Package 'midasml'

May 20, 2021

Type Package

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

Version 0.1.5-1

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

Description The 'midasml' package implements estimation and prediction methods for high-dimensional mixed-frequency (MIDAS) time-series and panel data regression models. The regularized MIDAS models are estimated using orthogonal (e.g. Legendre) polynomials and sparse-group LASSO (sg-LASSO) estimator. For more information on the 'midasml' approach see Babii, Ghysels, and Striaukas (2021, JBES forthcoming) <doi:10.1080/07350015.2021.1899933>. The package is equipped with the fast implementation of the sg-LASSO estimator by means of proximal block coordinate descent. High-dimensional mixed frequency time-series data can also be easily manipulated with functions provided in the package.

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

License GPL (>= 2)

Depends Matrix, R (>= 3.5.0)

Imports foreach, graphics, mcGlobaloptim, methods, lubridate, stats

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation yes

Author Jonas Striaukas [cre, aut],
Andrii Babii [aut],
Eric Ghysels [aut],
Alex Kostrov [ctb] (Contributions to analytical gradients for non-linear low-dimensional MIDAS estimation code)

Repository CRAN

Date/Publication 2021-05-20 08:30:02 UTC

R topics documented:

`midasml-package` *midasml*

Description

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

Author(s)

Jonas Striaukas (maintainer) <jonas.striaukas@gmail.com>, Andrii Babii <andrii@email.unc.edu>, Eric Ghysels <eghysels@unc.edu>

alfred_vintages *ALFRED monthly and quarterly series vintages*

Description

ALFRED monthly and quarterly series vintages

Usage

```
data(alfred_vintages)
```

Format

A list objects

Source

ALFRED

Examples

```
data(alfred_vintages)
i <- 1
alfred_vintages[[i]] # ith variable
```

cv.panel.sglfit *Cross-validation fit for panel sg-LASSO*

Description

Does k-fold cross-validation for panel data sg-LASSO regression model.

The function runs sglfit `nfolds+1` times; the first to get the path solution in `lambda` sequence, the rest to compute the fit with each of the folds omitted. The average error and standard deviation over the folds is computed, and the optimal regression coefficients are returned for `lam.min` and `lam.1se`. Solutions are computed for a fixed γ.

Usage

```
cv.panel.sglfit(x, y, lambda = NULL, gamma = 1.0, gindex = 1:p, nfolds = 10,
  foldid, method = c("pooled", "fe"), nf = NULL, parallel = FALSE, ...)
```

Arguments

x	NT by p data matrix, where NT and p respectively denote the sample size of pooled data and the number of regressors.
y	NT by 1 response variable.
lambda	a user-supplied lambda sequence. By leaving this option unspecified (recommended), users can have the program compute its own λ sequence based on nlambda and γ lambda.factor. It is better to supply, if necessary, a decreasing sequence of lambda values than a single (small) value, as warm-starts are used in the optimization algorithm. The program will ensure that the user-supplied lambda sequence is sorted in decreasing order before fitting the model.
gamma	sg-LASSO mixing parameter. γ = 1 gives LASSO solution and γ = 0 gives group LASSO solution.
gindex	p by 1 vector indicating group membership of each covariate.
nfolds	number of folds of the cv loop. Default set to 10.
foldid	the fold assignments used.
method	choose between 'pooled' and 'fe'; 'pooled' forces the intercept to be fitted in sglfit, 'fe' computes the fixed effects. User must input the number of fixed effects nf for method = 'fe', and it is recommended to do so for method = 'pooled'. Program uses supplied nf to construct foldsid. Default is set to method = 'pooled'.
nf	number of fixed effects. Used only if method = 'fe'.
parallel	if TRUE, use parallel foreach to fit each fold. Must register parallel before hand, such as doMC or others. See the example below.
...	Other arguments that can be passed to sglfit.

Details

The cross-validation is run for sg-LASSO linear model. The sequence of linear regression models implied by λ vector is fit by block coordinate-descent. The objective function is either (case method='pooled')

$$\|y - \iota\alpha - x\beta\|_{NT}^2 + 2\lambda\Omega_\gamma(\beta),$$

where $\iota \in R^{NT}$ and α is common intercept to all N items or (case method='fe')

$$\|y - B\alpha - x\beta\|_{NT}^2 + 2\lambda\Omega_\gamma(\beta),$$

where $B = I_N \times \iota$ and $\|u\|_{NT}^2 = \langle u, u\rangle/NT$ is the empirical inner product. 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

cv.panel.sglfit object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
cv.panel.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5, method = "fe", nf = 10,
  standardize = FALSE, intercept = FALSE)
## Not run:
# Parallel
require(doMC)
registerDoMC(cores = 2)
x = matrix(rnorm(1000 * 20), 1000, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(1000)
gindex = sort(rep(1:4,times=5))
system.time(cv.panel.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5, method = "fe", nf = 10,
  standardize = FALSE, intercept = FALSE))
system.time(cv.panel.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5, method = "fe", nf = 10,
  standardize = FALSE, intercept = FALSE, parallel = TRUE))

## End(Not run)
```

`cv.sglfit` *Cross-validation fit for sg-LASSO*

Description

Does k-fold cross-validation for sg-LASSO regression model.

The function runs sglfit `nfolds+1` times; the first to get the path solution in `lambda` sequence, the rest to compute the fit with each of the folds omitted. The average error and standard deviation over the folds is computed, and the optimal regression coefficients are returned for `lam.min` and `lam.1se`. Solutions are computed for a fixed γ.

Usage

```
cv.sglfit(x, y, lambda = NULL, gamma = 1.0, gindex = 1:p,
  nfolds = 10, foldid, parallel = FALSE, ...)
```

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 response variable.
lambda	a user-supplied lambda sequence. By leaving this option unspecified (recommended), users can have the program compute its own λ sequence based on nlambda and γ lambda.factor. It is better to supply, if necessary, a decreasing sequence of lambda values than a single (small) value, as warm-starts are used in the optimization algorithm. The program will ensure that the user-supplied lambda sequence is sorted in decreasing order before fitting the model.
gamma	sg-LASSO mixing parameter. γ = 1 gives LASSO solution and γ = 0 gives group LASSO solution.
gindex	p by 1 vector indicating group membership of each covariate.
nfolds	number of folds of the cv loop. Default set to 10.
foldid	the fold assignments used.
parallel	if TRUE, use parallel foreach to fit each fold. Must register parallel before hand, such as doMC or others. See the example below.
...	Other arguments that can be passed to sglfit.

Details

The cross-validation is run for sg-LASSO linear model. The sequence of linear regression models implied by λ vector is fit by block coordinate-descent. The objective function is

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

where $\iota \in R^T$ and $\|u\|_T^2 = \langle u, u\rangle/T$ is the empirical inner product. 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

cv.sglfit object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
```

```
gindex = sort(rep(1:4,times=5))
cv.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE)
## Not run:
# Parallel
require(doMC)
registerDoMC(cores = 2)
x = matrix(rnorm(1000 * 20), 1000, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(1000)
gindex = sort(rep(1:4,times=5))
system.time(cv.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE))
system.time(cv.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE, parallel = TRUE))

## End(Not run)
```

`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)
```

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 α = 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)
```

`ic.panel.sglfit` *Information criteria fit for panel sg-LASSO*

Description

Does information criteria for panel data sg-LASSO regression model.

The function runs sglfit 1 time; computes the path solution in `lambda` sequence. Solutions for `BIC`, `AIC` and `AICc` information criteria are returned.

Usage

```
ic.panel.sglfit(x, y, lambda = NULL, gamma = 1.0, gindex = 1:p,
                method = c("pooled","fe"), nf = NULL, ...)
```

Arguments

`x`	NT by p data matrix, where NT and p respectively denote the sample size of pooled data and the number of regressors.
`y`	NT by 1 response variable.
`lambda`	a user-supplied lambda sequence. By leaving this option unspecified (recommended), users can have the program compute its own λ sequence based on `nlambda` and `lambda.factor`. It is better to supply, if necessary, a decreasing sequence of lambda values than a single (small) value, as warm-starts are used in the optimization algorithm. The program will ensure that the user-supplied λ sequence is sorted in decreasing order before fitting the model.
`gamma`	sg-LASSO mixing parameter. $\gamma = 1$ gives LASSO solution and $\gamma = 0$ gives group LASSO solution.
`gindex`	p by 1 vector indicating group membership of each covariate.
`method`	choose between 'pooled' and 'fe'; 'pooled' forces the intercept to be fitted in sglfit, 'fe' computes the fixed effects. User must input the number of fixed effects `nf` for `method = 'fe'`. Default is set to `method = 'pooled'`.
`nf`	number of fixed effects. Used only if `method = 'fe'`.
`...`	Other arguments that can be passed to sglfit.

Details

The sequence of linear regression models implied by λ vector is fit by block coordinate-descent. The objective function is either (case `method='pooled'`)

$$\|y - \iota\alpha - x\beta\|_{NT}^2 + 2\lambda\Omega_\gamma(\beta),$$

where $\iota \in R^{NT}$ and α is common intercept to all N items or (case `method='fe'`)

$$\|y - B\alpha - x\beta\|_{NT}^2 + 2\lambda\Omega_\gamma(\beta),$$

where $B = I_N \times \iota$ and $\|u\|_{NT}^2 = \langle u, u \rangle / NT$ is the empirical inner product. 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

ic.panel.sglfit object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
ic.panel.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE)
```

---

ic.sglfit *Information criteria fit for sg-LASSO*

Description

Does information criteria for sg-LASSO regression model.

The function runs sglfit 1 time; computes the path solution in `lambda` sequence. Solutions for `BIC`, `AIC` and `AICc` information criteria are returned.

Usage

```
ic.sglfit(x, y, lambda = NULL, gamma = 1.0, gindex = 1:p,  ...)
```

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 response variable.

`lambda`	a user-supplied lambda sequence. By leaving this option unspecified (recommended), users can have the program compute its own λ sequence based on `nlambda` and `lambda.factor`. It is better to supply, if necessary, a decreasing sequence of lambda values than a single (small) value, as warm-starts are used in the optimization algorithm. The program will ensure that the user-supplied λ sequence is sorted in decreasing order before fitting the model.
`gamma`	sg-LASSO mixing parameter. γ = 1 gives LASSO solution and γ = 0 gives group LASSO solution.
`gindex`	p by 1 vector indicating group membership of each covariate.
`...`	Other arguments that can be passed to sglfit.

Details

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

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

where $\iota \in R^T$ and $\|u\|_T^2 = \langle u, u\rangle/T$ is the empirical inner product. 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

ic.sglfit object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
ic.sglfit(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE)
```

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)
```

market_ret *SNP500 returns*

Description

SNP500 returns

Usage

```
data(market_ret)
```

Format

A data.frame object.a

Source

market_ret - FRED

Examples

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

---

midas.ardl *MIDAS regression*

Description

Fits MIDAS regression model with single high-frequency covariate. Options include linear-in-parameters polynomials (e.g. Legendre) or non-linear polynomials (e.g. exponential Almon). Non-linear polynomial optimization routines are equipped with analytical gradients, which allows fast and accurate optimization.

Usage

```
midas.ardl(y, x, z = NULL, loss_choice = c("mse","logit"),
           poly_choice = c("legendre","expalmon","beta"),
           poly_spec = 0, legendre_degree = 3, nbtrials = 500)
```

Arguments

y	response variable. Continuous for loss_choice = "mse", binary for loss_choice = "logit".
x	high-frequency covariate lags.
z	other lower-frequency covariate(s) or AR lags (both can be supplied in an appended matrix). Either must be supplied.
loss_choice	which loss function to fit: loss_choice="mse" fits least squares MIDAS regression, loss_choice="logit" fits logit MIDAS regression.
poly_choice	which MIDAS lag polynomial function to use: poly_choice="expalmon" - exponential Almon polynomials, poly_choice="beta" - Beta density function (need to set poly_spec), poly_choice="legendre" - legendre polynomials (need to set legendre_degree). Default is set to poly_choice="expalmon".
poly_spec	which Beta density function specification to apply (applicable only for poly_choice="beta"). poly_spec = 0 - all three parameters are fitted, poly_spec = 1 (θ_2, θ_3) are fitted, poly_spec = 2 (θ_1, θ_2) are fitted, poly_spec = 3 (θ_2) is fitted. Default is set to poly_spec = 0.

`legendre_degree`
: the degree of legendre polynomials (applicable only for `legendre="beta"`). Default is set to 3.

`nbtrials`
: number of initial values tried in multistart optimization. Default is set to `poly_spec = 500`.

Details

Several polynomial functional forms are available (`poly_choice`):
- `beta`: Beta polynomial
- `expalmon`: Exp Almon polynomial
- `legendre`: Legendre polynomials.

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

midas.ardl object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
z = rnorm(100)
y = rnorm(100)
midas.ardl(y = y, x = x, z = z)
```

---

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	m by 1 high-frequency time series data vector.
data.xdate	m 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_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	m by 1 high-frequency time series data vector.
data.xdate	m 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"))
```

`predict.cv.panel.sglfit` *Computes prediction*

Description

Similar to other predict methods, this functions predicts fitted values from a fitted sglfit object.

Usage

```
## S3 method for class 'cv.panel.sglfit'
predict(
  object,
  newx,
  s = c("lam.min", "lam.1se"),
  type = c("response"),
  method = c("pooled", "fe"),
  ...
)
```

Arguments

`object`	fitted `cv.panel.sglfit` model object.
`newx`	matrix of new values for x at which predictions are to be made. NOTE: `newx` must be a matrix, predict function does not accept a vector or other formats of newx.
`s`	choose between 'lam.min' and 'lam.1se'.
`type`	type of prediction required. Only response is available. Gives predicted response for regression problems.
`method`	choose between 'pooled', and 'fe'.
`...`	Not used. Other arguments to predict.

Details

`s` is the new vector at which predictions are to be made. If `s` is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right $lambda$ indices.

Value

The object returned depends on type.

`predict.cv.sglfit` *Computes prediction*

Description

Similar to other predict methods, this functions predicts fitted values from a fitted sglfit object.

Usage

```
## S3 method for class 'cv.sglfit'
predict(object, newx, s = c("lam.min", "lam.1se"), type = c("response"), ...)
```

Arguments

`object`	fitted `cv.sglfit` model object.
`newx`	matrix of new values for x at which predictions are to be made. NOTE: `newx` must be a matrix, predict function does not accept a vector or other formats of newx.
`s`	choose between 'lam.min' and 'lam.1se'.
`type`	type of prediction required. Only response is available. Gives predicted response for regression problems.
`...`	Not used. Other arguments to predict.
`method`	choose between 'single', 'pooled', and 'fe'.

Details

`s` is the new vector at which predictions are to be made. If `s` is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right $lambda$ indices.

Value

The object returned depends on type.

`predict.ic.panel.sglfit` *Computes prediction*

Description

Similar to other predict methods, this functions predicts fitted values from a fitted sglfit object.

Usage

```
## S3 method for class 'ic.panel.sglfit'
predict(
  object,
  newx,
  s = c("bic", "aic", "aicc"),
  type = c("response"),
  method = c("pooled", "fe"),
  ...
)
```

Arguments

`object`	fitted `ic.panel.sglfit` model object.
`newx`	matrix of new values for x at which predictions are to be made. NOTE: `newx` must be a matrix, predict function does not accept a vector or other formats of newx.
`s`	choose between 'bic', 'aic', and 'aicc'.
`type`	type of prediction required. Only response is available. Gives predicted response for regression problems.
`method`	choose between 'pooled', and 'fe'.
`...`	Not used. Other arguments to predict.

Details

`s` is the new vector at which predictions are to be made. If s is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right $lambda$ indices.

Value

The object returned depends on type.

predict.ic.sglfit *Computes prediction*

Description

Similar to other predict methods, this functions predicts fitted values from a fitted sglfit object.

Usage

```
## S3 method for class 'ic.sglfit'
predict(object, newx, s = c("bic", "aic", "aicc"), type = c("response"), ...)
```

Arguments

object	fitted cv.sglfit model object.
newx	matrix of new values for x at which predictions are to be made. NOTE: newx must be a matrix, predict function does not accept a vector or other formats of newx.
s	choose between 'bic', 'aic', and 'aicc'.
type	type of prediction required. Only response is available. Gives predicted response for regression problems.
...	Not used. Other arguments to predict.

Details

s is the new vector at which predictions are to be made. If s is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right $lambda$ indices.

Value

The object returned depends on type.

predict.sglpath *Computes prediction*

Description

Similar to other predict methods, this functions predicts fitted values from a fitted sglfit object.

Usage

```
## S3 method for class 'sglpath'
predict(
  object,
  newx,
  s = NULL,
  type = c("response"),
  method = c("single", "pooled", "fe"),
  ...
)
```

Arguments

object	fitted sglfit model object.
newx	matrix of new values for x at which predictions are to be made. NOTE: newx must be a matrix, predict function does not accept a vector or other formats of newx.
s	value(s) of the penalty parameter $lambda$ at which predictions are to be made. Default is the entire sequence used to create the model.
type	type of prediction required. Only response is available. Gives predicted response for regression problems.
method	choose between 'single', 'pooled', and 'fe'.
...	Not used. Other arguments to predict.

Details

s is the new vector at which predictions are to be made. If s is not in the lambda sequence used for fitting the model, the predict function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right $lambda$ indices.

Value

The object returned depends on type.

---

reg.panel.sgl *Regression fit for panel sg-LASSO*

Description

Fits panel data sg-LASSO regression model.

The function fits sg-LASSO regression based on chosen tuning parameter selection method_choice. Options include cross-validation and information criteria.

Usage

```
reg.panel.sgl(x, y, gamma = NULL, gindex, intercept = TRUE,
              method_choice = c("ic","cv"), nfolds = 10,
              method = c("pooled", "fe"), nf = NULL,
              verbose = FALSE, ...)
```

Arguments

x	NT by p data matrix, where NT and p respectively denote the sample size of pooled data and the number of regressors.
y	NT by 1 response variable.
gamma	sg-LASSO mixing parameter. γ = 1 gives LASSO solution and γ = 0 gives group LASSO solution.
gindex	p by 1 vector indicating group membership of each covariate.
intercept	whether intercept be fitted (TRUE) or set to zero (FALSE). Default is TRUE.
method_choice	choose between ic and cv. ic gives fit based on information criteria (BIC, AIC or AICc) by running ic.fit, while cv gives fit based on cross-validation by running cv.sglfit. If cv is chosen, optional number of folds nfolds can be supplied.
nfolds	number of folds of the cv loop. Default set to 10.
method	choose between 'pooled' and 'fe'; 'pooled' forces the intercept to be fitted in sglfit, 'fe' computes the fixed effects. User must input the number of fixed effects nf for method = 'fe', and it is recommended to do so for method = 'pooled'. Program uses supplied nf to construct foldsid if method_choice = 'cv' is chosen. Default is set to method = 'pooled'.
nf	number of fixed effects. Used only if method = 'fe'.
verbose	flag to print information.
...	Other arguments that can be passed to sglfit.

Details

The sequence of linear regression models implied by λ vector is fit by block coordinate-descent. The objective function is either (case method='pooled')

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

where $\iota \in R^{NT}$ and α is common intercept to all N items or (case method='fe')

$$\|y - B\alpha - x\beta\|_T^2 + 2\lambda\Omega_\gamma(\beta),$$

where $B = I_N \times \iota$ and $\|u\|_{NT}^2 = \langle u, u\rangle/NT$ is the empirical inner product. 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

reg.panel.sgl object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
reg.panel.sgl(x = x, y = y,
  gindex = gindex, gamma = 0.5,
  method = "fe", nf = 10,
  standardize = FALSE, intercept = FALSE)
```

`reg.sgl` *Fit for sg-LASSO regression*

Description

Fits sg-LASSO regression model.

The function fits sg-LASSO regression based on chosen tuning parameter selection `method_choice`. Options include cross-validation and information criteria.

Usage

```
reg.sgl(x, y, gamma = NULL, gindex, intercept = TRUE,
        method_choice = c("ic","cv"), nfolds = 10,
        verbose = FALSE, ...)
```

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 response variable.
`gamma`	sg-LASSO mixing parameter. $\gamma = 1$ gives LASSO solution and $\gamma = 0$ gives group LASSO solution.
`gindex`	p by 1 vector indicating group membership of each covariate.
`intercept`	whether intercept be fitted (`TRUE`) or set to zero (`FALSE`). Default is `TRUE`.

`method_choice`	choose between `ic` and `cv`. `ic` gives fit based on information criteria (BIC, AIC or AICc) by running `ic.fit`, while `cv` gives fit based on cross-validation by running `cv.sglfit`. If `cv` is chosen, optional number of folds `nfolds` can be supplied.
`nfolds`	number of folds of the cv loop. Default set to `10`.
`verbose`	flag to print information.
`...`	Other arguments that can be passed to `sglfit`.

Details

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

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

where $\iota \in R^T$ and $\|u\|_T^2 = \langle u, u\rangle/T$ is the empirical inner product. 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

reg.sgl object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
reg.sgl(x = x, y = y, gindex = gindex, gamma = 0.5,
  standardize = FALSE, intercept = FALSE)
```

rgdp_dates *Real GDP release dates*

Description

Real GDP release dates

Usage

```
data(rgdp_dates)
```

Format

A list objects

Source

ALFRED

Examples

```
data(rgdp_dates)
rgdp_dates$Quarter_q # reference quarters in quarters
rgdp_dates$Quarter_m # reference quarters in months
rgdp_dates$Quarter_d # reference quarters in days
rgdp_dates$`First release` # first release date for the reference
rgdp_dates$`Second release` # second release date for the reference
rgdp_dates$`Third release` # third release date for the reference
```

rgdp_vintages *Real GDP vintages*

Description

Real GDP vintages

Usage

```
data(rgdp_vintages)
```

Format

A list objects

Source

ALFRED

Examples

```
data(rgdp_vintages)
rgdp_vintages$date # dates
rgdp_vintages$time_series # series, q-q annual rate
rgdp_vintages$realtime_period # real time dates
```

`sglfit` *Fits sg-LASSO regression*

Description

Fits sg-LASSO regression model. The function fits sg-LASSO regression model for a sequence of λ tuning parameter and fixed γ tuning parameter. The optimization is based on block coordinate-descent. Optionally, fixed effects are fitted.

Usage

```
sglfit(x, y, gamma = 1.0, nlambda = 100L, method = c("single", "pooled", "fe"),
       nf = NULL, lambda.factor = ifelse(nobs < nvars, 1e-02, 1e-04),
       lambda = NULL, pf = rep(1, nvars), gindex = 1:nvars,
       dfmax = nvars + 1, pmax = min(dfmax * 1.2, nvars), standardize = FALSE,
       intercept = FALSE, eps = 1e-08, maxit = 1000000L, peps = 1e-08)
```

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 response variable.
`gamma`	sg-LASSO mixing parameter. $\gamma = 1$ gives LASSO solution and $\gamma = 0$ gives group LASSO solution.
`nlambda`	number of λ's to use in the regularization path; used if `lambda = NULL`.
`method`	choose between 'single', 'pooled' and 'fe'; 'single' implies standard sg-LASSO regression, 'pooled' forces the intercept to be fitted, 'fe' computes the fixed effects. User needs to input the number of fixed effects `nf`. Default is set to 'single'.
`nf`	number of fixed effects. Used only if `method = 'fe'`.

`lambda.factor` The factor for getting the minimal λ in the λ sequence, where `min(lambda) = lambda.factor * max(lambda)`. max(lambda) is the smallest value of lambda for which all coefficients are zero. λ_{max} is determined for each γ tuning parameter separately. The default depends on the relationship between `T` (the sample size) and `p` (the number of predictors). If `T < p`, the default is `0.01`. If `T > p`, the default is `0.0001`, closer to zero. The smaller the value of `lambda.factor` is, the denser is the fit for λ_{min}. Used only if `lambda = NULL`.

`lambda` a user-supplied lambda sequence. By leaving this option unspecified (recommended), users can have the program compute its own `lambda` sequence based on `nlambda` and `lambda.factor`. It is better to supply, if necessary, a decreasing sequence of lambda values than a single (small) value, as warm-starts are used in the optimization algorithm. The program will ensure that the user-supplied λ sequence is sorted in decreasing order before fitting the model.

`pf` the ℓ_1 penalty factor of length `p` used for the adaptive sg-LASSO. Separate ℓ_1 penalty weights can be applied to each coefficient to allow different ℓ_1 + $\ell_{2,1}$ shrinkage. Can be 0 for some variables, which imposes no shrinkage, and results in that variable always be included in the model. Default is 1 for all variables.

`gindex` p by 1 vector indicating group membership of each covariate.

`dfmax` the maximum number of variables allowed in the model. Useful for very large `p` when a partial path is desired. Default is `p+1`. In case `method='fe'`, `dfmax` is ignored.

`pmax` the maximum number of coefficients allowed ever to be nonzero. For example, once $\beta_i \neq 0$ for some $i \in [p]$, no matter how many times it exits or re-enters the model through the path, it will be counted only once. Default is `min(dfmax*1.2,p)`.

`standardize` logical flag for variable standardization, prior to fitting the model sequence. The coefficients are always returned to the original scale. It is recommended to keep `standardize=TRUE`. Default is `FALSE`.

`intercept` whether intercept be fitted (`TRUE`) or set to zero (`FALSE`). Default is `FALSE`. In case `method='pooled'`, `intercept=TRUE` is forced. In case `method='fe'`, `intercept=FALSE` is forced and `entity` specific intercepts are fitted in a separate output variable `a0`.

`eps` convergence threshold for block coordinate descent. Each inner block coordinate-descent loop continues until the maximum change in the objective after any coefficient update is less than thresh times the null deviance. Defaults value is `1e-8`.

`maxit` maximum number of outer-loop iterations allowed at fixed lambda values. Default is `1e6`. If the algorithm does not converge, consider increasing `maxit`.

`peps` convergence threshold for proximal map of sg-LASSO penalty. Each loop continues until G group difference sup-norm, $\|\beta_G^k - \beta_G^{k-1}\|_\infty$, is less than `peps`. Defaults value is `1e-8`.

Details

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

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

where $\iota \in R^T$ and $\|u\|_T^2 = \langle u, u \rangle / T$ is the empirical inner product. 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

sglfit object.

Author(s)

Jonas Striaukas

Examples

```
set.seed(1)
x = matrix(rnorm(100 * 20), 100, 20)
beta = c(5,4,3,2,1,rep(0, times = 15))
y = x%*%beta + rnorm(100)
gindex = sort(rep(1:4,times=5))
sglfit(x = x, y = y, gindex = gindex, gamma = 0.5)
```

---

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 `list` object.a

Source

rgdp
cfnai
payems
ads

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