Package ‘ZIM’

October 12, 2022

Title Zero-Inflated Models (ZIM) for Count Time Series with Excess Zeros

Version 1.1.0

Description Analyze count time series with excess zeros.
Two types of statistical models are supported: Markov regression by Yang et al. (2013) <doi:10.1016/j.stamet.2013.02.001> and state-space models by Yang et al. (2015) <doi:10.1177/1471082X14535530>. They are also known as observation-driven and parameter-driven models respectively in the time series literature. The functions used for Markov regression or observation-driven models can also be used to fit ordinary regression models with independent data under the zero-inflated Poisson (ZIP) or zero-inflated negative binomial (ZINB) assumption. Besides, the package contains some miscellaneous functions to compute density, distribution, quantile, and generate random numbers from ZIP and ZINB distributions.

License GPL-3

Encoding UTF-8

RoxygenNote 6.1.0

Imports MASS

Suggests pscl, TSA

URL https://github.com/biostatstudio/ZIM

BugReports https://github.com/biostatstudio/ZIM/issues

NeedsCompilation no

Author Ming Yang [aut, cre],
Gideon Zamba [aut],
Joseph Cavanaugh [aut]

Maintainer Ming Yang <mingyang@biostatstudio.com>

Repository CRAN

Date/Publication 2018-08-28 13:04:25 UTC

R topics documented:

`ZIM-package` *Zero-Inflated Models for Count Time Series with Excess Zeros*

Description

Fits observation-driven and parameter-driven models for count time series with excess zeros.

Details

The package `ZIM` contains functions to fit statistical models for count time series with excess zeros (Yang et al., 2013, 2015). The main function for fitting observation-driven models is `zim`, and the main function for fitting parameter-driven models is `dzim`.

Note

The observation-driven models for zero-inflated count time series can also be fit using the function `zeroinfl` from the `pscl` package (Zeileis et al., 2008). Fitting parameter-driven models is based on sequential Monte Carlo (SMC) methods, which are computer intensive and could take several hours to estimate the model parameters.

References

Yang, M., Cavanaugh, J. E., and Zamba, G. K. D. (2015). State-space models for count time series with excess zeros. *Statistical Modelling*, **15**:70-90

Yang, M., Zamba, G. K. D., and Cavanaugh, J. E. (2013). Markov regression models for count time series with excess zeros: A partial likelihood approach. *Statistical Methodology*, **14**:26-38.

Zeileis, A., Kleiber, C., and Jackman, S. (2008). Regression models for count data in `R`. *Journal of Statistical Software*, **27**(8).

`bshift` *Backshift Operator*

Description

Apply the backshift operator or lag operator to a time series objective.

Usage

```
bshift(x, k = 1)
```

Arguments

`x`	univariate or multivariate time series.
`k`	number of lags.

See Also

`lag`, `zlag`

Examples

```
x <- arima.sim(model = list(ar = 0.8, sd = 0.5), n = 120)
bshift(x, k = 12)
```

dzim *Fitting Dynamic Zero-Inflated Models*

Description

`dzim` is used to fit dynamic zero-inflated models.

Usage

```
dzim(formula, data, subset, na.action, weights = 1, offset = 0,
  control = dzim.control(...), ...)
```

Arguments

`formula`	an objective of class "`formula`".
`data`	an optional dataframe, list or environment containing the variables in the model.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s.
`weights`	an optional vector of 'prior weights' to be used in the fitting process.
`offset`	this can be used to specify a priori known component to be included in the linear predictor during fitting.
`control`	control arguments from `dzim.control`
`...`	additional arguments

See Also

`dzim.fit`, `dzim.filter`, `dzim.smooth`, `dzim.control`, `dzim.sim`, `dzim.plot`

dzim.control *Auxiliary for Controlling DZIM Fitting*

Description

Auxiliary function for `dzim` fitting. Typically only used internally by `dzim.fit`, but may be used to construct a control argument for either function.

Usage

```
dzim.control(dist = c("poisson", "nb", "zip", "zinb"), trace = FALSE,
  start = NULL, order = 1, mu0 = rep(0, order), Sigma0 = diag(1,
  order), N = 1000, R = 1000, niter = 500)
```

Arguments

`dist`	count model family
`trace`	logical; if TRUE, display iteration history.
`start`	initial parameter values.
`order`	autoregressive order.
`mu0`	mean vector for initial state.
`Sigma0`	covariance matrix for initial state.
`N`	number of particiles in particle filtering.
`R`	number of replications in particle smoothing.
`niter`	number of iterations.

Note

The default values of `N`, `R`, and `niter` are chosen based on our experience. In some cases, `N` = 500, `R` = 500, and `niter` = 200 might be sufficient. The `dzim.plot` function should always be used for convergence diagnostics.

See Also

`dzim`, `dzim.fit`, `dzim.filter`, `dzim.smooth`, `dzim.sim`, `dzim.plot`

`dzim.filter` *Particle Filtering for DZIM*

Description

Function to implement the particle filtering method proposed by Gordsill et al. (1993).

Usage

```
dzim.filter(y, X, w, para, control)
```

Arguments

`y`	response variable.
`X`	design matrix.
`w`	`log(w)` is used as an offset variable in the linear predictor.
`para`	model parameters.
`control`	control arguments.

References

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEEE Proceedings*, **140**, 107-113.

See Also

dzim, dzim.fit, dzim.smooth, dzim.control, dzim.sim, dzim.plot

---

dzim.fit *Fitter Function for Dynamic Zero-Inflated Models*

Description

dzim.fit is the basic computing engine called by dzim used to fit dynamic zero-inflated models. This should usually *not* be used directly unless by experienced users.

Usage

```
dzim.fit(y, X, offset = rep(0, n), control = dzim.control(...), ...)
```

Arguments

y	response variable.
X	design matrix.
offset	offset variable.
control	control arguments.
...	additional arguments.

See Also

dzim, dzim.control, dzim.filter, dzim.smooth, dzim.sim, dzim.plot

---

dzim.plot *Trace Plots from DZIM*

Description

Function to display trace plots from a dynamic zero-inflated model.

Usage

```
dzim.plot(object, k.inv = FALSE, sigma.sq = FALSE, ...)
```

Arguments

object	objective from dzim or dzim.fit.
k.inv	logical; indicating whether an inverse transformation is needed for the dispersion parameter.
sigma.sq	logical; indicating whether a square transformation is needed for the standard deviation parameter.
...	additional arguments.

dzim.sim *Simulate Data from DZIM*

Description

Simulate data from a dynamic zero-inflated model.

Usage

```
dzim.sim(X, w, omega, k, beta, phi, sigma, mu0, Sigma0)
```

Arguments

X	design matrix.
w	log(w) is used as an offset variable in the linear predictor.
omega	zero-inflation parameter.
k	dispersion parameter.
beta	regression coefficients.
phi	autoregressive coefficients.
sigma	standard deviation.
mu0	mean vector of initial state.
Sigma0	covariance matrix of initial state.

See Also

dzim, dzim.fit, dzim.filter, dzim.smooth, dzim.control, dzim.plot

dzim.smooth *Particle Smoothing for DZIM*

Description

Function to implement the particle smoothing method proposed by Gordsill et al. (2004).

Usage

```
dzim.smooth(y, X, w, para, control)
```

Arguments

y	response variable.
X	design matrix.
w	log(w) is used as an offset variable in the linear predictor.
para	model parameters.
control	control arguments.

References

Gordsill, S. J., Doucet, A., and West, M. (2004). Monte Carlo smoothing for nonlinear time series. *Journal of the American Statistical Association*, **99**, 156-168.

See Also

`dzim`, `dzim.fit`, `dzim.filter`, `dzim.control`, `dzim.sim`, `dzim.plot`

---

injury — *Example: Injury Series from Occupational Health*

Description

Monthly number of injuries in hospitals from July 1988 to October 1995.

Source

Numbers from Figure 1 of Yau et al. (2004).

References

Yau, K. K. W., Lee, A. H. and Carrivick, P. J. W. (2004). Modeling zero-inflated count series with application to occupational health. *Computer Methods and Programs in Biomedicine*, **74**, 47-52.

Examples

```
data(injury)
plot(injury, type = "o", pch = 20, xaxt = "n", yaxt = "n", ylab = "Injury Count")
  axis(side = 1, at = seq(1, 96, 8))
  axis(side = 2, at = 0:9)
  abline(v = 57, lty = 2)
  mtext("Pre-intervention",  line = 1, at = 25, cex = 1.5)
  mtext("Post-intervention", line = 1, at = 80, cex = 1.5)
```

---

pvalue — *Function to Compute P-value.*

Description

Function to compute p-value based on a t-statistic.

Usage

```
pvalue(t, df = Inf, alternative = c("two.sided", "less", "greater"))
```

Arguments

`t`	t-statistic.
`df`	degree of freedoms.
`alternative`	type of alternatives.

Examples

```
pvalue(1.96, alternative = "greater")
```

---

syph *Example: Syphilis Series*

Description

Weekly number of syphilis cases in the United States from 2007 to 2010.

Format

A data frame with 209 observations on the following 69 variables.

`year`	Year
`week`	Week
`a1`	**United States**
`a2`	**New England**
`a3`	Connecticut
`a4`	Maine
`a5`	Massachusetts
`a6`	New Hampshire
`a7`	Rhode Island
`a8`	Vermont
`a9`	**Mid. Atlantic**
`a10`	New Jersey
`a11`	New York (Upstate)
`a12`	New York City
`a13`	Pennsylvania
`a14`	**E.N. Central**
`a15`	Illinois
`a16`	Indiana
`a17`	Michigan
`a18`	Ohio
`a19`	Wisconsin
`a20`	**W.N. Central**
`a21`	Iowa
`a22`	Kansas
`a23`	Minnesota
`a24`	Missouri

a25	Nebraska
a26	North Dakota
a27	South Dakota
a28	**S. Atlantic**
a29	Delaware
a30	District of Columbia
a31	Florida
a32	Georgia
a33	Maryland
a34	North Carolina
a35	South Carolina
a36	Virginia
a37	West Virginia
a38	**E.S. Central**
a39	Alabama
a40	Kentucky
a41	Mississippi
a42	Tennessee
a43	**W.S. Central**
a44	Arkansas
a45	Louisana
a46	Oklahoma
a47	Texas
a48	**Moutain**
a49	Arizona
a50	Colorado
a51	Idaho
a52	Montana
a53	Nevada
a54	New Mexico
a55	Utah
a56	Wyoming
a57	**Pacific**
a58	Alaska
a59	California
a60	Hawaii
a61	Oregon
a62	Washington
a63	American Samoa
a64	C.N.M.I.
a65	Guam
a66	Peurto Rico
a67	U.S. Virgin Islands

Note

C.N.M.I.: Commonwealth of Northern Mariana Islands.

Source

CDC Morbidity and Mortality Weekly Report (http://www.cdc.gov/MMWR/).

Examples

```
data(syph)
plot(ts(syph$a33), main = "Maryland")
```

---

zim *Fitting Zero-Inflated Models*

Description

`zim` is used to fit zero-inflated models.

Usage

```
zim(formula, data, subset, na.action, weights = 1, offset = 0,
  control = zim.control(...), ...)
```

Arguments

`formula`	an objective of class "`formula`".
`data`	an optional dataframe, list or environment containing the variables in the model.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s.
`weights`	an optional vector of 'prior weights' to be used in the fitting process.
`offset`	this can be used to specify a priori known component to be included in the linear predictor during fitting.
`control`	control arguments.
`...`	additional arguments.

Note

`zim` is very similar to `zeroinfl` from the `pscl` package. Both functions can be used to fit observation-driven models for zero-inflated time series.

See Also

`zim.fit`, `zim.control`

zim.control — *Auxiliary for Controlling ZIM Fitting*

Description

Auxiliary function for zim fitting. Typically only used internally by zim.fit, but may be used to construct a control argument for either function.

Usage

```
zim.control(dist = c("zip", "zinb"), method = c("EM-NR", "EM-FS"),
  type = c("solve", "ginv"), robust = FALSE, trace = FALSE,
  start = NULL, minit = 10, maxit = 10000, epsilon = 1e-08)
```

Arguments

dist	count model family.
method	algorithm for parameter estimation.
type	type of matrix inverse.
robust	logical; if TRUE, robust standard errors will be calculated.
trace	logical; if TRUE, display iteration history.
start	initial parameter values.
minit	minimum number of iterations.
maxit	maximum number of iterations.
epsilon	positive convergence tolerance.

See Also

zim, zim.fit

zim.fit — *Fitter Function for Zero-Inflated Models*

Description

zim.fit is the basic computing engine called by zim used to fit zero-inflated models. This should usually *not* be used directly unless by experienced users.

Usage

```
zim.fit(y, X, Z, weights = rep(1, nobs), offset = rep(0, nobs),
  control = zim.control(...), ...)
```

Arguments

`y`	response variable.
`X`	design matrix for log-linear part.
`Z`	design matrix for logistic part.
`weights`	an optional vector of 'prior weights' to be used in the fitting process.
`offset`	offset variable
`control`	control arguments from `zim.control`.
`...`	additional argumetns.

See Also

`zim`, `zim.control`

ZINB — *The Zero-Inflated Negative Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the zero-inflated negative binomial (ZINB) distribution with parameters `k`, `lambda`, and `omega`.

Usage

```
dzinb(x, k, lambda, omega, log = FALSE)
pzinb(q, k, lambda, omega, lower.tail = TRUE, log.p = FALSE)
qzinb(p, k, lambda, omega, lower.tail = TRUE, log.p = FALSE)
rzinb(n, k, lambda, omega)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of random values to return.
`k`	dispersion parameter.
`lambda`	vector of (non-negative) means.
`omega`	zero-inflation parameter.
`log, log.p`	logical; if TRUE, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Value

`dzinb` gives the density, `pzinb` gives the distribution function, `qzinb` gives the quantile function, and `rzinb` generates random deviates.

See Also

dzip, pzip, qzip, and rzip for the zero-inflated Poisson (ZIP) distribution.

Examples

```
dzinb(x = 0:10, k = 1, lambda = 1, omega = 0.5)
pzinb(q = c(1, 5, 9), k = 1, lambda = 1, omega = 0.5)
qzinb(p = c(0.25, 0.50, 0.75), k = 1, lambda = 1, omega = 0.5)
set.seed(123)
rzinb(n = 100, k = 1, lambda = 1, omega = 0.5)
```

ZIP *The Zero-Inflated Poisson Distribution*

Description

Density, distribution function, quantile function and random generation for the zero-inflated Poisson (ZIP) distribution with parameters `lambda` and `omega`.

Usage

```
dzip(x, lambda, omega, log = FALSE)
pzip(q, lambda, omega, lower.tail = TRUE, log.p = FALSE)
qzip(p, lambda, omega, lower.tail = TRUE, log.p = FALSE)
rzip(n, lambda, omega)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of random values to return.
`lambda`	vector of (non-negative) means.
`omega`	zero-inflation parameter.
`log, log.p`	logical; if TRUE, probabilities `p` are given as `log(p)`.
`lower.tail`	logical; if TRUE (default), probabilities are `P[X <= x]`, otherwise, `P[X > x]`.

Value

`dzip` gives the density, `pzip` gives the distribution function, `qzip` gives the quantile function, and `rzip` generates random deviates.

See Also

dzinb, pzinb, qzinb, and rzinb for the zero-inflated negative binomial (ZINB) distribution.

Examples

```
dzip(x = 0:10, lambda = 1, omega = 0.5)
pzip(q = c(1, 5, 9), lambda = 1, omega = 0.5)
qzip(p = c(0.25, 0.50, 0.75), lambda = 1, omega = 0.5)
set.seed(123)
rzip(n = 100, lambda = 1, omega = 0.5)
```

Index