Package ‘tscount’

February 20, 2015

Type Package
Title Analysis of Count Time Series
Version 1.0.0
Date 2015-02-10
Description Likelihood-based methods for model fitting and assessment, prediction and intervention analysis of count time series following generalized linear models are provided. Models with the identity and with the logarithmic link function are allowed. The conditional distribution can be Poisson or Negative Binomial.

Imports parallel, ltsa
Suggests Matrix, xtable, gamlss.data
License GPL-2 | GPL-3
URL http://tscount.r-forge.r-project.org
ByteCompile true
LazyData true
Encoding UTF-8
NeedsCompilation no

Author Tobias Liboschik [aut, cre],
Roland Fried [aut],
Konstantinos Fokianos [aut],
Philipp Probst [aut],
Jonathan Rathjens [ctb]

Maintainer Tobias Liboschik <liboschik@statistik.tu-dortmund.de>
Repository CRAN
Date/Publication 2015-02-11 10:40:44

R topics documented:
tscout-package ......................................................... 2
campy ................................................................. 3
ecoli ................................................................. 4
Analysis of Count Time Series

Description

Collection of R functions for analysis of count time series. Currently the focus is on count time series following generalised linear models.

Details

See the main function `tsglm` for more details on the usage of the package. There is a vignette available which introduces the functionality of the package and its underlying statistical methods (vignette("tsglm", package="tscout").)

Author(s)

Tobias Liboschik <liboschik@statistik.tu-dortmund.de>
References


campy  
Campylobacter Infections Time Series

Description

Time series with the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. Campylobacterosis is an acute bacterial infectious disease attacking the digestive system.

Usage

campy

Format

A time series of class "ts".

Source


See Also

coli, ehec, influenza, measles in this package, polio in package gamlss.data
Examples

plot(campy)

# Fit the INGARCH model used in Ferland et al. (2006):
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
summary(campyfit)
plot(campyfit)
# Note that these parameter estimations differ from those obtained by
# Ferland et al. (2006). This might be due to a different initialisation
# of pre-sample values and different optimisation algorithms (they use
# Microsoft Excel Solver Macro).

ecoli

E. coli Infections Time Series

Description

Weekly number of reported disease cases caused by Escherichia coli in the state of North Rhine-Westphalia (Germany) from January 2001 to May 2013, excluding cases of EHEC and HUS.

Usage

ecoli

Format

A data frame with variables year and week giving the year and calendar week of observation, and with a variable cases giving the number of reported cases in the respective week.

Source


The data are provided with kind permission of the Robert Koch Institute. Further details and terms of usage are given at https://survstat.rki.de. More data reported under the German Infectious Diseases Protection Act is available via the SurvStat@RKI web application linked above.

See Also

campy, ehec, influenza, measles in this package, polio in package gamlss.data
Description

Weekly number of reported EHEC/HUS infections in the state of North Rhine-Westphalia (Germany) from January 2001 to May 2013.

Usage

ehec

Format

A data frame with variables year and week giving the year and calendar week of observation, and with a variable cases giving the number of reported cases in the respective week.

Source

Robert Koch Institute: SurvStat@RKI, https://survstat.rki.de, accessed on 10th June 2013. The data are provided with kind permission of the Robert Koch Institute. Further details and terms of usage are given at https://survstat.rki.de. More data reported under the German Infectious Diseases Protection Act is available via the SurvStat@RKI web application linked above.

See Also

campy, ecoli, influenza, measles in this package, polio in package gamlss.data

Description

Weekly number of reported influenza cases in the state of North Rhine-Westphalia (Germany) from January 2001 to May 2013.

Usage

influenza

Format

A data frame with variables year and week giving the year and calendar week of observation, and with a variable cases giving the number of reported cases in the respective week.
Source


The data are provided with kind permission of the Robert Koch Institute. Further details and terms of usage are given at https://survstat.rki.de. More data reported under the German Infectious Diseases Protection Act is available via the SurvStat@RKI web application linked above.

See Also
campy, ecoli, ehec, measles in this package, polio in package gamlss.data

--------------------------------------------------------------------------------
ingarch.analytical  Analytical Mean, Variance and Autocorrelation of an INGARCH Process
--------------------------------------------------------------------------------

Description

Functions to calculate the analytical mean, variance and autocorrelation / partial autocorrelation / autocovariance function of an integer-valued generalised autoregressive conditional heteroscedasticity (INGARCH) process.

Usage

ingarch.mean(intercept, past_obs=NULL, past_mean=NULL)
ingarch.var(intercept, past_obs=NULL, past_mean=NULL)
ingarch.acf(intercept, past_obs=NULL, past_mean=NULL, lag.max=10,
   type=c("acf", "pacf", "acvf"), plot=TRUE, ...)

Arguments

intercept  numeric positive value for the intercept $\beta_0$.
past_obs  numeric non-negative vector containing the coefficients $\beta_1, \ldots, \beta_p$ for regression on previous observations (see Details).
past_mean  numeric non-negative vector containing the coefficients $\alpha_1, \ldots, \alpha_q$ for regression on previous conditional means (see Details).
lag.max  integer value indicating how many lags of the (partial) autocorrelation / autocovariance function should be calculated.
type  character. If type="acf" (the default) the autocorrelation function is calculated, "pacf" gives the partial autocorrelation function and "acvf" the autocovariance function.
plot  logical. If plot=TRUE (the default) the values are plotted and returned invisible.
...  additional arguments to be passed to function plot.
**Details**

The INGARCH model of order $p$ and $q$ used here follows the definition

$$Z_t | F_{t-1} \sim \text{Poi}(\kappa_t),$$

where $F_{t-1}$ is the history of the process up to time $t-1$ and Poi is the Poisson distribution parametrised by its mean (cf. Ferland et al., 2006). The conditional mean $\kappa_t$ is given by

$$\kappa_t = \beta_0 + \beta_1 Z_{t-1} + \ldots + \beta_p Z_{t-p} + \alpha_1 \kappa_{t-1} + \ldots + \alpha_q \kappa_{t-q}.$$

The function `ingarch.acf` depends on the function `tacvfARMA` from package `ltsa`, which needs to be installed.

**Author(s)**

Tobias Liboschik

**References**


**See Also**

`tsglm` for fitting a more general GLM for time series of counts of which this INGARCH model is a special case. `tsglm.sim` for simulation from such a model.

**Examples**

```r
ingarch.mean(0.3, c(0.1,0.1), 0.1)
# Not run:
ingarch.var(0.3, c(0.1,0.1), 0.1)
ingarch.acf(0.3, c(0.1,0.1,0.1), 0.1, type="acf", lag.max=15)
# End(Not run)
```

---

**Description**

Generates covariates describing certain types of intervention effects according to the definition by Fokianos and Fried (2010).

**Usage**

```r
interv_covariate(n, tau, delta)
```
Arguments

n  integer value giving the number of observations the covariates should have.

tau integer vector giving the times where intervention effects occur.

delta numeric vector with constants specifying the type of intervention (see Details). Must be of the same length as tau.

Details

The intervention effect occurring at time $\tau$ is described by the covariate

$$X_t = \delta^{t - \tau} I_{[\tau, \infty)}(t),$$

where $I_{[\tau, \infty)}(t)$ is the indicator function which is 0 for $t < \tau$ and 1 for $t \geq \tau$. The constant $\delta$ with $0 \leq \delta \leq 1$ specifies the type of intervention. For $\delta = 0$ the intervention has an effect only at the time of its occurrence, for $0 < \delta < 1$ the effect decays exponentially and for $\delta = 1$ there is a persistent effect of the intervention after its occurrence.

If tau and delta are vectors, one covariate is generated with tau[1] as $\tau$ and delta[1] as $\delta$, another covariate for the second elements and so on.

Value

A matrix with $n$ rows and length(tau) columns. The generated covariates describing the interventions are the columns of the matrix.

Author(s)

Tobias Liboschik

References


See Also

tsglm for fitting a GLM for time series of counts. interv_test, interv_detect and interv_multiple for tests and detection procedures for intervention effects.

Examples

interv_covariate(n=140, tau=c(84,100), delta=c(1,0))
Description

Detection procedure for an intervention of given type occurring at unknown time as proposed by Fokianos and Fried (2010, 2012).

Usage

```r
## S3 method for class 'tsglm'
interv_detect(fit, taus=2:length(fit$ts), delta, external=FALSE, 
B=0L, info=c("score"), start.control_bootstrap, 
final.control_bootstrap, inter.control_bootstrap, 
parallel=FALSE, est_interv=TRUE, ...)
```

Arguments

- `fit`: an object of class "tsglm". Usually the result of a call to `tsglm`.
- `taus`: integer vector of time points which are considered for the possible intervention to occur. Default is to consider all possible time points.
- `delta`: numeric value that determines the type of intervention (see Details).
- `external`: logical value specifying whether the intervention's effect is external or not (see Details).
- `B`: positive integer value giving the number of bootstrap samples for estimation of the p-value. For `B=0` (the default) no p-value is returned.
- `info`: character value that determines how to calculate the information matrix, see `tsglm`. Currently "score" is the only possible choice.
- `start.control_bootstrap`: named list that determines how to make initial estimation in the bootstrap, see argument `start.control` in `tsglm`. If missing, the same settings as for the regular estimation are used.
- `final.control_bootstrap`: named list that determines how to make final maximum likelihood estimation in the bootstrap, see argument `final.control` in `tsglm`. If missing, the same settings as for the regular estimation are used. If `final.control_bootstrap=NULL`, then the model is not re-fitted for each bootstrap sample. Instead the parameters of the original fit which have been used for simulating the bootstrap samples are used. This approach saves computation time at the cost of a more conservative procedure, see Fokianos and Fried (2012).
- `inter.control_bootstrap`: named list determining how to maximise the log-likelihood function in an intermediate step, see argument `inter.control` in `tsglm`. If missing, the same settings as for the regular estimation are used.
parallel logical value. If parallel=TRUE, the bootstrap is distributed to multiple cores parallely. Requires a computing cluster to be initialised and registered as the default cluster by makeCluster and setDefaultCluster from package parallel.

est_interv logical value. If est_interv=TRUE a fit for the model with the intervention effect with the largest test statistic is computed and additionally returned.

... additional arguments passed to the fitting function tsglm.

Details

For each time in taus the score test statistic for an intervention effect occurring at that time is computed, see interv_test. The time with the maximum test statistic is considered as a candidate for a possible intervention effect at that time. The type of the intervention effect is specified by delta as described in interv_covariate. The intervention is included as an additional covariate according to the definition in tsglm. It can have an internal (the default) or external (external=TRUE) effect (see Liboschik et al., 2014).

If argument b is not NULL, the null hypothesis that there is no intervention effect at any time is tested. Test statistic for this test is the maximum test statistic of the score test (see above). The p-value is computed by a parametric bootstrap with b bootstrap samples. It is recommended to use at least several hundred bootstrap samples. Note that this bootstrap procedure is very time-consuming.

Value

An object of class "interv_detect", which is a list with at least the following components:

test_statistic maximum value of the score test statistics for all considered times in taus.
test_statistic_tau numeric vector of all score test statistics at the considered times in taus.
tau_max time at which the score test statistic has its maximum.
fit_H0 object of class "tsglm" with the fitted model under the null hypothesis of no intervention, see tsglm.
model_interv model specification of the model with the specified intervention at time tau_max.

If argument est_interv=TRUE (the default), the following component is additionally returned:

fit_interv object of class "tsglm" with the fitted model with the specified intervention at time tau_max, see tsglm.

Author(s)

Tobias Liboschik, Philipp Probst, Konstantinos Fokianos and Roland Fried

References


**See Also**

S3 methods `print` and `plot`.

`tsglm` for fitting a GLM for time series of counts. `interv_test` for testing on intervention effects and `interv_multiple` for iterative detection of multiple interventions of unknown types. `interv_covariate` for generation of deterministic covariates describing intervention effects.

**Examples**

```r
## Campylobacter infections in Canada (see help("campy"))
# Searching for a potential intervention effect:
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
campyfit_intervdetect <- interv_detect(fit=campyfit, taus=80:120, delta=1)
plot(campyfit_intervdetect)
```

```r
## Not run:
# Parallel computation for shorter run time on a cluster:
library(parallel)
ntasks <- 3
clust <- makeCluster(ntasks)
setDefaultCluster(cl=clust)
interv_detect(fit=campyfit, taus=80:120, delta=1, B=500, parallel=TRUE)
```

---

**interv_multiple.tsglm**  
*Detecting Multiple Interventions in Count Time Series Following Generalised Linear Models*

**Description**

Iterative detection procedure for multiple interventions of unknown types occurring at unknown times as proposed by Fokianos and Fried (2010, 2012).

**Usage**

```r
## S3 method for class 'tsglm'
interv_multiple(fit, taus=2:length(fit$ts), deltas=c(0,0.8,1),
               external=FALSE, B=10, signif_level=0.05,
               start.control_bootstrap, final.control_bootstrap,
               inter.control_bootstrap, parallel=FALSE, ...)
```
Arguments

- **fit**: an object of class "tsglm". Usually the result of a call to `tsglm`.
- **taus**: integer vector of times which are considered for the possible intervention to occur. Default is to consider all times.
- **deltas**: numeric vector that determines the types of intervention to be considered (see Details).
- **external**: logical value specifying whether the interventions effect is external or not (see Details).
- **B**: positive integer value giving the number of bootstrap samples for estimation of the p-value.
- **signif_level**: numeric value with $0 < \text{signif\_level} < 1$ giving a significance level for the procedure.
- **start.control_bootstrap**: named list that determines how to make initial estimation in the bootstrap, see argument `start.control` in `tsglm`. If missing, the same settings as for the regular estimation are used.
- **final.control_bootstrap**: named list that determines how to make final maximum likelihood estimation in the bootstrap, see argument `final.control` in `tsglm`. If missing, the same settings as for the regular estimation are used. If `final.control_bootstrap=NULL`, then the model is not re-fitted for each bootstrap sample. Instead the parameters of the original fit which have been used for simulating the bootstrap samples are used. This approach saves computation time at the cost of a more conservative procedure, see Fokianos and Fried (2012).
- **inter.control_bootstrap**: named list determining how to maximise the log-likelihood function in an intermediate step, see argument `inter.control` in `tsglm`. If missing, the same settings as for the regular estimation are used.
- **parallel**: logical value. If `parallel=TRUE`, the bootstrap is distributed to multiple cores parallely. Requires a computing cluster to be initialised and registered as the default cluster by `makeCluster` and `setDefaultCluster` from package `parallel`.
- **...**: additional arguments passed to the function for detection of single intervention effects `interv_detect` and via this function some of the arguments are passed to the fitting function `tsglm`.

Details

This function performs an iterative procedure for detection of multiple intervention effects. In each step the function `interv_detect` is applied for each of the possible intervention types provided in the argument `deltas`. If there is (after a Bonferroni correction) no significant intervention effect the procedure stops. Otherwise the type of intervention with the minimum p-value is chosen. In case of equal p-values preference is given to a level shift (i.e. $\delta = 1$) and then to the type of intervention with the largest test statistic. The effect of the chosen intervention is removed from the time series. The time series cleaned from the intervention effect is tested for further interventions in a next step.

For each time in `taus` the test statistic of a score test on an intervention effect occurring at that time is computed, see `interv_test`. The time with the maximum test statistic is considered as
a candidate for a possible intervention effect at that time. The type of the intervention effect is specified by delta as described in \textit{interv_covariate}. The intervention is included as an additional covariate according to the definition in \textit{tsglm}. It can have an internal (the default) or external (\texttt{external=TRUE}) effect (see Liboschik et al., 2014). All p-values given in the output are multiplied by the number of intervention types considered to account for the multiple testing in each step by a Bonferroni correction. Note that this correction can lead to p-values greater than one.

Note that this bootstrap procedure is very time-consuming.

\textbf{Value}

An object of class "interv_multiple", which is a list with the following components:

- \texttt{interventions} data frame giving the detected interventions, which has the variables tau, delta, size, test_statistic and p-value.
- \texttt{fit_H0} object of class "tsglm" with the fitted model under the null hypothesis of no intervention, see \textit{tsglm}.
- \texttt{fit_cleaned} object of class "tsglm" with the fitted model for the cleaned time series after the last step of the iterative procedure, see \textit{tsglm}.
- \texttt{model_interv} model specification of the model with all detected interventions at their respective times.
- \texttt{fit_interv} object of class "tsglm" with the fitted model with all detected interventions at their respective times, see \textit{tsglm}.
- \texttt{track} named list of matrices with the detailed results of the iterative detection procedure. Element tau_max gives the times where the test statistic has its maximum for each type of intervention and in each iteration step and element size gives the estimated sizes of the respective intervention effects. Elements test_statistic and p_value require no further explanation.

\textbf{Author(s)}

Tobias Liboschik, Philipp Probst, Konstantinos Fokianos and Roland Fried

\textbf{References}


\textbf{See Also}

S3 methods \texttt{print} and \texttt{plot}.
\textit{tsglm} for fitting a GLM for time series of counts. \textit{interv_test} for testing for intervention effects and \textit{interv_detect} for detection of single interventions of given type. \textit{interv_covariate} for generation of deterministic covariates describing intervention effects.
**Examples**

```r
# Not run:
### Campylobacter infections in Canada (see help("campy"))
# Searching for potential intervention effects (runs several hours!):
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13))
campyfit_intervmultiple <- interv_multiple(fit=campyfit, taus=80:120, deltas=c(0,0.8,1), B=500, signif_level=0.05)
campyfit_intervmultiple
plot(campyfit_intervmultiple)
# Parallel computation for shorter run time on a cluster:
library(parallel)
ntasks <- 3
clust <- makeCluster(ntasks)
setDefaultCluster(cl=clust)
interv_multiple(fit=campyfit, taus=80:120, deltas=c(0,0.8,1), B=500,
               signif_level=0.05, parallel=TRUE)
## End(Not run)
```

---

**interv_test.tsglm**  
*Testing for Interventions in Count Time Series Following Generalised Linear Models*

**Description**

Test for one or more interventions of given type at given time as proposed by Fokianos and Fried (2010, 2012).

**Usage**

```r
## S3 method for class 'tsglm'
interv_test(fit, tau, delta, external, info=c("score"), est_interv=FALSE, ...)
```

**Arguments**

- `fit` an object of class "tsglm". Usually the result of a call to `tsglm`.
- `tau` integer vector of times at which the interventions occur which are tested for.
- `delta` numeric vector that determines the types of the interventions (see Details). Must be of the same length as `tau`.
- `external` logical vector of length `length(tau)` specifying for each intervention whether its effect is external or not (see Details). If this is only a scalar this choice will be used for all interventions. If this is only a scalar this choice will be used for all interventions. If omitted all interventions will have an internal effect (i.e. `external=FALSE`).
- `info` character value that determines how to calculate the information matrix, see `tsglm`. Currently "score" is the only possible choice.
est_interv logical value. If est_interv=TRUE a fit for the model with all specified interventions is computed and additionally returned.

... additional arguments passed to the fitting function tsglm.

Details

A score test on the null hypothesis of no interventions is done. The null hypothesis is that the data are generated from the model specified in the argument model, see definition in tsglm. Under the alternative there are one or more intervention effects occurring at times tau. The types of the intervention effects are specified by delta as defined in interv_covariate. The interventions are included as additional covariates according to the definition in tsglm. It can have an internal (the default) or external (external=TRUE) effect (see Liboschik et al., 2014).

Under the null hypothesis the test statistic has asymptotically a chi-square distribution with length(tau) (i.e. the number of breaks) degrees of freedom. The returned p-value is based on this and approximately valid for long time series, i.e. when length(ts) large.

Value

An object of class "interv_test", which is a list with at least the following components:

test_statistic value of the test statistic.
df degrees of freedom of the chi-squared distribution the test statistic is compared with.
pMvalue p-value of the test.
fit_H0 object of class "tsglm" with the fitted model under the null hypothesis of no intervention, see tsglm.
model_interv model specification of the model with the specified interventions.

If argument est_interv=TRUE, the following component is additionally returned:

fit_interv object of class "tsglm" with the fitted model with the specified interventions, see tsglm.

Author(s)

Tobias Liboschik, Philipp Probst, Konstantinos Fokianos and Roland Fried

References


invertinfo

Compute a Covariance Matrix from a Fisher Information Matrix

Description

Stable function for computing a covariance matrix from a given Fisher information matrix by inversion.

Usage

invertinfo(mat, silent=TRUE, stopOnError=FALSE)

Arguments

mat  a Fisher Information Matrix.
silent logical value. If FALSE, errors in the computation of the inverse while using the Cholesky decomposition algorithm are printed. If TRUE, errors can be seen only in the value error_message.
stopOnError logical value. If TRUE only an error message is printed in case of error.

Details

A Cholesky decomposition is used to obtain the covariance matrix. This can be done because the Fisher information matrix is symmetric and positive definite.

This function is meant to be a more stable alternative to the function solve, which does not take into account, that the matrix is symmetric and positive definite.
Value

A list containing the following components:

- `vcov`: the covariance matrix.
- `error_message`: possible error messages that occurred when inverting the Fisher information matrix.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

- `chol` and `chol2inv`.

Examples

```r
library(Matrix)
invertinfo(Hilbert(5), stopOnError=TRUE)
invertinfo(Hilbert(100))
invertinfo(Hilbert(100), silent=FALSE)
## Not run: invertinfo(Hilbert(100), stopOnError=TRUE)
```

marcal.tsglm  
Predictive Model Assessment with a Marginal Calibration Plot for Time Series Following Generalised Linear Models

Description

The function produces a marginal calibration plot.

Usage

```r
## S3 method for class 'tsglm'
marcal(object, plot=TRUE, ...)
```

Arguments

- `object`: an object of class "tsglm".
- `plot`: logical. If plot=TRUE (the default), the marginal calibration is plotted and the underlying data are returned invisibly only.
- `...`: additional arguments to be passed to `plot`.

Details

Marginal Calibration can be assessed by taking the difference between the average predictive cumulative distribution function (c.d.f.) and the empirical c.d.f. of the observations. Minor fluctuations about zero are expected if the marginal calibration hypothesis is true. For more information about marginal calibration see the references listed below.
Value

Produces a plot of the difference between the average predictive cumulative distribution function (c.d.f.) and the empirical c.d.f. of the observations at each value between the highest and lowest observation of the time series (only for plot=TRUE).

Returns a list with elements x and y, where x are the threshold values and y the respective differences of predictive and empirical cumulative distribution function (invisibly for plot=TRUE).

Author(s)

Philipp Probst and Tobias Liboschik

References


See Also

tsglm for fitting a GLM for time series of counts.
pit and scoring for other predictive model assessment tools for time series following GLMs.

Examples

```r
###Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
marcal(campyfit)
```

measles | Measles Infections Time Series

Description

Weekly number of reported measles infections in the state of North Rhine-Westphalia (Germany) from January 2001 to May 2013.

Usage

measles
Format

A data frame with variables year and week giving the year and calendar week of observation, and with a variable cases giving the number of reported cases in the respective week.

Source

Robert Koch Institute: SurvStat@RKI, https://survstat.rki.de, accessed on 10th June 2013. The data are provided with kind permission of the Robert Koch Institute. Further details and terms of usage are given at https://survstat.rki.de. More data reported under the German Infectious Diseases Protection Act is available via the SurvStat@RKI web application linked above.

See Also

campy, ecoli, ehec, influenza in this package, polio in package gamlss.data

Description

The function allows a probabilistic calibration check with a Probability Integral Transform (PIT) histogram.

Usage

## S3 method for class 'tsglm'
pit(object, bins=10, ...)

Arguments

object an object of class "tsglm".

bins number of bins in the histogram. Default value is 10.

... additional arguments passed to plot.

Details

A PIT histogram is a tool for evaluating the statistical consistency between the probabilistic forecast and the observation. The predictive distributions of the observations are compared with the actual observations. If the predictive distribution is ideal the result should be a flat PIT histogram with no bin having an extraordinary high or low level. For more information about PIT histograms see the references listed below.

Author(s)

Philipp Probst and Tobias Liboschik
References


See Also

*tsglm* for fitting a GLM for time series of counts.

*marcal* and *scoring* for other predictive model assessment tools for time series following GLMs.

Examples

```r
### Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
plot(campyfit)
```

---

**plot.interv_detect**

*Plot Test Statistic of Intervention Detection Procedure for Count Time Series Following Generalised Linear Models*

Description

Provides a plot of the test statistics of a test on an intervention in GLM-type count time series (as returned by *interv_detect.tsglm*) against time.

Usage

```r
## S3 method for class 'interv_detect'
plot(x, ...)
```

Arguments

- `x` an object of class "interv_detect", usually a result of a call to *interv_detect.tsglm*.
- `...` additional arguments to be passed to function *plot*.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

*interv_detect* for detecting an intervention effect in GLM-type count time series and *tsglm* for fitting such a model.
Examples

```r
## Not run:
## Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
campyfit_intervdetect <- interv_detect(fit=campyfit, taus=80:120,
                                      delta=1, external=FALSE)
# This example runs about 20 minutes on a single processing unit,
# of course depending on its speed.
plot(campyfit_intervdetect)
## End(Not run)
```

Description

Provides a plot with the intervention effects detected by an iterative procedure (as returned by `interv_multiple.tsglm`) and the time series cleaned from these intervention effects.

Usage

```r
## S3 method for class 'interv_multiple'
plot(x, ...
```

Arguments

- `x` an object of class "interv_multiple", usually a result of a call to `interv_detect`.
- `...` additional arguments to be passed to function `plot`.

Details

The vertical red lines indicate where possible interventions were found and the dashed blue line is the time series cleaned from all detected intervention effects.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

- `interv_multiple` for detecting multiple intervention effects in GLM-type count time series and `tsglm` for fitting such a model.
Examples

```r
## Not run:
## Not run (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7, 13)))
campyfit_intervmultiple <- interv_multiple(fit=campyfit, taus=80:120, deltas=c(0, 0.8, 1), external=FALSE, B=2, signif_level=0.05) # runs several hours!
plot(campyfit_intervmultiple)
## End(Not run)
```

plot.tsglm

Diagnostic Plots for a Fitted GLM-type Model for Time Series of Counts

Description

Produces several diagnostic plots to assess the fit of a GLM-type model for time series of counts.

Usage

```r
## S3 method for class 'tsglm'
plot(x, ask = TRUE, ...)
```

Arguments

- `x`: an object of class "tsglm". Usually the result of a call to `tsglm`.
- `ask`: logical value. If TRUE (and the R session is interactive) the user is asked for input, before a new figure is drawn (see `devAskNewPage`).
- `...`: further arguments are currently ignored. Only for compatibility with generic function.

Details

Produces plots of the acf of the Pearson residuals, the Pearson residuals plotted against time, a cumulative periodogram of the Pearson residuals, a probability integral transform (PIT) histogram (see function `pit`) and a marginal calibration plot (see function `marcal`). The cumulative periodogram is plotted with the function `cpgram` from package MASS and is omitted with a warning if this package is not available.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

`tsglm` for fitting a GLM for time series of counts.
Examples

```r
## Campylobacter infections in Canada (see help("campy"))
interventions <- interv_covariate(n=\text{length(campy)}, \text{tau}=c(84, 100),
delta=c(1, 0)) # detected by Fokianos and Fried (2010, 2012)
# Linear link function with Negative Binominal distribution:
campyfit <- tsglm(campy, model=list(past_obs=1, past_mean=13),
xreg=interventions, dist="nbinom")
plot(campyfit)
```

---

### predict.tsglm

**Predicts Method for Time Series of Counts Following Generalised Linear Models**

Description

Predict future observations based on a fitted GLM-type model for time series of counts.

Usage

```
## S3 method for class 'tsglm'
predict(object, n.ahead = 1, newobs = NULL, newxreg = NULL, level=0.95, B, ...)
```

Arguments

- **object**
  - an object of class "tsglm". Usually the result of a call to tsglm.

- **n.ahead**
  - positive integer value giving the number of steps ahead for which predictions should be made.

- **newobs**
  - integer vector of known future observations of the time series. This argument is only relevant if more than one observation ahead is to be predicted (\text{n.ahead} greater than 1). The \text{h}-step-ahead prediction for \text{h} > 1 is computed as a 1-step-ahead prediction given all previous values, which can be observations of the original time series or new observations provided in this argument. Previous observations which are not available are replaced by their respective 1-step-ahead prediction.

- **newxreg**
  - matrix or vector containing new values for the covariates to be used for prediction. If newxreg is omitted or contains less rows than the value of n.ahead, the last known values of the covariates are used for prediction. This is often not reasonable and it is strongly advised to explicitly make assumptions on future covariates and to specify the argument xreg accordingly.

- **level**
  - numeric value determining the confidence level for prediction intervals. Note that this level holds only for each of the prediction intervals but not globally.

- **B**
  - positive integer value giving the number of bootstrap samples to use for numerical determination of prediction intervals with a confidence level of level each. If missing no prediction intervals are computed.

- **...**
  - further arguments are currently ignored. Only for compatibility with generic function.
Details

Returns predictions for the \( n_{\text{ahead}} \) observations following the fitted time series contained in argument \texttt{object}. The 1-step-ahead prediction is the conditional expectation of the observation to be predicted given the past. The true parameters are replaced by their estimations given in argument \texttt{object}. For a 2-step-ahead-prediction the true previous observation is used when given in argument \texttt{newobs}, otherwise it is replaced by the 1-step-ahead prediction computed before. For a 3-step-prediction this holds for the previous two observations, which are replaced by their respective predictions if not available, and so on.

The function additionally returns prediction intervals numerically determined by a parametric bootstrap, if the number of bootstrap replicates is specified by the argument \texttt{B}. Note that the prediction intervals do not reflect the additional uncertainty induced by the parameter estimation. However, for sufficiently long time series it is expected that this uncertainty is negligible compared to the uncertainty of the predictive distribution.

Value

A list with the following element:

\texttt{pred} a numeric vector of the predictions. Has class "ts" if the response used for fitting has this class.

If the predictive distribution is approximated by simulation (i.e. argument \texttt{B} specifying the number of simulated trajectories of the process is provided), the list has the additional elements:

\texttt{median} a vector giving the empirical median of the simulated predictive distribution for each of the future time points. Has class "ts" if the response used for fitting has this class.

\texttt{interval\_shortest} a matrix with the columns "lower" and "upper" giving the lower and upper boundaries of prediction intervals for the future time points, each with an intended coverage rate as given in argument \texttt{level}. The prediction intervals are obtained from the simulated predictive distribution such that they have minimal length. Has class "ts" if the response used for fitting has this class.

\texttt{interval\_quantiles} like \texttt{interval\_shortest}, but the prediction intervals are given by the empirical \( a \)- and \( 1-a \)-quantiles of the simulated predictive distribution, with \( a=(1-\texttt{level})/2 \).

\texttt{type} a character value saying which method was used for computing the prediction intervals. Currently this is always the method "bootstrap".

\texttt{B} an integer value giving the number of bootstrap samples which were used for computing prediction intervals.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

tsglm for fitting a GLM for time series of counts.
residuals.tsglm

Examples

```r
### Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
predict(campyfit, n.ahead=5)
```

---

residuals.tsglm  Residuals of a Generalised Linear Model for Time Series of Counts

Description

Returns the residuals of a fitted GLM-type model for time series of counts.

Usage

```r
# S3 method for class 'tsglm'
residuals(object, type = c("response", "pearson", "anscombe"), ...)
```

Arguments

- `object` an object of class "tsglm". Usually the result of a call to `tsglm`.
- `type` character value giving the type of residuals which should be returned. Choose type="response" for raw residuals, type="pearson" for Pearson residuals and type="anscombe" for Anscombe residuals.
- `...` further arguments are currently ignored. Only for compatibility with generic function.

Details

Computes a vector with the respective residuals of the fit given in argument `object`.

Value

Numerical vector of the residuals.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

tsglm for fitting a GLM for time series of counts.

Examples

```r
### Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
campyfit_resid <- residuals(campyfit, type="pearson")
plot(campyfit_resid)
acf(campyfit_resid)
```
Computes scoring rules for the assessment of sharpness of a fitted GLM-type model for time series of counts.

Usage

```r
# S3 method for class 'tsglm'
scoring(object, cutoff=1000, ...)
```

Arguments

- `object`: an object of class "tsglm".
- `cutoff`: positive integer. Summation over the infinite sample space \{0,1,2,...\} of a distribution is cut off at this value. This affects the quadratic, spherical and ranked probability score.
- `...`: further arguments are currently ignored. Only for compatibility with generic function.

Details

The scoring rules are penalties that should be minimised for a better forecast, so a smaller scoring value means better sharpness. Different competing forecast models can be ranked via these scoring rules. They are computed as follows: For each score \( s \) and time \( t \) the value \( s(P_t,Y_t) \) is computed, where \( P_t \) is the predictive c.d.f. and \( Y_t \) is the observation at time \( t \). To obtain the overall score for one model the average of the score of all observations \( \frac{1}{n} \sum_{t=1}^{n} s(P_t,Y_t) \) is calculated.

For all \( t \geq 1 \), let \( p_y = P(Y_t = y|F_{t-1}) \) be the density function of the predictive distribution at \( y \) and \( ||p||^2 = \sum_{y=0}^{\infty} p_y^2 \) be a quadratic sum over the whole sample space \( y = 0,1,2,... \) of the predictive distribution. \( \mu_{P_t} \) and \( \sigma_{P_t} \) are the mean and the standard deviation of the predictive distribution, respectively.

Then the scores are defined as follows:

- Logarithmic score: \( \text{logs}(P_t,Y_t) = -log p_y \)
- Quadratic or Brier score: \( \text{qs}(P_t,Y_t) = -2p_y + ||p||^2 \)
- Spherical score: \( \text{sphs}(P_t,Y_t) = \frac{-p_y}{||p||} \)
- Ranked probability score: \( \text{rps}(P_t,Y_t) = \sum_{x=0}^{\infty} (P_t(x) - 1(Y_t \leq x))^2 \) (sum over the whole sample space \( x = 0,1,2,... \))
- Dawid-Sebastiani score: \( \text{dss}(P_t,Y_t) = \left( \frac{Y_t - \mu_{P_t}}{\sigma_{P_t}} \right)^2 + 2log\sigma_{P_t} \)
- Normalized squared error score: \( \text{nse}(P_t,Y_t) = \left( \frac{Y_t - \mu_{P_t}}{\sigma_{P_t}} \right)^2 \)
Squared error score: $ses(P_t, Y_t) = (Y_t - \mu_{P_t})^2$

For more information on scoring rules see the references listed below.

Value

Returns a named vector of the following scoring rules:

- **logarithmic**: Logarithmic score
- **quadratic**: Quadratic or Brier score
- **spherical**: Spherical score
- **rankprob**: Ranked probability score
- **dawseb**: Dawid-Sebastiani score
- **normsq**: Normalized squared error score
- **sqerror**: Squared error score

Author(s)

Philipp Probst and Tobias Liboschik

References


See Also

tsglm for fitting a GLM for time series of counts.

pit and marcal for other predictive model assessment tools for time series following GLMs.

Examples

```r
###Campylobacter infections in Canada (see help("campy"))
campyfit <- tsglm(ts=campy, model=list(past_obs=1, past_mean=c(7,13)))
scoring(campyfit)
```
se.tsglm  

Standard Errors of a Fitted Generalised Linear Model for Time Series of Counts

Description

Computes the standard errors for the parameters of a fitted GLM-type model for time series of counts.

Usage

## S3 method for class 'tsglm'
se(object, B, parallel, ...)

Arguments

- **object**: an object of class "tsglm". Usually the result of a call to tsglm.
- **B**: positive integer value giving the number of bootstrap samples to use for estimation of the standard errors. If missing the standard errors are based on a normal approximation.
- **parallel**: logical value. If parallel=TRUE, the bootstrap is distributed to multiple cores parallelly. Requires a computing cluster to be initialised and registered as the default cluster by makeCluster and setDefaultCluster from package parallel.
- **...**: additional arguments to be passed to the fitting function tsglm. Only made use of if the standard errors are computed by a bootstrap procedure.

Details

By default the standard errors are based on a normal approximation of the (quasi) maximum likelihood estimator. They are the square roots of the diagonal elements of the inverse of the information matrix. There is no analytical approximation of the standard error for the overdispersion coefficient sigmasq, which is therefore set to NA.

If the number of bootstrap samples B is given, the standard errors are computed by a parametric bootstrap. The standard errors are the empirical standard deviation of the parameter estimations of B random samples drawn from the fitted model given in argument object.

Value

A list with the following components:

- **est**: a vector of the maximum likelihood estimated coefficients.
- **se**: a vector of the standard errors of each estimated coefficient.
- **type**: a character value "normapprox" or "bootstrap" giving how the standard errors are computed.
If the standard errors are computed by a parametric bootstrap procedure, the following component is additionally returned:

- positive integer value giving the number of bootstrap samples used for estimation of the standard errors.

**Author(s)**

Tobias Liboschik and Philipp Probst

**See Also**

- `tsglm` for fitting a GLM for time series of counts.

**Examples**

```r
### Road casualties in Great Britain (see help("Seatbelts"))
timeseries <- Seatbelts[, "VanKilled"]
regressors <- cbind(PetrolPrice=Seatbelts[, c("PetrolPrice")],
                    linearTrend=seq(along=timeseries)/12)
# Logarithmic link function with Poisson distribution:
seatbeltsfit <- tsglm(ts=timeseries, link="log",
                      model=list(past_obs=c(1, 12)), xreg=regressors, distr="pois")

## Not run: system.time(stderror <- se(seatbeltsfit, B=100)) # by bootstrap
# This estimation of bootstrap standard errors takes several minutes on a single
# processing unit, of course depending on its speed.
# Parallel computation for shorter run time on a cluster:
library(parallel)
ntasks <- 3
clust <- makeCluster(ntasks)
setDefaultCluster(cl=clust)
system.time(stderror <- se(seatbeltsfit, B=100, parallel=TRUE))
## End(Not run)
```

**Summary**

`summary.tsglm` 

*Summarising Fits of Count Time Series following Generalised Linear Models*

**Description**

summary method for class "tsglm".

**Usage**

```r
### S3 method for class 'tsglm'
summary(object, B, parallel=FALSE, ...)
```
Arguments

- object: an object of class "tsglm". Usually the result of a call to `tsglm`.
- `B`: controls the computation of standard errors. Is passed to `se`.
- `parallel`: controls the computation of standard errors. Is passed to `se`.
- `...`: further arguments are currently ignored. Only for compatibility with generic function.

Details

Computes and returns a list of summary statistics of the fitted model given in argument `object`.

Value

A named list with the following elements:

- `call`: see `tsglm`.
- `link`: see `tsglm`.
- `distr`: see `tsglm`.
- `residuals`: see `tsglm`.
- `coefficients`: data frame with estimated parameters and their standard errors (based on a normal approximation or a parametric bootstrap, see `se`).
- `numberCoeff`: number of coefficients.
- `se.type`: type of standard errors, see `se.tsglm`.
- `se.bootstrap.samples`: number of bootstrap samples used for estimation of the standard errors, see `se.tsglm`. Is omitted if the standard errors are not obtained by a bootstrap procedure.
- `logLik`: value of the maximised log-likelihood function.
- `AIC`: Akaike’s information criterion (AIC).
- `BIC`: Bayesian information criterion (BIC).
- `pearson.resid`: Pearson residuals, see `residuals.tsglm`.

Author(s)

Tobias Liboschik and Philipp Probst

See Also

S3 method `print`.

`tsglm` for fitting a GLM for time series of counts.
Examples

```r
### Road casualties in Great Britain (see help("Seatbelts"))
timeseries <- Seatbelts[, "VanKilled"]
regressors <- cbind(PetrolPrice=Seatbelts[, c("PetrolPrice")],
                   linearTrend=seq(along=timeseries)/12)
# Logarithmic link function with Poisson distribution:
seatbeltsfit <- tsglm(ts=timeseries, link="log",
                      model=list(past_obs=c(1, 12)), xreg=regressors, distr="pois")
summary(seatbeltsfit)
```

\section*{tsglm} \hspace{1cm} \textit{Count Time Series Following Generalised Linear Models}

\section*{Description}

The function \textsf{tsglm} fits a generalised linear model (GLM) for time series of counts. The specification of the linear predictor allows for regressing on past observations, past values of the linear predictor and covariates as defined in the Details section. There is the so-called INGARCH model with the identity link \cite[see for example]{ferland2006}, \cite{fokianos2009} and another model with the logarithmic link \cite[see for example]{fokianos2011}, which also differ in the specification of the linear predictor. The conditional distribution can be chosen to be either Poisson or negative binomial.

Estimation is done by conditional maximum likelihood for the Poisson distribution or by a conditional quasi-likelihood approach based on the Poisson likelihood function for the negative binomial distribution.

There is a vignette available which introduces the functionality of \textsf{tsglm} and related functions of this package and its underlying statistical methods \cite{vignette(tsglm, package=tscount)}.

The function \textsf{ingarch.fit} is a lower level function to fit the mean specification of an INGARCH model assuming a Poisson distribution. It is called by \textsf{tsglm}. It has additional arguments allowing for a finer control of the fitting procedure, which can be handed over from the function \textsf{tsglm} by its \ldots argument. Analogously, \textsf{loglin.fit} is the lower level fitting function for the model with the logarithmic link. Note that it is usually not necessary for a user to call these lower level functions nor to worry about the additional arguments provided by these functions. The defaults of these arguments have been chosen wisely by the authors of this package and should perform well in most applications.

\section*{Usage}

```r
tsglm(ts, model = list(past_obs = NULL, past_mean = NULL,
                   external = NULL), xreg = NULL, link = c("identity", "log"),
                   distr = c("poisson", "nbinom"), ...)

ingarch.fit(ts, model = list(past_obs = NULL, past_mean = NULL,
                     external = NULL), xreg = NULL, score = TRUE,
               info = c("score", "none", "hessian", "sandwich"),
               init.method=c("marginal", "iid", "firstobs", "zero"),
```


**Arguments**

- `ts` a univariate time series.
- `model` a named list specifying the model for the linear predictor, which can be of the following elements:
  - `past_obs` integer vector giving the previous observations to be regressed on (autoregression). This is a vector with the elements $i_1, \ldots, i_p$ (see Details). If omitted, or of length zero, there will be no regression on previous observations.
  - `past_mean` integer vector giving the previous conditional means to be regressed on. This is a vector with the elements $j_1, \ldots, j_q$ (see Details). If omitted, or of length zero, there will be no regression on previous conditional means.
  - `external` logical vector of length `ncol(xreg)` specifying for each covariate whether its effect should be external or not (see Details). If this is a scalar this choice will be used for all covariates. If omitted, all covariates will have an internal effect (i.e. `external=FALSE`).
  - `xreg` matrix with covariates in the columns, i.e. its number of rows must be `length(ts)`. This is the matrix $X$ (see Details). If omitted no covariates will be included. For the identity link the covariates have to be non-negative.
  - `link` character giving the link function. Default is "identity", fitting an INGARCH model. Another possible choice is "log", fitting a log-linear model.
  - `distr` character giving the conditional distribution. Default is "poisson", i.e. a Poisson distribution.
  - `...` additional arguments to be passed to the lower level fitting functions `ingarch.fit` or `loglin.fit`. See below.
  - `score` logical value indicating whether the score vector should be computed.
  - `info` character that determines if and how to compute the information matrix. Can be set to "score" (the default) for calculation via the outer product of the score vector, or to "hessian" for calculation via the Hessian matrix of second derivatives. For `info="sandwich"` the information matrix is estimated by a sandwich formula using both the outer score product and the Hessian matrix. If set to "none", no information matrix is computed. For `link="log"` the alternatives "hessian" and "sandwich" are currently not available and will result in an error. For `distr="nbinom"` one can only use `info="score"`. 

```r
loglin.fit(ts, model = list(past_obs = NULL, past_mean = NULL, external = NULL), xreg = NULL, score = TRUE, info = c("score", "none"), init.method=c("marginal", "iid", "firstobs", "zero"), init.drop = FALSE, epsilon = 1e-06, slackvar = 1e-06, start.control = list(), final.control = list(), inter.control = NULL)
```
init.method character that determines how the recursion of the conditional mean (and possibly of its derivatives) is initialised. If set to "marginal" (the default), the marginal mean of a model without covariates and its derivatives are used. If set to "iid", all values are initialised by the marginal mean under the assumption of i.i.d. data, which depends on the intercept only. If set to "firstobs" the first observation is used. If set to "zero", the recursions are initialised by the value zero.

init.drop logical value that determines which observations are considered for computation of the log-likelihood, the score vector and, if applicable, the information matrix. If TRUE, the first max(model$past.obs) observations, which are needed for the autoregression, are not considered. If FALSE (the default), all observations are considered and pre-sample values determined by the method specified by the argument itinit.method are used for the autoregression. Note that in the first case the effective number of observations used for maximum likelihood estimation is lower than the total number of observations of the original time series. Consequently only this lower number of observations is considered in the output. Note that for init.drop=TRUE the log-likelihood function for models of different orders might not be comparable if the effective number of observations is different.

epsilon numeric positive but small value determining how close the parameters may come to the limits of the parameter space.

slackvar numeric positive but small value determining how true inequalities among the parameter restrictions are treated; a true inequality x < y will be transformed to x + slackvar <= y.

start.control named list with optional elements that determine how to make the start estimation. Possible list elements are:

use integer vector of length one or two giving the number of observations from the beginning (if of length one) or the range of observations (if of length two) used for start estimation. For use = Inf all observations are used, which is the default.

method character specifying how start estimators should be estimated. Possible values are "iid", "CSS", "CSS-ML", "ML", "MM", "GLM" and "fixed". If method is "iid" (the default), a moment estimator assuming an iid model without covariates is used. If method="MM", the start estimate is the ARMA(1,1) fit by moment estimators and parameters of higher order than one are set to zero. For this method the starting parameter values for the covariates are zero by default and can be set by the list element xreg. If method is "CSS", "CSS-ML" or "ML", the start estimate is based on an ARMA fit using the function arima, and list element method is passed to its argument of the same name. If method="GLM", the estimated parameters of a generalised linear model with regression on the specified past observations and covariates, but not on past conditional means, are used as start estimates. Initial estimates for the coefficients of past conditional means are set to zero. If method="fixed", parameters given in further named list elements of start.control are used when available, else the predefined values given in the following are used.
intercept numeric value with the start value for the intercept parameter. Default value is 1.
past_obs numeric vector with the start values for parameters for regression on previous observations. Default values are zero.
past_mean numeric vector with the start values for parameters for regression on previous conditional means. Default values are zero.
xreg numeric vector with the start values for the regression parameters. These values will also be used if method="MM". Default values are zero.

final.control named list with optional elements that determine how to make the final maximum likelihood estimation. If final.control=NULL, only start estimates are computed and a list with fewer elements which has not the class "ts glm" is returned. Possible list elements of this argument are:

- constrained named list whose elements are passed to function constrOptim with possible elements mu, outer.iterations and outer.eps (see constrOptim for details). If constrained=NULL, an unconstrained optimisation is made with function optim. Note that this is likely to result in a fitted model which is non-stationary, which might cause further problems.
- optim.method character which is passed to functions constrOptim or optim as argument method.
- optim.control named list which is passed to function constrOptim or optim as the argument control. Must not contain the list element fnscale.

inter.control named list determining how to maximise the log-likelihood function in a first step. This intermediate optimisation will start from the start estimation and be followed by the final optimisation, which will in turn start from the intermediate optimisation result. This intermediate optimisation is intended to use a very quick but imprecise optimisation algorithm. Possible elements are the same as for final.control. The default is inter.control=NULL, which skips this intermediate optimisation step.

Details

The INGARCH model (argument link="identity") used here follows the definition

\[ Z_t | F_{t-1} \sim \text{Poi}(\nu_t) \quad \text{or} \quad Z_t | F_{t-1} \sim \text{NegBin}(\nu_t, \phi), \]

where \( F_{t-1} \) denotes the history of the process up to time \( t-1 \), Pois and NegBin is the Poisson respectively the negative binomial distribution with the parametrisation as specified below. For the model with covariates having an internal effect (the default) the linear predictor of the INGARCH model (which is in that case identical to the conditional mean) is given by

\[ \nu_t = \beta_0 + \beta_1 Z_{t-i_1} + \ldots + \beta_p Z_{t-i_p} + \alpha_1 \nu_{t-j_1} + \ldots + \alpha_q \nu_{t-j_q} + \eta_1 X_{t,1} + \ldots + \eta_r X_{t,r}. \]

The log-linear model (argument link="log") used here follows the definition

\[ Z_t | F_{t-1} \sim \text{Poi}(\lambda_t) \quad \text{or} \quad Z_t | F_{t-1} \sim \text{NegBin}(\lambda_t, \phi), \]

with \( \lambda_t = \exp(\nu_t) \) and \( F_{t-1} \) as above. For the model with covariates having an internal effect (the default) the linear predictor \( \nu_t = \log(\lambda_t) \) of the log-linear model is given by

\[ \nu_t = \beta_0 + \beta_1 \log(Z_{t-i_1} + 1) + \ldots + \beta_p \log(Z_{t-i_p} + 1) + \alpha_1 \nu_{t-j_1} + \ldots + \alpha_q \nu_{t-j_q} + \eta_1 X_{t,1} + \ldots + \eta_r X_{t,r}. \]
Note that because of the logarithmic link function the effect of single summands in the linear predictor on the conditional mean is multiplicative and hence the parameters play a different role than in the INGARCH model, although they are denoted by the same letters.

The Poisson distribution is parametrised by the mean $\lambda$ according to the definition in Poisson. The negative binomial distribution is parametrised by the mean $\mu$ with an additional dispersion parameter $size$ according to the definition in NegBinomial. In the notation above its mean parameter $\mu$ is $\nu_t$ and its dispersion parameter $size$ is $\phi$.

This function allows to include covariates in two different ways. A covariate can have a so-called internal effect as defined above, where its effect propagates via the regression on past values of the linear predictor and on past observations. Alternatively, it can have a so-called external effect, where its effect does not directly propagates via the feedback on past values of the linear predictor, but only via past observations. For external effects of the covariates, the linear predictor for the model with identity link is given by

$$\nu_t = \mu_t + \eta_1 X_{t,1} + \ldots + \eta_r X_{t,r},$$

$$\mu_t = \beta_0 + \beta_1 Z_{t-i_1} + \ldots + \beta_p Z_{t-i_p} + \alpha_1 \mu_t - j_1 + \ldots + \alpha_q \mu_t - j_q,$$

and analogously for the model with logarithmic link by

$$\nu_t = \mu_t + \eta_1 X_{t,1} + \ldots + \eta_r X_{t,r},$$

$$\mu_t = \beta_0 + \beta_1 \log(Z_{t-i_1} + 1) + \ldots + \beta_p \log(Z_{t-i_p} + 1) + \alpha_1 \mu_t - j_1 + \ldots + \alpha_q \mu_t - j_q.$$

This is described in more detail by Liboschik et al. (2014) for the case of deterministic covariates for modelling interventions. It is also possible to model a combination of external and internal covariates, which can be defined straightforwardly by adding each covariate either to the linear predictor $\nu_t$ itself (for an internal effect) or to $\mu_t$ defined above (for an external effect).

**Value**

An object of class "tsglm", which is a list with at least the following elements:

- **coefficients**: a named vector of the maximum likelihood estimated coefficients, which can be extracted by the `coef` method.
- **start**: a named vector of the start estimation for the coefficients.
- **residuals**: a vector of residuals, which can be extracted by the `residuals` method.
- **fitted.values**: the fitted values, which can be extracted by the `fitted` method.
- **linear.predictors**: the linear fit on link scale.
- **response**: a vector of the response values (this is usually the original time series but possibly without the first few observations used for initialization if argument `init.drop=TRUE`).
- **logLik**: the log-likelihood of the fitted model, which can be extracted by the `logLik` method. This is the complete log-likelihood including all constant terms. It is based on $n_eff$ observations (see below).
- **score**: the score vector at the maximum likelihood estimation.
- **info.matrix**: the information matrix at the maximum likelihood estimation assuming a Poisson distribution.
info.matrix_corrected
the information matrix at the maximum likelihood estimation assuming the distribution specified in distr.
call
the matched call.
n_obs
the number of observations.
n_eff
the effective number of observations used for maximum likelihood estimation (might be lower than n_obs if argument init.droop=TRUE).
ts
the original time series.
model
the model specification.
xreg
the given covariates.
distr
a character giving the fitted conditional distribution.
distrcoefs
a named vector of the estimated additional coefficients specifying the conditional distribution. Is NULL in case of a Poisson distribution.
sigmasq
the estimated overdispersion coefficient. Is zero in case of a Poisson distribution.

The functions ingarch.fit and loglin.fit have the same output except the elements distr, distrcoefs and sigmasq. In addition, they return the following list elements:

inter
some details on the intermediate estimation of the coefficients as returned by constrOptim or optim.

final
some details on the final estimation of the coefficients as returned by constrOptim or optim.

durations
named vector of the durations of the model fit (in seconds).

outerscoreprod
array of outer products of score vectors at each time point.

Author(s)
Tobias Liboschik, Philipp Probst, Konstantinos Fokianos and Roland Fried

References


**See Also**

S3 methods `print`, `summary`, `residuals`, `plot`, `fitted`, `coef`, `predict`, `logLik`, `vcov`, `AIC` and `BIC` for the class "tsglm". The S3 method `se` computes the standard errors of the parameter estimates. Additionally, there are the S3 methods `pit`, `marcal` and `scoring` for predictive model assessment.

S3 methods `interv_test`, `interv_detect` and `interv_multiple` for tests and detection procedures for intervention effects. `tsglm.sim` for simulation from GLM-type model for time series of counts. `ingarch.mean`, `ingarch.var` and `ingarch.acf` for calculation of analytical mean, variance and autocorrelation function of an INGARCH model (i.e. with identity link) without covariates.

Example time series of counts are `campy`, `ecoli`, `ehec`, `influenza`, `measles` in this package, `polio` in package `gamlss.data`.

**Examples**

```r
### Campylobacter infections in Canada (see help("campy"))
interventions <- interv_c covariate(n=length(campy), tau=c(84, 100),
  delta=c(1, 0)) #detected by Fokianos and Fried (2010, 2012)
# Linear link function with Negative Binomial distribution:
campyfit <- ts glm(campy, model=list(past_obs=1, past_mean=13),
  xreg=interventions, dist="nbinom")
campyfit
plot(campyfit)

### Road casualties in Great Britain (see help("Seatbelts"))
timeseries <- Seat belts[, "VanKilled"]
regressors <- cbind(PetrolPrice=Seat belts[, c("PetrolPrice")],
  linearTrend=seq(along=timeseries)/12)
# Logarithmic link function with Poisson distribution:
seatbeltsfit <- ts glm(ts=timeseries, link="log",
  model=list(past_obs=c(1, 12)), xreg=regressors, distr="pois")
summary(seatbeltsfit)
```

---

**tsglm.sim**

*Simulate a Time Series Following a Generalised Linear Model*

**Description**

Generates a simulated time series from a GLM-type model for time series of counts (see `tsglm` for details).
Usage

tsglm.sim(n, param = list(intercept = 1, past_obs = NULL, past_mean = NULL, xreg = NULL), model = list(past_obs = NULL, past_mean = NULL, external = FALSE), xreg = NULL, link = c("identity", "log"), distr = c("poisson", "nbinom"), distrcoefs, fit, n_start = 50)

Arguments

n integer value giving the number of observations to be simulated.
param a named list giving the parameters for the linear predictor of the model, which has the following elements:
intercept numeric positive value for the intercept $\beta_0$.
past_obs numeric non-negative vector containing the coefficients $\beta_1, \ldots, \beta_p$ for regression on previous observations (see Details).
past_mean numeric non-negative vector containing the coefficients $\alpha_1, \ldots, \alpha_q$ for regression on previous conditional means (see Details).
xreg numeric non-negative vector specifying the size $\nu_1, \ldots, \nu_r$ of each intervention
model a named list specifying the model for the linear predictor, which has the elements past_obs, past_mean and external (see function tsglm for details). This model specification must be in accordance to the parameters given in argument param.
xreg matrix with covariates in the columns (see tsglm for details). Its number of rows must be equal to the number of observations which should be simulated.
link character giving the link function. Default is "identity", simulating from a so-called INGARCH model. Another possible choice is "log", simulating from a log-linear model.
distr character giving the conditional distribution. Default is "poisson", i.e. a Poisson distribution.
distrcoefs numeric vector of additional coefficients specifying the conditional distribution. For distr="poisson" no additional parameters need to be provided. For distr="nbinom" the additional parameter size needs to be specified (i.e. by distrcoefs=2), see tsglm for details.
fit an object of class "tsglm". Usually the result of a call to tsglm. If argument fit is not missing, the specification of the linear predictor, the link function and the estimated parameters from this argument are used instead of those in arguments model, link and param. The length of the simulated time series is only taken from argument fit, if no argument n is provided. The same holds for arguments xreg, distr and distrcoefs, which are also prefered over the respective information provided in argument fit if both are provided.
n_start number of observations used as a burn-in.

Details

The definition of the model used here is like in function tsglm.
Note that during the burn-in period covariates are set to zero.
If a previous model fit is given in argument fit and the length of the burn-in period n_start is set to zero, then the a continuation of the original time series is simulated.

Value
A list with the following components:

- ts
  an object of class "ts" with the simulated time series.
- linear.predictors
  an object of class "ts" with the simulated conditional means $\kappa_t$ for all $t = 1, \ldots, n$.
- xreg.effects
  an object of class "ts" with the cumulated effect of the covariates $\eta_1 X_{t,1} + \ldots + \eta_r X_{t,r}$ for all $t = 1, \ldots, n$.

Author(s)
Tobias Liboschik and Philipp Probst

See Also
tsglm for fitting a GLM for time series of counts.

Examples
#Simulate from an INGARCH model with two interventions:
interventions <- interv_covariate(n=200, tau=c(50, 150), delta=c(1, 0.8))
model <- list(past_obs=1, past_mean=c(1, 7), external=FALSE)
param <- list(intercept=2, past_obs=0.3, past_mean=c(0.2, 0.1), xreg=c(3, 10))
tsglm.sim(n=200, param=param, model=model, xreg=interventions, link="identity",
          distr="nbinom", distrcoef=c(size=1))
Index

*Topic Data
  campy, 3
ecoli, 4
ehec, 5
influenza, 5
measles, 18
*Topic Inference
  se.tsglm, 28
  summary.tsglm, 29
tcount-package, 2
*Topic Intervention detection
  interv_covariate, 7
  interv_detect.tsglm, 9
  interv_multiple.tsglm, 11
  interv_test.tsglm, 14
  plot.interv_detect, 20
  plot.interv_multiple, 21
tcount-package, 2
*Topic Model assessment
  marcal.tsglm, 17
  pit.tsglm, 19
  plot.tsglm, 22
  residuals.tsglm, 25
  scoring.tsglm, 26
  summary.tsglm, 29
tcount-package, 2
*Topic Prediction
  predict.tsglm, 23
  tcount-package, 2
*Topic Simulation
  tcount-package, 2
tsglm.sim, 37

AIC, 37
arima, 33

BIC, 37

campy, 3, 4–6, 19, 37
chol, 17

chol2inv, 17
coeff, 35, 37
constrOptim, 34, 36
cpgram, 22
devAskNewPage, 22
ecoli, 3, 4, 5, 6, 19, 37
ehec, 3, 4, 5, 6, 19, 37
fitted, 35, 37
influenza, 3–5, 5, 19, 37
ingarch.acf, 37
ingarch.acf (ingarch.analytical), 6
ingarch.analytical, 6
ingarch.fit(tsglm), 31
ingarch.mean, 37
ingarch.mean (ingarch.analytical), 6
ingarch.var, 37
ingarch.var (ingarch.analytical), 6
interv_covariate, 7, 10, 11, 13, 15, 16
interv_detect, 8, 12, 13, 16, 20, 21, 37
interv_detect (interv.detect.tsglm), 9
interv_detect.tsglm, 9, 20
interv_multiple, 8, 11, 16, 21, 37
interv_multiple (interv_multiple.tsglm), 11
interv_multiple.tsglm, 11, 21
interv_test, 8, 10–13, 37
interv_test (interv.test.tsglm), 14
interv_test.tsglm, 14
invertinfo, 16

logLik, 35, 37
logLik.tsglm(tsglm), 31
loglin.fit(tsglm), 31

makeCluster, 10, 12, 28
marcal, 20, 22, 27, 37
marcal (marcal.tsglm), 17
INDEX

marcal.tsglm, 17
measles, 3–6, 18, 37

NegBinomial, 35

optim, 34, 36

pit, 18, 22, 27, 37
pit (pit.tsglm), 19
pit.tsglm, 19
plot, 6, 11, 13, 17, 19–21, 37
plot.interv_detect, 20
plot.interv_multiple, 21
plot.tsglm, 22
Poisson, 35
polio, 3–6, 19, 37
predict, 37
predict.tsglm, 23
print, 11, 13, 16, 30, 37
print.summary.tsglm(summary.tsglm), 29
print.tsglm (tsglm), 31

residuals, 35, 37
residuals.tsglm, 25, 30

scoring, 18, 20, 37
scoring (scoring.tsglm), 26
scoring.tsglm, 26
se, 30, 37
se (se.tsglm), 28
se.tsglm, 28, 30
setDefaultCluster, 10, 12, 28
solve, 16
summary, 37
summary.tsglm, 29

tacvfARMA, 7
tscount (tscount-package), 2
tscount-package, 2
tsglm, 2, 7–16, 18, 20–25, 27–30, 31, 37–39
tsglm.sim, 7, 37, 37

vcov, 37
vcov.tsglm (tsglm), 31