Package 'midasml'

July 5, 2020

Type Package

Title Estimation and Prediction Methods for High-Dimensional Mixed Frequency Time Series Data

Version 0.0.5

Author Jonas Striaukas [aut, cre]

Maintainer Jonas Striaukas <jonas.striaukas@gmail.com>

Description The 'midasml' estimation and prediction methods for high dimensional time series regression models under mixed data sampling data structures using structured-sparsity penalties and orthogonal polynomials. For more information on the 'midasml' approach see Babii, Ghysels, and Striaukas (2020) <arXiv:2005.14057>. Functions that compute MIDAS data structures were inspired by MIDAS 'Matlab' toolbox (v2.3) written by Eric Ghysels.

BugReports https://github.com/jstriaukas/midasml/issues

License GPL (>= 2)

Depends foreach (>= 1.4.4)

Imports Rcpp (>= 1.0.3), lubridate (>= 1.7.4), parallel (>= 3.5.2), doSNOW (>= 1.0.18), stats (>= 3.5.2), optimx (>= 2020-4.2), quantreg (>= 5.34)

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-07-05 21:20:07 UTC

R topics documented:

`midasml-package` *Estimation and Prediction Methods for High-Dimensional Mixed Frequency Time Series Data*

Description

The 'midasml' estimation and prediction methods for high dimensional time series regression models under mixed data sampling data structures using structured-sparsity penalties and orthogonal polynomials. For more information on the 'midasml' approach see Babii, Ghysels, and Striaukas (2020) <arXiv:2005.14057>. Functions that compute MIDAS data structures were inspired by MIDAS 'Matlab' toolbox (v2.3) written by Eric Ghysels.

Details

midasml package contains functions to run classical MIDAS regression models under several loss functions as well as MIDAS machine learning methods. The main functions to run predictive midasml methods are `midasml_forecast` and `qtarget.sort_midasml`, while to estimate the midasml regressions the main functions are `reg_sgl` and `panel_sgl`.

Author(s)

Jonas Striaukas [aut, cre]

Maintainer: Jonas Striaukas <jonas.striaukas@gmail.com>

---

apply_transform *Time series matrix transformation*

Description

Transform time series a matrix based on transformation code.

Usage

```
apply_transform(data.in, tcode.all)
```

Arguments

data.in	time series matrix.
tcode.all	transformation code vector. 1 - not transformed, 2 - first difference, 3 - second difference, 4 - natural log, 5 - first difference of natural log, 6 - second difference of natural log, 7 - first difference of percent change.

Value

transformed time series matrix.

Examples

```
data.in <- matrix(rnorm(100*2,1),nrow = 100, ncol = 2)
apply_transform(data.in, tcode = rep(1,times=2))
```

---

beta_w *Beta density polynomial weights*

Description

For a given set of parameters θ and the number of high-frequency lags, returns weights implied by Beta density functional form.

Usage

```
beta_w(param, dayLag)
```

Arguments

param	two-dimensional parameter vector θ.
dayLag	number of high-frequency lags.

Value

(normalized) weights vector.

Author(s)

Jonas Striaukas

Examples

```
param <- c(2,2)
dayLag <- 22
beta_w(param, dayLag)
```

---

dateMatch *Match dates*

Description

Change the date to the beginning of the month date.

Usage

```
dateMatch(x, y)
```

Arguments

x	date vector to match with y date vector.
y	date vector.

Value

changed date vector.

Author(s)

Jonas Striaukas

Examples

```
x <- seq(as.Date("2020-01-01"),as.Date("2020-12-01"), by = "day")
set.seed(100)
x <- x[-sample(1:336, 100)]
y <- seq(as.Date("2020-01-01"),as.Date("2020-12-01"), by = "month")
dateMatch(x,y)
```

expalmon_w *Exponential Almon polynomial weights*

Description

For a given set of parameters θ and the number of high-frequency lags, returns weights implied by exponential Almon functional form.

Usage

```
expalmon_w(param, dayLag)
```

Arguments

param	two-dimensional parameter vector θ.
dayLag	number of high-frequency lags.

Value

(normalized) weights vector.

Author(s)

Jonas Striaukas

Examples

```
param <- c(0.02,-0.2)
dayLag <- 22
expalmon_w(param, dayLag)
```

gb *Gegenbauer polynomials shifted to [a,b]*

Description

For a given set of points in X, computes the orthonormal Gegenbauer polynomials basis of L2 [a,b] for a given degree and α parameter. The Gegenbauer polynomials are a special case of more general Jacobi polynomials. In turn, you may get Legendre polynomials from Gegenbauer by setting $\alpha = 0$, or Chebychev's polynomials by setting α = 1/2 or -1/2.

Usage

```
gb(degree, alpha, a = 0, b = 1, jmax = NULL, X = NULL)
```

Arguments

degree	polynomial degree.
alpha	Gegenbauer polynomials parameter.
a	lower shift value (default - 0).
b	upper shift value (default - 1).
jmax	number of high-frequency lags.
X	optional evaluation grid vector.

Value

Psi weight matrix with Gegenbauer functions upto degree.

Author(s)

Jonas Striaukas

Examples

```
degree <- 3
alpha <- 1
jmax <- 66
gb(degree = degree, alpha = alpha, a = 0, b = 1, jmax = jmax)
```

---

lb *Legendre polynomials shifted to [a,b]*

Description

For a given set of points in X, computes the orthonormal Legendre polynomials basis of L2 [a,b] for a given degree.

Usage

```
lb(degree, a = 0, b = 1, jmax = NULL, X = NULL)
```

Arguments

degree	polynomial degree.
a	lower shift value (default - 0).
b	upper shift value (default - 1).
jmax	number of high-frequency lags.
X	optional evaluation grid vector.

Value

Psi weight matrix with Legendre functions upto degree.

Author(s)

Jonas Striaukas

Examples

```
degree <- 3
jmax <- 66
lb(degree = degree, a = 0, b = 1, jmax = jmax)
```

`macro_midasml` *GDP nowcasting using midasML approach example data*

Description

US real GDP, FRED-MD monthly dataset, SPF survey and textual analysis data.

Usage

```
data(market_ret)
```

Format

A `list` object

Source

`macro_midasml$rgdp` - FRED
`macro_midasml$md.data` - St. Louis Fed M. W. McCracken website
`macro_midasml$text.data` - The Structure of Economic News website
`macro_midasml$survey.data` - Philadelphia Fed SPF data

Examples

```
data(macro_midasml)
macro_midasml$rgdp.data # GDP data
macro_midasml$md.data # FRED-MD data
macro_midasml$text.data # textual analysis data
macro_midasml$survey.data # SPF data
```

market_ret *SNP500 returns*

Description

SNP500 returns

Usage

```
data(market_ret)
```

Format

A data.frame object.a

Source

rgdp - FRED

Examples

```
data(market_ret)
market_ret$snp500ret
```

midasml_forecast *MIDAS ML regression prediction function*

Description

Predicts from a high-dimensional MIDAS model.

Usage

```
midasml_forecast(y_in, y_out, x_in, x_out, group_index,
  gamma_w, y_out_dates, scheme, verbose = FALSE, ...)
```

Arguments

y_in	response variable (in-sample).
y_out	response variable (out-of-sample).
x_in	predictor variables (in-sample).
x_out	predictor variables (out-of-sample).
group_index	group membership of each covariate.

gamma_w	sg-LASSO mixing parameter. gamma_w = 1 is LASSO and gamma_w = 0 group LASSO.
y_out_dates	out-of-sample dates.
scheme	prediction scheme. Choices are: expand - expanding window scheme, rolling - rolling window scheme.
verbose	flag to print information.
...	optional parameters to feed into reg_sgl.

Details

Based on desired computation of the tuning parameter λ (ic/cv), the MIDAS ML regression is fit and predictions are computed based on the chosen scheme (rolling/expanding). The regression function that is fit is

$$||y - \alpha\iota - X\beta||_T^2 + 2\lambda\Omega_\gamma(\beta)$$

,
where X (x_in and x_out) contains autoregressive lags and MIDAS covariates of sort_midasml class. The group index vector should be used from data sorting function qtarget.sort_midasml. The penalty function $\Omega_\gamma(.)$ is applied on β coefficients and is

$$\Omega_\gamma(\beta) = \gamma|\beta|_1 + (1-\gamma)||\beta||_{2,1},$$

a convex combination of LASSO and group LASSO penalty functions.

Value

out-of-sample predictions.

Author(s)

Jonas Striaukas

Examples

```
data(macro_midasml)
est.start <- as.Date("1990-12-01")
est.end <- as.Date("2017-03-01")
rgdp.data <- macro_midasml$rgdp.data
rgdp.data <- rgdp.data[rgdp.data$DATE<=as.Date("2017-06-01"),]
data <- qtarget.sort_midasml(y.data = rgdp.data, x.macro.data = macro_midasml$md.data,
      x.real.time = macro_midasml$text.data, x.quarterly_group = macro_midasml$survey.data,
         x.lag = 12, legendre_degree = 3,
         horizon = 1, macro_delay = 1, est.start, est.end,
         standardize = TRUE, group_ar_lags = FALSE, disp.flag = FALSE)
midasml_forecast(y_in = data$y_in, y_out = data$y_out,
         x_in = data$x_str, x_out = data$x_str_out,
         group_index = data$group_index, gamma_w = 0.65,
```

```
y_out_dates = data$y_out_dates, scheme = "expand",
method_choice = "ic", num_cores = 2)
```

`midas_ardl` *ARDL-MIDAS regression*

Description

Estimates and predicts using a single variate ARDL-MIDAS model.

Usage

```
midas_ardl(data.y, data.ydate, data.x, data.xdate, x.lag,
  y.lag, est.start, est.end, horizon = 1,
 polynomial = c("legendre_w", "beta_w", "rbeta_w", "expalmon_w", "umidas_w","step_fun"),
  scheme = c("fixed", "rolling", "expand"), loss = c("mse", "rq", "als"), ...)
```

Arguments

`data.y`	response variable series.
`data.ydate`	response variable dates.
`data.x`	predictor variable series.
`data.xdate`	predictor variable dates.
`x.lag`	number of high-frequency data lags.
`y.lag`	number of low-frequency data lags.
`est.start`	start date of the estimation sample (referenced with data.xdate).
`est.end`	end date of the estimation sample (referenced with data.xdate).
`horizon`	forecast horizon measured in predictor variable sampling frequency (default set 1 unit ahead).
`polynomial`	MIDAS lag polynomial specification. Options are: Legendre (`legendre_w`), Beta density (`beta_w`), restricted Beta density (`rbeta_w`), exponential Almon (`expalmon_w`), unrestricted MIDAS (`umidas_w`), step functions (`step_fun`).
`scheme`	forecasting scheme. Options are: fixed scheme (fixed), rolling window scheme (rolling), expanding window scheme (expand).
`loss`	loss function. Options are: mean squared error (mse), quantile (rq), asymmetric least squares (als).
`...`	optional parameters to feed into other functions. `forecast.flag` - TRUE/FALSE to compute out-of-sample predictions (default TRUE), disp.flag - TRUE/FALSE to display MIDAS data structures (default FALSE) `num.evals` - number of objective function evaluations using random starting parameter values in the case of non-linear MIDAS polynomial (default 1e4) `num.coef` - number of best coefficients to use as starting values in nonlinear optimization (default 10)

seed - value used in set.seed for randomly drawing initial starting values around OLS optimal solution
profiling - TRUE/FALSE to use MIDAS parameter profiling, coded only for rbeta_w polynomial, (default FALSE)
step_idx - index of step function lags. If step_fun is used as a polynomial, it is best to specify this option too, otherwise, the program figures out the sampling frequency ratio and computes step_idx accordingly (message is displayed in this case)
legendre_degree - degree of Legendre polynomials. If legendre_w is used as a polynomial, it is best to specify this option too, otherwise, the value is set to 3 (message is displayed in this case)
tau - quantile level for als and rq regressions. If either als or rq loss is used, this option must be specified, program stops if no value is provided.

Details

Several polynomial functional forms are available (input variable polynomial):
- beta_w: Beta polynomial
- rbeta_w: restricted Beta polynomial
- expalmon_w: Exp Almon polynomial
- umidas_w: unrestricted lags (U-MIDAS)
- step_fun: polynomial with step functions
- legendre_w: Legendre polynomials

different forecasting schemes (input variable scheme):
- fixed: fixed scheme
- rolling: rolling window
- expand: expanding window

and different loss functions (input variable loss)
- mse: least squares
- als: asymmetric least squares
- rq: quantile.

The ARDL-MIDAS model is:

$$y_t = \mu + \sum_p \rho_p y_{t-p} + \beta \sum_j \omega_j(\theta) x_{t-1}$$

where μ, β, θ, ρ_p are model parameters, p is number of low-frequency and ω is the weight function.

Value

returns midas_ardl list which contains parameter estimates, in- and out-of-sample statistics and predictions, and some information about the specification of the method used.

Author(s)

Jonas Striaukas

Examples

```
data(us_rgdp)
rgdp <- us_rgdp$rgdp
cfnai <- us_rgdp$cfnai
rgdp[-1, 2] <- ((rgdp[-1, 2]/rgdp[-dim(rgdp)[1], 2])^4-1)*100
rgdp <- rgdp[-1, ]
data.y <- rgdp[,2]
data.ydate <- rgdp[,1]
est.start <- as.Date("1990-01-01")
est.end <- as.Date("2002-03-01")
data.x <- cfnai[,2]
data.xdate <- cfnai[,1]
midas_ardl(data.y, data.ydate, data.x, data.xdate,
   x.lag = 12, y.lag = 4, est.start, est.end, horizon = 1,
   polynomial = "legendre_w", legendre_degree = 3)
```

`midas_dl` *DL-MIDAS regression*

Description

Estimates and predicts using a single variate DL-MIDAS model.

Usage

```
midas_dl(data.x, data.xdate, data.y, data.ydate, x.lag, est.start, est.end, horizon = 1,
 polynomial = c("legendre_w", "beta_w", "rbeta_w", "expalmon_w", "umidas_w", "step_fun"),
  scheme = c("fixed", "rolling", "expand"), loss = c("mse", "rq", "als"), ...)
```

Arguments

`data.x`	predictor variable series.
`data.xdate`	predictor variable dates.
`data.y`	dependent variable series (can leave unspecified, see `midas_gen` option).
`data.ydate`	dependent variable dates (can leave unspecified, see `midas_gen` option).
`x.lag`	number of high-frequency data lags.
`est.start`	start date of the estimation sample (referenced with data.xdate).
`est.end`	end date of the estimation sample (referenced with data.xdate).
`horizon`	forecast horizon measured in predictor variable sampling frequency (default set 1 unit ahead).
`polynomial`	MIDAS lag polynomial specification. Options are: Legendre (`legendre_w`), Beta density (`beta_w`), restricted Beta density (`rbeta_w`), exponential Almon (`expalmon_w`), unrestricted MIDAS (`umidas_w`), step functions (`step_fun`).
`scheme`	forecasting scheme. Options are: fixed scheme (fixed), rolling window scheme (rolling), expanding window scheme (expand).

`loss` loss function. Options are: mean squared error (mse), quantile (rq), asymmetric least squares (als).

`...` optional parameters to feed into other functions:
`forecast.flag` - TRUE/FALSE to compute out-of-sample predictions (default TRUE)
`disp.flag` - TRUE/FALSE to display MIDAS data structures (default FALSE)
`num.evals` - number of objective function evaluations using random starting parameter values in the case of non-linear MIDAS polynomial (default 1e4)
`num.coef` - number of best coefficients to use as starting values in nonlinear optimization (default 10)
`seed` - value used in set.seed for randomly drawing initial starting values around OLS optimal solution
`profiling` - TRUE/FALSE to use MIDAS paramater profiling, coded only for rbeta_w polynomial, (default FALSE)
`step_idx` - index of step function lags. If `step_fun` is used as a polynomial, it is best to specify this option too, otherwise, the program figures out the sampling frequency ratio and computes `step_idx` accordingly (message is displayed in this case)
`legendre_degree` - a degree of legendre polynomials. If `legendre_w` is used as a polynomial, it is best to specify this option too, otherwise, the value is set to 3 (message is displayed in this case)
`tau` - quantile level for als and rq regressions. If eithr als or rq loss is used, this option must be specified, program stops if no value is provided
`midas_gen` - option on how to generate the low-frequency variable. `from_hf` - computes from high-frequency variable (see `mixed_freq_data_mhorizon`, `aggregation` method could be specified as an additional input) or `as_ref` - computes MIDAS data structures using low-frequency variable (default 'from_hf').

Details

Several polynomial functional forms are available (input variable `polynomial`):
- `beta_w`: Beta polynomial
- `rbeta_w`: restricted Beta polynomial
- `expalmon_w`: Exp Almon polynomial
- `umidas_w`: unrestricted lags (U-MIDAS)
- `step_fun`: polynomial with step functions
- `legendre_w`: Legendre polynomials

different forecasting schemes (input variable `scheme`):
- `fixed`: fixed scheme
- `rolling`: rolling window
- `expand`: expanding window

and different loss functions (input variable `loss`)
- `mse`: least squares
- `als`: asymmetric least squares
- `rq`: quantile.

The DL-MIDAS model is:

$$y_t = \mu + \beta \sum_j \omega_j(\theta) x_{t-1}$$

where μ, β and θ are model parameters and ω is the weight function.

Value

returns `midas_dl` list which contains parameter estimates, in- and out-of-sample statistics and predictions, and some information about the specification of the method used.

Author(s)

Jonas Striaukas

Examples

```
data(market_ret)
data.x <- market_ret$snp500ret
data.xdate <- market_ret$DATE
est.start <- as.Date("2005-01-01")
est.end <- as.Date("2008-12-31")
midas_dl(data.x, data.xdate, x.lag = 5,
         est.start = est.start, est.end = est.end,
         horizon = 1, polynomial = "legendre_w", legendre_degree = 3,
         scheme = "fixed", loss = "mse", midas_gen = "from_hf")
```

---

mixed_freq_data *MIDAS data structure*

Description

Creates a MIDAS data structure for a single high-frequency covariate and a single low-frequency dependent variable.

Usage

```
mixed_freq_data(data.y, data.ydate, data.x, data.xdate, x.lag, y.lag,
  horizon, est.start, est.end, disp.flag = TRUE)
```

Arguments

`data.y`	n by 1 low-frequency time series data vector.
`data.ydate`	n by 1 low-frequency time series date vector.
`data.x`	nm by 1 high-frequency time series data vector.
`data.xdate`	nm by 1 high-frequency time series date vector.
`x.lag`	number of high-frequency lags to construct in high-frequency time units.

y.lag	number of low-frequency lags to construct in low-frequency time units.
horizon	forecast horizon relative to data.ydate date in high-frequency time units.
est.start	estimation start date, taken as the first ... .
est.end	estimation end date, taken as the last ... . Remaining data after this date is dropped to out-of-sample evaluation data.
disp.flag	display flag to indicate whether or not to display obtained MIDAS data structure in console.

Value

a list of MIDAS data structure.

Author(s)

Jonas Striaukas

Examples

```
data(us_rgdp)
rgdp <- us_rgdp$rgdp
payems <- us_rgdp$payems
payems[-1, 2] <- log(payems[-1, 2]/payems[-dim(payems)[1], 2])*100
payems <- payems[-1, ]
rgdp[-1, 2] <- ((rgdp[-1, 2]/rgdp[-dim(rgdp)[1], 2])^4-1)*100
rgdp <- rgdp[-1, ]
est.start <- as.Date("1990-01-01")
est.end <- as.Date("2002-03-01")
mixed_freq_data(rgdp[,2], as.Date(rgdp[,1]), payems[,2],
  as.Date(payems[,1]), x.lag = 9, y.lag = 4, horizon = 1,
  est.start, est.end, disp.flag = FALSE)
```

---

mixed_freq_data_mhorizon

MIDAS data structure

Description

Creates a MIDAS data structure for a single high-frequency covariate and a single low-frequency dependent variable computed from high-frequency covariate (e.g. stock returns).

Usage

```
mixed_freq_data_mhorizon(data.x, data.xdate, x.lag, est.start, est.end,
  horizon, disp.flag = TRUE, ...)
```

Arguments

data.x	nm by 1 high-frequency time series data vector.
data.xdate	nm by 1 high-frequency time series date vector.
x.lag	number of high-frequency lags to construct in high-frequency time units.
est.start	estimation start date, taken as the first ... .
est.end	estimation end date, taken as the last ... . Remaining data after this date is dropped to out-of-sample evaluation data.
horizon	forecast horizon relative to data.ydate date in high-frequency time units.
disp.flag	display flag to indicate whether or not to display obtained MIDAS data structure in console.
...	an optional parameter aggregation specifying the aggregation method of high-frequency data to get low-frequency target (non-overlapping) sum - sum of high-frequency lags sum&abs - sum of high-frequency lags which after aggregation are taken in absolute value sum&sq - sum of high-frequency lags which after aggregation are taken in squares mean - average of high-frequency lags first_val - the most recent lag value of high-frequency lags.

Value

a list of midas data structure.

Author(s)

Jonas Striaukas

Examples

```
data(market_ret)
data.x <- market_ret$snp500ret
data.xdate <- market_ret$DATE
est.start <- as.Date("2005-01-01")
est.end <- as.Date("2017-12-31")
mixed_freq_data_mhorizon(data.x, data.xdate, x.lag = 5, est.start, est.end,
                         horizon = 1, disp.flag = FALSE, aggregation = "sum")
```

---

mixed_freq_data_single

MIDAS data structure

Description

Creates a MIDAS data structure for a single high-frequency covariate based on low-frequency reference date.

Usage

```
mixed_freq_data_single(data.refdate, data.x, data.xdate, x.lag, horizon,
  est.start, est.end, disp.flag = TRUE)
```

Arguments

data.refdate	n by 1 date vector.
data.x	nm by 1 high-frequency time series data vector.
data.xdate	nm by 1 high-frequency time series date vector.
x.lag	number of high-frequency lags to construct in high-frequency time units.
horizon	forecast horizon relative to data.refdate date in high-frequency time units.
est.start	estimation start date, taken as the first ... .
est.end	estimation end date, taken as the last ... . Remaining data after this date is dropped to out-of-sample evaluation data.
disp.flag	display flag to indicate whether or not to display obtained MIDAS data strcuture in console.

Value

a list of midas data structure.

Author(s)

Jonas Striaukas

Examples

```
data(us_rgdp)
rgdp <- us_rgdp$rgdp
cfnai <- us_rgdp$cfnai
data.refdate <- rgdp$date
data.x <- cfnai$cfnai
data.xdate <- cfnai$date
est.start <- as.Date("1990-01-01")
est.end <- as.Date("2002-03-01")
mixed_freq_data_single(data.refdate, data.x, data.xdate, x.lag = 12, horizon = 1,
 est.start, est.end, disp.flag = FALSE)
```

monthBegin *Beginning of the month date*

Description

Change the date to the beginning of the month date.

Usage

```
monthBegin(x)
```

Arguments

x	date value.

Value

changed date value.

Author(s)

Jonas Striaukas

Examples

```
monthBegin(as.Date("2020-05-15"))
```

monthEnd *End of the month date*

Description

Change the date to the end of the month date.

Usage

```
monthEnd(x)
```

Arguments

x	date value.

Value

changed date value.

Author(s)

Jonas Striaukas

Examples

```
monthEnd(as.Date("2020-05-15"))
```

`panel_sgl` *Panel sg-LASSO regression model*

Description

Fits panel data regression model, random and fixed effects, under sg-LASSO penalty function. Options include random effects and fixed effects models, cross-validation and information criteria for λ penalty parameter selection.

Usage

```
panel_sgl(X, Z = NULL, y, index, entity_indices, gamma_w = NULL, l1_factor = NULL,
  l21_factor = NULL, dummies_index = NULL, full_est = NULL,
  regress_choice = c("re", "fe"), method_choice = c("ic", "cv", "initial"),
  nlam = 100, lambdas = NULL, min_frac = NULL, nfolds = 10,
  lambda_choice = c("min", "1se"), ic_choice = c("bic", "aic", "aicc"),
  num_cores = NULL, verbose = FALSE, thresh = NULL,
  outer_thresh = NULL, inner_iter = NULL, outer_iter = NULL)
```

Arguments

`X`	NT by p data matrix, where n, t and p respectively denote the number of individuals, sample size and the number of regressors.
`Z`	dummies matrix for random effects or fixed effects panel data model. If left unspecified, it is computed based on `regress_choice` choice.
`y`	NT by 1 vector of outcome.
`index`	p by 1 vector indicating group membership of each covariate.
`entity_indices`	NT by 1 vector of individual indices.
`gamma_w`	sg-LASSO mixing parameter. `gamma_w = 1` is LASSO and `gamma_w = 0` group LASSO.
`l1_factor`	ℓ_1 norm penalty factor for random or fixed effects (default value is zero which means α is left unpenalized in ℓ_1 norm).
`l21_factor`	ℓ_1 norm penalty factor for random or fixed effects (default value is zero which means α is left unpenalized in ℓ_1 norm).
`dummies_index`	vector indicating group membership of α (default - no grouping).
`full_est`	pre-estimated parameters based on full sample and `regress_choice` for a sequence of λ's.

regress_choice	choose between ‘re‘ and ‘fe‘. ‘re‘ computes random effects regression with sg-LASSO penalty (default). ‘fe‘ computes fixed effects regression with sg-LASSO penalty.
method_choice	choose between ‘initial‘, ‘ic‘ and ‘cv‘. ‘initial‘ pre-computes initial estimates. ‘ic‘ comptes solution based on information criteria (BIC, AIC or AICc). ‘cv‘ computes solution based on cross-validation (cv).
nlam	number of λ’s to use in the regularization path.
lambdas	user specified sequence of λ values for fitting. We recommend leaving this to NULL and letting function to self-select values.
min_frac	the minimum value of the penalty parameter, as a fraction of the maximum value.
nfolds	number of folds of the cv loop.
lambda_choice	chose between ‘min‘ and ‘1se‘. ‘min‘ computes solution that minimizes the cv error. ‘1se‘ computes solution such that the cv error is within 1 standard error of the minimum ‘min‘.
ic_choice	choose between ‘bic‘, ‘aic‘ and ‘aicc‘. ‘bic‘ computes solution that minimizes Bayesian information criterion. ‘aic‘ computes solution that minimizes Akaike information criterion. ‘aicc‘ omputes solution that minimizes Akaike information corrected criterion.
num_cores	number of cores used to compute cv loop.
verbose	flag to print information.
thresh	convergence threshold for change in beta. We recommend leaving this to NULL.
outer_thresh	outer loop convergence threshold. We recommend leaving this to NULL.
inner_iter	the maximum number of inner sg-LASSO loop iterations. We recommend leaving this to NULL.
outer_iter	the maximum number of outer sg-LASSO loop iterations. We recommend leaving this to NULL.

Details

The sequence of panel data models implied by lambdas vector is fit by block coordinate-descent. The objective function is

$$RSS(\alpha, \beta)/NT + 2\lambda * \Omega_\gamma(\beta),$$

where $RSS(\alpha, \beta)$ is either random or fixed effects model fit. The penalty function $\Omega_\gamma(.)$ is applied on coefficients of time-varying covariates β and is

$$\Omega_\gamma(\beta) = \gamma|\beta|_1 + (1-\gamma)||\beta||_{2,1},$$

a convex combination of LASSO and group LASSO penalty functions. There are additional options to apply sg-LASSO structures on fixed effects coefficient vector α (see description of input variables for more details).

Value

Parameter estimates of panel data regression model under sg-LASSO penalty.

Author(s)

Jonas Striaukas

Examples

```
# simulate DGP
set.seed(1)
t <- 10; n = 5; p = 20; size.groups = 4
index <- ceiling(1:p / size.groups)
X <- matrix(rnorm(n * t * p), ncol = p, nrow = n*t)
beta <- c(5,4,3,2,1)
y <- X[,1:5] %*% beta + 5*rnorm(n*t)
entity_indices <- sort(rep(1:n,times=t-1))
panel_sgl(X = X, Z = NULL, y = y, index = index,
  entity_indices = entity_indices, gamma_w = 1,
  regress_choice = "fe", method_choice = "initial",
  num_cores = 2, verbose = FALSE)
```

---

plot_weights *MIDAS weights plot function*

Description

Based on specification in obj, plots a basic R figure of estimated MIDAS weights.

Usage

```
plot_weights(obj)
```

Arguments

obj	midas_ardl or midas_dl object with parameter estimates and model specfication inputs.

Details

MIDAS regression pecifcation is picked up from obj, see midas_dl or midas_ardl function descriptions for more details.

Value

returns R figure of estimated MIDAS weights.

Author(s)

Jonas Striaukas

Examples

```
data(us_rgdp)
rgdp <- us_rgdp$rgdp
cfnai <- us_rgdp$cfnai
rgdp[-1, 2] <- ((rgdp[-1, 2]/rgdp[-dim(rgdp)[1], 2])^4-1)*100
rgdp <- rgdp[-1, ]
data.y <- rgdp[,2]
data.ydate <- rgdp[,1]
est.start <- as.Date("1990-01-01")
est.end <- as.Date("2002-03-01")
data.x <- cfnai[,2]
data.xdate <- cfnai[,1]
fit <- midas_ardl(data.y, data.ydate, data.x, data.xdate,
                  x.lag = 12, y.lag = 4, est.start, est.end, horizon = 1,
                  polynomial = "legendre_w", legendre_degree = 3)
plot_weights(obj = fit$est.obj)
```

---

predict.panel_sgl *Computes prediction for the sg-LASSO panel regression model*

Description

Computes prediction for the sg-LASSO panel regression model

Usage

```
## S3 method for class 'panel_sgl'
predict(object, newX, newZ = NULL, regress_choice = c("re", "fe"), ...)
```

Arguments

object	fit object from panel_sgl.
newX	matrix of out-of-sample covariate observations.
newZ	optional matrix of dummies for panel data model.
regress_choice	choose between ‘re‘ and ‘fe‘. Must be consistent with object.
...	currently ignored optional parameters.

Value

a list of these variables:

pred - overall prediction.

predZ - dummies prediction.

predX - covariates prediction.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
t <- 10; n = 5; p = 20; size.groups = 4
index <- ceiling(1:p / size.groups)
X <- matrix(rnorm(n * t * p), ncol = p, nrow = n*t)
beta <- c(5,4,3,2,1)
y <- X[,1:5] %*% beta + 5*rnorm(n*t)
Z <- kronecker(diag(n), rep(1, times = t))
entity_indices <- sort(rep(1:n,times=t-1))
fit <- panel_sgl(X = X, Z = Z, y = y, index = index,
         entity_indices = entity_indices, gamma_w = 1,
         regress_choice = "fe", method_choice = "ic",
         num_cores = 2, verbose = FALSE)
predict.panel_sgl(object = fit, newX = X, newZ = Z, regress_choice = "fe")$pred
```

---

predict.reg_sgl *Computes prediction for the sg-LASSO linear regression*

Description

Computes prediction for the sg-LASSO linear regression

Usage

```
## S3 method for class 'reg_sgl'
predict(object, newX, ...)
```

Arguments

object	fit object from sglassofit.
newX	matrix of out-of-sample covariate observations.
...	currently ignored optional parameters.

Value

a list of these variables:

pred - overall prediction.

predZ - intercept prediction.

predX - covariates prediction.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)
index <- 1:20
fit <- reg_sgl(X = x, y = y, index = index, gamma_w = 1, method_choice = "ic",
       num_cores = 2, verbose = FALSE)
predict.reg_sgl(object = fit, newX = x)
```

---

qtarget.sort_midasml *High-dimensional mixed frequency data sort function*

Description

Sorts high-dimensional mixed frequency data for quarterly target variable.

Usage

```
qtarget.sort_midasml(y.data, x.macro.data = NULL, x.real.time = NULL,
  x.quarterly_group = NULL, x.lag = NULL, legendre_degree, horizon,
  macro_delay = 1, est.start, est.end, standardize = TRUE, group_ar_lags = FALSE,
  real_time_predictions = FALSE, disp.flag = TRUE)
```

Arguments

y.data	response variable data.
x.macro.data	macro data which is not real-time, i.e. is used with publication delay defined in macro_delay.
x.real.time	real-time data.
x.quarterly_group	quarterly data currently taken as real-time data.
x.lag	single value or vector of size of the total number of variables defining the number of lags for each high-frequency variable in x.macro.data,x.real.time.
legendre_degree	single value or vector of size of the total number of variables defining the polynomial degree for each each high-frequency variable in x.macro.data,x.real.time.
horizon	forecast horizon relative to y.data date column in high-frequency time units.
macro_delay	number of months that macro series in x.macro.data are delayed.
est.start	estimation start date, taken as the first ... .

est.end	estimation end date, taken as the last ... . Remainig data after this date is dropped to out-of-sample evaluation data.
standardize	TRUE/FALSE to standardize high-frequneyc covariates in high-frequency units.
group_ar_lags	TRUE/FALSE to group AR lags.
real_time_predictions	TRUE/FALSE in case real-time data is used for predictions
disp.flag	display flag to indicate whether or not to display obtained MIDAS data structure in console.

Details

Sorts high-dimensional mixed frequency data for quarterly target variable read to be inputed into `midasml_forecast` function to produce nowcasts & forecasts.

Value

MIDAS covariates and group memberships based on desired specification.

Author(s)

Jonas Striaukas

Examples

```
data(macro_midasml)
est.start <- as.Date("1990-12-01")
est.end <- as.Date("2017-03-01")
rgdp.data <- macro_midasml$rgdp.data
rgdp.data <- rgdp.data[rgdp.data$DATE<=as.Date("2017-06-01"),]
qtarget.sort_midasml(y.data = rgdp.data, x.macro.data = macro_midasml$md.data,
      x.real.time = macro_midasml$text.data, x.quarterly_group = macro_midasml$survey.data,
         x.lag = 12, legendre_degree = 3,
         horizon = 1, macro_delay = 1, est.start, est.end,
         standardize = TRUE, group_ar_lags = FALSE, disp.flag = FALSE)
```

---

rbeta_w *Restricted Beta density polynomial weights*

Description

For a given set of parameters θ and the number of high-frequency lags, returns weights implied by Restricted Beta density functional form.

Usage

```
rbeta_w(param, dayLag)
```

Arguments

param	one-dimensional parameter vector θ.
dayLag	number of high-frequency lags.

Value

(normalized) weights vector.

Author(s)

Jonas Striaukas

Examples

```
param <- 3
dayLag <- 22
rbeta_w(param, dayLag)
```

---

reg_sgl *Linear sg-LASSO regression*

Description

Fits sg-LASSO least squares regression model. Options include cross-validation and information criteria for λ penalty parameter selection.

Usage

```
reg_sgl(X, y, index, gamma_w = NULL, full_est = NULL,
  method_choice = c("ic", "cv", "initial"), nlam = 100, lambdas = NULL,
  min_frac = NULL, nfolds = 10, lambda_choice = c("min", "1se"),
  ic_choice = c("bic", "aic", "aicc"), num_cores = NULL, verbose = FALSE,
  thresh = NULL, outer_thresh = NULL, inner_iter = NULL, outer_iter = NULL)
```

Arguments

X	T by p data matrix, where t and p respectively denote the sample size and the number of regressors.
y	T by 1 vector of outcome.
index	p by 1 vector indicating group membership of each covariate.
gamma_w	sg-LASSO mixing parameter. `gamma_w = 1` is LASSO and `gamma_w = 0` group LASSO.
full_est	pre-estimated parameters based on full sample and `regress_choice` for a sequence of λ's.

method_choice	choose between ‘initial‘, ‘ic‘ and ‘cv‘. ‘initial‘ pre-computes initial estimates. ‘ic‘ comptes solution based on information criteria (BIC, AIC or AICc). ‘cv‘ computes solution based on cross-validation (cv).
nlam	number of λ’s to use in the regularization path.
lambdas	user specified sequence of λ values for fitting. We recommend leaving this to NULL and letting function to self-select values.
min_frac	the minimum value of the penalty parameter, as a fraction of the maximum value.
nfolds	number of folds of the cv loop.
lambda_choice	chose between ‘min‘ and ‘1se‘. ‘min‘ computes solution that minimizes the cv error. ‘1se‘ computes solution such that the cv error is within 1 standard error of the minimum ‘min‘.
ic_choice	choose between ‘bic‘, ‘aic‘ and ‘aicc‘. ‘bic‘ computes solution that minimizes Bayesian information criterion. ‘aic‘ computes solution that minimizes Akaike information criterion. ‘aicc‘ omputes solution that minimizes Akaike information corrected criterion.
num_cores	number of cores used to compute cv loop.
verbose	flag to print information.
thresh	convergence threshold for change in beta. We recommend leaving this to NULL.
outer_thresh	outer loop convergence threshold. We recommend leaving this to NULL.
inner_iter	the maximum number of inner sg-LASSO loop iterations. We recommend leaving this to NULL.
outer_iter	the maximum number of outer sg-LASSO loop iterations. We recommend leaving this to NULL.

Details

The sequence of linear regression models implied by lambdas vector is fit by block coordinate-descent. The objective function is

$$RSS(\alpha, \beta)/T + 2\lambda * \Omega_\gamma(\beta),$$

where $RSS(\alpha, \beta)$ is the least squares fit. The penalty function $\Omega_\gamma(.)$ is applied on β coefficients and is

$$\Omega_\gamma(\beta) = \gamma|\beta|_1 + (1-\gamma)||\beta||_{2,1},$$

a convex combination of LASSO and group LASSO penalty functions.

Value

Parameter estimates of linear regression model under sg-LASSO penalty.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
y = rnorm(100)
index = 1:20
reg_sgl(X = x, y = y, index = index, gamma_w = 1, method_choice = "initial",
  num_cores = 2, verbose = FALSE, lambdas = c(1,2,3))
```

---

sgl_fit *sg-LASSO regression*

Description

Fits mse sg-LASSO regression model.

Usage

```
sgl_fit(X, Z, y, index, lambdas, gamma_w = NULL, l1_factor = NULL, l21_factor = NULL,
  dummies_index = NULL, inner_iter = NULL, outer_iter = NULL, thresh = NULL,
  outer_thresh = NULL)
```

Arguments

X	T by p data matrix, where t and p respectively denote the sample size and the number of regressors.
Z	dummies matrix.
y	T by 1 vector of outcome.
index	p by 1 vector indicating group membership of each covariate.
lambdas	user specified sequence of λ values for fitting. We recommend leaving this to NULL and letting function to self-select values.
gamma_w	sg-LASSO mixing parameter. gamma_w = 1 is LASSO and gamma_w = 0 group LASSO.
l1_factor	ℓ_1 norm penalty factor for random or fixed effects (default value is zero which means α is left unpenalized in ℓ_1 norm).
l21_factor	ℓ_1 norm penalty factor for random or fixed effects (default value is zero which means α is left unpenalized in ℓ_1 norm).
dummies_index	vector indicating group membership of α (default - no grouping).
inner_iter	the maximum number of inner sg-LASSO loop iterations. We recommend leaving this to NULL.
outer_iter	the maximum number of outer sg-LASSO loop iterations. We recommend leaving this to NULL.
thresh	convergence threshold for change in beta. We recommend leaving this to NULL.
outer_thresh	outer loop convergence threshold. We recommend leaving this to NULL.

Details

The sequence of linear regression models implied by `lambdas` vector is fit by block coordinate-descent. The objective function is

$$RSS(\alpha, \beta)/T + 2\lambda * \Omega_\gamma(\beta),$$

where $RSS(\alpha, \beta)$ is the least squares fit. The penalty function $\Omega_\gamma(.)$ is applied on β coefficients and is

$$\Omega_\gamma(\beta) = \gamma|\beta|_1 + (1-\gamma)||\beta||_{2,1},$$

a convex combination of LASSO and group LASSO penalty functions.

Value

sg-LASSO regression fitted coefficients.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
y = rnorm(100)
index = 1:20
Z <- as.matrix(rep(1,times=length(y)))
sgl_fit(X = x, Z = Z, y = y, index = index, lambdas = c(1,2,3), gamma_w = 1)
```

`transform_dt` *Time series vector transformation*

Description

Transform time series a vector based on transformation code.

Usage

```
transform_dt(x, tcode)
```

Arguments

`x`	time series vector.
`tcode`	transformation code. 1 - not transformed, 2 - first difference, 3 - second difference, 4 - natural log, 5 - first difference of natural log, 6 - second difference of natural log, 7 - first difference of percent change.

Value

transformed time series vector.

Examples

```
x <- rnorm(100,1)
transform_dt(x, tcode = 1)
```

---

us_rgdp *US real GDP data with several high-frequency predictors*

Description

US real GDP, Chicago National Activity Index, Nonfarm payrolls and ADS Index

Usage

```
data(us_rgdp)
```

Format

A data.frame object.a

Source

rgdp - FRED
cfnai - Chicago Fed website
payems - FRED
ads - Philadelphia Fed website

Examples

```
data(us_rgdp)
us_rgdp$rgdp # - GDP data
us_rgdp$cfnai # - CFNAI predictor data
us_rgdp$payems # - Nonfarm payrolls predictor data
us_rgdp$ads # - ADS predictor data
```

Index