Package ‘midasr’

February 23, 2021

Title Mixed Data Sampling Regression

Description Methods and tools for mixed frequency time series data analysis. Allows estimation, model selection and forecasting for MIDAS regressions.

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

Version 0.8

Maintainer Vaidotas Zemlys-Balevičius <zemlys@gmail.com>

Author Virmantas Kvedaras <virmantas.kvedaras@ec.europa.eu>, Vaidotas Zemlys-Balevicius <zemlys@gmail.com>

Depends R (>= 2.11.0), sandwich, optimx, quantreg

Imports MASS, numDeriv, Matrix, forecast, zoo, stats, graphics, utils, Formula, texreg, methods

License GPL-2 | MIT + file LICENCE

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

Suggests testthat, lubridate, xts

RoxygenNote 7.1.1

Encoding UTF-8

Collate 'deriv.R' 'imidasreg.R' 'lagspec.R' 'midas_nlpr.R'
  'midas_r_methods.R' 'midas_nlpr_methods.R' 'midas_qr_methods.R'
  'midas_sp.R' 'midas_sp_methods.R' 'midaslag.R' 'midasqr.R'
  'midasr-package.R' 'midasreg.R' 'modsel.R' 'nonparametric.R'
  'simulate.R' 'tests.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2021-02-23 09:40:05 UTC

R topics documented:

midasr-package .......................................................... 4
+.lws_table ............................................................ 4
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>agk.test</td>
</tr>
<tr>
<td>almonp</td>
</tr>
<tr>
<td>almonp_gradient</td>
</tr>
<tr>
<td>amidas_table</td>
</tr>
<tr>
<td>amweights</td>
</tr>
<tr>
<td>average_forecast</td>
</tr>
<tr>
<td>check_mixfreq</td>
</tr>
<tr>
<td>coef.midas_nlpr</td>
</tr>
<tr>
<td>coef.midas_r</td>
</tr>
<tr>
<td>coef.midas_sp</td>
</tr>
<tr>
<td>deriv_tests</td>
</tr>
<tr>
<td>deviance.midas_nlpr</td>
</tr>
<tr>
<td>deviance.midas_r</td>
</tr>
<tr>
<td>deviance.midas_sp</td>
</tr>
<tr>
<td>dmls</td>
</tr>
<tr>
<td>expand_amidas</td>
</tr>
<tr>
<td>expand_weights_lags</td>
</tr>
<tr>
<td>extract.midas_r</td>
</tr>
<tr>
<td>fitted.midas_nlpr</td>
</tr>
<tr>
<td>fitted.midas_sp</td>
</tr>
<tr>
<td>fmls</td>
</tr>
<tr>
<td>forecast.midas_r</td>
</tr>
<tr>
<td>genexp</td>
</tr>
<tr>
<td>genexp_gradient</td>
</tr>
<tr>
<td>get_estimation_sample</td>
</tr>
<tr>
<td>gompertzp</td>
</tr>
<tr>
<td>gompertzp_gradient</td>
</tr>
<tr>
<td>hAh_r_test</td>
</tr>
<tr>
<td>hAh_test</td>
</tr>
<tr>
<td>harstep</td>
</tr>
<tr>
<td>harstep_gradient</td>
</tr>
<tr>
<td>hf_lags_table</td>
</tr>
<tr>
<td>imidas_r</td>
</tr>
<tr>
<td>lcauchyp</td>
</tr>
<tr>
<td>lcauchyp_gradient</td>
</tr>
<tr>
<td>lf_lags_table</td>
</tr>
<tr>
<td>lstr</td>
</tr>
<tr>
<td>midas_auto_sim</td>
</tr>
<tr>
<td>midas_lstr_plain</td>
</tr>
<tr>
<td>midas_lstr_sim</td>
</tr>
<tr>
<td>midas_mmm_plain</td>
</tr>
<tr>
<td>midas_mmm_sim</td>
</tr>
<tr>
<td>midas_nlpr</td>
</tr>
<tr>
<td>midas_nlpr.fit</td>
</tr>
<tr>
<td>midas_pl_plain</td>
</tr>
<tr>
<td>midas_pl_sim</td>
</tr>
<tr>
<td>midas_qr</td>
</tr>
<tr>
<td>midas_r</td>
</tr>
</tbody>
</table>
R topics documented:

midas_r.fit ................................................................. 57
midas_r_ic_table .................................................... 58
midas_r_np ................................................................. 60
midas_r_plain ............................................................ 61
midas_sim ................................................................. 62
midas_si_plain .......................................................... 63
midas_si_sim .............................................................. 64
midas_sp ................................................................. 65
midas_u ................................................................. 67
mls .................................................................. 69
mlsd ................................................................. 70
mmm ................................................................. 71
modsel ................................................................. 71
nakagamip ....................................................... 72
nakagamip_gradient ............................................. 73
nbeta ............................................................... 74
nbetaMT ............................................................. 74
nbetaMT_gradient ................................................... 75
nbeta_gradient .......................................................... 76
nealmon ............................................................... 76
nealmon_gradient .................................................. 77
oos_prec .............................................................. 78
plot_lstr ................................................................. 79
plot_midas_coef ....................................................... 80
plot_midas_coef.midas_nlpr ....................................... 81
plot_sp ................................................................. 82
polystep ............................................................... 83
polystep_gradient .................................................. 84
predict.midas_nlpr .................................................... 84
predict.midas_r .......................................................... 85
predict.midas_sp .......................................................... 86
prep_hAh ................................................................. 87
rvsp500 ................................................................. 88
select_and_forecast .................................................. 88
simulate.midas_r ....................................................... 90
split_data ............................................................... 92
update_weights ....................................................... 93
UScpiqs ................................................................. 94
USEffrw ................................................................. 94
USpayems .............................................................. 94
USqgdp ................................................................. 95
USrealgdp .............................................................. 95
USunempr .............................................................. 96
weights_table .......................................................... 96

Index ................................................................. 98
midasr-package

Mixed Data Sampling Regression

Description

Package for estimating, testing and forecasting MIDAS regression.

Details

Methods and tools for mixed frequency time series data analysis. Allows estimation, model selection and forecasting for MIDAS regressions.

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 rbinds them.

Value

lws_table object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
### Examples

```r
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
```

---

**Description**

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

**Usage**

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

**Arguments**

- `x` MIDAS regression object of class `midas_r`

**Value**

A htest object

**Author(s)**

Virmantas Kvedaras, Vaidotas Zemlys

**References**


**Examples**

```r
## 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)
```
almonp  

*Almon polynomial MIDAS weights specification*

**Description**

Calculate Almon polynomial MIDAS weights

**Usage**

```r
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**

```r
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
amidas_table

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**

```r
amidas_table(
  formula,
  data,
  weights,
  wstart,
  type,
  start = NULL,
  from,
  to,
  IC = c("AIC", "BIC"),
  test = c("hAh_test"),
  Ofunction = "optim",
  weight_gradients = NULL,
  ...
)
```

**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 first 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

weight_gradients
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 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))
Description

Produces weights for aggregates based MIDAS regression

Usage

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

Arguments

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

Details

Suppose a weight function \( w(\beta, \theta) \) satisfies the following equation:

\[
   w(\beta, \theta) = \beta g(\theta)
\]

The following combinations are defined, corresponding to structure types A, B and C respectively:

- \((w(\beta_1, \theta_1), ..., w(\beta_k, \theta_k))\)
- \((w(\beta_1, \theta), ..., w(\beta_k, \theta))\)
- \(\beta(w(1, \theta), ..., w(1, \theta))\),

where \(k\) is the number of low frequency lags, i.e. \(d/m\). If the latter value is not whole number, the error is produced.

The starting values \(p\) should be supplied then as follows:

- \((\beta_1, \theta_1), ..., \beta_k, \theta_k)\)
- \((\beta_1, ..., \beta_k, \theta)\)
- \((\beta, \theta)\)

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"),  
show_progress = 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  
show_progress 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**

```r
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"))
```

**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**

```r
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

c coef.midas_nlpr

Details
MIDAS regression has two sets of coefficients. The first set is the coefficients associated with the
parameters of weight functions associated with MIDAS regression terms. These are the coefficients
of the NLS problem associated with MIDAS regression. The second is the coefficients of the linear
model, i.e the values of weight functions of terms, or so called MIDAS coefficients. By default the
function returns the first set of the coefficients.

Value
a vector with coefficients

Author(s)
Vaidotas Zemlys
coef.midas_r

Extract coefficients of MIDAS regression

Description

Extracts various coefficients of MIDAS regression

Usage

## S3 method for class 'midas_r'
coef(object, midas = FALSE, term_names = NULL, ...)

Arguments

object midas_r object
midas logical, if TRUE, MIDAS coefficients are returned, if FALSE (default), coefficients of NLS problem are returned
term_names a character vector with term names. Default is NULL, which means that coefficients of all the terms are returned
... not used currently

Details

MIDAS regression has two sets of coefficients. The first set is the coefficients associated with the parameters of weight functions associated with MIDAS regression terms. These are the coefficients of the NLS problem associated with MIDAS regression. The second is the coefficients of the linear model, i.e. the values of weight functions of terms, or so called MIDAS coefficients. By default the function returns the first set of the coefficients.

Value

a vector with coefficients

Author(s)

Vaidotas Zemlys

Examples

#Simulate MIDAS regression
n<-250
trend<-c(1:n)
x<-rnorm(4*n)
z<-rnorm(12*n)
fn.x <- nealmon(p=c(1,-0.5),d=8)
fn.z <- nealmon(p=c(2,0.5,-0.1),d=17)
y<-2+0.1*trend+mls(x,0:7,4)*%*%fn.x+mls(z,0:16,12)*%*%fn.z+rnorm(n)
eqr<-midas_r(y ~ trend + mls(x, 0:7, 4, nealmon) +
    mls(z, 0:16, 12, nealmon),
    start = list(x = c(1, -0.5), z = c(2, 0.5, -0.1)))

coeff(eqr)
coeff(eqr, term_names = "x")
coeff(eqr, midas = TRUE)
coeff(eqr, midas = TRUE, term_names = "x")

description
Extract coefficients of MIDAS regression

Usage
## S3 method for class 'midas_sp'
coef(object, type = c("plain", "midas", "bw"), term_names = NULL, ...)

Arguments
  object midas_nlpr object
  type one of plain, midas, or nlpr. Returns appropriate coefficients.
  term_names a character vector with term names. Default is NULL, which means that coefficients of all the terms are returned
  ... not used currently

Details
MIDAS regression has two sets of coefficients. The first set is the coefficients associated with the parameters of weight functions associated with MIDAS regression terms. These are the coefficients of the NLS problem associated with MIDAS regression. The second is the coefficients of the linear model, i.e. the values of weight functions of terms, or so called MIDAS coefficients. By default the function returns the first set of the coefficients.

Value
a vector with coefficients

Author(s)
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_nlpr Non-linear parametric MIDAS regression model deviance

Description

Returns the deviance of a fitted MIDAS regression object

Usage

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

Arguments

object  a midas_r object
...

Value

The sum of squared residuals

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

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
...

Value

The sum of squared residuals

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
deviance.midas_sp  

Semi-parametric MIDAS regression model deviance

Description

Returns the deviance of a fitted MIDAS regression object

Usage

```r
## S3 method for class 'midas_sp'
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

```r
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**

```r
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 \(k_{\text{min}}\) to \(k_{\text{max}}\). 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**

```r
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
```
**extract.midas_r**

*Extract coefficients and GOF measures from MIDAS regression object*

**Description**

Extract coefficients and GOF measures from MIDAS regression object

**Usage**

```r
extract.midas_r(
  model, 
  include.rsquared = TRUE, 
  include.nobs = TRUE, 
  include.rmse = TRUE, 
  ...
)
```

**Arguments**

- `model` a MIDAS regression object
- `include.rsquared` If available: should R-squared be reported?
- `include.nobs` If available: should the number of observations be reported?
- `include.rmse` If available: should the root-mean-square error (= residual standard deviation) be reported?
- `...` additional parameters passed to summary

**Value**

texreg object

**Author(s)**

Virmantas Kvedaras, Vaidotas Zemlys

---

**fitted.midas_nlpr**

*Fitted values for non-linear parametric MIDAS regression model*

**Description**

Returns the fitted values of a fitted non-linear parametric MIDAS regression object

**Usage**

```r
## S3 method for class 'midas_nlpr'
fitted(object, ...)
```
fitted.midas_sp

Arguments

- `object` a `midas_r` object
- `...` currently nothing

Value

the vector of fitted values

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

---

fitted.midas_sp  Fitted values for semi-parametric MIDAS regression model

Description

Returns the fitted values of a fitted semi-parametric MIDAS regression object

Usage

```r
## S3 method for class 'midas_sp'
fitted(object, ...)
```

Arguments

- `object` a `midas_r` object
- `...` currently nothing

Value

the vector of fitted values

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
**Description**

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

**Usage**

```r
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`

---

**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

```r
## S3 method for class 'midas_r'
forecast(
  object,
  newdata = NULL,
  se = FALSE,
  level = c(80, 95),
  fan = FALSE,
  npaths = 999,
  method = c("static", "dynamic"),
  insample = get_estimation_sample(object),
  show_progress = TRUE,
  add_ts_info = FALSE,
  ...
)
```

Arguments

- **object**: midas_r object
- **newdata**: a named list containing future values of mixed frequency regressors. The default is `NULL`, meaning that only in-sample data is used.
- **se**: logical, if `TRUE`, the prediction intervals are calculated.
- **level**: confidence level for prediction intervals
- **fan**: if `TRUE`, level is set to `seq(50,99,by=1)`. This is suitable for fan plots
- **npaths**: the number of samples for simulating prediction intervals
- **method**: the forecasting method, either "static" or "dynamic"
- **insample**: a list containing the historic mixed frequency data
- **show_progress**: logical, if `TRUE`, the progress bar is shown if `se = TRUE`
- **add_ts_info**: logical, if `TRUE`, the forecast is cast as `ts` object. Some attempts are made to guess the correct start, by assuming that the response variable is a `ts` object of frequency 1. If `FALSE`, then the result is simply a numeric vector.

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

an object of class "forecast", a list containing following elements:
- **method**: the name of forecasting method: MIDAS regression, static or dynamic
- **model**: original object of class `midas_r`
- **mean**: point forecasts
lower limits for prediction intervals
upper limits for prediction intervals
fitted values, one-step forecasts
residuals from the fitted model
the original response variable

The methods print, summary and plot from package forecast can be used on the object.

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*h)
##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")

##Use print, summary and plot methods from package forecast
fmr <- forecast(mr, list(trend = trendn, x = xn), method = "static")
fmr
summary(fmr)
plot(fmr)
Description

Calculates the MIDAS coefficients for generalized exponential MIDAS lag specification

Usage

genexp(p, d, m)

Arguments

p  a vector of parameters
d  number of coefficients
m  the frequency, currently ignored

Details

Generalized exponential MIDAS lag specification is a generalization of exponential Almon lag. It is defined as a product of first order polynomial with exponent of the second order polynomial. This specification was used by V. Kvedaras and V. Zemlys (2012).

Value

a vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

Description

Calculates the gradient of generalized exponential MIDAS lag specification

Usage

genexp_gradient(p, d, m)

Arguments

p a vector of parameters
d number of coefficients
m the frequency, currently ignored

Details

Generalized exponential MIDAS lag specification is a generalization of exponential Almon lag. It is defined as a product of first order polynomial with exponent of the second order polynomial. This specification was used by V. Kvedaras and V. Zemlys (2012).

Value

a vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References

get_estimation_sample

Get the data which was used to estimate 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

Description

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
**gompertzp_gradient**

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

**Description**

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

**Usage**

```r
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
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)
xx <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)

## Simulate the response variable
y <- midas_sim(500, xx, theta0)

x <- window(xx, 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)
hAh_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)),
  weight_gradients=list())

## Use exact gradient. Note the
hAh_test(mr)
**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 \( h = 0, \ldots, (k + 1)m \).

**Value**

a htest object

**Author(s)**

Virmantas Kvedaras, Vaidotas Zemlys

**References**


**See Also**

hAhr_test

**Examples**

```r
##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 <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)

## Simulate the response variable
y <- midas_sim(500, xx, theta0)
x <- window(xx, 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)),
               weight_gradients=list())

# 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)

---

**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 Volatility (HAR-RV). It is assumed that month has 20 days.

Value
vector of coefficients

Author(s)
Virmantas Kvedaras, Vaidotas Zemlys

References

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 Volatility (HAR-RV). It is assumed that month has 20 days.
Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References


---

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",
  weight_gradients = NULL,
  ...
)

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

weight_gradients 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)),
                      data = list(y=y, x=x, trend = trend),
                      from=c(x=0),to=list(x=c(4,4)))
mlr
**Description**

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

**Usage**

```r
imidas_r(
  formula,
  data,
  start,
  Ofunction = "optim",
  weight_gradients = NULL,
  ...
)
```

**Arguments**

- `formula` : formula for restricted MIDAS regression. 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`
- `weight_gradients` : a named list containing gradient functions of weights. The weight gradient function must return the matrix with dimensions `d_k × q`, where `d_k` and `q` are the number of coefficients in unrestricted and restricted regressions correspondingly. The names of the list should coincide with the names of weights used in formula. The default value is NULL, which means that the numeric approximation of weight function gradient is calculated. If the argument is not NULL, but the name of the weight used in formula is not present, it is assumed that there exists an R function which has the name of the weight function appended with `.gradient`
- `...` : additional arguments supplied to optimisation function

**Details**

Given MIDAS regression:

\[ y_t = \sum_{j=0}^{k} \sum_{i=0}^{m-1} \theta_{jm+i} x(t-j)m-i + z_t \beta + u_t \]
estimate the parameters of the restriction

\[ \theta_h = g(h, \lambda), \]

where \( h = 0, ..., (k + 1)m \), together with coefficients \( \beta \) 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 \texttt{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 \texttt{midas_r} object which is the list with the following elements:

- **coefficients** the estimates of parameters of restrictions
- **midas_coefficients** the estimates of MIDAS coefficients of MIDAS regression
- **model** model data
- **unrestricted** unrestricted regression estimated using \texttt{midas_u}
- **term_info** the named list. Each element is a list with the information about the term, such as its frequency, function for weights, gradient function of weights, etc.
- **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
- **gen_midas_coef** the function which generates the MIDAS 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
- **start_list** the starting values as a list
- **call** the call to the function
- **terms** terms object
- **gradient** gradient of NLS objective function
- **hessian** hessian of NLS objective function
- **gradD** gradient function of MIDAS weight functions
- **Zenv** the environment in which data is placed
- **use_gradient** TRUE if user supplied gradient is used, FALSE otherwise
- **nobs** the number of effective observations
- **convergence** the convergence message
- **fitted.values** the fitted values of MIDAS regression
- **residuals** the residuals of MIDAS regression
Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

See Also

midas_r

Examples

```r
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 <- ts(cumsum(rnorm(600*12)), frequency = 12)

## Simulate the response variable
y <- midas_sim(500, xx, theta0)

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)))
```

Description

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

Usage

```r
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
**lcauchyp_gradient**

*Author(s)*

Julius Vainora

---

**Description**

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

```r
lf_lags_table(
    formula, 
    data, 
    start, 
    from, 
    to, 
    IC = c("AIC", "BIC"), 
    test = c("hAh_test"), 
    Ofunction = "optim", 
    weight_gradients = NULL, 
    ...
)
```

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`
- `weight_gradients`: see `midas_r`
- `...`: additional parameters to optimisation function, see `midas_r`

Details

This function estimates models sequentially increasing the midas lag from $k_{min}$ to $k_{max}$ 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
lstr

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

lstr

Compute LSTR term for high frequency variable

Description

Compute LSTR term for high frequency variable

Usage

lstr(X, theta, beta, sd_x = sd(c(X), na.rm = TRUE))

Arguments

X matrix, high frequency variable embedded in low frequency, output of mls
theta vector, restriction coefficients for high frequency variable
beta vector of length 4, parameters for LSTR term, slope and 3 LSTR parameters
sd_x vector of length 1, defaults to standard deviation of X.

Value

a vector
midas_auto_sim

Simulate simple autoregressive MIDAS model

Description

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

Usage

midas_auto_sim(
  n, 
  alpha, 
  x, 
  theta, 
  rand_gen = rnorm, 
  innov = rand_gen(n, ...), 
  n_start = NA, 
  ... 
)

Arguments

  n                sample size. 
  alpha            autoregressive coefficients. 
  x                a high frequency predictor variable. 
  theta            a vector with MIDAS weights for predictor variable. 
  rand_gen         a function to generate the innovations, default is the normal distribution. 
  innov            an optional time series of innovations. 
  n_start          number of observations to omit for the burn.in. 
  ...              additional arguments to function rand_gen. 

Value

  a ts object

Author(s)

  Virmantas Kvedaras, Vaidotas Zemlys
Examples

```r
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 <- ts(arima.sim(model = list(ar = 0.6), 1000 * 12), frequency = 12)

y <- midas_auto_sim(500, 0.5, xx, theta0, n_start = 200)
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_lstr_plain

**LSTR (Logistic Smooth TRansition) MIDAS regression**

Description

Function for fitting LSTR MIDAS regression without the formula interface

Usage

```r
midas_lstr_plain(
  y,
  X,
  z = NULL,
  weight,
  start_lstr,
  start_x,
  start_z = NULL,
  method = c("Nelder-Mead"),
  ...
)
```

Arguments

- `y` model response
- `X` prepared matrix of high frequency variable lags for LSTR term
- `z` additional low frequency variables
- `weight` the weight function
- `start_lstr` the starting values for lstr term
- `start_x` the starting values for weight function
midas_lstr_sim

start_z the starting values for additional low frequency variables
method a method passed to optim
... additional parameters to optim

Value

an object similar to midas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

midas_lstr_sim  Simulate LSTR MIDAS regression model

Description

Simulate LSTR MIDAS regression model

Usage

midas_lstr_sim(
  n,
  m,
  theta,
  intercept,
  plstr,
  ar.x,
  ar.y,
  rand.gen = rnorm,
  n.start = NA,
  ...
)

Arguments

n number of observations to simulate.
m integer, frequency ratio
theta vector, restriction coefficients for high frequency variable
intercept vector of length 1, intercept for the model.
plstr vector of length 4, slope for the LSTR term and LSTR parameters
ar.x vector, AR parameters for simulating high frequency variable
ar.y vector, AR parameters for AR part of the model
rand.gen function, a function for generating the regression innovations, default is rnorm
n.start integer, length of a 'burn-in' period. If NA, the default, a reasonable value is computed.
... additional parameters to rand.gen
Value

a list

Examples

```r
nnbeta <- function(p, k) nbeta(c(1,p),k)

dgp <- midas_lstr_sim(250, m = 12, theta = nnbeta(c(2, 4), 24),
  intercept = c(1), plstr = c(1.5, 1, log(1), 1),
  ar.x = 0.9, ar.y = 0.5, n.start = 100)
z <- cbind(1, mls(dgp$y, 1:2, 1))
colnames(z) <- c("Intercept", "y1", "y2")
X <- mls(dgp$x, 0:23, 12)
lstr_mod <- midas_lstr_plain(dgp$y, X, z, nnbeta,
  start_lstr = c(1.5, 1, 1, 1),
  start_x = c(2, 4), start_z = c(1, 0.5, 0))
coef(lstr_mod)
```

---

**Description**

Function for fitting MMM MIDAS regression without the formula interface

**Usage**

```r
midas_mmm_plain(  
y, 
X, 
z = NULL,  
weight,  
start_mmm,  
start_x,  
start_z = NULL,  
method = c("Nelder-Mead"),  
...  
)
```

**Arguments**

- `y` model response
- `X` prepared matrix of high frequency variable lags for MMM term
- `z` additional low frequency variables
midas_mmm_sim

weight
start_mmm
start_x
start_z
method
... 

Value

an object similar to midas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

midas_mmm_sim Simulate MMM MIDAS regression model

Description

Simulate MMM MIDAS regression model

Usage

midas_mmm_sim(
  n,
  m,
  theta,
  intercept,
  pmmm,
  ar.x,
  ar.y,
  rand.gen = rnorm,
  n.start = NA,
  ...
)

Arguments

n number of observations to simulate.
m integer, frequency ratio
theta vector, restriction coefficients for high frequency variable
intercept vector of length 1, intercept for the model.
pmmm vector of length 2, slope for the MMM term and MMM parameter
midas_nlpr

Description

Estimate restricted MIDAS regression using non-linear least squares.

Usage

midas_nlpr(formula, data, start, Ofunction = "optim", ...)

Arguments

formula 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 arguments method="Nelder-Mead" and control=list(maxit=5000). Other supported functions are nls, optimx.

... additional arguments supplied to optimisation function

Details

Given MIDAS regression:

\[ y_t = \sum_{j=1}^{p} \alpha_j y_{t-j} + \sum_{i=0}^{k} \sum_{j=0}^{l_i} \beta_j^{(i)} x_{tm_i-j}^{(i)} + u_t, \]

estimate the parameters of the restriction

\[ \beta_j^{(i)} = g^{(i)}(j, \lambda). \]

Such model is a generalisation of so called ADL-MIDAS regression. It is not required that all the coefficients should be restricted, i.e the function \( g^{(i)} \) might be an identity function. Model with no restrictions is called U-MIDAS model. The regressors \( x_{(i)}^{(i)} \) must be of higher (or of the same) frequency as the dependent variable \( y_t \).

Value

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

- coefficients the estimates of parameters of restrictions
- midas_coefficients the estimates of MIDAS coefficients of MIDAS regression
- model model data
- unrestricted unrestricted regression estimated using midas_u
- term_info the named list. Each element is a list with the information about the term, such as its frequency, function for weights, gradient function of weights, etc.
- 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
- gen_midas_coef the function which generates the MIDAS 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
- start_list the starting values as a list
- call the call to the function
midas_nlpr.fit

terms terms object
gradient gradient of NLS objective function
hessian hessian of NLS objective function
gradD gradient function of MIDAS weight functions
Zenv the environment in which data is placed
nobs the number of effective observations
convergence the convergence message
fitted.values the fitted values of MIDAS regression
residuals the residuals of MIDAS regression

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

midas_nlpr.fit Fit restricted MIDAS regression

Description

Workhorse function for fitting restricted MIDAS regression

Usage

midas_nlpr.fit(x)

Arguments

x midas_r object

Value

midas_r object

Author(s)

Vaidotas Zemlys
**midas_pl_plain**  
*MIDAS Partialy linear non-parametric regression*

**Description**

Function for fitting PL MIDAS regression without the formula interface

**Usage**

```r
midas_pl_plain(
  y,
  X,
  z,
  p.ar = NULL,
  weight,
  degree = 1,
  start_bws,
  start_x,
  start_ar = NULL,
  method = c("Nelder-Mead"),
  ...
)
```

**Arguments**

- `y`: model response
- `X`: prepared matrix of high frequency variable lags for MMM term
- `z`: a vector, data for the non-parametric part
- `p.ar`: length of AR part
- `weight`: the weight function
- `degree`: the degree of local polynomial
- `start_bws`: the starting values bandwith
- `start_x`: the starting values for weight function
- `start_ar`: the starting values for AR part. Should be the same length as `p`
- `method`: a method passed to `optim`
- `...`: additional parameters to `optim`

**Value**

an object similar to `midas_r` object

**Author(s)**

Virmantas Kvedaras, Vaidotas Zemlys
**midas_pl_sim**  
*Simulate PL MIDAS regression model*

**Description**

Simulate PL MIDAS regression model

**Usage**

```r
midas_pl_sim(
  n, 
  m, 
  theta, 
  gfun, 
  ar.x, 
  ar.y, 
  rand.gen = rnorm, 
  n.start = NA, 
  ...
)
```

**Arguments**

- `n`: number of observations to simulate.
- `m`: integer, frequency ratio
- `theta`: vector, restriction coefficients for high frequency variable
- `gfun`: function, a function which takes a single index
- `ar.x`: vector, AR parameters for simulating high frequency variable
- `ar.y`: vector, AR parameters for AR part of the model
- `rand.gen`: function, a function for generating the regression innovations, default is `rnorm`
- `n.start`: integer, length of a 'burn-in' period. If NA, the default, a reasonable value is computed.
- `...`: additional parameters to `rand.gen`

**Value**

a list

**Examples**

```r
nnbeta <- function(p, k) nbeta(c(1,p),k)

dgp <- midas_pl_sim(250, m = 12, theta = nnbeta(c(2, 4), 24), 
  gfun = function(x) 0.25*x^3, 
  ar.x = 0.9, ar.y = 0.5, n.start = 100)
```
**midas_qr**  
*Restricted MIDAS quantile regression*

**Description**

Estimate restricted MIDAS quantile regression using nonlinear quantile regression

**Usage**

```r
midas_qr(
  formula,
  data,
  tau = 0.5,
  start,
  Ofunction = "nlrq",
  weight_gradients = NULL,
  guess_start = TRUE,
  ...
)
```

**Arguments**

- **formula**: formula for restricted MIDAS regression or `midas_qr` object. Formula must include `mls` function
- **data**: a named list containing data with mixed frequencies
- **tau**: quantile
- **start**: 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`
- **weight_gradients**: a named list containing gradient functions of weights. The weight gradient function must return the matrix with dimensions $d_k \times q$, where $d_k$ and $q$ are the number of coefficients in unrestricted and restricted regressions correspondingly. The names of the list should coincide with the names of weights used in formula. The default value is `NULL`, which means that the numeric approximation of weight function gradient is calculated. If the argument is `NULL`, but the name of the weight used in formula is not present, it is assumed that there exists an R function which has the name of the weight function appended with `_gradient`.
- **guess_start**: logical, if `TRUE` tries certain strategy to improve starting values
- **...**: additional arguments supplied to optimisation function
Value

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

- **coefficients**: the estimates of parameters of restrictions
- **midas_coefficients**: the estimates of MIDAS coefficients of MIDAS regression
- **model**: model data
- **unrestricted**: unrestricted regression estimated using midas_u
- **term_info**: the named list. Each element is a list with the information about the term, such as its frequency, function for weights, gradient function of weights, etc.
- **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
- **gen_midas_coef**: the function which generates the MIDAS 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
- **start_list**: the starting values as a list
- **call**: the call to the function
- **terms**: terms object
- **gradient**: gradient of NLS objective function
- **hessian**: hessian of NLS objective function
- **gradD**: gradient function of MIDAS weight functions
- **Zenv**: the environment in which data is placed
- **use_gradient**: TRUE if user supplied gradient is used, FALSE otherwise
- **nobs**: the number of effective observations
- **convergence**: the convergence message
- **fitted.values**: the fitted values of MIDAS regression
- **residuals**: the residuals of MIDAS regression

Author(s)

Vaidotas Zemlys-Balevicius

Examples

```r
# Take the same example as in midas_r

theta_h0 <- function(p, dk, ...) {
  i <- (1:dk-1)/100
```
## Generate coefficients
theta0 <- theta_h0(c(-0.1,10,-10,-10),4*12)

## Plot the coefficients
plot(theta0)

## Generate the predictor variable
xx <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)

## Simulate the response variable
y <- midas_sim(500, xx, theta0)

x <- window(xx, start=start(y))

## Fit quantile regression. All the coefficients except intercept should be constant.
## Intercept coefficient should correspond to quantile function of regression errors.
mr <- midas_qr(y~fmls(x,4*12-1,12,theta_h0), tau = c(0.1, 0.5, 0.9),
              list(y=y,x=x),
              start=list(x=c(-0.1,10,-10,-10)))

## Estimation

### Description
Estimate restricted MIDAS regression using non-linear least squares.

### Usage

midas_r(
  formula,
  data,
  start,
  Ofunction = "optim",
  weight_gradients = NULL,
  ...
)

### Arguments

- **formula**: 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.
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`.

**Details**

Given MIDAS regression:

\[
y_t = \sum_{j=1}^{p} \alpha_j y_{t-j} + \sum_{i=0}^{k} \sum_{j=0}^{l_i} \beta^{(i)}_j x^{(i)}_{tm_i-j} + u_t,
\]

estimate the parameters of the restriction

\[
\beta^{(i)}_j = g^{(i)}(j, \lambda).
\]

Such model is a generalisation of so called ADL-MIDAS regression. It is not required that all the coefficients should be restricted, i.e the function \( g^{(i)} \) might be an identity function. Model with no restrictions is called U-MIDAS model. The regressors \( x^{(i)} \) must be of higher (or of the same) frequency as the dependent variable \( y_t \).

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_coefficients`: the estimates of MIDAS coefficients of MIDAS regression
- `model`: model data
- `unrestricted`: unrestricted regression estimated using `midas_u`
term_info  the named list. Each element is a list with the information about the term, such as its frequency, function for weights, gradient function of weights, etc.
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
gen_midas_coef  the function which generates the MIDAS 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
start_list  the starting values as a list
call  the call to the function
terms  terms object
gradient  gradient of NLS objective function
hessian  hessian of NLS objective function
gradD  gradient function of MIDAS weight functions
Zenv  the environment in which data is placed
use_gradient  TRUE if user supplied gradient is used, FALSE otherwise
nobs  the number of effective observations
convergence  the convergence message
fitted.values  the fitted values of MIDAS regression
residuals  the residuals of MIDAS regression

Author(s)
Virmantas Kvedaras, Vaidotas Zemlys

References

Examples
```r
#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

```r
plot(theta0)
```

## Generate the predictor variable

```r
xx <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)
```

## Simulate the response variable

```r
y <- midas_sim(500, xx, theta0)
```

```r
x <- window(xx, start = start(y))
```

## Fit restricted model

```r
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

```r
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)))
```

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

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

## Fit AR(1) model

```r
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

```r
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

```r
midas_r.fit(x)
```
Arguments

x  
midas_r object

Value

midas_r object

Author(s)

Vaidotas Zemlys

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,
  data = NULL,
  start = NULL,
  table,
  IC = c("AIC", "BIC"),
  test = c("hAh_test"),
  Ofunction = "optim",
  weight_gradients = NULL,
  show_progress = 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
midas_r_ic_table

weight_gradients
   see midas_r
show_progress  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 re-
             stricted 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(formula, data, lambda = NULL)

Arguments

- formula: 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 minimizing AIC.

Details

Estimates non-parametric MIDAS regression according to Breitung et al.

Value

a midas_r_np object

Author(s)

Vaidotas Zemlys

References


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_plain**

**Restricted MIDAS regression**

**Description**

Function for fitting MIDAS regression without the formula interface

**Usage**

```r
midas_r_plain(
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_plain(y,X,trend,weight=nealmon,startx=c(0,0,0))

midas_sim

Simulate simple MIDAS regression response variable

Description

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

Usage

midas_sim(n, x, theta, rand_gen = rnorm, innov = rand_gen(n, ...), ...)

Arguments

n
The sample size

x
a ts object with MIDAS regression predictor variable

theta
a vector with MIDAS regression coefficients

rand_gen
the function which generates the sample of innovations, the default is rnorm

innov
the vector with innovations, the default is NULL, i.e. innovations are generated using argument rand_gen

... additional arguments to rand_gen.

Details

MIDAS regression with one predictor variable has the following form:

\[ y_t = \sum_{j=0}^{h} \theta_j x_{tm-j} + u_t, \]

where \( m \) is the frequency ratio and \( h \) is the number of high frequency 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

```r
## 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, leave 4 low frequency lags of data for burn-in.
xx <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)

## Simulate the response variable
y <- midas_sim(500, xx, theta0)

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_si_plain**

### MIDAS Single index regression

**Description**

Function for fitting SI MIDAS regression without the formula interface

**Usage**

```r
midas_si_plain(
  y,
  X,
  p.ar = NULL,
  weight,
  degree = 1,
  start_bws,
  start_x,
)```
Arguments

y model response
x prepared matrix of high frequency variable lags for MMM term
p.ar length of AR part
weight the weight function
degree the degree of local polynomial
start_bws the starting values bandwith
start_x the starting values for weight function
start_ar the starting values for AR part. Should be the same length as p
method a method passed to optim, defaults to Nelder-Mead
... additional parameters to optim

Value

an object similar to midas_r object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Description

Simulate SI MIDAS regression model

Usage

midas_si_sim(
  n,
  m,
  theta,
  gfun,
  ar.x,
  ar.y,
  rand.gen = rnorm,
  n.start = NA,
  ...
)
Arguments

n  number of observations to simulate.

m  integer, frequency ratio

theta  vector, restriction coefficients for high frequency variable

gfun  function, a function which takes a single index

ar.x  vector, AR parameters for simulating high frequency variable

ar.y  vector, AR parameters for AR part of the model

rand.gen  function, a function for generating the regression innovations, default is rnorm

n.start  integer, length of a 'burn-in' period. If NA, the default, a reasonable value is computed.

...  additional parameters to rand.gen

Value

a list

Examples

nnbeta <- function(p, k) nbeta(c(1,p),k)

dgp <- midas_si_sim(250, m = 12, theta = nnbeta(c(2, 4), 24),
                      gfun = function(x) 0.03*x^3,
                      ar.x = 0.9, ar.y = 0.5, n.start = 100)

midas_sp  

Semi-parametric MIDAS regression

Description

Estimate semi-parametric MIDAS regression using non-linear least squares.

Usage

midas_sp(formula, data, bws, start, degree = 1, Ofunction = "optim", ...)

Arguments

formula  formula for restricted MIDAS regression or midas_r object. Formula must include fmls function

data  a named list containing data with mixed frequencies

bws  a bandwith specification. Note you need to supply logarithm value of the bandwith.
midas_sp

start
the starting values for optimisation. Must be a list with named elements.

degree
the degree of local polynomial. 0 corresponds to local-constant, 1 local-linear.
For univariate models higher values can be provided.

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 arguments method="Nelder-Mead"
and control=list(maxit=5000). Other supported functions are nls, optimx.

... additional arguments supplied to optimisation function

Details
Given MIDAS regression:

\[ y_t = \sum_{j=1}^{p} \alpha_j y_{t-j} + \sum_{i=0}^{k} \sum_{j=0}^{l_i} \beta_{ij}^{(i)} x_{tm_i-j} + u_t, \]

estimate the parameters of the restriction

\[ \beta_{ij}^{(i)} = g^{(i)}(j, \lambda). \]

Such model is a generalisation of so called ADL-MIDAS regression. It is not required that all the
coefficients should be restricted, i.e the function \( g^{(i)} \) might be an identity function. The regressors
\( x^{(i)}_t \) must be of higher (or of the same) frequency as the dependent variable \( y_t \).

Value
a midas_sp object which is the list with the following elements:

coefficients the estimates of parameters of restrictions

midas_coefficients the estimates of MIDAS coefficients of MIDAS regression

model model data

unrestricted unrestricted regression estimated using midas_u
term_info the named list. Each element is a list with the information about the term, such
as its frequency, function for weights, gradient function of weights, etc.

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
gen_midas_coef the function which generates the MIDAS 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
**midas_u**

- `start_list` the starting values as a list
- `call` the call to the function
- `terms` terms object
- `gradient` gradient of NLS objective function
- `hessian` hessian of NLS objective function
- `gradD` gradient function of MIDAS weight functions
- `Zenv` the environment in which data is placed
- `nobs` the number of effective observations
- `convergence` the convergence message
- `fitted.values` the fitted values of MIDAS regression
- `residuals` the residuals of MIDAS regression

**Author(s)**
Virmantas Kvedaras, Vaidotas Zemlys-Balevičius

---

**midas_u**

*Estimate unrestricted MIDAS regression*

---

**Description**

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

**Usage**

```r
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:

\[
y_t = \sum_{j=1}^{p} \alpha_j y_{t-j} + \sum_{i=0}^{k} \sum_{j=0}^{l_i} \beta_j(i) x_{tm_i-j} + u_t,
\]

where \( x_{tm_i-j} \), \( i = 0, \ldots k \), are regressors of higher (or similar) frequency than \( y_t \). Given certain assumptions the coefficients can be estimated using usual OLS and they have the familiar properties associated with simple linear regression.
Value

\texttt{lm} object.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

References


Examples

```r
##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
xx <- ts(arima.sim(model = list(ar = 0.6), 600 * 12), frequency = 12)

##Simulate the response variable
y <- midas_sim(500, xx, theta0)

x <- window(xx, start=start(y))

##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))

---

### Description

Create a matrix of selected MIDAS lags

### Usage

```r
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

```r
## 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)
```
mlsd

Description

MIDAS lag structure with dates

Usage

mlsd(x, k, datey, ...)

Arguments

x a vector
k lags, a vector
datey low frequency dates
... 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-Balevičius

Examples

x <- c(1:144)
y <- c(1:12)
datey <- (y-1)*12+1

# mlsd and mls should give the same results
m1 <- mlsd(x, 0:5, datey)
m2 <- mls(x, 0:5, 12)
sum(abs(m1 - m2))
**mmm**

*Compute MMM term for high frequency variable*

**Description**

Compute MMM term for high frequency variable

**Usage**

```r
mmm(X, theta, beta, ...)
```

**Arguments**

- **X**
  - matrix, high frequency variable embedded in low frequency, output of mls
- **theta**
  - vector, restriction coefficients for high frequency variable
- **beta**
  - vector of length 2, parameters for MMM term, slope and MMM parameter.
- **...**
  - currently not used

**Value**

a vector

**modsel**

*Select the model based on given information criteria*

**Description**

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

**Usage**

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

**Arguments**

- **x**
  - a `midas_r_ic_table` object
- **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

\texttt{lf\_lags\_table}, \texttt{hf\_lags\_table}, \texttt{amidas\_table} and \texttt{midas\_r\_ic\_table}.

Value

(invisibly) the best model based on information criteria, \texttt{midas\_r} object

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```r
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)))

modsels(mhfr,"BIC","unrestricted")

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

nakagamip

\textit{Normalized Nakagami probability density function MIDAS weights specification}

Description

Calculate MIDAS weights according to normalized Nakagami probability density function specification

Usage

\texttt{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 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*

**Description**

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

---

**nbetaMT**

*Normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible)*

**Description**

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
nbetaMT_gradient

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

---

nbetaMT_gradient

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient function for normalized beta probability density function MIDAS weights specification (MATLAB toolbox compatible)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>nbetaMT_gradient(p, d, m)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>m</td>
</tr>
</tbody>
</table>

Value

vector of coefficients

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
nbeta_gradient  Gradient function for normalized beta probability density function MIDAS weights specification

Description
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

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

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

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 \( \delta \) 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

```r
# 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

```r
## 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((coef(x, midas = TRUE)-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"))

### plot_lstr

#### Description

Plots logistic function for LSTR MIDAS regression
Usage

plot_lstr(x, term_name, title = NULL, compare = NULL, ...)

Arguments

x  
midas_r object

term_name  
the term name for which the coefficients are plotted. Default is NULL, which selects the first MIDAS term

title  
the title string of the graph. The default is NULL for the default title.

compare  
the parameters for weight function to compare with the model, default is NULL

...  
not used

Details

Plots logistic function for LSTR MIDAS regression of unrestricted MIDAS regression

Value

a data frame with restricted MIDAS coefficients, unrestricted MIDAS coefficients and lower and upper confidence interval limits. The data frame is returned invisibly.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

plot_midas_coef  
**Plot MIDAS coefficients**

Description

Plots MIDAS coefficients of a MIDAS regression for a selected term.

Usage

plot_midas_coef(x, term_name, title, ...)

## S3 method for class 'midas_r'
plot_midas_coef(
x,  
term_name = NULL,  
title = NULL,  
vcov. = sandwich,  
unrestricted = x$unrestricted,  
...  
)


Arguments

- **x**: midas_r object
- **term_name**: the term name for which the coefficients are plotted. Default is NULL, which selects the first MIDAS term.
- **title**: the title string of the graph. The default is NULL for the default title.
- **...**: additional arguments passed to vcov.
- **vcov.**: the covariance matrix to calculate the standard deviation of the coefficients.
- **unrestricted**: the unrestricted model, the default is unrestricted model from the x object. Set NULL to plot only the weights.

Details

Plots MIDAS coefficients of a selected MIDAS regression term together with corresponding MIDAS coefficients and their confidence intervals of unrestricted MIDAS regression.

Value

A data frame with restricted MIDAS coefficients, unrestricted MIDAS coefficients and lower and upper confidence interval limits. The data frame is returned invisibly.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

Examples

```r
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)))

plot_midas_coef(mr)
```

Description

Plots MIDAS coefficients of a MIDAS regression for a selected term.
plot_sp

Plot non-parametric part of the single index MIDAS regression

Description

Plot non-parametric part of the single index MIDAS regression of unrestricted MIDAS regression

Usage

plot_sp(x, term_name, title = NULL, compare = NULL, ...)

Arguments

- **x**: midas_r object
- **term_name**: the term name for which the coefficients are plotted. Default is NULL, which selects the first MIDAS term
- **title**: the title string of the graph. The default is NULL for the default title.
- **compare**: the parameters for weight function to compare with the model, default is NULL
- **normalize**: logical, if FALSE use the weight from the model, if TRUE, set the normalization coefficient of the weight function to 1.
- **...**: not used

Details

Plots MIDAS coefficients of a selected MIDAS regression term together with corresponding MIDAS coefficients and their confidence intervals of unrestricted MIDAS regression

Value

a data frame with restricted MIDAS coefficients, unrestricted MIDAS coefficients and lower and upper confidence interval limits. The data frame is returned invisibly.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
polystep

Arguments

\[ x \quad \text{midas_r object} \]

\[ \text{term_name} \quad \text{the term name for which the coefficients are plotted. Default is NULL, which selects the first MIDAS term} \]

\[ \text{title} \quad \text{the title string of the graph. The default is NULL for the default title.} \]

\[ \text{compare} \quad \text{the parameters for weight function to compare with the model, default is NULL not used} \]

Value

a data frame with restricted MIDAS coefficients, unrestricted MIDAS coefficients and lower and upper confidence interval limits. The data frame is returned invisibly.

Author(s)

Virmantas Kvedaras, Vaidotas Zemlys

---

polystep \quad \text{Step function specification for MIDAS weights}

Description

Step function specification for MIDAS weights

Usage

\[ \text{polystep}(p, d, m, a) \]

Arguments

\[ p \quad \text{vector of parameters} \]

\[ d \quad \text{number of coefficients} \]

\[ m \quad \text{the frequency ratio, currently ignored} \]

\[ a \quad \text{vector of increasing positive integers indicating the steps} \]

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_nlpr  Predict method for non-linear parametric MIDAS regression fit

**Description**

Predicted values based on midas_nlpr object.

**Usage**

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

**Arguments**

- `object`  `midas_nlpr` object
- `newdata` a named list containing data for mixed frequencies. If omitted, the in-sample 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

---

**predict.midas_r** 
*Predict method for MIDAS regression fit*

Description

Predicted values based on `midas_r` object.

Usage

```r
## 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 in-sample 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`.
predict.midas_sp

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))

predict.midas_sp

Predict method for semi-parametric MIDAS regression fit

Description

Predicted values based on midas_sp object.

Usage

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

Arguments

object midas_nlpr object
newdata a named list containing data for mixed frequencies. If omitted, the in-sample 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_sp 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-Balevičius

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**

[https://realized.oxford-man.ox.ac.uk/images/oxfordmanrealizedvolatilityindices-0.2-final.zip](https://realized.oxford-man.ox.ac.uk/images/oxfordmanrealizedvolatilityindices-0.2-final.zip)

**References**


**Examples**

```r
## Do not run:
## Download the data from
## https://realized.oxford-man.ox.ac.uk/images/oxfordmanrealizedvolatilityindices-0.2-final.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
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.
test             argument to modsel
ftype             which type of forecast to use.
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

```r
### 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=cbind(c(14,19),c(18,23),c(22,27)),z=cbind(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")
## )
```

Simulate MIDAS regression response

Description

Simulates one or more responses from the distribution corresponding to a fitted MIDAS regression object.
## S3 method for class 'midas_r'

```r
simulate(
  object,
  nsim = 999,
  seed = NULL,
  future = TRUE,
  newdata = NULL,
  insample = NULL,
  method = c("static", "dynamic"),
  innov = NULL,
  show_progress = TRUE,
  ...
)
```

### Arguments

- **object**: midas_r object
- **nsim**: number of simulations
- **seed**: either NULL or an integer that will be used in a call to set.seed before simulating the time series. The default, NULL will not change the random generator state.
- **future**: logical, if TRUE forecasts are simulated, if FALSE in-sample simulation is performed.
- **newdata**: a named list containing future values of mixed frequency regressors. The default is NULL, meaning that only in-sample data is used.
- **insample**: a list containing the historic mixed frequency data
- **method**: the simulation method, if "static" in-sample values for dependent variable are used in autoregressive MIDAS model, if "dynamic" the dependent variable values are calculated step-by-step from the initial in-sample values.
- **innov**: a matrix containing the simulated innovations. The default is NULL, meaning that innovations are simulated from model residuals.
- **show_progress**: logical, TRUE to show progress bar, FALSE for silent evaluation
- **...**: not used currently

### Details

Only the regression innovations are simulated, it is assumed that the predictor variables and coefficients are fixed. The innovation distribution is simulated via bootstrap.

### Value

A matrix of simulated responses. Each row contains a simulated response.

### Author(s)

Virmantas Kvedaras, Vaidotas Zemlys
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

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 + fnls(x, 23, 12, nealmon), start = list(x = rep(0, 3)))
simulate(mr, nsim=10, future=FALSE)

#Forecast horizon
h <- 3
#Declining unemployment
xn <- rep(-0.1, 12*h)
#New trend values
trendn <- length(y) + 1:h
simulate(mr, nsim = 10, future = TRUE, newdata = list(trend = trendn, x = xn))
**update_weights**

**Author(s)**

Virmantas Kvedaras, Vaidotas Zemlys

**Examples**

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

---

**Description**

Updates weights in a expression with MIDAS term

**Usage**

```r
update_weights(expr, tb)
```

**Arguments**

- `expr`  
  expression with MIDAS term
- `tb`  
  a named list with redefined weights

**Details**

For a MIDAS term `fmls(x,6,1,nealmon)` change weight `nealmon` to another weight.

**Value**

an expression with changed weights

**Author(s)**

Vaidotas Zemlys

**Examples**

```r
update_weights(y=trend+mls(x,0:7,4,nealmon)+mls(z,0:16,12,nealmon),list(x = "nbeta", z = ""))
```
UScpiqs  
*US quarterly seasonaly adjusted consumer price index*

**Description**

US quarterly CPI from 1960Q1 to 2017Q3. Seasonaly adjusted, Index 2015=1

**Format**

A *data.frame* object.

**Source**

FRED

USeffrw  
*US weekly effective federal funds rate.*

**Description**

US weekly effective federal funds rate from 1954-07-07 to 2017-12-13

**Format**

A *data.frame* object.

**Source**

FRED

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

**Description**


**Format**

A *ts* object.
**USqgdp**

**Source**

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

**Examples**

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

<table>
<thead>
<tr>
<th>USqgdp</th>
<th>United States gross domestic product, quarterly, seasonaly adjusted annual rate.</th>
</tr>
</thead>
</table>

**Description**


**Format**

A `ts` object.

**Source**

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

**Examples**

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

<table>
<thead>
<tr>
<th>USrealgdp</th>
<th>US annual gross domestic product in billions of chained 2005 dollars</th>
</tr>
</thead>
</table>

**Description**

The annual gross domestic product in billions of chained 2005 dollars for US from 1948 to 2011. This data is kept for historical purposes, newer data is in 2012 chained dollars.

**Format**

A `ts` object.

**Source**

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

USunempr  US monthly unemployment rate

Description

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

Format

A ts object.

Source

FRED

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",
  weight_gradients = NULL,
  ...
)

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
weights_table

0f function see midas
weight gradients see midas_r
... additional parameters to optimisation function, see midas_r

Details

This function estimates models sequentially increasing the midas lag from \( k_{\text{min}} \) to \( k_{\text{max}} \) 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

```r
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
```
Index

* datasets
  - oos_prec, 78
  - rvsp500, 88
  - UScpiqs, 94
  - USeffrw, 94
  - USpayems, 94
  - USqgdp, 95
  - USrealgdp, 95
  - USunempr, 96

* package
  - midasr-package, 4
  + .lws_table, 4
  - agk.test, 5
  - almonp, 6
  - almonp_gradient, 6
  - amidas_table, 7, 72
  - amweights, 7, 9, 18
  - average_forecast, 10
  - check_mixfreq, 11
  - coef.midas_nlpr, 12
  - coef.midas_r, 13
  - coef.midas_sp, 14
  - data.frame, 94
  - deriv_tests, 15
  - deviance.midas_nlpr, 15
  - deviance.midas_r, 16
  - deviance.midas_sp, 17
  - dmls, 17
  - expand_amidas, 18
  - expand_weights_lags, 19, 58
  - extract.midas_r, 20
  - fitted.midas_nlpr, 20
  - fitted.midas_sp, 21
  - fmls, 22, 36, 47, 54, 65
  - forecast (forecast.midas_r), 22
  - forecast.midas_r, 22, 83, 87

  - genexp, 25
  - genexp_gradient, 26
  - get_estimation_sample, 27
  - gompertzp, 27
  - gompertzp_gradient, 28
  - hAh_test, 30, 87
  - hAhr_test, 29, 87
  - harstep, 32
  - harstep_gradient, 33
  - hf_lags_table, 34, 72
  - imidas_r, 36
  - lcauchyp, 38
  - lcauchyp_gradient, 39
  - lf_lags_table, 39, 72
  - lm, 67, 68
  - lstr, 41
  - midas_auto_sim, 42
  - midas_lstr_plain, 43
  - midas_lstr_sim, 44
  - midas_mmm_plain, 45
  - midas_mmm_sim, 46
  - midas_nlpr, 47, 84, 86
  - midas_nlpr.fit, 49
  - midas_pl_plain, 50
  - midas_pl_sim, 51
  - midasqr, 52
  - midas_r, 5, 8, 15–17, 21, 29, 31, 35, 37, 40, 49, 54, 58, 59, 72, 85, 89, 91, 97
  - midas_r.fit, 57
  - midas_r_ic_table, 19, 58, 71, 72
  - midas_r_np, 60
  - midas_r_plain, 61
  - midas_si_plain, 63
  - midas_si_sim, 64
  - midas_sim, 62
  - midas_sp, 65
INDEX

midas_u, 37, 48, 53, 55, 66, 67
midasr, 8, 35, 40, 58, 97
midasr (midasr-package), 4
midasr-package, 4
mls, 52, 69
mlsd, 70
mmm, 71
mods, 71
nakagamip, 72
nakagamip_gradient, 73
nbeta, 74
nbeta_gradient, 76
nbetaMT, 74
nbetaMT_gradient, 75
nealmon, 76
nealmon_gradient, 77
nls, 36, 48, 52, 55, 66

oos_prec, 78
optim, 36, 48, 50, 52, 55, 64, 66
optimx, 44, 46, 48, 61, 66
plot_lstr, 79
plot_midas_coef, 80
plot_midas_coef.midas_nlpr, 81
plot_sp, 82
polystep, 83
polystep_gradient, 84
predict.midas_nlpr, 84
predict.midas_r, 85
predict.midas_sp, 86
prep_hAh, 87

rnorm, 62
rvsp500, 88

select_and_forecast, 88
simulate(simulate.midas_r), 90
simulate.midas_r, 90
split_data, 92

ts, 62, 94–96

update_weights, 93
UScpiqs, 94
USEffrw, 94
USpayems, 94
USqgdp, 95
USrealgdp, 95

weights_table, 96