR-RV). It is assumed that month has 20 days.

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Corsi, F., *A Simple Approximate Long-Memory Model of Realized Volatility*, Journal of Financial Econometrics Vol. 7 No. 2 (2009) 174-196

`hf_lags_table` *Create a high frequency lag selection table for MIDAS regression model*

Description

Creates a high frequency lag selection table for MIDAS regression model with given information criteria and minimum and maximum lags.

Usage

```
hf_lags_table(formula, data, start, from, to, IC = c("AIC", "BIC"),
  test = c("hAh.test"), Ofunction = "optim", user.gradient = FALSE, ...)
```

Arguments

`formula`	the formula for MIDAS regression, the lag selection is performed for the last MIDAS lag term in the formula
`data`	a list containing data with mixed frequencies
`start`	the starting values for optimisation
`from`	a named list, or named vector with lag numbers which are the beginings of MIDAS lag structures. The names should correspond to the MIDAS lag terms in the formula for which to do the lag selection. Value NA indicates lag start at zero
`to`	a named list where each element is a vector with two elements. The first element is the lag number from which the lag selection starts, the second is the lag number at which the lag selection ends. NA indicates lowest (highest) lag numbers possible.
`IC`	the information criteria which to compute

test	the names of statistical tests to perform on restricted model, p-values are reported in the columns of model selection table
Ofunction	see midasr
user.gradient	see midas_r
...	additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentially increasing the midas lag from kmin to kmax of the last term of the given formula

Value

a midas_r_iclagtab object which is the list with the following elements:

table	the table where each row contains calculated information criteria for both restricted and unrestricted MIDAS regression model with given lag structure
candlist	the list containing fitted models
IC	the argument IC

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)

mlr <- hf_lags_table(y~trend+fmls(x,12,12,nealmon),
                     start=list(x=rep(0,3)),
                     from=c(x=0),to=list(x=c(4,4)))
mlr
```

---

imidas_r *Restricted MIDAS regression with I(1) regressors*

Description

Estimate restricted MIDAS regression using non-linear least squares, when the regressor is I(1)

Usage

```
imidas_r(x, ...)

## Default S3 method:
imidas_r(x, data, start, Ofunction = "optim",
  user.gradient = FALSE, ...)
```

Arguments

x	either formula for restricted MIDAS regression or midas_r object. Formula must include fmls function
data	a named list containing data with mixed frequencies
start	the starting values for optimisation. Must be a list with named elements.
Ofunction	the list with information which R function to use for optimisation. The list must have element named Ofunction which contains character string of chosen R function. Other elements of the list are the arguments passed to this function. The default optimisation function is optim with argument method="BFGS". Other supported functions are nls
user.gradient	the default value is FALSE, which means that the numeric approximation of weight function gradient is calculated. If TRUE it is assumed that the R function for weight function gradient has the name of the weight function appended with .gradient. This function must return the matrix with dimensions $d_k \times q$, where d_k and q are the numbers of coefficients in unrestricted and restricted regressions correspondingly.
...	additional arguments supplied to optimisation function

Details

Given MIDAS regression:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + \mathbf{z_t}\beta + u_t$$

estimate the parameters of the restriction

$$\theta_h = g(h, \lambda),$$

where $h = 0, ..., (k+1)m$, together with coefficients β corresponding to additional low frequency regressors.

It is assumed that x is a I(1) process, hence the special transformation is made. After the transformation midas_r is used for estimation.

MIDAS regression involves times series with different frequencies.

The restriction function must return the restricted coefficients of the MIDAS regression.

Value

a midas_r object which is the list with the following elements:

coefficients	the estimates of parameters of restrictions
midas.coefficientas	the estimates of restricted coefficients of MIDAS regression
model	model data
weights	the restriction function(s) used in estimation.
unrestricted	unrestricted regression estimated using midas_u
param.map	parameter map for optimisation function
fn0	optimisation function for non-linear least squares problem solved in restricted MIDAS regression
rhs	the function which evaluates the right-hand side of the MIDAS regression
allcoef	the function which evaluates the restricted coefficients of MIDAS regression
opt	the output of optimisation procedure
argmap.opt	the list containing the name of optimisation function together with arguments for optimisation function
start.opt	the starting values used in optimisation

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

See Also

midas_r.midas_r

Examples

```
theta.h0 <- function(p, dk) {
  i <- (1:dk-1)/100
  pol <- p[3]*i + p[4]*i^2
 (p[1] + p[2]*i)*exp(pol)
}

theta0 <- theta.h0(c(-0.1,10,-10,-10),4*12)

xx <- simplearma.sim(list(ar=1),1500*12,1,12)
y <- midas.sim(500,theta0,xx,1)
x <- window(xx,start=start(y))

imr <- imidas_r(y~fmls(x,4*12-1,12,theta.h0)-1,start=list(x=c(-0.1,10,-10,-10)))
```

imidas_r.imidas_r *Restricted MIDAS regression with I(1) regressors*

Description

Reestimate the MIDAS regression with I(1) regressors with different starting values

Usage

```
## S3 method for class 'imidas_r'
imidas_r(x, start = coef(x),
  Ofunction = x$argmap.opt$Ofunction, ...)
```

Arguments

x	imidas_r object
start	the starting values
Ofunction	a character string of the optimisation function to use. The default value is to use the function of previous optimisation.
...	further arguments to optimisation function. If none are supplied, the arguments of previous optimisation are used.

Value

imidas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

See Also

imidas_r

lcauchyp *Normalized log-Cauchy probability density function MIDAS weights specification Calculate MIDAS weights according to normalized log-Cauchy probability density function specification*

Description

Normalized log-Cauchy probability density function MIDAS weights specification Calculate MIDAS weights according to normalized log-Cauchy probability density function specification

Usage

```
lcauchyp(p, d, m)
```

Arguments

p	parameters for normalized log-Cauchy probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

---

lcauchyp.gradient *Gradient function for normalized log-Cauchy probability density function MIDAS weights specification Calculate gradient function for normalized log-Cauchy probability density function specification of MIDAS weights.*

Description

Gradient function for normalized log-Cauchy probability density function MIDAS weights specification Calculate gradient function for normalized log-Cauchy probability density function specification of MIDAS weights.

Usage

```
lcauchyp.gradient(p, d, m)
```

Arguments

p	parameters for normalized log-Cauchy probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

`lf_lags_table` *Create a low frequency lag selection table for MIDAS regression model*

Description

Creates a low frequency lag selection table for MIDAS regression model with given information criteria and minimum and maximum lags.

Usage

```
lf_lags_table(formula, data, start, from, to, IC = c("AIC", "BIC"),
  test = c("hAh.test"), Ofunction = "optim", user.gradient = FALSE, ...)
```

Arguments

`formula`	the formula for MIDAS regression, the lag selection is performed for the last MIDAS lag term in the formula
`data`	a list containing data with mixed frequencies
`start`	the starting values for optimisation
`from`	a named list, or named vector with high frequency (NB!) lag numbers which are the beginnings of MIDAS lag structures. The names should correspond to the MIDAS lag terms in the formula for which to do the lag selection. Value NA indicates lag start at zero
`to`	a named list where each element is a vector with two elements. The first element is the low frequency lag number from which the lag selection starts, the second is the low frequency lag number at which the lag selection ends. NA indicates lowest (highest) lag numbers possible.
`IC`	the information criteria which to compute
`test`	the names of statistical tests to perform on restricted model, p-values are reported in the columns of model selection table
`Ofunction`	see midasr
`user.gradient`	see midas_r
`...`	additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentially increasing the midas lag from `kmin` to `kmax` of the last term of the given formula

Value

a `midas_r_ic_table` object which is the list with the following elements:

`table`	the table where each row contains calculated information criteria for both restricted and unrestricted MIDAS regression model with given lag structure
`candlist`	the list containing fitted models
`IC`	the argument IC

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)

mlr <- lf_lags_table(y~trend+fmls(x,12,12,nealmon),
                     start=list(x=rep(0,3)),
                     from=c(x=0),to=list(x=c(3,4)))
mlr
```

---

midas.auto.sim *Simulate autoregressive MIDAS model*

Description

Given the predictor variable, the weights and autoregressive coefficients calculate MIDAS regression response variable.

Usage

```
midas.auto.sim(n, theta, alpha, x, eps.sd, n.start = NA)
```

Arguments

`n`	sample size
`theta`	a vector with MIDAS weights for predictor variable
`alpha`	autoregressive coefficients
`x`	a high frequency predictor variable
`eps.sd`	the standard error of the regression disturbances, which are assumed to be independent normal zero mean random variables
`n.start`	number of observations to omit for the burn.in

Value

a ts object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
theta.h0 <- function(p, dk) {
  i <- (1:dk-1)/100
  pol <- p[3]*i + p[4]*i^2
  (p[1] + p[2]*i)*exp(pol)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,10,-10,-10),4*12)

##Generate the predictor variable
xx <- simplearma.sim(list(ar=0.6),3000*12,1,12)

y <- midas.auto.sim(500,theta0,c(0.5),xx,1,n.start=100)
x <- window(xx,start=start(y))
midas_r(y~mls(y,1,1)+fmls(x,4*12-1,12,theta.h0),start=list(x=c(-0.1,10,-10,-10)))
```

---

midas.sim *Simulate MIDAS regression response variable*

Description

Given the predictor variable and the coefficients calculate MIDAS regression response variable.

Usage

```
midas.sim(n, theta, x, eps.sd)
```

Arguments

n	The sample size
theta	a vector with MIDAS regression coefficients
x	a ts object with MIDAS regression predictor variable
eps.sd	the standard error of the regression disturbances, which are assumed to be independent normal zero mean random variables

Details

MIDAS regression has the following form:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + u_t$$

or alternatively

$$y_t = \sum_{h=0}^{(k+1)m} \theta_h x_{tm-h} + u_t,$$

where m is the frequency ratio and k is the number of lags included in the regression.

MIDAS regression involves times series with different frequencies. In R the frequency property is set when creating time series objects `ts`. Hence the frequency ratio m which figures in MIDAS regression is calculated from frequency property of time series objects supplied.

Value

a `ts` object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
##The parameter function
theta.h0 <- function(p, dk) {
   i <- (1:dk-1)/100
   pol <- p[3]*i + p[4]*i^2
   (p[1] + p[2]*i)*exp(pol)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,10,-10,-10),4*12)

##Plot the coefficients
plot(theta0)

##Generate the predictor variable
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)

##Simulate the response variable
y <- midas.sim(500,theta0,x,1)
```

midas_coef *Return the coefficients of MIDAS regression*

Description

A helper function for working with output of midas_r. Returns the regression coefficients.

Usage

```
midas_coef(x)
```

Arguments

x	an output from midas_r

Value

vector with coefficients of MIDAS regression

Author(s)

Vaidotas Zemlys

midas_r *Restricted MIDAS regression*

Description

Estimate restricted MIDAS regression using non-linear least squares.

Usage

```
midas_r(x, ...)

## Default S3 method:
midas_r(x, data, start, Ofunction = "optim",
  user.gradient = FALSE, ...)
```

Arguments

x	either formula for restricted MIDAS regression or midas_r object. Formula must include fmls function
data	a named list containing data with mixed frequencies
start	the starting values for optimisation. Must be a list with named elements.
Ofunction	the list with information which R function to use for optimisation. The list must have element named Ofunction which contains character string of chosen R function. Other elements of the list are the arguments passed to this function. The default optimisation function is optim with argument method="BFGS". Other supported functions are nls
user.gradient	the default value is FALSE, which means that the numeric approximation of weight function gradient is calculated. If TRUE it is assumed that the R function for weight function gradient has the name of the weight function appended with .gradient. This function must return the matrix with dimensions $d_k \times q$, where d_k and q are the numbers of coefficients in unrestricted and restricted regressions correspondingly.
...	additional arguments supplied to optimisation function

Details

Given MIDAS regression:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + \mathbf{z_t}\beta + u_t$$

estimate the parameters of the restriction

$$\theta_h = g(h, \lambda),$$

where $h = 0, ..., (k+1)m$, together with coefficients β corresponding to additional low frequency regressors.

MIDAS regression involves times series with different frequencies.

MIDAS-AR* (a model with a common factor, see (Clements and Galvao, 2008)) can be estimated by specifying additional argument, see an example.

The restriction function must return the restricted coefficients of the MIDAS regression.

Value

a midas_r object which is the list with the following elements:

coefficients	the estimates of parameters of restrictions
midas.coefficientas	the estimates of restricted coefficients of MIDAS regression
model	model data

weights	the MIDAS weights used in estimation.
unrestricted	unrestricted regression estimated using midas_u
param.map	parameter map for optimisation function
fn0	optimisation function for non-linear least squares problem solved in restricted MIDAS regression
rhs	the function which evaluates the right-hand side of the MIDAS regression
allcoef	the function which evaluates the restricted coefficients of MIDAS regression
opt	the output of optimisation procedure
argmap.opt	the list containing the name of optimisation function together with arguments for optimisation function
start.opt	the starting values used in optimisation
call	the call to the function
terms	terms object
gradient	gradient of NLS objective function
hessian	hessian of NLS objective function
Zenv	the environment in which data is placed
user.gradient	the value of supplied argument user.gradient

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Clements, M. and Galvao, A., *Macroeconomic Forecasting With Mixed-Frequency Data: Forecasting Output Growth in the United States*, Journal of Business and Economic Statistics, Vol.26 (No.4), (2008) 546-554

See Also

midas_r.midas_r

Examples

```
##The parameter function
theta.h0 <- function(p, dk, ...) {
   i <- (1:dk-1)/100
   pol <- p[3]*i + p[4]*i^2
   (p[1] + p[2]*i)*exp(pol)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,10,-10,-10),4*12)

##Plot the coefficients
plot(theta0)
```

```
##Generate the predictor variable
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)

##Simulate the response variable
y <- midas.sim(500,theta0,x,1)

##Remove unnecessary history of x
x <- window(x,start=start(y))

##Fit restricted model
mr <- midas_r(y~fmls(x,4*12-1,12,theta.h0)-1,
              list(y=y,x=x),
              start=list(x=c(-0.1,10,-10,-10)))

##Include intercept and trend in regression
mr.it <- midas_r(y~fmls(x,4*12-1,12,theta.h0)+trend,
                 list(data.frame(y=y,trend=1:500),x=x),
                 start=list(x=c(-0.1,10,-10,-10)))

data("USrealgdp")
data("USunempr")

y.ar <- diff(log(USrealgdp))
xx <- window(diff(USunempr), start = 1949)
trend <- 1:length(y.ar)

##Fit AR(1) model
mr.ar <- midas_r(y.ar ~ trend + mls(y.ar, 1, 1) +
                 fmls(xx, 11, 12, nealmon),
                 start = list(xx = rep(0, 3)))

##First order MIDAS-AR* restricted model
mr.arstar <-  midas_r(y.ar ~ trend + mls(y.ar, 1, 1, "*")
                      + fmls(xx, 11, 12, nealmon),
                      start = list(xx = rep(0, 3)))
```

`midas_r.fit` *Fit restricted MIDAS regression*

Description

Workhorse function for fitting restricted MIDAS regression

Usage

```
## S3 method for class 'fit'
midas_r(x)
```

Arguments

x	midas_r object

Value

midas_r object

Author(s)

Vaidotas Zemlys

---

midas_r.midas_r *Restricted MIDAS regression*

Description

Reestimate the MIDAS regression with different starting values

Usage

```
## S3 method for class 'midas_r'
midas_r(x, start = coef(x),
  Ofunction = x$argmap.opt$Ofunction, ...)
```

Arguments

x	midas_r object
start	the starting values
Ofunction	a character string of the optimisation function to use. The default value is to use the function of previous optimisation.
...	further arguments to optimisation function. If none are supplied, the arguments of previous optimisation are used.

Value

midas_r object

Author(s)

Vaidotas Zemlys

See Also

midas_r

`midas_r_ic_table` *Create a weight and lag selection table for MIDAS regression model*

Description

Creates a weight and lag selection table for MIDAS regression model with given information criteria and minimum and maximum lags.

Usage

```
midas_r_ic_table(formula, ...)

## Default S3 method:
midas_r_ic_table(formula, data = NULL, start = NULL,
  table, IC = c("AIC", "BIC"), test = c("hAh.test"), Ofunction = "optim",
  user.gradient = FALSE, showprogress = TRUE, ...)
```

Arguments

`formula`	the formula for MIDAS regression, the lag selection is performed for the last MIDAS lag term in the formula
`data`	a list containing data with mixed frequencies
`start`	the starting values for optimisation excluding the starting values for the last term
`table`	an wls_table object, see expand_weights_lags
`IC`	the names of information criteria which to compute
`test`	the names of statistical tests to perform on restricted model, p-values are reported in the columns of model selection table
`Ofunction`	see midasr
`user.gradient`	see midas_r
`showprogress`	logical, TRUE to show progress bar, FALSE for silent evaluation
`...`	additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentially increasing the midas lag from `kmin` to `kmax` and varying the weights of the last term of the given formula

Value

a `midas_r_ic_table` object which is the list with the following elements:

`table`	the table where each row contains calculated information criteria for both restricted and unrestricted MIDAS regression model with given lag structure
`candlist`	the list containing fitted models
`IC`	the argument IC

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)


mwlr <- midas_r_ic_table(y~trend+fmls(x,12,12,nealmon),
                  table=list(x=list(weights=
                  as.list(c("nealmon","nealmon","nbeta")),
                  lags=list(0:4,0:5,0:6),
                  starts=list(rep(0,3),rep(0,3,),c(1,1,1,0)))))

mwlr
```

`midas_r_np` *Estimate non-parametric MIDAS regression*

Description

Estimates non-parametric MIDAS regression

Usage

```
midas_r_np(x, data, lambda = NULL)
```

Arguments

`x`	formula specifying MIDAS regression
`data`	a named list containing data with mixed frequencies
`lambda`	smoothing parameter, defaults to `NULL`, which means that it is chosen by minimising AIC.

Details

Estimates non-parametric MIDAS regression accodring Breitung et al.

Value

a `midas_r_np` object

Author(s)

Vaidotas Zemlys

References

Breitung J, Roling C, Elengikal S (2013). *Forecasting inflation rates using daily data: A non-parametric MIDAS approach* Working paper, URL http://www.ect.uni-bonn.de/mitarbeiter/joerg-breitung/npmidas.

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)
midas_r_np(y~trend+fmls(x,12,12))
```

---

midas_r_simple *Restricted MIDAS regression*

Description

Function for fitting MIDAS regression without the formula interface

Usage

```
midas_r_simple(y, X, z = NULL, weight, grw = NULL, startx, startz = NULL,
  method = c("Nelder-Mead", "BFGS"), ...)
```

Arguments

y	model response
X	prepared matrix of high frequency variable lags
z	additional low frequency variables
weight	the weight function
grw	the gradient of weight function
startx	the starting values for weight function
startz	the starting values for additional low frequency variables
method	a method passed to optimx
...	additional parameters to optimx

Value

an object similar to midas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)

X<-fmls(x,11,12)

midas_r_simple(y,X,trend,weight=nealmon,startx=c(0,0,0))
```

`midas_u` *Estimate unrestricted MIDAS regression*

Description

Estimate unrestricted MIDAS regression using OLS. This function is a wrapper for `lm`.

Usage

```
midas_u(formula, data, ...)
```

Arguments

`formula`	MIDAS regression model formula
`data`	a named list containing data with mixed frequencies
`...`	further arguments, which could be passed to `lm` function.

Details

MIDAS regression has the following form:

$$y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x_{(t-j)m-i} + \mathbf{z_t}\beta + u_t$$

or alternatively

$$y_t = \sum_{h=0}^{(k+1)m} \theta_h x_{tm-h} + \mathbf{z_t}\beta + u_t,$$

where m is the frequency ratio and k is the number of lags included in the regression.

Given certain assumptions the coefficients can be estimated using usual OLS and they have the familiar properties associated with simple linear regression.

MIDAS regression involves times series with different frequencies.

Value

lm object.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Kvedaras V., Zemlys, V. *Testing the functional constraints on parameters in regressions with variables of different frequency* Economics Letters 116 (2012) 250-254

Examples

```
##The parameter function
theta.h0 <- function(p, dk, ...) {
   i <- (1:dk-1)/100
   pol <- p[3]*i + p[4]*i^2
   (p[1] + p[2]*i)*exp(pol)
}

##Generate coefficients
theta0 <- theta.h0(c(-0.1,10,-10,-10),4*12)

##Plot the coefficients
##Do not run
#plot(theta0)

##Generate the predictor variable
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)

##Simulate the response variable
y <- midas.sim(500,theta0,x,1)

##Create low frequency data.frame
ldt <- data.frame(y=y,trend=1:length(y))

##Create high frequency data.frame

hdt <- data.frame(x=window(x, start=start(y)))

##Fit unrestricted model
mu <- midas_u(y~fmls(x,2,12)-1, list(ldt, hdt))

##Include intercept and trend in regression

mu.it <- midas_u(y~fmls(x,2,12)+trend, list(ldt, hdt))

##Pass data as partialy named list

mu.it <- midas_u(y~fmls(x,2,12)+trend, list(ldt, x=hdt$x))
```

`mls` *MIDAS lag structure*

Description

Create a matrix of selected MIDAS lags

Usage

```
mls(x, k, m, ...)
```

Arguments

`x`	a vector
`k`	a vector of lag orders, zero denotes contemporaneous lag.
`m`	frequency ratio
`...`	further arguments used in fitting MIDAS regression

Details

The function checks whether high frequency data is complete, i.e. `m` must divide `length(x)`.

Value

a matrix containing the lags

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
## Quarterly frequency data
x <- 1:16
## Create MIDAS lag for use with yearly data
mls(x,0:3,4)

## Do not use contemporaneous lag
mls(x,1:3,4)

## Compares with embed when m=1
embed(x,2)
mls(x,0:1,1)
```

mls_coef *Return the coefficients for fmls variables*

Description

Extracts the coefficients and returns those coefficients which name has string `fmls` or `mls` in it.

Usage

```
mls_coef(x)
```

Arguments

`x`	an output from `midas_u`

Value

a vector

Author(s)

Vaidotas Zemlys

modsel *Select the model based on given information criteria*

Description

Selects the model with minimum of given information criteria and model type

Usage

```
modsel(x, IC = x$IC[1], test = x$test[1], type = c("restricted",
  "unrestricted"), print = TRUE)
```

Arguments

`x`	and output from iclagtab function
`IC`	the name of information criteria to base the choosing of the model
`test`	the name of the test for which to print out the p-value
`type`	the type of MIDAS model, either restricted or unrestricted
`print`	logical, if TRUE, prints the summary of the best model.

Details

This function selects the model from the model selection table for which the chosen information criteria achieves the smallest value. The function works with model tables produced by functions lf_lags_table, hf_lags_table, amidas_table and midas_r_ic_table.

Value

(invisibly) the best model based on information criteria, midas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)

mhfr <- hf_lags_table(y~trend+fmls(x,12,12,nealmon),
                      start=list(x=rep(0,3)),
                      from=list(x=0),to=list(x=c(4,6)))

mlfr <- lf_lags_table(y~trend+fmls(x,12,12,nealmon),
                      start=list(x=rep(0,3)),
                      from=list(x=0),to=list(x=c(2,3)))

modsel(mhfr,"BIC","unrestricted")

modsel(mlfr,"BIC","unrestricted")
```

---

nakagamip

Normalized Nakagami probability density function MIDAS weights specification Calculate MIDAS weights according to normalized Nakagami probability density function specification

---

Description

Normalized Nakagami probability density function MIDAS weights specification Calculate MIDAS weights according to normalized Nakagami probability density function specification

Usage

```
nakagamip(p, d, m)
```

Arguments

p	parameters for normalized Nakagami probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

---

nakagamip.gradient *Gradient function for normalized Nakagami probability density function MIDAS weights specification Calculate gradient function for normalized Nakagami probability density function specification of MIDAS weights.*

Description

Gradient function for normalized Nakagami probability density function MIDAS weights specification Calculate gradient function for normalized Nakagami probability density function specification of MIDAS weights.

Usage

```
nakagamip.gradient(p, d, m)
```

Arguments

p	parameters for normalized Nakagami probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Julius Vainora

`nbeta` *Normalized beta probability density function MIDAS weights specification Calculate MIDAS weights according to normalized beta probability density function specification*

Description

Normalized beta probability density function MIDAS weights specification Calculate MIDAS weights according to normalized beta probability density function specification

Usage

```
nbeta(p, d, m)
```

Arguments

`p`	parameters for normalized beta probability density function
`d`	number of coefficients
`m`	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

`nbeta.gradient` *Gradient function for normalized beta probability density function MIDAS weights specification Calculate gradient function for normalized beta probability density function specification of MIDAS weights.*

Description

Gradient function for normalized beta probability density function MIDAS weights specification Calculate gradient function for normalized beta probability density function specification of MIDAS weights.

Usage

```
nbeta.gradient(p, d, m)
```

Arguments

p	parameters for normalized beta probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

`nbetaMT` *Normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible) Calculate MIDAS weights according to normalized beta probability density function specification. Compatible with the specification in MATLAB toolbox.*

Description

Normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible) Calculate MIDAS weights according to normalized beta probability density function specification. Compatible with the specification in MATLAB toolbox.

Usage

```
nbetaMT(p, d, m)
```

Arguments

p	parameters for normalized beta probability density function
d	number of coefficients
m	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

`nbetaMT.gradient` *Gradient function for normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible) Calculate gradient function for normalized beta probability density function specification of MIDAS weights.*

Description

Gradient function for normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible) Calculate gradient function for normalized beta probability density function specification of MIDAS weights.

Usage

```
nbetaMT.gradient(p, d, m)
```

Arguments

`p`	parameters for normalized beta probability density function
`d`	number of coefficients
`m`	the frequency ratio, currently ignored

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

`nealmon` *Normalized Exponential Almon lag MIDAS coefficients*

Description

Calculate normalized exponential Almon lag coefficients given the parameters and required number of coefficients.

Usage

```
nealmon(p, d, m)
```

Arguments

p	parameters for Almon lag
d	number of the coefficients
m	the frequency, currently ignored.

Details

Given unrestricted MIDAS regression

$$y_t = \sum_{h=0}^{d} \theta_h x_{tm-h} + \mathbf{z_t}\beta + u_t$$

normalized exponential Almon lag restricts the coefficients $theta_h$ in the following way:

$$\theta_h = \delta \frac{\exp(\lambda_1(h+1) + \ldots + \lambda_r(h+1)^r)}{\sum_{s=0}^{d} \exp(\lambda_1(s+1) + \ldots + \lambda_r(h+1)^r)}$$

The parameter δ should be the first element in vector p. The degree of the polynomial is then decided by the number of the remaining parameters.

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
##Load data
data("USunempr")
data("USrealgdp")

y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
t <- 1:length(y)

midas_r(y~t+fmls(x,11,12,nealmon),start=list(x=c(0,0,0)))
```

nealmon.gradient *Gradient function for normalized exponential Almon lag weights*

Description

Gradient function for normalized exponential Almon lag weights

Usage

```
nealmon.gradient(p, d, m)
```

Arguments

p	hyperparameters for Almon lag
d	number of coefficients
m	the frequency ratio, currently ignored

Value

the gradient matrix

Author(s)

Vaidotas Zemlys

oos_prec *Out-of-sample prediction precision data on simulation example*

Description

The code in the example generates the out-of-sample prediction precision data for correctly and incorrectly constrained MIDAS regression model compared to unconstrained MIDAS regression model.

Format

A `data.frame` object with four columns. The first column indicates the sample size, the second the type of constraint, the third the value of the precision measure and the fourth the type of precision measure.

Examples

```
## Do not run:
## set.seed(1001)

## gendata<-function(n) {
##     trend<-c(1:n)
##     z<-rnorm(12*n)
##     fn.z <- nealmon(p=c(2,0.5,-0.1),d=17)
##     y<-2+0.1*trend+mls(z,0:16,12)%*%fn.z+rnorm(n)
##     list(y=as.numeric(y),z=z,trend=trend)
## }

## nn <- c(50,100,200,300,500,750,1000)

## data_sets <- lapply(n,gendata)

## mse <- function(x) {
##     mean(residuals(x)^2)
## }

## bnorm <- function(x) {
##     sqrt(sum((midas_coef(x)-c(2,0.1,nealmon(p=c(2,0.5,-0.1),d=17)))^2))
## }

## rep1 <- function(n) {
##     dt <- gendata(round(1.25*n))
##     ni <- n
##     ind <- 1:ni
##     mind <- 1:(ni*12)
##     indt<-list(y=dt$y[ind],z=dt$z[mind],trend=dt$trend[ind])
##     outdt <- list(y=dt$y[-ind],z=dt$z[-mind],trend=dt$trend[-ind])
##     um <- midas_r(y~trend+mls(z,0:16,12),data=indt,start=NULL)
##     nm <- midas_r(y~trend+mls(z,0:16,12,nealmon),data=indt,start=list(z=c(1,-1,0)))
##     am <- midas_r(y~trend+mls(z,0:16,12,almonp),data=indt,start=list(z=c(1,0,0,0)))
##     modl <- list(um,nm,am)
##     names(modl) <- c("um","nm","am")
##     list(norms=sapply(modl,bnorm),
##          mse=sapply(modl,function(mod)mean((forecast(mod,newdata=outdt)-outdt$y)^2)))
## }

## repr <- function(n,R) {
##     cc <- lapply(1:R,function(i)rep1(n))
##     list(norms=t(sapply(cc,"[[","norms")),mse=t(sapply(cc,"[[","mse")))
## }

## res <- lapply(nn,repr,R=1000)

## norms <- data.frame(nn,t(sapply(lapply(res,"[[","norms"),function(l)apply(l,2,mean))))
## mses <- data.frame(nn,t(sapply(lapply(res,"[[","mse"),function(l)apply(l,2,mean))))


## msd <- melt(mses[-1,],id=1)
```

```
## colnames(msd)[2] <- "Constraint"
## nmd <- melt(norms[-1,],id=1)
## colnames(nmd)[2] <- "Constraint"

## msd$Type <- "Mean squared error"
## nmd$Type <- "Distance from true values"
## oos_prec <- rbind(msd,nmd)
## oos_prec$Type <- factor(oos_prec$Type,levels=c("Mean squared error","Distance from true values"))
```

`polystep` *Step function specification for MIDAS weights*

Description

Step function specification for MIDAS weights

Usage

```
polystep(p, d, m, a)
```

Arguments

`p`	vector of parameters
`d`	number of coefficients
`m`	the frequency ratio, currently ignored
`a`	vector of increasing positive integers indicating the steps

Value

vector of coefficients

Author(s)

Vaidotas Zemlys

polystep.gradient *Gradient of step function specification for MIDAS weights*

Description

Gradient of step function specification for MIDAS weights

Usage

```
polystep.gradient(p, d, m, a)
```

Arguments

p	vector of parameters
d	number of coefficients
m	the frequency ratio, currently ignored
a	vector of increasing positive integers indicating the steps

Value

vector of coefficients

Author(s)

Vaidotas Zemlys

predict.midas_r *Predict method for MIDAS regression fit*

Description

Predicted values based on midas_r object.

Usage

```
## S3 method for class 'midas_r'
predict(object, newdata, na.action = na.omit, ...)
```

Arguments

object	midas_r object
newdata	a named list containing data for mixed frequencies. If omitted, the fitted values are used.
na.action	function determining what should be done with missing values in newdata. The most likely cause of missing values is the insufficient data for the lagged variables. The default is to omit such missing values.
...	additional arguments, not used

Details

predict.midas_r produces predicted values, obtained by evaluating regression function in the frame newdata. This means that the appropriate model matrix is constructed using only the data in newdata. This makes this function not very convenient for forecasting purposes. If you want to supply the new data for forecasting horizon only use the function forecast.midas_r. Also this function produces only static predictions, if you want dynamic forecasts use the forecast.midas_r.

Value

a vector of predicted values

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USrealgdp")
data("USunempr")

y <- diff(log(USrealgdp))
x <- window(diff(USunempr), start = 1949)

##24 high frequency lags of x included
mr <- midas_r(y ~ fmls(x, 23, 12, nealmon), start = list(x = rep(0, 3)))

##Declining unemployment
xn <- rnorm(2 * 12, -0.1, 0.1)

##Only one predicted value, historical values discarded
predict(mr, list(x = xn))

##Historical values taken into account
forecast(mr, list(x = xn))
```

---

prepmidas_r *Prepare necessary objects for fitting of the MIDAS regression*

Description

Prepare necessary objects for fitting of the MIDAS regression

Usage

```
prepmidas_r(y, X, mt, Zenv, cl, args, start, Ofunction, user.gradient,
  lagsTable, unrestricted = NULL, guess_start = TRUE)
```

Arguments

y	the response
X	the model matrix
mt	the terms of the formula
Zenv	the environment to evaluate the formula
cl	call of the function
args	additional argument
start	starting values
Ofunction	the optimisation function
user.gradient	see midas_r documentation
lagsTable	the lagstable from checkARstar
unrestricted	the unrestricted model
guess_start	if TRUE, get the initial values for non-MIDAS terms via OLS, if FALSE, initialize them with zero.

Author(s)

Vaidotas Zemlys

---

prep_hAh *Calculate data for hAh.test and hAhr.test*

Description

Workhorse function for calculating necessary matrices for hAh.test and hAhr.test. Takes the same parameters as hAh.test

Usage

```
prep_hAh(x)
```

Arguments

x	midas_r object

Value

a list with necessary matrices

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

See Also

hAh.test, hAhr.test

rvsp500 *Realized volatility of S&P500 index*

Description

Realized volatility of S&P500(Live) index of the period 2000 01 03 - 2013 11 22

Format

A `data.frame` object with two columns. First column contains date id, and the second the realized volatility for S&P500 index.

Source

http://realized.oxford-man.ox.ac.uk/media/1366/oxfordmanrealizedvolatilityindices.zip

References

Heber, Gerd and Lunde, Asger, and Shephard, Neil and Sheppard, Kevin *Oxford-Man Institute's realized library*, Oxford-Man Institute, University of Oxford (2009)

Examples

```
## Do not run:
## Download the data from
## http://realized.oxford-man.ox.ac.uk/media/1366/oxfordmanrealizedvolatilityindices.zip
## It contains the file OxfordManRealizedVolatilityIndices.csv.

## rvi <- read.csv("OxfordManRealizedVolatilityIndices.csv",check.names=FALSE,skip=2)
## ii <- which(rvi$DateID=="20131112")
## rvsp500 <- na.omit(rvi[1:ii,c("DataID","SPX2.rv")]
```

select_and_forecast *Create table for different forecast horizons*

Description

Creates tables for different forecast horizons and table for combined forecasts

Usage

```
select_and_forecast(formula, data, from, to, insample, outsample, weights,
  wstart, start = NULL, IC = "AIC", seltype = c("restricted",
  "unrestricted"), test = "hAh.test", ftype = c("fixed", "recursive",
  "rolling"), measures = c("MSE", "MAPE", "MASE"), fweights = c("EW",
  "BICW", "MSFE", "DMSFE"), ...)
```

Arguments

formula	initial formula for the
data	list of data
from	a named list of starts of lags from where to fit. Denotes the horizon
to	a named list for lag selections
insample	the low frequency indexes for in-sample data
outsample	the low frequency indexes for out-of-sample data
weights	names of weight function candidates
wstart	starting values for weight functions
start	other starting values
IC	name of information criteria to choose model from
seltype	argument to modsel, "restricted" for model selection based on information criteria of restricted MIDAS model, "unrestricted" for model selection based on unrestricted (U-MIDAS) model.
ftype	which type of forecast to use.
test	argument to modsel
measures	the names of goodness of fit measures
fweights	names of weighting schemes
...	additional arguments for optimisation method, see midas_r

Details

Divide data into in-sample and out-of-sample. Fit different forecasting horizons for in-sample data. Calculate accuracy measures for individual and average forecasts.

Value

a list containing forecasts, tables of accuracy measures and the list with selected models

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
### Sets a seed for RNG ###
set.seed(1001)
## Number of low-frequency observations
n<-250
## Linear trend and higher-frequency explanatory variables (e.g. quarterly and monthly)
trend<-c(1:n)
x<-rnorm(4*n)
z<-rnorm(12*n)
## Exponential Almon polynomial constraint-consistent coefficients
fn.x <- nealmon(p=c(1,-0.5),d=8)
```

```
fn.z <- nealmon(p=c(2,0.5,-0.1),d=17)
## Simulated low-frequency series (e.g. yearly)
y<-2+0.1*trend+mls(x,0:7,4)%*%fn.x+mls(z,0:16,12)%*%fn.z+rnorm(n)
##Do not run
## cbfc<-select_and_forecast(y~trend+mls(x,0,4)+mls(z,0,12),
## from=list(x=c(4,8,12),z=c(12,24,36)),
## to=list(x=rbind(c(14,19),c(18,23),c(22,27)),z=rbind(c(22,27),c(34,39),c(46,51))),
## insample=1:200,outsample=201:250,
## weights=list(x=c("nealmon","almonp"),z=c("nealmon","almonp")),
## wstart=list(nealmon=rep(1,3),almonp=rep(1,3)),
## IC="AIC",
## seltype="restricted",
## ftype="fixed",
## measures=c("MSE","MAPE","MASE"),
## fweights=c("EW","BICW","MSFE","DMSFE")
## )
```

`simplearma.sim` *Simulate AR(1) or MA(1) model*

Description

Simulate MIDAS regressor as a AR(1) or MA(1) time series

Usage

```
simplearma.sim(model, n, innov.sd, frequency, n.start = 300)
```

Arguments

`model`	A named vector of length one. Name is either "ar", or "ma" depending on which AR(1) or MA(1) process should be generated
`n`	the length of output series
`innov.sd`	the standard error of innovations, which are zero mean normal random variables
`frequency`	the frequency of the regressor, should be larger than one.
`n.start`	the length of the burn.in period, the default is 300.

Value

a time-series object of class `ts`

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
#Generate AR(1) model with rho=0.6, with frequency 12
x <- simplearma.sim(list(ar=0.6),1500*12,1,12)
```

split_data *Split mixed frequency data into in-sample and out-of-sample*

Description

Splits mixed frequency data into in-sample and out-of-sample datasets given the indexes of the low frequency data

Usage

```
split_data(data, insample, outsample)
```

Arguments

data	a list containing mixed frequency data
insample	the low frequency indexes for in-sample data
outsample	the low frequency indexes for out-of-sample data

Details

It is assumed that data is a list containing mixed frequency data. Then given the indexes of the low frequency data the function splits the data into two subsets.

Value

a list with elements indata and outdata containing respectively in-sample and out-of-sample data sets

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
#Monthly data
x <- 1:24
#Quartely data
z <- 1:8
#Yearly data
y <- 1:2
split_data(list(y=y,x=x,z=z),insample=1,outsample=2)
```

USpayems *United States total employment non-farms payroll, monthly, seasonally adjusted.*

Description

United States total employment non-farms payroll, monthly, seasonally adjusted. Retrieved from FRED, symbol "PAYEMS" at 2014-04-25.

Format

A ts object.

Source

FRED, Federal Reserve Economic Data, from the Federal Reserve Bank of St. Louis

Examples

```
## Do not run:
## library(quantmod)
## USpayems <- ts(getSymbols("PAYEMS",src="FRED",auto.assign=FALSE),start=c(1939,1),frequency=12)
```

USqgdp *United States gross domestic product, quarterly, seasonaly adjusted annual rate.*

Description

United States gross domestic product, quarterly, seasonaly adjusted annual rate. Retrieved from FRED, symbol "GDP" at 2014-04-25.

Format

A ts object.

Source

FRED, Federal Reserve Economic Data, from the Federal Reserve Bank of St. Louis

Examples

```
## Do not run:
## library(quantmod)
## USqgdp <- ts(getSymbols("GDP",src="FRED",auto.assign=FALSE),start=c(1947,1),frequency=4)
```

USrealgdp — *US annual gross domestic product in billions of chained 2005 dollars*

Description

The annual gross domestic product in billions of chained 2005 dollars for US from 1948 to 2011.

Format

A ts object.

Source

U.S. Department of Commerce, Bureau of Economic Analysis

USunempr — *US monthly unemployment rate*

Description

The monthly unemployment rate for United States from 1948 to 2011.

Format

A ts object.

Source

U.S. Bureau of Labor Statistics

weights_table — *Create a weight function selection table for MIDAS regression model*

Description

Creates a weight function selection table for MIDAS regression model with given information criteria and weight functions.

Usage

```
weights_table(formula, data, start = NULL, IC = c("AIC", "BIC"),
  test = c("hAh.test"), Ofunction = "optim", user.gradient = FALSE, ...)
```

Arguments

formula	the formula for MIDAS regression, the lag selection is performed for the last MIDAS lag term in the formula
data	a list containing data with mixed frequencies
start	the starting values for optimisation
IC	the information criteria which to compute
test	the names of statistical tests to perform on restricted model, p-values are reported in the columns of model selection table
Ofunction	see midasr
user.gradient	see midas_r
...	additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentially increasing the midas lag from kmin to kmax of the last term of the given formula

Value

a midas_r_ic_table object which is the list with the following elements:

table	the table where each row contains calculated information criteria for both restricted and unrestricted MIDAS regression model with given lag structure
candlist	the list containing fitted models
IC	the argument IC

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```
data("USunempr")
data("USrealgdp")
y <- diff(log(USrealgdp))
x <- window(diff(USunempr),start=1949)
trend <- 1:length(y)
mwr <- weights_table(y~trend+fmls(x,12,12,nealmon),
                     start=list(x=list(nealmon=rep(0,3),
                     nbeta=c(1,1,1,0))))

mwr
```

weight_coef *Return the restricted coefficients generated by restriction function(s)*

Description

A helper function for working with output of midas_r. Returns the restricted coefficients generated by restriction function(s)

Usage

```
weight_coef(x, name = weight_names(x))
```

Arguments

x	an output from midas_r
name	name(s) of the restriction function(s), the default value is the name(s) of the restriction function(s) supplied to midas_r

Value

a list if length(name)>1, a vector otherwise

Author(s)

Vaidotas Zemlys

weight_names *Return the names of restriction function(s)*

Description

A helper function for working with output of midas_r. Returns the name(s) of restriction function(s) used in call to midas_r

Usage

```
weight_names(x)
```

Arguments

x	an output from midas_r

Value

a character vector

Author(s)

Vaidotas Zemlys

`weight_param` *Return the estimated hyper parameters of the weight function(s)*

Description

A helper function for working with output of `midas_r`. Returns the estimated parameters of restriction function(s)

Usage

```
weight_param(x, name = weight_names(x))
```

Arguments

`x`	an output from `midas_r`
`name`	name of the restriction function, the default value is the names of the restriction functions supplied to `midas_r`

Value

a list if `length(name)>1`, a vector otherwise

Author(s)

Vaidotas Zemlys

Index