Package: ConsReg (via r-universe)

November 6, 2024

Type Package

Title Fitting Regression Models & Regression with Arma Errors, Subject to Constraints and Restrictions to the Coefficients

Version 0.1.0

Author Josep Puig Sallés

Maintainer Josep Puig <puigjos@gmail.com>

Description Fits or generalized linear models either a regression with arma errors (Time series) where the parameters could be subject to constraints or restrictions. The model is specified by an objective function (Gaussian, Binomial or Poisson) or an Arma order (p,q), a vector of bound constraints for the coefficients and the possibility to incorporate restrictions among coefficients (i.e b1 > b2).

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown

Imports data.table (>= 1.10), forecast (>= 8.0), rlang (>= 0.4), nloptr (>= 1.2), FME (>= 1.3), MCMCpack (>= 1.4), Rsolnp (>= 1.15), DEoptim (>= 2.2), dfoptim, GA (>= 3.0), GenSA (>= 1.1), Metrics, ggplot2, adaptMCMC, Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Config/pak/sysreqs cmake make libssl-dev

Repository https://puigjos.r-universe.dev

RemoteUrl https://github.com/puigjos/consreg

RemoteRef HEAD

RemoteSha 50b7df1faaccc888036bed3622c4651e669be672

Contents

`ConsReg` *Fit a regression model with gaussian or binomial objective function*

Description

ConsReg is a function that allows to estimate a regression model: linear regression (gaussian), logistic regression (binomial) or poisson regression. It allows the introduction of restrictions (both lower and upper limits) and restrictions between the coefficients (in the form, for example, of a>b).

Usage

```
ConsReg(...)

## Default S3 method:
ConsReg(x, y, family, optimizer, ini.pars.coef = NULL,
  constraints = NULL, LOWER = NULL, UPPER = NULL, penalty = 1000,
  ...)

## S3 method for class 'formula'
ConsReg(formula, data = list(), optimizer = "solnp",
  family = c("gaussian", "binomial"), constraints = NULL,
  LOWER = NULL, UPPER = NULL, penalty = 1000,
  na.action = "na.omit", ini.pars.coef = NULL, ...)
```

Arguments

`...`	additional parameters passed in the optimizer (number of iterations, ...)
`x`	matrix of predictive variables
`y`	vector of outcome variable
`family`	a description of the error distribution and link function to be used in the model. Possible values are: "gaussian" (linear regression) or "binomial" (logistic regression) and "poisson"

optimizer	Optimizer package used for fit the model (include bayesian and genetic algorithm optimization). Possible values are: "solnp" (default) (Rsolnp), "gosonlp" (Rsolnp), "optim" (stats::optim), "nloptr" (nloptr), DEoptim ("DEoptim"), "dfoptim" (dfoptim), "mcmc" (FME::modMCMC), "MCMCmetrop" (MCMCpack::MCMCmetrop1R),'adaptM "GA" (GA package), "GenSA" (GenSA package)
ini.pars.coef	vector of initial parameters. In case there is some constraint, then the ini.pars.coef should fulfill the constraints
constraints	vector of constraints (see details)
LOWER	(default NULL) vector of lower bounds for the coefficients. If the lenght of LOWER is not equal with the length of the coeefficients, then, the rest will be equal to -Inf
UPPER	(default NULL) vector of lower bounds for the coefficients. If the lenght of UPPER is not equal with the length of the coeefficients, then, the rest will be equal to +Inf
penalty	(default 1000) penalty to the objective function if some constraints do not fullfill
formula	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
na.action	na.action to the data

Details

Several optimizers of various R packages are implemented, including methods typically used in Bayesian regressions like Markov Chain Monte Carlo simulation.

Constraints will be a string: For example, if x1 and x2 are two coefficient names, then a constraint could be: "x1 > x2" or "x1+x2 > 2". For some constraints, one can write: "x1+x2>2, x1 > 1". Each constraint will be separate by commas.

Important: if there are some constraints that do not fulfill in a model without those constraints, it is recommended to use `ini.pars.coef` parameter to set initial values that fulfill constraints. See the example

Value

An object of class "ConsReg".

coefficients	Coefficients of the regression
hessian	hessian matrix if the optimizer can return it
family	Model family function
optimizer	optimizer object return (see details of each optimization package)
optimizer.name	name of the optimizer
df	nrow(data) - number of coefficients

rank	number of coefficients
residuals	residuals of the model
fitted	fitted values of the model
metrics	Accuracy metrics of the model
call	the matched call
y	objective variable
x	regressors
formula	formula term

Author(s)

Josep Puig Sallés

Examples

```
data('fake_data')
fit1 = ConsReg(formula = y~x1+x2+x3+ I(x3^2) + x4, family = 'gaussian',
                     optimizer = 'mcmc',
                     data = fake_data)
summary(fit1)

# We impose constraints to x3 and x3^2 and x4
fit2 = ConsReg(formula = y~x1+x2+x3+ I(x3^2) + x4, data = fake_data,
            family = 'gaussian',
            constraints = '(x3 + `I(x3^2)`) > .01, x4 < .2',
            optimizer = 'mcmc',
            ini.pars.coef = c(-1.65, .12, -.004, 0.1, 0.1, .15))

fit1$coefficients
fit2$coefficients
```

ConsRegArima *Fit regression model with Arma errors to univariate time series*

Description

ConsRegArima is a function that allows to estimate a regression model with errors following an ARMA process (p,q). It allows the introduction of restrictions (both lower and upper limits) and restrictions between the coefficients (in the form, for example, of a>b).

Usage

```
ConsRegArima(...)

## Default S3 method:
ConsRegArima(x, y, order, seasonal, optimizer,
  LOWER = NULL, UPPER = NULL, penalty = 1000, constraints = NULL,
  ini.pars.coef, model_fit = NULL, ...)

## S3 method for class 'formula'
ConsRegArima(formula, data = list(),
  optimizer = c("solnp"), order = c(0, 0), seasonal = list(order =
  c(0, 0), period = NA), LOWER = NULL, UPPER = NULL, penalty = 1000,
  constraints = NULL, ini.pars.coef = NULL, na.action = "na.omit",
  ...)
```

Arguments

`...`	additional parameters passed in the optimizer (number of iterations, ...)
`x`	matrix of predictive variables
`y`	vector of outcome variable
`order`	Arma component (p, q)
`seasonal`	A specification of the seasonal part of the ARMA model (P,Q), plus the period (which defaults to 1).
`optimizer`	Optimizer package used for fit the model (include bayesian and genetic algorithm optimization). Possible values are: "solnp" (default) (Rsolnp), "gosonlp" (Rsolnp), "optim" (stats::optim), "nloptr" (nloptr), DEoptim ("DEoptim"), "dfoptim" (dfoptim), "mcmc" (FME::modMCMC), "MCMCmetrop" (MCMCpack::MCMCmetrop1R), 'adaptMCMC'(adaptMCMC::MCMC), "GA" (GA package), "GenSA" (GenSA package)
`LOWER`	(default NULL) vector of lower bounds for the coefficients. If the lenght of LOWER is not equal with the length of the coeefficients, then, the rest will be equal to -Inf
`UPPER`	(default NULL) vector of lower bounds for the coefficients. If the lenght of UPPER is not equal with the length of the coeefficients, then, the rest will be equal to +Inf
`penalty`	(default 1000) penalty to the objective function if some constraints do not fullfill
`constraints`	vector of constraints (see details)
`ini.pars.coef`	vector of initial parameters. In case there is some constraint, then the ini.pars.coef should fulfill the constraints. This vector is only for regression component.
`model_fit`	object of class `ConsRegArima` to update the Arma part and fix the coefficient from a previous model
`formula`	an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted

data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.
na.action	na.action to the data

Details

Several optimizers of various R packages are implemented, including methods typically used in Bayesian regressions like Markov Chain Monte Carlo simulation.

Constraints will be a string: For example, if x1 and x2 are two coefficient names, then a constraint could be: "x1 > x2" or "x1+x2 > 2". For some constraints, one can write: "x1+x2>2, x1 > 1". Each constraint will be separate by commas.

Important: if there are some constraints that do not fulfill in a model without those constraints, it is recommended to use `ini.pars.coef` parameter to set initial values that fulfill constraints. See the example

On the other hand, aic value is computed as auto.arima function computes the AIC when method == 'CSS':

$$AIC = n * log(sigma2) + npar * 2$$

Where npar I set the number of coefficients.

Value

An object of class "`ConsRegArima`".

coefficients	Coefficients (regression + arma errors)
hessian	hessian matrix if the optimizer can return it
optimizer	optimizer object return (see details of each optimization package)
optimizer.name	name of the optimizer
df	nrow(data) - number of coefficients
rank	number of coefficients
objective_function	objective_function used
model	A list representing the Kalman Filter used in the fitting
sigma2	the MLE of the innovations variance
residuals	residuals of the model
fitted	fitted values of the model
fitted_regression	fitted values only of the regression part
fitted_arima	fitted values only of the arma part
metrics	Accuracy metrics of the model (accuracy function of the forecast package)
call	the matched call
y	objective series

x	regressors
formula	formula term
aic	the AIC value (see details)
bic	the BIC value
aicc	the AICc value

Author(s)

Josep Puig Salles

References

Peiris, M. & Perera, B. (1988), On prediction with fractionally Hyndman RJ, Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R."

Examples

```
data('series')
fit1 = ConsRegArima(formula = y ~ x1+x2 +x3+x4,
                    order = c(2, 1), data = series)
summary(fit1)
fit2 = ConsRegArima(formula = y ~ x1+x2 +x3+x4, order = c(2, 1),
                    data = series, constraints = '(x3 +.1) > x1',
                    ini.pars.coef = c(.96, .2, -.8, .4), UPPER = 1, LOWER = -1)

fit1$coefficients
fit2$coefficients
```

---

fake_data *Fake data for regression*

Description

Fake data to show gaussian model example

Usage

```
fake_data
```

Format

data

Examples

```
data('fake_data')
```

plot.roll.ConsRegArima *Plot an roll object plot an roll.ConsRegArima object*

Description

Plot an roll object plot an roll.ConsRegArima object

Usage

```
## S3 method for class 'roll.ConsRegArima'
plot(x, ...)
```

Arguments

x	roll.ConsRegArima object
...	Additional params passed to ggplot2::labs function

predict.ConsReg *Predict or fitted values of object* ConsReg

Description

Predict or fitted values of object ConsReg

Usage

```
## S3 method for class 'ConsReg'
predict(object, newdata = NULL, components = F, ...)
```

Arguments

object	object of class ConsReg
newdata	New data to predict the objective function. If is NULL (default), then the fitted values will be returned
components	if its TRUE, it will return the predictions for each regression component
...	Additional argument passed to family. In particular, at this moment, if type = 'link', then for binomial family, it will return the link values

Value

predictions

Examples

```
data('fake_data')
data = fake_data
data$y = 1/(1+exp(-data$y))
data$y = ifelse(data$y > .5, 1, 0)
table(data$y)

fit5 = ConsReg(y~x1+x2+x3+x4, data = data,
               family = 'binomial', penalty = 10000,
               LOWER = -.5, UPPER = .2,
               optimizer = 'gosolnp')
pr = predict(fit5, newdata = data[1:3,], type = 'probability')
pr
```

`predict.ConsRegArima` *Predict function for ConsRegArima object*

Description

Obtains predictions of ConsRegArima object

Usage

```
## S3 method for class 'ConsRegArima'
predict(object, h = ifelse(is.null(newdata), 1,
  nrow(newdata)), newdata = NULL, intervals = 90, origdata = NULL,
  ...)

## S3 method for class 'predict.ConsRegArima'
print(x, ...)

## S3 method for class 'predict.ConsRegArima'
plot(x, ...)
```

Arguments

`object`	ConsRegArima object
`h`	horizont to predict
`newdata`	data frame in which to look for variables with which to predict. In case there is no regression part, this parameter could be set NULL
`intervals`	Confidence level for prediction intervals (default 90)
`origdata`	Original data (default NULL). Useful if lagged predictive variables are used in the formula
`...`	Additional params passed to the function ggplot2::labs
`x`	object of class predict.ConsRegArima

Value

Returns an object of class predict.ConsRegArima

predict	dataframe with the predictions
table	dataframe with the predictions as well as the fitted values
intervals	Interval level
object	original object

---

rolling *rolling: Back-test your model*

Description

Function for creating rolling density forecast from ConsRegArima models with option for refitting every n periods.

Usage

```
rolling(object, used.sample, refit, h = 1, orig.data, ...)
```

Arguments

object	ConsRegArima object
used.sample	The starting point in the dataset from which to initialize the rolling forecast.
refit	Determines every how many periods the model is re-estimated. If refit=0, then no refit is doing
h	The number of periods to forecast
orig.data	data original which was used to estimate the ConsRegArima object
...	Additional params for predict function

Value

results	data.frame with Real, Prediction, Prediction_High, Prediction_Low and fitted values of the object
refitT	how many periods the model is re-estimated
metrics	Main metrics of the predictions

See Also

plot.roll.ConsRegArima

Examples

```
data('series')
fit1 = ConsRegArima(formula = y ~ x1+x2 +x3, order = c(2, 1),
                    optimizer = 'solnp', data = series)
roll = rolling(fit1, used.sample = 40,
               refit = 5, orig.data = series, h=3)
roll
plot(roll)
```

---

sales *Sales data set*

Description

Sales data set for time series models

Usage

```
sales
```

Format

data

Examples

```
data('series')
```

---

series *Fake data for time series*

Description

Fake data to show Arima example

Usage

```
series
```

Format

data

Index