Package ‘carx’

November 20, 2017

Title Censored Autoregressive Model with Exogenous Covariates

Version 0.7.1

Description A censored time series class is designed. An estimation procedure is implemented to estimate the Censored AutoRegressive time series with eXogenous covariates (CARX), assuming normality of the innovations. Some other functions that might be useful are also included.

Depends R (>= 1.9.0)

License GPL-3

LazyData true

Imports tmvtnorm, mvtnorm, matrixStats, xts, zoo, nlme, grDevices, graphics, stats

Author Chao Wang [aut, cre], Kung-Sik Chan [aut]

Maintainer Chao Wang <`chao-wang@uiowa.edu`>

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-11-20 04:24:20 UTC

R topics documented:

`AIC.carx` *Compute the AIC of a fitted* `carx` *object*

Description

Return the AIC of a fitted `carx` object where the maximum log-likelihood is replaced by the maximum quasi-log-likelihood.

Usage

```
## S3 method for class 'carx'
AIC(object, ..., k = 2)
```

Arguments

`object`	a fitted `carx` object.
`...`	not used.
`k`	penalty multiplier for the number of parameters. Default = 2.

Value

the AIC value = -2*maximum quasi-log-likelihood*+*k*number of parameters

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
ic = AIC(mdl)
```

`carx` *`carx`: A package to fit Censored Auto-Regressive model with eXogenous covariates (CARX)*

Description

`carx` is a package to estimate the parameters of the Censored AutoRegressive model with eXogenous covariates (CARX), which can also be viewed as regression models with censored responses and autoregressive residuals. `carx` allows left, right, or interval censoring for the response variable. The regression errors are assumed to follow an autoregressive model with normal innovations. In addition to the estimation method, the package also contains functions to predict future values, diagnose whether the model is adequate, and plot functions to illustrate the data and model.

Details

More specifically, we estimate the parameters assumed in the following model. Let (Y_t) be a censored time series with the latent process denoted by (Y_t^*). For each Y_t^*, it can be censored by either $(-\infty, c_{l,t})$ or $(c_{u,t}, \infty)$, and if it is censored, Y_t will be recorded as $c_{l,t}$ or $c_{u,t}$ respectively.

The latent process (Y_t^*) is modelled as

$$Y_t^* = X_t'\beta + \eta_t,$$

and

$$\eta_t = \sum_{i=1}^{p} \psi_i \eta_{t-i} + \varepsilon_t,$$

where (X_t) is a covariate process with all values observable, and the innovations (ε_t) are independent and identically normally distributed with mean 0 and variance σ^2.

In this package we implemented the quasi-maximum likelihood estimator proposed by Wang and Chan (2017), for more details, please refer to the paper.

References

Wang C, Chan KS (2017). "Quasi-likelihood estimation of a censored autoregressive model with exogenous variables." Journal of the American Statistical Association. 2017 Mar 20(just-accepted).

`carx.default` *The default estimation method for a CARX model*

Description

Estimate a CARX model, and compute the standard errors and confidence intervals of the parameter estimates by parametric bootstrap.

Usage

```
## Default S3 method:
carx(y, x = NULL, ci = NULL, lcl = NULL, ucl = NULL,
  p = 1, prmtrX = NULL, prmtrAR = NULL, sigma = NULL,
  y.na.action = c("skip", "as.censored"), addMu = TRUE, tol = 1e-04,
  max.iter = 500, CI.compute = FALSE, CI.level = 0.95, b = 1000,
  b.robust = FALSE, b.show.progress = FALSE, init.method = c("biased",
  "consistent"), cenTS = NULL, verbose = FALSE, seed = NULL, ...)
```

Arguments

`y`	a vector of possibly censored responses.
`x`	a matrix of covariates, or some object which can be coerced to matrix. Default=`NULL`, indicating a pure AR model for `y`.
`ci`	a vector of numeric values. If its i-th value is zero (negative, positive), then the corresponding element of `y` is uncensored (left, right censored). Default = `NULL`, indicating no censoring.
`lcl`	a vector of lower censoring limits, or a number assuming constant limit. Default = `NULL`, indicating no lower censoring limit.
`ucl`	a vector of upper censoring limits, or a number assuming constant limit. Default = `NULL`, indicating no upper censoring limit.
`p`	the AR order for the regression errors. Default = 1.
`prmtrX`	the initial values of the regression parameters for `x`. Default = `NULL`.
`prmtrAR`	the initial values of the AR coefficients. Default = `NULL`.
`sigma`	the initial value of the innovation (noise) standard deviation. Default = `NULL`.
`y.na.action`	a string indicating how to deal with missing (NA) values in `y`. If it is set to "skip" (default), cases containing a missing value will be skipped, so that the estimating equation of future cases will be conditional on the most recent `p` complete cases after the skipped case. If "as.censored", the `y` value will be treated as left-censored with lower censoring limit replaced by positive infinity. The user may choose to use "skip" if there exist few long gaps in the time series of response. Use "as.censored" if the missing values in `y` are many and scattered in time. N.B.: The presence of any missing values in `x` will automatically hard-code `y.na.action` to be "skip".
`addMu`	logical, indicating whether to include an intercept in case `x=NULL`. Default = `TRUE`.
`tol`	the tolerance level used in the stopping criterion. Default = 1.0e-4.
`max.iter`	maximum number of iterations allowed in the optimization. Default = 100.
`CI.compute`	logical value to indicate whether to compute the confidence intervals for the parameters. Default = `FALSE`, as it can be time-consuming to run the parametric bootstrap.
`CI.level`	numeric value in (0,1) representing the confidence level. Default = 0.95.
`b`	number of bootstrap replicates used for computing the boostrap confidence intervals. Default = 1000.

`b.robust`	logical, if `TRUE`, the innovations are re-sampled from the simulated residuals; otherwise they are re-sampled from a centered normal distribution with the estimated standard deviation. Default = `FALSE`.
`b.show.progress`	logical, if `TRUE`, a text bar will be displayed to show the progress of bootstrap when computing the confidence intervals of the parameter estimates.
`init.method`	a string selecting a procedure ("biased", or "consistent") for deteriming the initial values, in case there are no initial values for some parameters, i.e., one of `prmtrX`, `prmtrAR`, `sigma` is `NULL`. The "biased" method is always available, as it uses maximum gaussian likelihood estimator with the data replacing any censored observation by its corresponding censoring limit. The "consistent" method is only available for left censored data. Note that the "consistent" method may produce initial estimates with non-stationary AR coefficients or negative innovation variance, in which case, the function will fall back to the "biased" method for constucting the initial values (when this happens, the attribute `fallBackToBiased` of the returned object will be set to `TRUE`.) Default="biased".
`cenTS`	an optional argument to pass a `cenTS` object which contain the data used, when `carx.formula` is invoked. Default = `NULL`.
`verbose`	logical value indicates whether to print intermediate results for monitoring the the progress of the iterative estimation procedure. Default = FALSE.
`seed`	the random seed initialized at the beginning of the function. If a valid seed is supplied, the function will first store the current seed and set the seed according to the supplied seed, then restore the seed upon exit. Default = `NULL`.
`...`	not used.

Value

a CARX object of the estimated model.

See Also

`cenTS` on how to construct a `cenTS` object.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y=dat$y, x=dat[,c("X1","X2")], ci=dat$ci, lcl=dat$lcl, ucl=dat$ucl, p=2)
#or simply call
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
```

`carx.formula` *A formula interface to the* `carx` *method*

Description

This interface uses the supplied `formula` and data provided by `data` and other arguments in `...` to invoke the `carx.default` method. This is the preferred way of calling the `carx.default` function.

Usage

```
## S3 method for class 'formula'
carx(formula, data = list(), ...)
```

Arguments

`formula`	a formula representing the regression part of the model, such as `y ~ x1 + x2`.
`data`	a `list`, `data.frame`, or a `cenTS` object which includes the following: • the response variable with variable name identified by the supplied formula. • any covariate(s) with with variable name(s) identified by the supplied formula. • `ci` whose components take values from -1, 0, 1, where -1 (0,1) indicates that the corresponding element in the response variable is left-censored (not censored, right censored). • `lcl` which denotes the vector of left (lower) censoring limits. • `ucl` which denotes the vector of right (upper) censoring limits.
`...`	other parameters accepted by `carx.default` except `y`, `x`, `ci`, `lcl`, and `ucl`.

Value

a CARX object of the estimated model.

See Also

`carx.default` for more options.

`cenTS` on how to construct a `cenTS` object.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
```

`carxSelect` *Select a* `carx` *model by the AIC*

Description

This function selects the `carx` model which minimizes the AIC among a set of `carx` models defined by a set of formulas or a list of regression formulas with a maximal AR order. The model specification is supplied by `formulas` which can be either a formula or a list of formulas. For each formula, the function will estimate the `carx` models with the AR order from 1 to `max.ar` inclusive. If `detect.outlier=TRUE`, outlier detection will be performed for each combination of model formula and AR order. The function returns a `list` which consists of: 1) `aicMat` which is a matrix of AIC values where each row contains the AICs of the model given by a specific regression formula with the AR order ranging from 1 to `mar.ar` (after incorporation of any found outlier if outlier detection if enabled), and 2) `fitted` which is the fitted object of the selected model.

Usage

```
carxSelect(formulas, max.ar, data = list(), detect.outlier = F, ...)
```

Arguments

`formulas`	a regression formula or a list of regression formulas.
`max.ar`	the maximal AR order.
`data`	a `CenTS` object containing the data and censored information.
`detect.outlier`	logical to specify whether outlier detection is performed (and incorporating in the `carx` model any found additive outliers) before computing the AIC for a model. Default = `FALSE`.
`...`	other arguments to be supplied, if not null, it will be called with the selected model and data. Examples include `CI.compute=TRUE`, which will cause the function to re-estimate the selected model with the confidence intervals computed, as in the selection part, no confidence interval is computed.

Value

a `carx` object with an additional element `selectionInfo` which is a list consisting of the information about the selection, in particular, `aicMat`, the matrix of AIC where rows correspond to the model formulas and columns correspond to the AR orders.

Examples

```
dataSim <- carxSimCenTS(nObs=100)
fmls <- list(M1=y~X1,M2=y~X1+X2,M3=y~X1+X2-1)
## Not run: cs = carxSelect(y~X1,max.ar=3,data=dataSim)
## Not run: cs = carxSelect(formulas=fmls,max.ar=3,data=dataSim)
## Not run:
  #To compute confidence intervals for the selected model, call with CI.compute=TRUE.
  cs = carxSelect(formulas=fmls,max.ar=3,data=dataSim,CI.compute=TRUE)

## End(Not run)
```

`carxSim` *Simulate data from a* `carx` *model*

Description

Use the provided parameters in the supplied `carx` model and other settings to simulate data from the `carx` model; see Wang and Chan (2017).

Usage

```
carxSim(nObs = 200, prmtrAR = c(-0.28, 0.25), prmtrX = c(0.2, 0.4),
  sigma = 0.6, lcl = -1, ucl = 1, x = NULL, seed = NULL,
  inno.dist = c("normal", "t"), t.df = 5, intercept = 0)
```

Arguments

`nObs`	number of observations to be simulated.
`prmtrAR`	the AR parameter.
`prmtrX`	the regression parameters for X.
`sigma`	the innovation standard deviation for the AR process.
`lcl`	the lower censoring limit.
`ucl`	the upper censoring limit.
`x`	optional matrix for X. Default = `NULL`, in which case X will be simulated from the standard normal distribution with dimensions determined by `nObs` and `prmtrX`.
`seed`	optional to set the seed of random number generator used by `R`, default=`NULL`.
`inno.dist`	innovation distribution, can be "normal" or "t", default="normal". If it is "t", its degree of freedom should be supplied in `t.df`.
`t.df`	the degree of freedom of the t distribution, used only if `inno.dist`="t". Default=5.
`intercept`	the intercept in the regression. Default=0.

Value

a data frame of simulated `y`, `x`, `ci`, `lcl` and `ucl`.

References

Wang C, Chan KS (2017). "Quasi-likelihood estimation of a censored autoregressive model with exogenous variables." Journal of the American Statistical Association. 2017 Mar 20(just-accepted). with exogenous variables." Submitted.

See Also

`carx` for model specification.

Examples

```
dat = carxSim()
```

`carxSimCenTS` *simulate a sample* `cenTS` *data for* `carx`

Description

Use provided parameters and other settings to simulate a series of data as a `cenTS` object.

Usage

```
carxSimCenTS(nObs = 200, prmtrAR = c(-0.28, 0.25), prmtrX = c(0.2, 0.4),
  sigma = 0.6, lcl = -1, ucl = 1, x = NULL, seed = NULL,
  value.name = "y", end.date = Sys.Date(), inno.dist = c("normal", "t"),
  t.df = 5, intercept = 0)
```

Arguments

`nObs`	number of observations to be simulated.
`prmtrAR`	the AR parameter.
`prmtrX`	the regression parameters for X.
`sigma`	the innovation standard deviation for the AR process.
`lcl`	the lower censoring limit.
`ucl`	the upper censoring limit.
`x`	optional matrix for X. Default = `NULL`, in which case X will be simulated from the standard normal distribution with dimensions determined by `nObs` and `prmtrX`.
`seed`	optional to set the seed of random number generator used by `R`, default=`NULL`.
`value.name`	the name of the response series
`end.date`	the date of the last observation, default = `Sys.date()`.
`inno.dist`	innovation distribution, can be "normal" or "t", default="normal". If it is "t", its degree of freedom should be supplied in `t.df`.
`t.df`	the degree of freedom of the t distribution, used only if `inno.dist`="t". Default=5.
`intercept`	the intercept in the regression. Default=0.

Value

a `cenTS` object with regressors.

See Also

`carxSim`.

Examples

```
cts = carxSimCenTS()
```

cenTS *Create a censored time series object of* cenTS *class*

Description

Create a censored time series response object of cenTS class. Default name of the response is "value", with the vectors of lower/upper censoring limits denoted by lcl and ucl respectively. The vector of censoring indicators, i.e., ci, is part of the cenTS object. Additional related variables can be stored and provided in the construction function, whose names are stored in xreg. All variable values are assumed to be of the same length of and thus aligned with the censored response time series. cenTS inherits from xts::xts.

Usage

```
cenTS(value, order.by, lcl = NULL, ucl = NULL, ci = NULL,
  value.name = "value", ...)
```

Arguments

value	the value vector.
order.by	the index vector, must be a vector of time/date.
lcl	the vector of lower censoring limits, or a single numeric representing the constant limit. Default = NULL indicating no lower limit.
ucl	the vector of upper censoring limits, or a single numeric representing the constant limit. Default = NULL indicating no upper limit.
ci	the vector of censoring indicators whose value is -1 (0, 1) if the corresponding response is left censored (observed, right censored). Default = NULL, in which case, the function will compute ci by value, lcl and ucl. If ci is not NULL, the function will check the consistency of the data, assuming the observed values less (greater) than or equal to left (right) censoring limits are censored, and are observed otherwise. The function will stop if inconsistent results are found.
value.name	the name of the value, default = "value".
...	additional variables, must be able to be coerced to a data.frame.

Value

a cenTS object, any censored observation will be replaced by its corresponding censoring limit.

Examples

```
strDates <- c("2000-01-01", "2000-01-02", "2000-01-03", "2000-01-04", "2000-01-05")
ts <- cenTS(value=c(1,-2,1,NA,0),
            order.by=as.Date(strDates,"%Y-%m-%d"),
            lcl=c(-3,-2,-1,-1,0),
            ucl=c(3,2,1,1,1),
            x=c(1,1,1,1,1),
           y=c(2,2,2,2,2))
 print(ts)
 print(xreg(ts))
 plot(ts)

## Not run:
#wrong call, case 1
ts <- cenTS(value=c(1,-2,1,NA,0),
           order.by=as.Date(strDates,"%Y-%m-%d"),
           lcl=c(-3,-2,-1,-1,0),
           ucl=c(3,2,1,1,1),
           ci =c(-1,-1,1,NA,-1)
)
#wrong call, case 2
ts <- cenTS(value=c(1,-2,1,NA,0),
           order.by=as.Date(strDates,"%Y-%m-%d"),
           lcl=c(-3,-2,-1,-1,0),
           ucl=c(3,2,1,1,1),
           ci =c(1,-1,1,NA,-1)
)


#wrong call, case 3
ts <- cenTS(value=c(1,-2,1,NA,0),
           order.by=as.Date(strDates,"%Y-%m-%d"),
           lcl=c(-3,-2,-1,-1,0),
           ucl=c(3,2,1,1,1),
           ci =c(0,-1,0,NA,-1)
)

## End(Not run)
```

`fitted.carx` *Fitted values of a* `carx` *object*

Description

Compute the fitted values from a `carx` object. Note that the existence of censoring invalidates the usual Markov property for an AR model. Instead, the conditional distribution of Y_t^* given the past Ys and current and past covariates is the same as the conditional distribution $D_t = D(Y_t^*|X_t,(Y_j,X_j)_{j=\tau}^{t-1})$, where $1 \leq \tau \leq t-p$ is the largest integer t such that none of $Y_t; t = \tau+p-1,...,\tau$ is censored. In the case that $\tau = t-p$, the fitted value can be readily computed; otherwise, the fitted value is computed as the mean of the distribution D_t by the function `mtmvnorm` from the package **tmvtnorm**.

Usage

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

Arguments

`object`	a fitted `carx` object.
`...`	not used.

Value

the fitted values.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
#compute the fitted values
fv = fitted(mdl)
```

`logLik.carx` *The quasi-log-likelihood of a* `carx` *object*

Description

The quasi-log-likelihood of a `carx` object

Usage

```
## S3 method for class 'carx'
logLik(object, ...)
```

Arguments

`object`	a fitted `carx` object.
`...`	not used.

Value

the quasi-log-likelihood value.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
lk = logLik(mdl)
```

outlier — *S3 method to detect outlier of a* `carx` *object*

Description

Detect all outliers of a `carx` object.

Usage

```
outlier(object, outlier.prefix = "OI_", seed = NULL)
```

Arguments

`object`	a `carx` object.
`outlier.prefix`	the prefix used to construct variable name for indicator variables representing the detected outliers, default = "OI_".
`seed`	the seed for randon number generator, default=`NULL`.

Value

an updated `carx` object. If any outlier is detected, its index will be stored in the `outlier.indices` attribute of the return object, and prefix for variable name is stored in the `outlier.prefix` attribute. Note that if the original object is fitted through a formula interface, the formula will also be updated.

See Also

`outlier.carx`.

outlier.carx — *Detect all outliers of a* `carx` *object*

Description

Detect all outliers of a `carx` object and update the model if any outlier is detected. It tests for the presence of outliers one at a time, for each time point, adjusted for multiplicity of testing, as described in Wang and Chan (2017).

Usage

```
## S3 method for class 'carx'
outlier(object, outlier.prefix = "OI_", seed = 131)
```

Arguments

`object`	a `carx` object.
`outlier.prefix`	the prefix used to construct variable name for indicator variables representing the detected outliers, default = "OI_".
`seed`	the seed for randon number generator, default=`NULL`.

Value

an updated `carx` object. If any outlier is detected, its index will be stored in the `outlier.indices` attribute of the return object, and prefix for variable name is stored in the `outlier.prefix` attribute. Note that if the original object is fitted through a formula interface, the formula will also be updated.

References

Wang C, Chan KS (2017). "Quasi-likelihood estimation of a censored autoregressive model with exogenous variables." Journal of the American Statistical Association. 2017 Mar 20(just-accepted).

Examples

```
sigma = 0.6
nObs = 100
dat = carxSimCenTS(nObs=nObs,sigma=sigma,ucl=Inf)
dat$y[as.integer(nObs/2)] = dat$y[as.integer(nObs/2)] + 4*sd(dat$y)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
oc = outlier(mdl)
#note the outlier indices in the output:
print(oc)
#note the updated formula:
print(formula(oc))
```

`plot.carx` *Plot a fitted* `carx` *object*

Description

`plot.carx` plots a fitted `carx` object.

Usage

```
## S3 method for class 'carx'
plot(x, FUN = identity, xAxisVar = NULL, xlab = "Index",
  ylab = "Response", ...)
```

Arguments

`x`	a fitted `carx` object.
`FUN`	an optional function to be applied to the transform the responses before plotting. This is useful for plotting the data on the original scale if the fitted `carx` object is based on transformed responses. For instance, if the `carx` object was fitted with log transformed responses, setting `FUN` to exp renders the original response data to be plotted. Default = `NULL`.
`xAxisVar`	an optional vector to be plotted as the x variable. Default = `NULL` corresponds to doing a time series plot.
`xlab`	the label of the x axis. Default = "Index".
`ylab`	the label of the y axis. Default = "Response".
`...`	other parameters supplied to the generic function `plot`.

Details

The y axis will be the values related to the response. If the fitted object contains the data and censored information in a `cenTS` object, the function will take advantage of the plot function for a `cenTS` object and superimpose the plot of the fitted values. Otherwise, the plot function will try to produce a plot similar to the previous case, while the x axis can be supplied by the user through `xAxisVar`, which must be ordinal and increasing.

Value

None. A plot will be displayed.

Examples

```
#case 1: plot with cenTS object in the object, note that the x-axis is in date.
dat = carxSimCenTS(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
#use default settings
plot(mdl)

#case 2: plot without cenTS object in the object, note that the x-axis is a vector of numbers.
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y=dat$y, x=dat[,c("X1","X2")], ci=dat$ci, lcl=dat$lcl, ucl=dat$ucl, p=2)
#or simply call
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
plot(mdl)
```

plot.cenTS *Plot a* cenTS *object*

Description

Plot a cenTS object

Usage

```
## S3 method for class 'cenTS'
plot(x, type = "l", auto.grid = TRUE,
  major.ticks = "auto", minor.ticks = TRUE, major.format = TRUE,
  bar.col = "grey", candle.col = "white", ann = TRUE, axes = TRUE,
  ylim = NULL, main = NULL, ...)
```

Arguments

x	a cenTS object.
type, auto.grid, major.ticks, minor.ticks, major.format, bar.col, candle.col, ann, axes, ylim, main	standard parameters to control the plot.

See Also

plot.xts.

Examples

```
strDates <- c("2000-01-01", "2000-01-02", "2000-01-03", "2000-01-04", "2000-01-05")
ts <- cenTS(value=c(1,-2,1,NA,0),
            order.by=as.Date(strDates,"%Y-%m-%d"),
            lcl=c(-3,-2,-1,-1,0),
            ucl=c(3,2,1,1,1),
            x=c(1,1,1,1,1),
            y=c(2,2,2,2,2))
 plot(ts)
```

predict.carx *Prediction with a fitted* carx *object*

Description

Predict the future values of an fitted carx object. If the model has non-null covariate x other than the constant mean, the new observations in x must be supplied via newxreg. The model prediction is done in a similar way as in fitted.carx by identifying the latest p consecutive observed responses in the data used to estimate model, then compute the mean of the conditional distribution of the future values given the information since the latest p consecutive observed values and the supplied new covariate values. For more details, see Wang and Chan (2017).

Usage

```
## S3 method for class 'carx'
predict(object, newxreg = NULL, n.ahead = 1,
  CI.level = 0.95, nRep = 1000, na.action = NULL,
  useSimulatedResidual = FALSE, ...)
```

Arguments

`object`	a fitted `carx` object.
`newxreg`	the new observations for the coverates `x`, default=NULL. If there is no covariate, the value can be assigned to be `NULL`. Otherwise, a matrix of new observations is required to compute predictions. Default=`NULL`.
`n.ahead`	the number of steps ahead to predict, default = 1.
`CI.level`	the CI.level to construct the confidence interval, default = 0.95.
`nRep`	the number of replications to be performed in the bootstrap for prediction confidence intervals when censoring exists in the last `p` observations, default = 1000.
`na.action`	how should `model.frame` deal with `NA` values.
`useSimulatedResidual`	if True, the innovation sequence will be sampled from the simulated residuals, otherwise, they will be sampled from normal distribution with mean 0 and standard deviation `object$sigma`. Set it `TRUE` if the normal distribution may be too restrictive. Default = FALSE.
`...`	not used.

Value

A list consisting of `fit`, `se.fit`, and `ci` representing the predictions, standard errors of predictions, and confidence intervals respectively.

References

Wang C, Chan KS (2017). "Quasi-likelihood estimation of a censored autoregressive model with exogenous variables." Journal of the American Statistical Association. 2017 Mar 20(just-accepted).

Examples

```
#This is the function to run a simulation study about the empirical coverage rate of
#the predictive confidence intervals,
runSimPredCR <- function(nRep=200,nObs=200,n.ahead=10,
                         saveRslt=FALSE,saveDir='./testPredictCR',
                         seed=NULL)
{
  if(!is.null(seed))
    set.seed(seed)
  crMat = matrix(nrow=n.ahead,ncol=nRep)
  if(saveRslt)
  {
    dir.create(saveDir,showWarnings=FALSE,recursive=TRUE)
    }

    replication = list()
    simSingle <- function(iRep)
    {
      message(sprintf("iRep: %04d",iRep))
      sdata = carxSim(nObs=nObs+n.ahead)
      trainingData = sdata[1:nObs,]
      testData = sdata[-(1:nObs),]
      mdl = carx(y~X1+X2-1,data=trainingData,p=2)
      newxreg = testData[,c('X1','X2')]
      predVal = predict(mdl,newxreg=newxreg,n.ahead=n.ahead)
      crInd = (predVal$ci[,1] <= testData$y) & (predVal$ci[,2] >= testData$y)
      crMat[,iRep] = crInd
      list(trainingData=trainingData,
           testData=testData,
           fitted=mdl,
           predVal = predVal,
           crInd= crInd)
    }
    replication = lapply(1:nRep,simSingle)
    crMat = sapply(replication,FUN=function(x){x$crInd})
    print(crMat)
    crPred = apply(crMat,1,mean)
    message("empirical coverage rate:")
    print(crPred)
    if(saveRslt)
    {
      save(replication,file=paste0(saveDir,'/replication.RData'))
      save(crMat,file=paste0(saveDir,'/crMat.RData'))
      save(crPred,file=paste0(saveDir,'/crPred.RData'))
    }

    list(replication=replication,crMat=crMat,meanCR=crPred)
  }
  #note that nRep=2 is for illustration only, for more stable result, use nRep>=500.
  simPredCR = runSimPredCR(nRep=2,nObs=100)
  ## Not run:
    # for more stable simulation result, run with nRep = 500.
    simPredCR = runSimPredCR(nRep=500,nObs=100)
    message("Empirical coverage rate:")
    print(simPredCR$meanCR)

## End(Not run)
```

`print.carx` *Print a short description of the fitted model*

Description

Print a short description of the fitted model

Usage

```
## S3 method for class 'carx'
print(x, ...)
```

Arguments

`x`	a fitted model object.
`...`	not used.

Value

none.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
print(mdl)
```

`print.cenTS` *Print a* `cenTS` *object*

Description

Print a `cenTS` object

Usage

```
## S3 method for class 'cenTS'
print(x, ...)
```

Arguments

`x`	a `cenTS` object.
`...`	not used.

Value

none.

Examples

```
strDates <- c("2000-01-01", "2000-01-02", "2000-01-03", "2000-01-04", "2000-01-05")
ts <- cenTS(value=c(1,-2,1,NA,0),
            order.by=as.Date(strDates,"%Y-%m-%d"),
            lcl=c(-3,-2,-1,-1,0),
            ucl=c(3,2,1,1,1),
            x=c(1,1,1,1,1),
            y=c(2,2,2,2,2))
 print(ts)
```

`print.summary.carx` *Print a summary of an* `carx` *object*

Description

Print a summary of an `carx` object

Usage

```
## S3 method for class 'summary.carx'
print(x, ...)
```

Arguments

`x`	a summary of an `carx` object.
`...`	not used.

Value

none.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
summary(mdl)
```

`pts`	*The total phosphorus concentration and river discharge data of West Fork Cedar River at Finchford, Iowa.*

Description

This data set contains the monthly total phosphorus concentration (P) and river discharge (Q) of West Fork Cedar River at Finchford, Iowa, USA, from 10/1998 to 10/2013. The P data were collected under the ambient water quality program conducted by the Iowa Department of Natural Resources (Iowa DNR), courtesy of Dr. K. E. Schilling from Iowa Geological Survey, University of Iowa. The Q data were obtained from the website of U.S. Geological Survey. A gap from 09/2008 to 03/2009 in the data are due to program suspension owing to lack of funding. The data contains `logP` and `logQ` which are the logarithm of the original P and Q respectively. The P data are left censored, with -1 or 0 in `ci` indicating that the corresponding observation is left censored or observed. The lower censoring limits are stored in `lcl`. `season` is the quarter to which a month belong.

Author(s)

Chao Wang (chao-wang@uiowa.edu), 08/2015

`residuals.carx`	*Residuals of a fitted* `carx` *object*

Description

Computes the residuals of fitted `carx` object. When no censoring is present, the ordinary residuals will be computed. Otherwise, the simulated residuals (Gourieroux, Monfort, Renault, and Trognon 1987) of a fitted `carx` object will be computed, as suggested in Wang and Chan (2017).

Usage

```
## S3 method for class 'carx'
residuals(object, type = c("raw", "pearson"), seed = NULL,
  ...)
```

Arguments

`object`	a fitted `carx` object.
`type`	a string indicates which type of residual is to be returned. "raw" returns the (simulated) residuals; "pearson" returns the raw residuals divided by estimated standard error of the residuals.
`seed`	the seed for the random number generator.
`...`	not used.

Details

The simulated residuals are constructed as follows. First, impute each unobserved Y_t^* by a (random) realization from the conditional distribution $D(Y_t^*|\{(Y_s, X_s)\}_{s=1}^t)$, evaluated at the parameter estimate. Then, refit the model with (Y_t^*, X_t) so obtained, via the method of conditional maximum likelihood; the residuals from the latter model are the simulated residuals ε_t.

Value

the simulated residuals.

References

Gourieroux C, Monfort A, Renault E, Trognon A (1987). "Simulated residuals." Journal of Econometrics, 34(1), 201-252.

Wang C, Chan KS (2017). "Quasi-likelihood estimation of a censored autoregressive model with exogenous variables." Journal of the American Statistical Association. 2017 Mar 20(just-accepted).

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
#compute the raw residuals
res = residuals(mdl,type="raw")
#compute the Pearson residuals
res = residuals(mdl,type="pearson")
```

`summary.carx` *Summarize the fitted* `carx` *object*

Description

Summarize the fitted `carx` object

Usage

```
## S3 method for class 'carx'
summary(object, ...)
```

Arguments

`object`	a fitted `carx` object.
`...`	not used.

Value

a summary.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
summary(mdl)
```

`tsdiag.carx` *Show diagnostic plots for a* `carx` *object*

Description

Four diagnostic plots will be shown, which are

- the time series plot of the residuals,
- the residuals versus the fitted values,
- the ACF plot of the residuals, and
- the Ljung-Box test statistics versus the lags.

Usage

```
## S3 method for class 'carx'
tsdiag(object, gof.lag, col = "red", omit.initial = TRUE,
  mfrow = c(4, 1), main = "Diagnostic Plots from Simulated Residuals", ...)
```

Arguments

`object`	a `carx` object.
`gof.lag`	the maximum number of lags in ACF and Ljung-Box goodness-of-fit test.
`col`	color of some warning lines in the figures, default=`red`.
`omit.initial`	whether initial residuals should be omitted, default = `TRUE`.
`mfrow`	`par` parameter indicating how the plots are to be arranged, default = `c(4,1)`.
`main`	The main title of the plot, default = "Diagnostic Plots".
`...`	Other arguments sent to `plot`.

Value

none.

Examples

```
dat = carxSim(nObs=100,seed=0)
mdl <- carx(y~X1+X2-1,data=dat, p=2, CI.compute = FALSE)
tsdiag(mdl)
```

xreg — *Return the* xreg *part of the* cenTS *object*

Description

Return the xreg part of the cenTS object

Usage

```
xreg(object)
```

Arguments

object	a cenTS object.

Value

the list in xreg.

See Also

cenTS.

Examples

```
strDates <- c("2000-01-01", "2000-01-02", "2000-01-03", "2000-01-04", "2000-01-05")
ts <- cenTS(value=c(1,-2,1,NA,0),
            order.by=as.Date(strDates,"%Y-%m-%d"),
            lcl=c(-3,-2,-1,-1,0),
            ucl=c(3,2,1,1,1),
            x=c(1,1,1,1,1),
            y=c(2,2,2,2,2))
 xreg(ts)
```

xreg.cenTS — *Return the* xreg *part of the* cenTS *object*

Description

Return the xreg part of the cenTS object

Usage

```
## S3 method for class 'cenTS'
xreg(object)
```

Arguments

`object`	a `cenTS` object.

Value

the list in `xreg`.

See Also

`cenTS`.

Examples

```
strDates <- c("2000-01-01", "2000-01-02", "2000-01-03", "2000-01-04", "2000-01-05")
ts <- cenTS(value=c(1,-2,1,NA,0),
            order.by=as.Date(strDates,"%Y-%m-%d"),
            lcl=c(-3,-2,-1,-1,0),
            ucl=c(3,2,1,1,1),
            x=c(1,1,1,1,1),
            y=c(2,2,2,2,2))
 xreg(ts)
```

Index