Package: tsoutliers (via r-universe)

October 10, 2024

Version 0.6-10

Date 2024-02-10

Title Detection of Outliers in Time Series

Description Detection of outliers in time series following the Chen and Liu (1993) <DOI:10.2307/2290724> procedure. Innovational outliers, additive outliers, level shifts, temporary changes and seasonal level shifts are considered.

Author Javier López-de-Lacalle <javlacalle@yahoo.es>

Maintainer Javier López-de-Lacalle <javlacalle@yahoo.es>

Depends R (>= 3.0.0)

Imports forecast, stats

NeedsCompilation no

LazyData true

Encoding UTF-8

License GPL-2

URL https://jalobe.com

Repository CRAN

Date/Publication 2024-02-12 12:00:02 UTC

Contents

`tsoutliers-package` *Automatic Detection of Outliers in Time Series*

Description

This package implements a procedure based on the approach described in Chen and Liu (1993) for automatic detection of outliers in time series. Innovational outliers, additive outliers, level shifts, temporary changes and seasonal level shifts are considered.

Details

Time series data often undergo sudden changes that alter the dynamics of the data transitory or permanently. These changes are typically non systematic and cannot be captured by standard time series models. That's why they are known as exogenous or outlier effects. Detecting outliers is important because they have an impact on the selection of the model, the estimation of parameters and, consequently, on forecasts.

Following the approach described in Chen & Liu (1993), an automatic procedure for detection of outliers in time series is implemented in the package **tsoutliers**. The procedure may in turn be run along with the automatic ARIMA model selection strategy available in the package **forecast**.

The function `tso` is the main interface for the automatic procedure. The functions `locate.outliers.oloop` and `remove.outliers` implement respectively the first and second stages of the procedure. In practice, the user may stick to use the function `tso`.

Although the purpose of the package is to provide an automatic procedure, the implementation allows the user to do a manual inspection of each step of the procedure. Thus, the package is also useful to track the behaviour of the procedure and come up with ideas for possible improvements or enhancements.

The functions `locate.outliers.oloop` and `remove.outliers` implement the major steps of the procedure. `tso` is the main interface to the automatic procedure. All the options at any stage of the procedure can be defined through the arguments passed to `tso`. Despite the user may stick to use the function `tso`, other functions called by this main interface are exported in the namespace of the package. They are helpful for debugging and allow the interested user to more easily track each step of the procedure.

Information supplemental to these help pages is given in the document that is provided with the package ('`tsoutliers/inst/doc/tsoutliers.pdf`' in the source files).

Author(s)

Javier López-de-Lacalle <javlacalle@yahoo.es>

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297. doi:10.2307/2290724

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Gómez, V. and Taguas, D. (1995). *Detección y Corrección Automática de Outliers con TRAMO: Una Aplicación al IPC de Bienes Industriales no Energéticos*. Ministerio de Economía y Hacienda. Document number D-95006. https://www.sepg.pap.hacienda.gob.es/sitios/sepg/es-ES/Presupuestos/DocumentacionEstadisticas/Documentacion/Documents/DOCUMENTOS%20DE%20TRABAJO/D95006.pdf

Hyndman, R.J. and Khandakar, Y. (2008). 'Automatic Time Series Forecasting: The **forecast** Package for R'. *Journal of Statistical Software*, **27**(3), pp. 1-22. https://www.jstatsoft.org/v27/i03

Hyndman, R.J. with contributions from George Athanasopoulos, Slava Razbash, Drew Schmidt, Zhenyu Zhou, Yousaf Khan, Christoph Bergmeir and Earo Wang (2014). '**forecast**: Forecasting functions for time series and linear models'. R package version 5.4. https://CRAN.R-project.org/package=forecast

---

bde9915 *Data Set: Working Paper 'bde9915'*

Description

Data set employed for illustration in the working paper referenced below.

Usage

```
bde9915
```

Format

A list containing the following monthly series:

- gipi: Italian industrial production index (1981:1-1996:12);
- euprin: European price index of industrial production (1981:1-1993:1) (*);
- metipi: Spanish production index of metal products (1980:1-1997:7).

(*) 1993:12 is reported in the reference document.

Note

The authors of the reference working paper probably use additional variables such as holidays to obtain the results reported in the document.

References

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915.

`calendar.effects` *Calendar Effects*

Description

This function creates regressor variables for trading day, Easter and leap year effects over the sample period where the input time series is sampled.

Usage

```
calendar.effects(x, trading.day = TRUE, easter = 6,
  leap.year = FALSE, holidays = NULL, easter.date = FALSE)
```

Arguments

`x`	a monthly time series.
`trading.day`	logical. If `TRUE`, trading day regressor variable is returned.
`easter`	numeric. The number of days before Easter over which the Easter effect spans. If it is set to zero Easter variable is not returned.
`leap.year`	logical. If `TRUE`, leap year regressor variable is returned.
`holidays`	an optional numeric vector of the same length as `x` containing the number of holidays for each time period.
`easter.date`	logical indicating whether the date of Easter should be returned.

Details

Let wd be the number of working days in a given month and nwd the number of non-working days. The trading day variable at time t is built as follows:

$$wd = wd - holidays$$

$$nwd = nwd + holidays$$

$$td_t = wd - (5/2) \times nwd$$

By default, working days are the days from Monday to Friday and non-working days are Saturdays and Sundays. If there are additional non-working days they can be defined in argument `holidays`.

For example, if the 1st of February is a local holiday, the user can define a variable containing zeros for all periods except for the periods related to February where the 1st of February falls within a working day (Monday to Friday); these data are set to one so that they are considered as non working days.

Easter effect is defined as the proportion of days before Easter (by default `easter = 6`) that lie in March and April, respectively for each month. It contains zeros for the remaining months.

The leap year is a vector of zeros for all months except February, where the variable takes on the value 0.75 if the year is a leap year and -0.25 otherwise.

Value

A `mts` matrix containing the selected calendar effects by columns.

If `easter.date` is `TRUE` a list is returned containing the `mts` matrix of calendar effects as well as the dates of Easter for each year of the sample.

Examples

```
# display calendar effects for a sample span period
# no data are actually necessary in the input series
# since calendar effects are concerned only with the dates
# at which the data are sampled
x <- ts(frequency = 12, start = c(1980, 1), end = c(2000, 12))
ce <- calendar.effects(x, leap.year = TRUE)
colnames(ce)
plot(ce, main = "calendar effects")
# Easter days for each year
calendar.effects(x, easter.date = TRUE)$easter
```

---

coefs2poly *Product of the Polynomials in an ARIMA Model*

Description

This function collapses the polynomials of an ARIMA model into two polynomials: the product of the autoregressive polynomials and the product of the moving average polynomials.

Usage

```
coefs2poly(x, add = TRUE)
```

Arguments

`x`	an object of class `Arima`, as returned by `arima`.
`add`	logical. If `TRUE`, the polynomial of the differencing filter (if present in the model) is multiplied by the stationary autoregressive polynomial. Otherwise only the coefficients of the product of the stationary polynomials is returned.

Value

A list containing the elements: `arcoefs`, the coefficients of the product of the autoregressive polynomials; `macoefs`, the coefficients of the product of the moving average polynomials. This list is of class `"ArimaPars"` so that it can be recognized by `outliers.tstatistics`.

Examples

```
# ARIMA(0,1,1)(0,1,1) model
fit <- arima(log(AirPassengers), order = c(0,1,1),
  seasonal = list(order = c(0,1,1)))
coefs <- coef(fit)

# "coefs2poly" returns the coefficients of the product of
# the non-seasonal and the seasonal moving average polynomials
a1 <- convolve(c(1, coefs[1]), rev(c(1, rep(0, 11), coefs[2])), type="open")[-1]
a2 <- coefs2poly(fit)$macoefs
a2
all.equal(a1, a2, check.names=FALSE)

# since the model does not contain an autoregressive part
# the product of the regular and the seasonal differencing
# filter is returned if "add = TRUE"
coefs2poly(fit)$arcoefs
# an empty set is returned if "add = FALSE"
coefs2poly(fit, add = FALSE)$arcoefs

# in a model with non-seasonal part and no differencing filter
# no multiplication of polynomials are involved and
# the coefficients are the same as those returned by "coef"
fit <- arima(log(AirPassengers), order = c(1,0,1))
coef(fit)
coefs2poly(fit)
```

---

discard.outliers *Stage II of the Procedure: Discard Outliers*

Description

This functions tests for the significance of a given set of outliers in a time series model that is fitted including the outliers as regressor variables.

Usage

```
discard.outliers(x, y, cval = NULL,
  method = c("en-masse", "bottom-up"),
  delta = 0.7, tsmethod.call = NULL,
  fdiff = NULL, logfile = NULL, check.rank = FALSE)
```

Arguments

x	a list. The output returned by locate.outliers.oloop.
y	a time series.
cval	a numeric. The critical value to determine the significance of each type of outlier.
method	a character. The method to discard/remove outliers. See details.
delta	a numeric. Parameter of the temporary change type of outlier.
tsmethod.call	an optional call object. The call to the function used to fit the time series model.
fdiff	currently ignored.
logfile	a character or NULL. It is the path to the file where tracking information is printed. Ignored if NULL.
check.rank	logical. If TRUE, variables generating perfect multicollinearity are removed (tentative implementation).

Details

In the regressions involved in this function, the variables included as regressors stand for the effects of the outliers on the data. These variables are the output returned by outliers.effects not by outliers.regressors, which returns the regressors used in the auxiliar regression where outliers are located (see second equation defined in locate.outliers).

The outliers are defined in input x. If there are regressor variables in tsmethod.call$xreg they are considered as other regressor variables that are included in the regression to test for the significance of outliers.

Given a set of potential outliers detected by locate.outliers and locate.outliers.oloop, three methods are considered in order to determine which outliers are not significant after refitting the model (including all the potential outliers):

- "en-masse": The complete set of outliers is included as regressor variables and the model is fitted again. Those outliers that turn out to be not significant for the critical value cval are discarded/removed. The procedure is iterated until all the outliers are significant in the final set of outliers.
- "bottom-up": First the, the outlier with larger t-statistic is included in the model. If it is significant the presence of the outlier is confirmed. Otherwise it is discarded. Then, the next outlier with larger t-statistic is included along with the first outlier and tested for significance. If after including a new outlier, e.g. the i-th outlier, the outliers that have been confirmed so far significant become not significant, then the i-th outlier is discarded regardless of the value of its t-statistic.

The option "en-masse" may be preferred to "bottom-up" when there are are several outliers, since it may be hard to fit an ARIMA model with many regressor variables.

Value

A list containing the following elements: xreg, the variables used as regressors; xregcoefs, the coefficients of the outlier regressors; xregtstats, the t-statistics of the outlier regressors;; iter, the number of iterations used by method "en-masse"; fit, the fitted model; outliers, the set of outliers after removing those that were not significant.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

See Also

locate.outliers, tso.

Examples

```
## Not run:
data("hicp")
y <- log(hicp[["011600"]])
fit <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)))
# initial set of outliers
res <- locate.outliers.oloop(y, fit, types = c("AO", "LS", "TC"))
res$outliers
# given the model fitted above, the effect on the data of some of
# the outliers is not significant (method = "en-masse")
discard.outliers(res, y, method = "en-masse",
  tsmethod.call = fit$call)$outliers
# in this case, using method = "bottom-up" the first four
# outliers with higher t-statistic are kept
discard.outliers(res, y, method = "bottom-up",
  tsmethod.call = fit$call)$outliers

## End(Not run)
```

---

find.consecutive.outliers

Find outliers at consecutive time points

Description

Find outliers at consecutive time points and discard those with lower t-statistic in absolute value.

Usage

```
find.consecutive.outliers(x, type)
```

Arguments

x	a data frame containing the type, index and t-statistic related to each outlier, named respectively by columns: `type`, `ind` and `tstat`
.	
type	a character, the type of outlier to be checked for runs at consecutive points.

Details

In the procedure, this function is applied by type of outliers, e.g., two or more consecutive LS are reduced to one LS. It is still possible to get two or more consecutive outliers of different type at consecutive time points. Alternatively consecutive outliers of any type could be reduced to a single outlier, For the time being, this is kept in this way so that for example, the following sequence AO1, LS2, LS3,LS4 can collapse to AO-1,LS-4, i.e., the AO is kept.

Value

The row names of input `x` pointing to the outliers to be discarded, i.e., outliers with lower significance than adjacent outliers of the same type.

Note

This function is intended for internal use; no check is done for correctness of the arguments passed as input.

`hicp` *Data Set: Harmonised Indices of Consumer Prices*

Description

Harmonised indices of consumer prices in the Euro area.

Usage

```
hicp
```

Format

A list containing the following monthly indices (2005=100).

Key	Description	Start
ICP.M.U2.N.000000.4.INX	Overall index	1990:1
ICP.M.U2.N.010000.4.INX	Food and non-alcoholic beverages	1990:1
ICP.M.U2.N.011000.4.INX	Food	1995:1
ICP.M.U2.N.011200.4.INX	Meat	1995:1
ICP.M.U2.N.011300.4.INX	Fish	1995:1
ICP.M.U2.N.011600.4.INX	Fruit	1995:1

ICP.M.U2.N.011700.4.INX	Vegetables	1995:1
ICP.M.U2.N.FOODPR.4.INX	Processed food incl. alcohol and tobacco	1990:1
ICP.M.U2.N.FOODUN.4.INX	Unprocessed food	1990:1
ICP.M.U2.N.IGXE00.4.INX	Industrial goods excluding energy	1990:1
ICP.M.U2.N.NRGY00.4.INX	Energy	1990:1
ICP.M.U2.N.SERV00.4.INX	Services	1990:1

All the series end in December 2013.

Source

European Central Bank. Statistical Data Warehouse. http://sdw.ecb.europa.eu/.

`ipi` *Data Set: Industrial Production Indices*

Description

Industrial production indices in the manufacturing sector of European Monetary Union countries.

Usage

```
ipi
```

Format

A list containing monthly indices (2010=100) for the following countries.

Country	Start	Country	Start
Belgium	2000:1	Luxembourg	2000:1
Germany	1999:1	Malta	2000:1
Estonia	1999:1	Netherlands	1999:1
Greece	2000:1	Austria	1999:1
Spain	1999:1	Portugal	1999:1
France	1999:1	Slovenia	1999:1
Italy	1999:1	Slovakia	2000:1
Cyprus	2000:1	Finland	1999:1
Latvia	2000:1		

All the series end in December 2013.

Note

Data base key: "sts_inpr_m", monthly short-term business statistics. Production in industry.

Gross data for Ireland is not available in Eurosta's data base, only seasonally or working day adjusted data are available for this country.

Source

EUROSTAT. Statistical Office of the European Communities. https://ec.europa.eu/eurostat.

`JarqueBera.test` *Jarque-Bera Test for Normality*

Description

This function applies the test for normality proposed in Jarque and Bera (1980).

Usage

```
JarqueBera.test(x, fc = 3.5, ...)
```

Arguments

`x`	a time series of residuals or an object of class `Arima`.
`fc`	a numeric. Factor to asses whether the first residual observations are to be omitted. Ignored if `x` is not an `Arima` object. See details.
`...`	further arguments. Currently omitted.

Details

This function is based on function `jarque.bera.test` available in package **tseries**. Here, the results are split in a test for the null hypothesis that the skewness is 0, the null that the kurtosis is 3 and the overall Jarque-Bera test.

The input can be a time series of residuals, `jarque.bera.test.default`, or an `Arima` object, `jarque.bera.test.Arima` from which the residuals are extracted. In the former case the whole input series of residuals is used. In the latter case, the first $n0$ (defined below) residuals are omitted if they are are equal to zero or if any of them are in absolute value larger than `fc` times the standard deviation of the remaining residuals. $n0$ is set equal to `x$arma[6] + x$arma[5] * x$arma[7]`, i.e. the number of regular differences times the periodicity of the data times the number of seasonal differences. If $n0$ happens to be equal to 1 it is set to 2.

If the latter trimming operation is not desired, the argument `fc` can be set to a high value to ensure the complete series of residuals in considered; or the function can be called as `jarque.bera.test(residuals(x))`.

Missing observations are omitted.

Value

A list containing one `htest` object for the null hypothesis that the kurtosis is 3, the skewness is 0 and a test combining both the kurtosis and the skewness to test for the normality of the input data.

References

Jarque, C. M. and Bera, A. K. (1980). 'Efficient test for normality, homoscedasticity and serial independence of residuals'. *Economic Letters*, **6**(3), pp. 255-259.

See Also

print.mhtest.

Examples

```
# fit an ARIMA model to the HICP 011600 series
# ARIMA(0,1,0)(2,0,1) was chosen by forecast::auto.arima(ic = "bic")
# normality of the residuals is rejected at the 5% significance level
# due to an excess of kurtosis
data("hicp")
y <- log(hicp[["011600"]])
fit1 <- arima(y, order = c(0, 1, 0), seasonal = list(order = c(2, 0, 1)))
JarqueBera.test(fit1)
JarqueBera.test(residuals(fit1))

# fit ARIMA model for the same series including outliers that were
# detected by "tso" and for the model chosen by "auto.arima"
# normality of the residuals is not rejected at the 5% significance level
# after including the presence of outliers
mo <- outliers(c("AO", "AO", "LS", "LS"), c(79, 210, 85, 225))
xreg <- outliers.effects(mo, length(y))
fit2 <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)),
  xreg = xreg)
JarqueBera.test(fit2)
```

---

locate.outliers *Stage I of the Procedure: Locate Outliers (Baseline Function)*

Description

This function applies the t-statistics for the significance of outliers at every time point and selects those that are significant given a critical value.

Usage

```
locate.outliers(resid, pars, cval = 3.5, types = c("AO", "LS", "TC"), delta = 0.7)
```

Arguments

resid	a time series. Residuals from a time series model fitted to the data.
pars	a list containing the parameters of the model fitted to the data. See details below.
cval	a numeric. The critical value to determine the significance of each type of outlier.
types	a character vector indicating the types of outliers to be considered.
delta	a numeric. Parameter of the temporary change type of outlier.

Details

Five types of outliers can be considered. By default: `"AO"` additive outliers, `"LS"` level shifts, and `"TC"` temporary changes are selected; `"IO"` innovative outliers and `"SLS"` seasonal level shifts can also be selected.

The approach described in Chen & Liu (1993) is followed to locate outliers. The original framework is based on ARIMA time series models. The extension to structural time series models is currently experimental.

Let us define an ARIMA model for the series y_t^* subject to m outliers defined as $L_j(B)$ with weights w:

$$y_t^* = \sum_{j=1}^{m} \omega_j L_j(B) I_t(t_j) + \frac{\theta(B)}{\phi(B)\alpha(B)} a_t \,,$$

where $I_t(t_j)$ is an indicator variable containing the value 1 at observation t_j where the j-th outlier arises; $\phi(B)$ is an autoregressive polynomial with all roots outside the unit circle; $\theta(B)$ is a moving average polynomial with all roots outside the unit circle; and $\alpha(B)$ is an autoregressive polynomial with all roots on the unit circle.

The presence of outliers is tested by means of t-statistics applied on the following regression equation:

$$\pi(B) y_t^* \equiv \hat{e}_t = \sum_{j=1}^{m} \omega_j \pi(B) L_j(B) I_t(t_j) + a_t \,.$$

where $\pi(B) = \sum_{i=0}^{\infty} \pi_i B^i$. The regressors of the above equation are created by the functions `outliers.regressors.arima` and the remaining functions described here.

The function `locate.outliers` computes all the t-statistics for each type of outlier and for every time point. See `outliers.tstatistics`. Then, the cases where the corresponding t-statistic are (in absolute value) below the threshold `cval` are removed. Thus, a potential set of outliers is obtained.

Some polishing rules are applied by `locate.outliers`:

- If level shifts are found at consecutive time points, only then point with higher t-statistic in absolute value is kept.
- If more than one type of outlier exceed the threshold `cval` at a given time point, the type of outlier with higher t-statistic in absolute value is kept and the others are removed.

The argument `pars` is a list containing the parameters of the model. In the framework of ARIMA models, the coefficients of the ARIMA must be defined in `pars` as the product of the autoregressive non-seasonal and seasonal polynomials (if any) and the differencing filter (if any). The function `coefs2poly` can be used to define the argument `pars`.

Value

A data frame defining by rows the potential set of outliers. The type of outlier, the observation, the coefficient and the t-statistic are given by columns respectively for each outlier.

Note

The default critical value, `cval`, is set equal to 3.5 and, hence, it is not based on the sample size as in functions `tso` or `locate.outliers.oloop`.

Currently the innovational outlier `"SLS"` is not available if `pars` is related to a structural time series model.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Gómez, V. and Taguas, D. (1995). *Detección y Corrección Automática de Outliers con TRAMO: Una Aplicación al IPC de Bienes Industriales no Energéticos*. Ministerio de Economía y Hacienda. Document number D-95006. https://www.sepg.pap.hacienda.gob.es/sitios/sepg/es-ES/Presupuestos/DocumentacionEstadisticas/Documentacion/Documents/DOCUMENTOS%20DE%20TRABAJO/D95006.pdf

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915.

See Also

`locate.outliers.oloop`, `locate.outliers.iloop`, `outliers.tstatistics`, `tso`.

Examples

```
data("hicp")
y <- log(hicp[["011600"]])
fit <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)))
resid <- residuals(fit)
pars <- coefs2poly(fit)
outliers <- locate.outliers(resid, pars)
outliers
```

---

locate.outliers.loops *Stage I of the Procedure: Locate Outliers (Loop Around Functions)*

Description

These functions implement the inner and outer loops based of the procedure to locate outliers following the approach described in Chen and Liu (1993) .

Usage

```
locate.outliers.oloop(y, fit, types = c("AO", "LS", "TC"),
  cval = NULL, maxit.iloop = 4, maxit.oloop = 4,
  delta = 0.7, logfile = NULL)
locate.outliers.iloop(resid, pars, cval = 3.5,
  types = c("AO", "LS", "TC"), maxit = 4, delta = 0.7, logfile = NULL)
```

Arguments

y	a time series.
fit	an Arima object. The output from arima or auto.arima
.	
resid	a time series. Residuals from a time series model fitted to the data.
pars	a list containing the parameters of the model fitted to the data. See details in locate.outliers.
cval	a numeric. The critical value to determine the significance of each type of outlier.
types	a character vector indicating the type of outlier to be considered by the detection procedure among the following: innovational outliers ("IO"), additive outliers ("AO"), level shifts ("LS"), temporary changes ("TC") and seasonal level shifts ("SLS").
maxit	a numeric. The maximum number of iterations in the inner loop.
maxit.iloop	a numeric. Same as argument as maxit to be passed to locate.outliers.iloop.
maxit.oloop	a numeric. The maximum number of iterations in the outer loop.
delta	a numeric. Parameter of the temporary change type of outlier.
logfile	a character or NULL. It is the path to the file where tracking information is printed. Ignored if NULL.

Details

See also the details section in locate.outliers.

The function locate.outliers.iloop iterates around the function locate.outliers until no additional outliers are found or the maximum number of iterations is reached. After each iteration, the effect of the outliers on the residuals of the fitted model is removed and the t-statistics are obtained again for the modified residuals. No model selection or refit of the model is conducted within this loop.

The function locate.outliers.oloop is the outer loop of the procedure to locate outliers. It iterates around the function locate.outliers.iloop. At the end of each iteration the detected outliers are removed from the original data. Then, the time series model is fitted (or selected) again for the adjusted series and a new search for outliers is executed. The outer loop stops when no additional outliers are detected.

In function locate.outliers.oloop, if no value is specified for argument cval a default value based on the sample size is used. Let n be the number of observations. If $n \leq 50$ then cval is set equal to 3.0; If $n \geq 450$ then cval is set equal to 4.0; otherwise cval is set equal to $3 + 0.0025 * (n - 50)$.

Value

`locate.outliers.iloop` returns a data frame defining by rows each detected outlier. The data frame follows the same format as the output from `locate.outliers`.

`locate.outliers.oloop` returns a list containing the following elements: `fit`: information from the last fitted model that will be required by other functions in the automatic procedure (parameter estimates, residuals and number of observations); `outliers`: a data frame defining by rows the detected outliers; `iter`: the number of iterations employed by the outer loop.

Note

In `locate.outliers.iloop` the default critical value, `cval`, is set equal to 3.5 and, hence, it is not based on the sample size. `locate.outliers.oloop` uses a default critical value based on the sampel size as in `tso`.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Gómez, V. and Taguas, D. (1995). *Detección y Corrección Automática de Outliers con TRAMO: Una Aplicación al IPC de Bienes Industriales no Energéticos*. Ministerio de Economía y Hacienda. Document number D-95006. https://www.sepg.pap.hacienda.gob.es/sitios/sepg/es-ES/Presupuestos/DocumentacionEstadisticas/Documentacion/Documents/DOCUMENTOS%20DE%20TRABAJO/D95006.pdf

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915.

See Also

`outliers.tstatistics`, `tso`.

Examples

```
# additional outliers may be detected in the inner or outlier loops
# in this case, the inner does not find further potential outliers
# and stops in the first iteration, while the outer loop detects
# a new outlier
data("hicp")
y <- log(hicp[["011600"]])
fit <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)))
resid <- residuals(fit)
pars <- coefs2poly(fit)
otypes <- c("AO", "LS", "TC")
mo0 <- locate.outliers(resid, pars, types = otypes)
mo0
mo1 <- locate.outliers.iloop(resid, pars, types = otypes)
```

```
mo1
mo2 <- locate.outliers.oloop(y, fit, types = otypes)
mo2$iter
mo2$outliers
```

`outliers` *Define Outliers in a Data Frame*

Description

This function is an interface to create a data frame defining the type, observation and weight of outliers. The output is properly designed to be used as input to other functions such as `outliers.effects` and `outliers.regressors`.

Usage

```
outliers(type, ind, weight = NULL)
```

Arguments

`type`	a character vector. The types of outliers ("IO", "AO", "LS", "TC", "SLS").
`ind`	a numeric vector. The observations at which each outlier takes effect.
`weight`	an optional numeric vector. The weights of the outliers. If `NULL` the weights are set equal to one.

Value

A data frame.

Examples

```
outliers(c("AO", "LS"), c(10, 20))
outliers(c("AO", "LS"), c(10, 20), c(2.1, 3.2))
```

`outliers.effects` *Create the Pattern of Different Types of Outliers*

Description

These functions create a unit or weighted impulse for the five types of outliers considered in the package.

Usage

```
outliers.effects(mo, n, weights = FALSE, delta = 0.7,
  pars = NULL, freq = 12)
```

Arguments

mo	a data frame defining the type, location and weight of the outliers to be created.
n	a numeric. The length of the variable that will contain the outlier.
weights	logical. If TRUE, the outliers are weighted by the values in column "coefhat" of the data frame mo. Otherwise, unit weights are considered.
delta	a numeric. Parameter of the temporary change type of outlier.
pars	a list containing the parameters of the time series model fitted to the data. Only for innovational outlier. See the details section in locate.outliers.
freq	a numeric. The periodicity of the data. Only for seasonal level shift.

Details

These functions delineate the effect of each type of outlier on the observed data. See the example below for a representation of the outliers.

The function outliers.effects operates directly on the output returned by time. The remaining functions are called by outliers.effects and can be used as a simpler interface to define some outliers. They generate the type of outliers indicated by their names.

The column names of the data frame mo must follow the same convention as the output returned by locate.outliers: the column containing the observation at which the outlier sparks is named "ind"; the column containing the weights is named "coefhat" and the type of outlier is specified in a column named "type". The column "ind" should contain the index time point, not the time point in terms of year and season as given by time.

Value

A n $\times$ nrow(mo) or n $\times$ length(ind) matrix containing by columns each outlier.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/99/Fic/dt9915e.pdf

See Also

locate.outliers, remove.outliers, tso.

Examples

```
n <- 30
# innovative outlier based on ARMA(3, 2) model
mo <- outliers("IO", 10)
io <- outliers.effects(mo, n, pars = list(arcoefs = c(0.8, -0.6, 0.2),
  macoefs = c(-0.5, 0.2)))
plot(c(io[seq.int(10)], rep(NA, 20)), type = "s", ylim = range(io),
  ylab = "io", main = "IO based on ARMA(3,2)")
lines(c(rep(NA, 9), io[-seq.int(9)]))

# innovative outlier based on Airlines model ARIMA(0,1,1)(0,1,1)
p1 <- c(1, -1)
p2 <- c(1, rep(0, 3), -1)
p1xp2 <- c(1, -1, 0, 0, -1, 1)
p1b <- c(1, -0.6)
p2b <- c(1, rep(0, 3), -0.6)
p2bxp2b <- c(1, -0.6, 0, 0, -0.6, 0.36)
io2 <- outliers.effects(mo, n, pars = list(arcoefs = -p1xp2[-1],
  macoefs = p2bxp2b[-1]))
plot(c(io2[seq.int(10)], rep(NA, 20)), type = "s", ylim = range(io2),
  main = "IO based on ARIMA(0,1,1)(0,1,1)", ylab = "io2")
lines(c(rep(NA, 9), io2[-seq.int(9)]))

# additive outlier
mo <- outliers("AO", 10)
ao <- outliers.effects(mo, n)
plot(ao, type = "h", main = "AO: additive outlier")

# level shift
mo <- outliers("LS", 10)
ls <- outliers.effects(mo, n)
plot(ls, type = "s", main = "LS: level shift")

# temporary change
mo <- outliers("TC", 10)
tc <- outliers.effects(mo, n)
plot(c(tc[seq.int(10)], rep(NA, 20)), type = "s",
  main = "TC: temporary change", ylab = "tc")
lines(c(rep(NA, 9), tc[-seq.int(9)]))

# the temporary change with parameter 0.7 is equivalent to
# an IO for an AR(1) model with coefficient 0.7
mo <- outliers("IO", 10)
io3 <- outliers.effects(mo, n = n, pars = list(arcoefs = c(0.7), macoefs = c(0)))
all.equal(io3, tc, check.attributes=FALSE)

# the temporary change with parameter 0.7 is equivalent to
# an IO for an AR(1) model with coefficient 0.7
mo <- outliers("IO", 10)
io4 <- outliers.effects(mo, n = n, pars = list(arcoefs = 1, macoefs = 0))
all.equal(io4, ls, check.attributes=FALSE)
```

```
# seasonal level shift (quarterly data)
mo <- outliers("SLS", 10)
sls <- outliers.effects(mo, n, freq = 4)
plot(sls, type = "h", main = "SLS: seasonal level shift")
```

`outliers.regressors` *Regressor Variables for the Detection of Outliers*

Description

These functions create regressor variables to be used included in the regression where tests for presence will be applied.

Usage

```
outliers.regressors(pars, mo, n, weights = TRUE,
  delta = 0.7, freq = 12)
```

Arguments

`pars`	a list containing the parameters of the model. See details section in `locate.outliers`.
`mo`	a data frame defining the type, location and weight of the outliers to be created.
`n`	a numeric. The length of the variable that will contain the outlier.
`weights`	logical. If `TRUE`, the variables are weighted by the values in column `"coefhat"` of the data frame `mo`. Otherwise, unit weights are considered.
`delta`	a numeric. Parameter of the temporary change type of outlier.
`freq`	a numeric. The periodicity of the data. Used only for the seasonal level shift, `"SLS"`.

Details

The variables returned by these functions are the regressors that take part in the second equation defined in `locate.outliers`, (equation (20) in Chen-Liu (1993), equation (3) in the documentat attached to the package).

Regressions are not actually run since the t-statistics can be obtained more conveniently as indicated in equation (14) in Chen-Liu (1993). These variables are used in function `locate.outliers.iloop` to adjust the residuals at each iteration.

The function `outliers` can be used to easily create the input argument `mo`.

Value

A matrix containing the regressors by columms.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/99/Fic/dt9915e.pdf

See Also

locate.outliers, outliers, outliers.tstatistics, tso.

Examples

```
# regression of the residuals from the ARIMA model
# on the corresponding regressors for three additive outliers
# at the 5% level, the first AO is not significant, the others are significant
data("hicp")
y <- log(hicp[["011600"]])
fit <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)))
resid <- residuals(fit)
pars <- coefs2poly(fit)
mo <- outliers(rep("AO", 3), c(10, 79, 224))
xreg <- outliers.regressors(pars, mo, length(y))
summary(lm(residuals(fit) ~ 0 + xreg))
```

---

outliers.tstatistics *Test Statistics for the Significance of Outliers*

Description

This function computes the t-statistics to assess the significance of different types of outliers at every possible time point. The statistics can be based either on an ARIMA model, arima or auto.arima.

Usage

```
outliers.tstatistics(pars, resid, types = c("AO", "LS", "TC"),
  sigma = NULL, delta = 0.7)
```

Arguments

pars	a list containing the parameters of the model. See details section in locate.outliers.
resid	a time series. Residuals of the ARIMA model fitted to the data.
types	a character vector indicating the types of outliers to be considered.
sigma	a numeric or NULL. Standard deviation of residuals.
delta	a numeric. Parameter of the temporary change type of outlier.

Details

Five types of outliers can be considered. By default: "AO" additive outliers, "LS" level shifts, and "TC" temporary changes are selected; "IO" innovative outliers and "SLS" seasonal level shifts can also be selected.

The test statistics are based on the second equation defined in locate.outliers.

These functions are the called by locate.outliers. The approach described in Chen & Liu (1993) is implemented to compute the t-statistics.

By default (sigma = NULL), the standard deviation of residuals is computed as the mean absolute deviation of resid.

Value

For each function, a two-column matrix is returned. The first column contains the estimate of the coefficients related to the type of outlier and the second column contains the t-statistics. The value of these statistics for each time point is stored by rows, thus the number of rows is equal to the length of resid.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Gómez, V. and Taguas, D. (1995). *Detección y Corrección Automática de Outliers con TRAMO: Una Aplicación al IPC de Bienes Industriales no Energéticos*. Ministerio de Economía y Hacienda. Document number D-95006. https://www.sepg.pap.hacienda.gob.es/sitios/sepg/es-ES/Presupuestos/DocumentacionEstadisticas/Documentacion/Documents/DOCUMENTOS%20DE%20TRABAJO/D95006.pdf

Kaiser, R., and Maravall, A. (1999). *Seasonal Outliers in Time Series*. Banco de España, Servicio de Estudios. Working paper number 9915. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/99/Fic/dt9915e.pdf

See Also

locate.outliers, outliers.regressors.

Examples

```
# given an ARIMA model detect potential outliers
# for a critical value equal to 3.5
data("hicp")
y <- log(hicp[["011600"]])
fit <- arima(y, order = c(1, 1, 0), seasonal = list(order = c(2, 0, 2)))
resid <- residuals(fit)
pars <- coefs2poly(fit)
tstats <- outliers.tstatistics(pars, resid)
```

```
# potential observations affected by an additive outliers
which(abs(tstats[,"AO","tstat"]) > 3.5)
# potential observations affected by a temporary change
which(abs(tstats[,"TC","tstat"]) > 3.5)
# potential observations affected by a level shift
which(abs(tstats[,"LS","tstat"]) > 3.5)
```

`plot.tsoutliers` *Display Outlier Effects Detected by* `tsoutliers`

Description

This function displays the output from function `tso`.

Usage

```
## S3 method for class 'tsoutliers'
plot(x,
  args.lines.y = list(col = "gray80"), args.lines.yadj = list(col = "blue"),
  args.lines.effects = list(type = "s", col = "red"),
  args.points = list(col = "gray80", bg = "red", pch = 21), plot.points = TRUE,
  args.x.axis = list(at = pretty(time(x$y)), tcl = -0.5, lwd = 0, lwd.ticks = 1),
  args.y.axis = list(at = pretty(x$y), tcl = -0.5, lwd = 0, lwd.ticks = 1),
 args.effects.axis = list(at = pretty(x$effects), tcl = -0.5, lwd = 0, lwd.ticks = 1),
  ...)
```

Arguments

`x`	a list of class `tsoutliers` as returned by `tso`.
`args.lines.y`	a list. Arguments passed to `lines` to customize the line displaying the original series.
`args.lines.yadj`	a list. Arguments passed to `lines` to customize the line displaying the series adjusted for outliers effects.
`args.lines.effects`	a list. Arguments passed to `lines` to customize the line displaying the for outliers effects.
`args.points`	a list. Arguments passed to `lines` to customize the points drawn if `plot.points = TRUE`.
`plot.points`	a logical indicating whether the time points of the outliers should be drawn as points over the original series.
`args.x.axis`	a list. Arguments to be passed to `axis` to customize the x-axis (time).
`args.y.axis`	a list. Arguments to be passed to `axis` to customize the y-axis for the original series.
`args.effects.axis`	a list. Arguments to be passed to `axis` to customize the y-axis for the outliers effects.
`...`	further arguments to be passed to `par`.

Details

Instead of using the ellipsis, ..., arguments passed to other functions are defined by means of a list. This approach is taken because there may be a single argument name to be used in different parts of the plot with a different value. For example, the argument `col` can be defined in `args.lines.y` to indicate the color of the original series, e.g. `col = "gray80"`; at the same time the color for the adjusted series can be defined in the list argument `args.lines.yadj`.

For further customizations, the source code of the function can be modified relatively easy. Alternatively, a similar plot can be displayed simply as: `plot(cbind(x$y, x$yadj, x$effects), plot.type = "multiple")`. In this way, the plot can be fully customized by setting the desired arguments to to `plot` or to ancillary functions that can be called afterwards.

Value

None.

See Also

`tso`.

`print.tsoutliers` *Print* `tsoutliers` *object*

Description

This function prints the output from function `tso`.

Usage

```
## S3 method for class 'tsoutliers'
print(x, digits = max(3L, getOption("digits") - 3L), call = FALSE, ...)
```

Arguments

`x`	a list of class `tsoutliers` as returned by `tso`.
`digits`	integer, the number of significant digits to be used.
`call`	logical, if `TRUE` the `call` to the fitting method is printed.
`...`	further arguments, currently ignored.

Details

When `arima` is run from `do.call`, the latter generates a large object for the `call` which includes the entire structure of the data. This often implies a long output which may not be desired. Setting `call=FALSE` avoids printint the `call`.

Value

None.

See Also

tso.

`remove.outliers-deprecated`

Stage II of the Procedure: Discard Outliers

Description

As of version 0.6-6, `remove.outliers` has been renamed as `discard.outliers`. `discard.outliers` should be used.

Usage

```
remove.outliers(x, y, cval = NULL,
  method = c("en-masse", "bottom-up"),
  delta = 0.7, tsmethod.call = NULL,
  fdiff = NULL, logfile = NULL)
```

Arguments

`x`	a list. The output returned by `locate.outliers.oloop`.
`y`	a time series.
`cval`	a numeric. The critical value to determine the significance of each type of outlier.
`method`	a character. The method to discard/remove outliers. See details.
`delta`	a numeric. Parameter of the temporary change type of outlier.
`tsmethod.call`	an optional `call` object. The call to the function used to fit the time series model.
`fdiff`	currently ignored.
`logfile`	a character or `NULL`. It is the path to the file where tracking information is printed. Ignored if `NULL`.

Value

A list.

See Also

`discard.outliers`.

tso *Automatic Procedure for Detection of Outliers*

Description

These functions are the interface to the automatic detection procedure provided in this package.

Usage

```
tso(y, xreg = NULL, cval = NULL, delta = 0.7,
  types = c("AO", "LS", "TC"),
  maxit = 1, maxit.iloop = 4, maxit.oloop = 4, cval.reduce = 0.14286,
  discard.method = c("en-masse", "bottom-up"), discard.cval = NULL,
  remove.method, remove.cval,
  tsmethod = c("auto.arima", "arima"),
  args.tsmethod = NULL, logfile = NULL, check.rank = FALSE)

tso0(x, xreg = NULL, cval = 3.5, delta = 0.7,
  types = c("AO", "LS", "TC"), maxit.iloop = 4, maxit.oloop = 4,
  discard.method = c("en-masse", "bottom-up"), discard.cval = NULL,
  tsmethod = c("auto.arima", "arima"), args.tsmethod = NULL,
  logfile = NULL, check.rank = FALSE)
```

Arguments

y	a time series where outliers are to be detected.
x	a time series object.
xreg	an optional matrix of regressors with the same number of rows as y.
cval	a numeric. The critical value to determine the significance of each type of outlier.
delta	a numeric. Parameter of the temporary change type of outlier.
types	a character vector indicating the type of outlier to be considered by the detection procedure: innovational outliers ("IO"), additive outliers ("AO"), level shifts ("LS"), temporary changes ("TC") and seasonal level shifts ("SLS").
maxit	a numeric. The maximum number of iterations.
maxit.iloop	a numeric. The maximum number of iterations in the inner loop. See locate.outliers.
maxit.oloop	a numeric. The maximum number of iterations in the outer loop.
cval.reduce	a numeric. Factor by which cval is reduced if the procedure is run on the adjusted series, if maxit > 1.
discard.method	a character. The method used in the second stage of the procedure. See discard.outliers.
discard.cval	a numeric. The critical value to determine the significance of each type of outlier in the second stage of the procedure (discard outliers). By default it is set equal to cval. See details.

remove.method	deprecated, argument discard.method should be used.
remove.cval	deprecated, argument discard.cval should be used.
tsmethod	a character. The framework for time series modelling. It basically is the name of the function to which the arguments defined in args.tsmethod are referred to.
args.tsmethod	an optional list containing arguments to be passed to the function invoking the method selected in tsmethod.
logfile	a character or NULL. It is the path to the file where tracking information is printed. Ignored if NULL.
check.rank	logical. If TRUE the regressors are checked for perfect collinearity. The variables related to coefficients that turn out to be NA due to possible perfect collinearity are discarded.

Details

Five types of outliers can be considered. By default: "AO" additive outliers, "LS" level shifts, and "TC" temporary changes are selected; "IO" innovative outliers and "SLS" seasonal level shifts can also be selected.

tso0 is mostly a wrapper function around the functions locate.outliers and discard.outliers.

tso iterates around tso0 first for the original series and then for the adjusted series. The process stops if no additional outliers are found in the current iteration or if maxit iterations are reached.

tso0 is an auxiliar function (it is the workhorse for tso but it is not intended to be called directly by the user, use tso(maxit = 1, ...) instead. tso0 does not check the arguments since they are assumed to be passed already checked by tso; the default value for cval is not based on the sample size. For the time being, tso0 is exported in the NAMESPACE since it is convenient for debugging.

If no value is specified for argument cval a default value based on the sample size is used. Let n be the number of observations. If $n \leq 50$ then cval is set equal to 3.0; If $n \geq 450$ then cval is set equal to 4.0; otherwise cval is set equal to $3 + 0.0025 * (n - 50)$.

If tsmethod is NULL, the following default arguments are used in the function selected in tsmethod: tsmethod = "auto.arima": allowdrift = FALSE, ic = "bic"; tsmethod = "arima" = order = c(0, 1, 1) seasonal = list(order = c(0, 1, 1)).

If args.tsmethod is NULL, the following lists are used by default, respectively for each method: auto.arima: list(allowdrift = FALSE, ic = "bic"); arima: list(order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1))).

xreg must be a matrix with time series attributes, tsp, that must be the same same as tsp(x). Column names are also compulsory. If there is only one regressor it may still have non-null dimension, i.e. it must be a one-column matrix.

The external regressors (if any) should be defined in the argument xreg. However, they may be also defined as an element in args.tsmethod since this list is passed to function that fits the model. The function tso deals with this possibility and returns a warning if "xreg" is defined twice with different values. No checks are done in tso0.

If maxit = 1 the procedure is run only once on the original series. If maxit > 1 the procedure is run iteratively, first for the original series and then for the adjusted series. The critical value used for the adjusted series may be reduced by the factor cval.reduce, equal to 0.14286 by default. The new critical value is defined as $cval * (1 - cval.reduce)$.

By default, the same critical value is used in the first stage of the procedure (location of outliers) and in the second stage (discard outliers). Under the framework of structural time series models I noticed that the default critical value based on the sample size is too high, since all the potential outliers located in the first stage were discarded in the second stage (even in simulated series with known location of outliers). In order to investigate this issue, the argument `discard.cval` has been added. In this way a different critical value can be used in the second stage. Alternatively, the argument `discard.cval` could be omitted and simply choose a lower critical value, `cval`, to be used in both stages. However, using the argument `discard.cval` is more convenient since it avoids locating too many outliers in the first stage. `discard.cval` is not affected by `cval.reduce`.

Value

A list of class `tsoutliers`.

References

Chen, C. and Liu, Lon-Mu (1993). 'Joint Estimation of Model Parameters and Outlier Effects in Time Series'. *Journal of the American Statistical Association*, **88**(421), pp. 284-297.

Gómez, V. and Maravall, A. (1996). *Programs TRAMO and SEATS. Instructions for the user*. Banco de España, Servicio de Estudios. Working paper number 9628. http://www.bde.es/f/webbde/SES/Secciones/Publicaciones/PublicacionesSeriadas/DocumentosTrabajo/96/Fich/dt9628e.pdf

Gómez, V. and Taguas, D. (1995). *Detección y Corrección Automática de Outliers con TRAMO: Una Aplicación al IPC de Bienes Industriales no Energéticos*. Ministerio de Economía y Hacienda. Document number D-95006. https://www.sepg.pap.hacienda.gob.es/sitios/sepg/es-ES/Presupuestos/DocumentacionEstadisticas/Documentacion/Documents/DOCUMENTOS%20DE%20TRABAJO/D95006.pdf

See Also

`locate.outliers`, `discard.outliers`, `plot.tsoutliers`, `print.tsoutliers`.

Examples

```
## Not run:
data("hicp")
tso(y = log(hicp[[1]]))

## End(Not run)
```

Index