Package 'bizicount'

May 1, 2024

Title Bivariate Zero-Inflated Count Models Using Copulas

Version 1.3.3

Description Maximum likelihood estimation of copula-based zero-inflated
(and non-inflated) Poisson and negative binomial count models, based on the
article <doi:10.18637/jss.v109.i01>. Supports Frank and Gaussian copulas.
Allows for mixed margins (e.g., one margin Poisson, the other zero-inflated
negative binomial), and several marginal link functions. Built-in methods for
publication-quality tables using 'texreg', post-estimation diagnostics using
'DHARMa', and testing for marginal zero-modification via <doi:10.1177/0962280217749991>.
For information on copula regression for count data, see Genest and Nešlehová (2007)
<doi:10.1017/S0515036100014963> as well as Nikoloulopoulos (2013) <doi:10.1007/978-3-
642-35407-6_11>.
For information on zero-inflated count regression generally, see Lambert (1992)
<https://www.jstor.org/stable/1269547?origin=crossref>. The author acknowledges
support by NSF DMS-1925119 and DMS-212324.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/jmniehaus/bizicount

BugReports https://github.com/jmniehaus/bizicount/issues

RoxygenNote 7.3.1

Imports pbivnorm (>= 0.6.0), Formula (>= 1.2-4), rlang (>= 0.4.7),
MASS (>= 7.3-54), numDeriv (>= 2016.8-1.1), methods, stats,
utils, DHARMa (>= 0.3.4), texreg (>= 1.37.5)

Depends R (>= 4.1.0)

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

NeedsCompilation no

Author John Niehaus [aut, cre]

Maintainer John Niehaus <jniehaus2257@gmail.com>

Repository CRAN

Date/Publication 2024-05-01 14:02:10 UTC

R topics documented:

---

bizicount-package   *bizicount: Copula-Based Bivariate Zero-Inflated Count Regression Models*

Description

The package provides regression functions for copula-based bivariate count models based on the paper [doi:10.18637/jss.v109.i01](https://doi.org/10.18637/jss.v109.i01), with and without zero-inflation, as well as regression functions for univariate zero-inflated count models. Generic methods from the [texreg-package](#) and [DHARMa](#) are extended to support this package, namely for the purposes of producing professional tables and carrying out post-estimation diagnostics. A generic for He et al. (2019)'s test for zero-modification is provided, with methods for both bizicount and glm-class objects.

Bivariate Functions

- [bizicount](#) – The primary function of this package. Carries out copula-based bivariate count regression via maximum likelihood using numerical optimization. Supports both zero-inflated and non-inflated distributions.

- [extract.bizicount](#) – Method for the texreg package's [extract](#) generic. Creates a list of texreg objects, one for each margin, for use with that package's other functions.

- [make_DHARMa](#) – Creates a list of DHARMa objects, one for each margin, for [bizicount](#) models. A wrapper around [createDHARMa](#).

- [simulate.bizicount](#) – Method that simulates observations using the fitted model's parameters, primarily for use with DHARMa.

- [zi_test](#) – Method for testing for marginal zero-modification using the esimated parameters from the model. This test is preferable to the Vuong, Wald, Score, and LR tests. See He et al. (2019).

Univariate Functions

- `zic.reg` – Univariate zero-inflated count regression models via maximum likelihood.
- `extract.zicreg` – Method for the texreg package's `extract` generic. Creates a texreg object that interfaces with that package's methods.
- `simulate.zicreg` – Method for simulating from the fitted model. Results are generally used for creating DHARMa objects.
  #'
- `zi_test` – Method for testing for univariate zero-modification using the esimated parameters from the model. This test is preferable to the Vuong, Wald, Score, and LR tests. See He et al. (2019).

Author(s)

John Niehaus

References

---

bizicount	*Bizicount: Maximum likelihood estimation of copula-based bivariate zero-inflated (and non-inflated) count models*

Description

The main bivariate regression function of the `bizicount-package` Estimates copula-based bivariate zero-inflated (and non-inflated) count models via maximum likelihood. Supports the Frank and Gaussian copulas, as well as zero-inflated Poisson and negative binomial margins (and their non-inflated counterparts). It's class has associated `simulate` methods for post-estimation diagnostics using the `DHARMa` package, as well as an `extract` method for printing professional tables using `texreg`, and a test for zero-modification using zi_test. See the 'See Also' section for links to these methods.

Usage

```
bizicount(
  fmla1,
  fmla2,
  data,
  cop = "gaus",
  margins = c("pois", "pois"),
  link.ct = c("log", "log"),
  link.zi = c("logit", "logit"),
  scaling = "none",
  starts = NULL,
  keep = TRUE,
```

```
  subset,
  na.action,
  weights,
  frech.min = 1e-07,
  pmf.min = 1e-07,
  ...
)
```

Arguments

fmla1, fmla2	[formula](#)s for the first margin and second margins, respectively. If non-inflated, of the form y ~ x_1 + x_2 + ... + x_k; if inflated, of the form y ~ x1 + x2 + ... + x_k| z1 + z2 + ... + z_p, where y is the outcome for the first margin, x are covariates for count parameters, and z are covariates for zero-inflated parameters in each margin. All covariates can be the same.
data	A [data.frame](#) containing the response variables, covariates, and offsets for the model. If NULL, these quantities are searched for in the parent environment.
cop	Character string specifying the copula to be used. One of c("gaus", "frank"). Partial matching supported.
margins	Length 2 character vector specifying the marginal distributions for each outcome. Each of the two elements must be one of c("pois", "nbinom", "zip", "zinb"), and must be consistent with its corresponding formula (i.e., zero-inflated margins with zero-inflated formulas).
link.ct	Length 2 character string specifying the link function used for the count portion of each margin. One of c("log", "identity", "sqrt").
link.zi	Length 2 character string specifying the link function used for the zero-inflation portion of each margin. One of c("logit", "probit", "cauchit", "log", "cloglog"). Ignored if corresponding margins entry is not zero-inflated.
scaling	Deprecated. It is recommended that users scale their covariates if they encounter convergence issues, which can be accomplished using the scale() function on their data before putting it into the bizicount() function.
starts	Numeric vector of starting values for parameter estimates. See 'Details' section regarding the correct order for the values in this vector. If NULL, starting values are obtained automatically by a univariate regression fit.
keep	Logical indicating whether to keep the model matrix in the returned model object. Defaults to TRUE, but can be set to FALSE to conserve memory. NOTE: Must be TRUE to use any post-estimation functions in this package, including zi_test.
subset	A vector indicating the subset of observations to use in estimation.
na.action	Deprecated.
weights	An optional numeric vector of weights for each observation.
frech.min	Lower boundary for Frechet-Hoeffding bounds on copula CDF. Used for computational purposes to prevent over/underflow in likelihood search. Must be in $[0, 1e-5]$, with 0 imposing the original FH bounds without computational consideration. See 'Details.'

pmf.min   Lower boundary on copula PMF evaluations. Used for computational purposes
                 to prevent over/underflow in likelihood search. Must be in $[0, 1e-5]$, with 0
                 imposing no bound. See 'Details.'

...   Additional arguments to be passed on to the quasi-newton fitting function, nlm.
                 See 'Details' for some parameters that may be useful to alter.

Details

- starts – Starting values should be organized as follows:
    1. count parameters for margin 1
    2. count parameters for margin 2
    3. zero-inflated parameters for margin 1 (if applicable),
    4. zero-inflated parameters for margin 2 (if applicable),
    5. inverse dispersion parameter for margin 1 (if applicable),
    6. inverse dispersion parameter for margin 2 (if applicable)

  Thus, in general count parameters should come first, followed by zero-inflation parameters, and finally inverse dispersion parameters.

- frech.min – Changing this argument should almost never be necessary. Frechet (1951) and Hoeffding (1940) showed that copula CDFs have bounds of the form $max\{u+v-1,0\} \le C(u,v) \le min\{u,v\}$, where u and v are uniform realizations derived from the probability integral transform. Due to numerical underflow, very small values of u and v can be rounded to zero. Particularly when evaluating the Gaussian copula CDF this is problematic, ultimately leading to infinite-valued likelihood evaluations. Therefore, we impose Frechet-Hoeffding bounds numerically as $max\{u+v-1, frech.min\} \le C(u,v) \le min\{u,v, 1-frech.min\}$. NOTE: Setting this to 0 imposes the original Frechet bounds mentioned above.

- pmf.min – Changing this argument should almost never be necessary. Observations can have likelihoods that are extremely close to 0. Numerically, these get rounded to 0 due to underflow. Then, taking logarithms results in an infinite likelihood. To avoid this, we bound PMF evaluations from below at pmf.min.

- ... – Sometimes it may be useful to alter nlm's default parameters. This can be done by simply passing those arguments into bizicount(). The two that seem to benefit the fitting process the most are stepmax and steptol. Readers are referred to the documentation on nlm for more details on these parameters. It can be useful to lower stepmax particularly when the Hessian is not negative definite at convergence, sometimes to a value between 0 and 1. It can also be beneficial to increase steptol.

Value

An S3 bizicount-class object, which is a list containing:

- coef – Coefficients of the model
- coef.nid – Coefficients without margin IDs
- coef.orig – Coefficients prior to transformations, for Gaussian dependence and negative binomial dispersion.
- coef.orig.nid – Coefficients prior to transforms, no margin IDs.
- se – Asymptotic normal-theory standard errors based on observed Fisher Information

- se.nid – Standard errors without margin IDs
- z – z-scores for parameter estimates
- z.nid – z-scores without margin IDs
- p – p-values for parameter estimates
- p.nid – p-values without margin IDs
- coefmats – A list containing coeficient matrices for each margin
- loglik – Scalar log-likelihood at convergence
- grad – Numerical gradient vector at convergence
- n.iter – Number of quasi-newton fitting iterations.
- covmat – Covariance matrix of parameter estimates based on observed Fisher Information
- aic – Model's Akaike information
- bic – Model's Bayesian information criterion
- nobs – Number of observations
- margins – Marginal distributions used in fitting
- link.zi, link.ct – Names of link functions used in fitting
- invlink.ct, invlink.zi – Inverse link functions used in fitting (the actual function, not their names)
- outcomes – Name of the response vector
- conv – Integer telling convergence status in nlm. See ?nlm.
- cop – The copula used in fitting
- starts – list of starting values used
- call – The model's call
- model – List containing model matrices, or NULL if keep = F.

Author(s)

John Niehaus

References

doi:10.18637/jss.v109.i01

Genest C, Nešlehová J (2007). "A primer on copulas for count data." ASTIN Bulletin: The Journal of the IAA, 37(2), 475–515.

Inouye DI, Yang E, Allen GI, Ravikumar P (2017). "A review of multivariate distributions for count data derived from the Poisson distribution." Wiley Interdisciplinary Reviews: Computational Statistics, 9(3).

Joe H (1997). Multivariate models and multivariate dependence concepts. CRC Press.

Nikoloulopoulos A (2013). "Copula-Based Models for Multivariate Discrete Response Data." In P Jaworski, F Durante, WK Härdle (eds.), Copulae in Mathematical and Quantitative Finance, chapter 11, pp. 231–250. Springer.

Nelsen RB (2007). An Introduction to Copulas. Springer Science & Business Media.

Trivedi P, Zimmer D (2017). "A note on identification of bivariate copulas for discrete countdata." Econometrics, 5(1), 10.

Trivedi PK, Zimmer DM (2007). Copula modeling: an introduction for practitioners. NowPublishers Inc.

See Also

[extract.bizicount](), [make_DHARMa](), [zi_test]()

Examples

```
### bizicount example

## SETUP
set.seed(123)
n = 300

# define a function to simulate from a gaussian copula
# first margin is zero-inflated negative binomial (zinb)
# second margin is zero-inflated poisson (zip)
# Note: marginal distributions are hard-coded in function, including
# inverse dispersion parameter for zinb.
gen = function(n,
               b1,
               b2,
               g1,
               g2,
               dep) {

    k1 = length(b1)
    k2 = length(b2)

    X1 = cbind(1, matrix(rbinom(n * (k1 - 1), 1, .5), ncol = k1 - 1))
    X2 = cbind(1, matrix(rexp(n * (k2 - 1), 3), ncol = k2 - 1))

    lam1 = exp(X1 %*% b1)
    lam2 = exp(X2 %*% b2)

    Z1 = cbind(1, matrix(runif(n * (k1 - 1), -1, 1), ncol = k1 - 1))
    Z2 = cbind(1, matrix(rnorm(n * (k2 - 1)), ncol = k2 - 1))

    psi1 = plogis(Z1 %*% g1)
    psi2 = plogis(Z2 %*% g2)

    norm_vars = MASS::mvrnorm(
        n,
        mu = c(0, 0),
        Sigma = matrix(c(1, dep, dep, 1), ncol =2)
        )

    U = pnorm(norm_vars)
```

```
        y1 =  qzinb(U[, 1],
                       mu = lam1,
                       psi = psi1,
                       size = .3)
        y2 =  qzip(U[, 2],
                       lambda = lam2,
                       psi = psi2)

        dat = data.frame(
            X1 = X1[, -1],
            X2 = X2[, -1],
            Z1 = Z1[, -1],
            Z2 = Z2[, -1],
            y1,
            y2,
            lam1,
            lam2,
            psi1,
            psi2
        )
        return(dat)
}


# define parameters
b1 = c(1, -2, 3)
b2 = c(-1, 3, 1)
g1 = c(2, -1.5, 2)
g2 = c(-1, -3.75, 1.25)
rho = .5


# generate data
dat = gen(n, b1, b2, g1, g2, rho)
f1 = y1 ~ X1.1 + X1.2 | Z1.1 + Z1.2
f2 = y2 ~ X2.1 + X2.2 | Z2.1 + Z2.2

## END SETUP

# estimate model

mod = bizicount(f1, f2, dat, cop = "g", margins = c("zinb", "zip"), keep = TRUE)

print(mod)
summary(mod)
```

---

bizicount-class   *The bizicount S4 Class*

Description

Note that `bizicount` objects are generally S3, and should use S3 syntax. This S4 class is defined only for compatability with `texreg`. However, the contents of `bizicount` objects is the same in both S3 and S4, so the descriptions below apply in both cases.

Slots

coef Coefficients of the model

coef.nid Coefficients without margin IDs

coef.orig Coefficients prior to transformations, for Gaussian dependence and negative binomial dispersion.

coef.orig.nid Coefficients prior to transforms, no margin IDs.

se Asymptotic standard errors based on observed Fisher Information

se.nid Standard errors without margin IDs

z z-scores for parameter estimates

z.nid z-scores without margin IDs

p p-values for parameter estimates

p.nid p-values without margin IDs

coefmats A list containing coefficient matrices for each margin

loglik Scalar log-likelihood at convergence

grad Gradient vector at convergence

n.iter Number of quasi-newton fitting iterations.

covmat Covariance matrix of parameter estimates based on observed Fisher Information

aic Model's Akaike information

bic Model's Bayesian information criterion

nobs Number of observations

margins Marginal distributions used in fitting

link.zi,link.ct Names of link functions used in fitting

invlink.ct,invlink.zi Inverse link functions used in fitting (the actual function, not their names)

outcomes Name of the response vector

conv Integer telling convergence status.

cop The copula used in fitting

starts list of starting values used

call The model's call

model List containing model matrices, or NULL if keep = F.

extract.bizicount *Texreg for bizicount objects*

Description

This is a method for the extract generic to be used with objects that are output from the bizicount function. The results can be used with any of the texreg-package generics.

Usage

```
## S3 method for class 'bizicount'
extract(model, CI = NULL, id = TRUE, ...)
```

Arguments

model	A bizicount-class model object (S3).
CI	The two-tailed confidence level, if confidence intervals are desired in the texreg object, otherwise NULL.
id	Logical indicating whether to prepend equation identifiers to coefficient names (ct_ for count parameters, zi_ for zero-inflated parameters)
...	Ignored.

Value

A texreg-class object, as produced by createTexreg, which can interface with all of that package's generics.

Note

Users can typically just call texreg directly on a bizicount-class object, instead of first extracting and then calling texreg.

Author(s)

John Niehaus

References

Leifeld, Philip (2013). texreg: Conversion of Statistical Model Output in R to LaTeX and HTML Tables. Journal of Statistical Software, 55(8), 1-24. URL http://dx.doi.org/10.18637/jss.v055.i08.

See Also

extract, createTexreg, bizicount

Examples

```
## SETUP
set.seed(123)
n = 500

# define a function to simulate from a gaussian copula
# first margin is zero-inflated negative binomial (zinb)
# second margin is zero-inflated poisson (zip)
# Note: marginal distributions are hard-coded in function, including
# inverse dispersion parameter for zinb.
gen = function(n,
               b1,
               b2,
               g1,
               g2,
               dep) {

    k1 = length(b1)
    k2 = length(b2)

    X1 = cbind(1, matrix(rbinom(n * (k1 - 1), 1, .5), ncol = k1 - 1))
    X2 = cbind(1, matrix(rexp(n * (k2 - 1), 3), ncol = k2 - 1))

    lam1 = exp(X1 %*% b1)
    lam2 = exp(X2 %*% b2)

    Z1 = cbind(1, matrix(runif(n * (k1 - 1), -1, 1), ncol = k1 - 1))
    Z2 = cbind(1, matrix(rnorm(n * (k2 - 1)), ncol = k2 - 1))

    psi1 = plogis(Z1 %*% g1)
    psi2 = plogis(Z2 %*% g2)

    norm_vars = MASS::mvrnorm(
        n,
        mu = c(0, 0),
        Sigma = matrix(c(1, dep, dep, 1), ncol =2)
    )

    U = pnorm(norm_vars)

    y1 =  qzinb(U[, 1],
                mu = lam1,
                psi = psi1,
                size = .3)
    y2 =  qzip(U[, 2],
               lambda = lam2,
               psi = psi2)

    dat = data.frame(
        X1 = X1[, -1],
        X2 = X2[, -1],
        Z1 = Z1[, -1],
```

```
        Z2 = Z2[, -1],
        y1,
        y2,
        lam1,
        lam2,
        psi1,
        psi2
    )
    return(dat)
}


# define parameters
b1 = c(1, -2, 3)
b2 = c(-1, 3, 1)
g1 = c(2, -1.5, 2)
g2 = c(-1, -3.75, 1.25)
rho = .5


# generate data
dat = gen(n, b1, b2, g1, g2, rho)
f1 = y1 ~ X1.1 + X1.2 | Z1.1 + Z1.2
f2 = y2 ~ X2.1 + X2.2 | Z2.1 + Z2.2

## END SETUP

# estimate model

mod = bizicount(f1, f2, dat, cop = "g", margins = c("zinb", "zip"), keep=TRUE)

# extract texreg objects, one with SEs, one with CIs
tr_obj_se = texreg::extract(mod)
tr_obj_ci = texreg::extract(mod, CI = .95)


# output to latex, single table.
# Note use of c(), because tr_obj_se, tr_obj_ci are lists.
texreg::texreg(c(tr_obj_se, tr_obj_ci))


# output as plaintext, two tables
texreg::screenreg(tr_obj_se)
texreg::screenreg(tr_obj_ci)
```

---

extract.zicreg   *Texreg for zicreg objects*

Description

This is a method for the [extract](#) generic to be used with objects that are output from the [zic.reg](#) function. The results can then interface with the [texreg-package](#), as shown in examples below.

Usage

```
## S3 method for class 'zicreg'
extract(model, CI = NULL, id = TRUE, ...)
```

Arguments

model	A zicreg model object, returned by [zic.reg](#).
CI	The two-tailed confidence level, if desired in the resulting [texreg](#) object.
id	Logical indicating whether to prepend equation identifiers to coefficient names (ct_ for count parameters, zi_ for zero-inflated parameters)
...	Ignored.

Value

A [texreg-class](#) object, as produced by [createTexreg](#), which can interface with all of that package's generics. See 'Examples.'

Author(s)

John Niehaus

References

Leifeld, Philip (2013). texreg: Conversion of Statistical Model Output in R to LaTeX and HTML Tables. Journal of Statistical Software, 55(8), 1-24. URL http://dx.doi.org/10.18637/jss.v055.i08.

See Also

[extract](#), [createTexreg](#), [zic.reg](#)

Examples

```
# Simulate some zip data
n=1000
x = cbind(1, rnorm(n))
z = cbind(1, rbeta(n, 4, 8))
b = c(1, 2.2)
g = c(-1, 1.7)
lam = exp(x %*% b)
psi = plogis(z %*% g)


y = bizicount::rzip(n, lambda = lam, psi=psi)
dat = cbind.data.frame(x = x[,-1], z = z[,-1], y = y)
```

```
# estimate model

mod = zic.reg(y ~ x | z, data = dat)


### Output to table with texreg

# extract information

tr_obj_se = texreg::extract(mod)
tr_obj_ci = texreg::extract(mod, CI = .95)

# output to latex, single table

texreg::texreg(list(tr_obj_se, tr_obj_ci))

# output to plain text, multiple tables

texreg::screenreg(tr_obj_se)
texreg::screenreg(tr_obj_ci)
```

---

make_DHARMa   *DHARMa-class objects from bizicount models*

Description

A wrapper around the DHARMa package's createDHARMa function. Creates a list of DHARMa objects, one for each margin of a bizicount-class object, using simulated responses from simulate.bizicount.

Usage

```
make_DHARMa(object, nsim = 250, seed = 123, method = "PIT")
```

Arguments

object	A bizicount-class object, as returned by bizicount.
nsim	Number of simulated responses from the fitted model to use for diagnostics.
seed	Random seed for simulating from fitted model.
method	See createDHARMa.

Value

A list of DHARMa objects.

Note

This is merely a wrapper around the createDHARMa function to conveniently get DHARMa objects for each margin of a bizicount model.

Author(s)

John Niehaus

References

Florian Hartig (2022). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.4.5. https://CRAN.R-project.org/package=DHARMa

See Also

DHARMa, createDHARMa, simulate.bizicount

Examples

```
## SETUP
set.seed(123)
n = 100

# define a function to simulate from a gaussian copula
# first margin is zero-inflated negative binomial (zinb)
# second margin is zero-inflated poisson (zip)
# Note: marginal distributions are hard-coded in function, including
# inverse dispersion parameter for zinb.
gen = function(n, b1, b2, g1, g2, dep) {

    k1 = length(b1)
    k2 = length(b2)

    X1 = cbind(1, matrix(rbinom(n * (k1 - 1), 1, .5), ncol = k1 - 1))
    X2 = cbind(1, matrix(rexp(n * (k2 - 1), 3), ncol = k2 - 1))

    lam1 = exp(X1 %*% b1)
    lam2 = exp(X2 %*% b2)

    Z1 = cbind(1, matrix(runif(n * (k1 - 1), -1, 1), ncol = k1 - 1))
    Z2 = cbind(1, matrix(rnorm(n * (k2 - 1)), ncol = k2 - 1))

    psi1 = plogis(Z1 %*% g1)
    psi2 = plogis(Z2 %*% g2)

    norm_vars = MASS::mvrnorm(
        n,
        mu = c(0, 0),
        Sigma = matrix(c(1, dep, dep, 1), ncol =2)
    )

    U = pnorm(norm_vars)

    y1 =  qzinb(U[, 1],
                mu = lam1,
                psi = psi1,
                size = .3)
```

```
        y2 =  qzip(U[, 2],
                    lambda = lam2,
                    psi = psi2)

        dat = data.frame(
              X1 = X1[, -1],
              X2 = X2[, -1],
              Z1 = Z1[, -1],
              Z2 = Z2[, -1],
              y1,
              y2,
              lam1,
              lam2,
              psi1,
              psi2
        )
        return(dat)
    }


    # define parameters
    b1 = c(1, -2, 3)
    b2 = c(-1, 3, 1)
    g1 = c(2, -1.5, 2)
    g2 = c(-1, -3.75, 1.25)
    rho = .5


    # generate data
    dat = gen(n, b1, b2, g1, g2, rho)
    f1 = y1 ~ X1.1 + X1.2 | Z1.1 + Z1.2
    f2 = y2 ~ X2.1 + X2.2 | Z2.1 + Z2.2

    ## END SETUP




    # estimate model

    mod = bizicount(f1, f2, dat, cop = "g", margins = c("zinb", "zip"), keep=TRUE)


    # diagnose model with DHARMa
    # see end for simulate.bizicount example.

    dharm = make_DHARMa(mod, nsim = 100)

    lapply(dharm, DHARMa::testResiduals)
```

---

predict.zicreg   *Predictions for univariate zero-inflated count regression models*

Description

Predicts the mean, probability, count mean, or zero-inflation probability for new data using parameters from a fitted zero-inflated count regression model.

Usage

```
## S3 method for class 'zicreg'
predict(object, newdata = NULL, y.new = NULL, type = "mean", ...)
```

Arguments

object	A fitted `zic.reg` object.
newdata	A `data.frame` containing new values of the same covariates appearing in fitted model.
y.new	An optional vector of new response values, used only for `type = "prob"`.
type	String, one of `c("mean", "prob", "psi", "lambda")`. `"mean"` will predict the conditional mean of the mixture distribution, `"prob"` will predict the probability of a new response value, `"psi"` will predict the probability of zero-inflation, and `"lambda"` will predict the mean of the count portion of the mixture distribution. NOTE: Setting `type = "mean"` and leaving `newdata = NULL` is the same as calling `fitted(object)`.
...	Ignored.

Value

A numeric vector containing the predictions using the model parameters.

Author(s)

John Niehaus

Examples

```
# Simulate some zip data
n=1000
x = cbind(1, rnorm(n))
z = cbind(1, rbeta(n, 4, 8))
b = c(1, 2.2)
g = c(-1, 1.7)
lam = exp(x %*% b)
psi = plogis(z %*% g)


y = bizicount::rzip(n, lambda = lam, psi=psi)
dat = cbind.data.frame(x = x[,-1], z = z[,-1], y = y)

# estimate model

mod = zic.reg(y ~ x | z, data = dat, keep = TRUE)
```

```
### Predict on observed/training data
# predict conditional mean (fitted values)
predict(mod, type = "mean")

# predict probabilty Y = y
probs_pred_obs = predict(mod, type = "prob")

# predict mean of count distribution (lambda)
lambda_pred_obs = predict(mod, type = "lambda")

# mse predicted vs true lambda values
mean((lam - lambda_pred_obs)**2)

# predict zero inflation probability (psi)
psi_pred_obs = predict(mod, type = "psi")

# MSE predicted vs true zero-inflation probabilities
mean((psi-psi_pred_obs)**2)


### Predict on test data
# simulate some test data

x = cbind(1, rnorm(n, mean = -0.5, sd = 1.25))
z = cbind(1, rbeta(n, 6, 12))
y = rzip(n, lambda = exp(x %*% coef(mod)[1:2]), psi = plogis(z %*% coef(mod)[3:4]))
dat_new = cbind.data.frame(x = x[,-1], z = z[,-1], y = y)

# predict conditional mean
mean_new = predict(mod, type = "mean", newdata = dat_new)
mean((y - mean_new)**2)

# predict probability of Y = y
probs_new = predict(mod, type = "prob", newdata = dat_new, y.new = y)

# predict lambda
lambda_new = predict(mod, type = "lambda", newdata = dat_new)

# predict zero inflation probability
psi_new = predict(mod, type = "psi", newdata = dat_new)
```

---

simulate.bizicount   *Simulating response values using parameters from fitted bizicount models*

Description

Simulates random response values using the fitted conditional mean function for each margin of a
bizicount-class object. Primarily for use with the DHARMa package.

Usage

```
## S3 method for class 'bizicount'
simulate(object, nsim = 250, seed = 123, ...)
```

Arguments

object	A fitted bizicount-class object, as returned by bizicount.
nsim	Number of simulated response values from the fitted model. E.g., nsim = 250 will simulate each observation 250 times, for $n \times 250$ total observations.
seed	Seed used for simulating from fitted model. If NULL, no seed is set.
...	Ignored.

Value

A length 2 list, with each entry containing a numeric $nXnsim$ matrix for each margin of the
bizicount model. Rows index the observation, and columns index the simulated dataset number.

Author(s)

John Niehaus

References

Florian Hartig (2022). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed)
Regression Models. R package version 0.4.5. https://CRAN.R-project.org/package=DHARMa

See Also

createDHARMa, simulateResiduals

Examples

```
## SETUP
set.seed(123)
n = 150

# define a function to simulate from a gaussian copula
# first margin is zero-inflated negative binomial (zinb)
# second margin is zero-inflated poisson (zip)
# Note: marginal distributions are hard-coded in function, including
# inverse dispersion parameter for zinb.
gen = function(n,
               b1,
               b2,
               g1,
```

```
              g2,
              dep) {

    k1 = length(b1)
    k2 = length(b2)

    X1 = cbind(1, matrix(rbinom(n * (k1 - 1), 1, .5), ncol = k1 - 1))
    X2 = cbind(1, matrix(rexp(n * (k2 - 1), 3), ncol = k2 - 1))

    lam1 = exp(X1 %*% b1)
    lam2 = exp(X2 %*% b2)

    Z1 = cbind(1, matrix(runif(n * (k1 - 1), -1, 1), ncol = k1 - 1))
    Z2 = cbind(1, matrix(rnorm(n * (k2 - 1)), ncol = k2 - 1))

    psi1 = plogis(Z1 %*% g1)
    psi2 = plogis(Z2 %*% g2)

    norm_vars = MASS::mvrnorm(
        n,
        mu = c(0, 0),
        Sigma = matrix(c(1, dep, dep, 1), ncol =2)
    )

    U = pnorm(norm_vars)

    y1 =  qzinb(U[, 1],
                  mu = lam1,
                 psi = psi1,
                 size = .3)
    y2 =  qzip(U[, 2],
                lambda = lam2,
                psi = psi2)

    dat = data.frame(
        X1 = X1[, -1],
        X2 = X2[, -1],
        Z1 = Z1[, -1],
        Z2 = Z2[, -1],
        y1,
        y2,
        lam1,
        lam2,
        psi1,
        psi2
    )
    return(dat)
}


# define parameters
b1 = c(1, -2, 3)
b2 = c(-1, 3, 1)
```

```
g1 = c(2, -1.5, 2)
g2 = c(-1, -3.75, 1.25)
rho = .5


# generate data
dat = gen(n, b1, b2, g1, g2, rho)
f1 = y1 ~ X1.1 + X1.2 | Z1.1 + Z1.2
f2 = y2 ~ X2.1 + X2.2 | Z2.1 + Z2.2

## END SETUP




# estimate model
mod = bizicount(f1, f2, dat, cop = "g", margins = c("zinb", "zip"), keep=TRUE)

# simulate from fitted model
sims = simulate(mod, nsim = 150)


# input sims to DHARMa for diagnostics
# margin 1
d1 = DHARMa::createDHARMa(
    simulatedResponse = sims[[1]],
    observedResponse = dat$y1,
    fittedPredictedResponse = fitted(mod)[,1],
    integerResponse = TRUE,
    method = "PIT"
)

# margin 2
d2 = DHARMa::createDHARMa(
    simulatedResponse = sims[[2]],
    observedResponse = dat$y2,
    fittedPredictedResponse = fitted(mod)[,2],
    integerResponse = TRUE,
    method = "PIT"
)

# test each margin
DHARMa::testResiduals(d1)
DHARMa::testResiduals(d2)
```

---

simulate.zicreg *Simulating response values from fitted univariate zero-inflated count regression model*

Description

Simulates responses using the fitted parameters from a `zicreg-class` object, as returned by `zic.reg`. Primarily useful for methods found in `DHARMa` package. See 'Examples.'

Usage

```
## S3 method for class 'zicreg'
simulate(object, nsim = 250, seed = 123, ...)
```

Arguments

object	A `zicreg-class` model object, as returned by `zic.reg`.
nsim	Number of simulated datasets to create.
seed	Random seed for random number generation in simulations. If NULL, no seed is set.
...	Ignored.

Value

A numeric $n x nsim$ matrix, with rows indexing observations, and columns indexing the simulation number.

Author(s)

John Niehaus

References

Florian Hartig (2022). DHARMa: Residual Diagnostics for Hierarchical (Multi-Level / Mixed) Regression Models. R package version 0.4.5. https://CRAN.R-project.org/package=DHARMa

Examples

```
# Simulate some zip data
n=300
x = cbind(1, rnorm(n))
z = cbind(1, rbeta(n, 4, 8))
b = c(1, 2.2)
g = c(-1, 1.7)
lam = exp(x %*% b)
psi = plogis(z %*% g)

y = bizicount::rzip(n, lambda = lam, psi=psi)
dat = cbind.data.frame(x = x[,-1], z = z[,-1], y = y)

# estimate model

mod = zic.reg(y ~ x | z, data = dat, keep = TRUE)

# simulate from fit for use in dharma
```

```
sims = simulate(mod)

### Make dharma object

dharm = DHARMa::createDHARMa(
     simulatedResponse = sims,
     observedResponse = y,
     fittedPredictedResponse = fitted(mod),
     integerResponse = TRUE,
     method = "PIT"
)

### Plot the DHARMa object, do other diagnostics
plot(dharm)
DHARMa::testResiduals(dharm)
```

---

terror   *Nigeria Terrorism Data*

Description

Data on terrorist attacks by Fulani Extremists and Boko Haram in Nigeria, from the year 2014. Attacks data from Global Terrorism Database, other variables from UCDP PRIO-Grid data.

Usage

    terror

Format

> terror:
>
> A data frame with 312 rows and 6 columns:
>
> **att.ful, att.bok** Integer number of attacks by Fulani Extremists and Boko Haram in 2014.
> **xcoord, ycoord** Longitude and Latidude of grid-cell centroid where attack occurs.
> **pop** Population in grid cell.
> **mtns** Proportion of terrain in grid cell that is considered mountainous.

---

zic.reg   *Univariate zero-inflated Poisson and negative binomial regression models*

Description

This function from the [bizicount](#) package estimates univariate zero-inflated Poisson and negative binomial regression models via maximum likelihood using either the [nlm](#) or [optim](#) optimization functions. It's class has associated [simulate](#) methods for post-estimation diagnostics using the [DHARMa](#) package, as well as an [extract](#) method for printing professional tables using [texreg](#). Visit the 'See Also' section for links to these methods for zicreg objects.

Usage

```
zic.reg(
  fmla = NULL,
  data,
  dist = "pois",
  link.ct = "log",
  link.zi = "logit",
  optimizer = "nlm",
  starts = NULL,
  subset,
  na.action,
  weights = rep(1, length(y)),
  X = NULL,
  z = NULL,
  y = NULL,
  offset.ct = NULL,
  offset.zi = NULL,
  warn.parent = T,
  keep = F,
  ...
)
```

Arguments

fmla	A [formula](#) of the form y ~ x_1 + x_2 + ... + x_n + offset(count_var) | z_1 + ... z_n + offset where the x values are covariates in the count portion of the model, and z are in the zero-inflation portion. The z and x variables can be the same. If NULL, design matrices, the response vector, and offsets can be entered directly; see X, z, y, offset.ct, and offset.zi below.
data	A [data.frame](#) containing all variables appearing in fmla, including offsets. If not specified, variables are searched for in parent environment.
dist	The distribution used for the count portion of the zero-inflated mixture. One of c("pois", "nbinom"), partial matching supported.
link.ct	String specifying the link function used for the count portion of the mixture distribution. One of c("log", "identity", "sqrt"). See [family](#).
link.zi	Character string specifying the link function used for the zero-inflation portion of the mixture distribution. One of c("logit", "probit", "cauchit", "log", "cloglog"). See [family](#).
optimizer	String specifying the optimizer to be used for fitting, one of c("nlm", "optim"). If "optim", defaults to method="BFGS".
starts	Optional vector of starting values used for the numerical optimization procedure. Should have count parameters first (with intercept first, if applicable), followed by zero-inflated parameters (with intercept first, if applicable), and the inverse dispersion parameter last (if applicable).
subset	Vector indicating the subset of observations on which to estimate the model

na.action	A function which indicates what should happen when the data contain NAs. Default is `na.omit`.
weights	An optional numeric vector of weights for each observation.
X, z	If `fmla = NULL`, these are the design matrices of covariates for the count and zero-inflation portions, respectively. Both require no missingness. Similar in spirit to `glm.fit` in that it can be faster for larger datasets because it bypasses model matrix creation.
y	If `fmla = NULL`, a vector containing the response variable.
offset.ct, offset.zi	
	If `fmla = NULL`, vectors containing the (constant) offset for the count and zero-inflated portions, respectively. Must be equal in length to y, and row-dim of X, z. If left NULL, defaults to `rep(0, length(y))`.
warn.parent	Logical indicating whether to warn about `data` not being supplied.
keep	Logical indicating whether to keep the model matrices in the returned model object. Must be TRUE to use `DHARMa` and `texreg` with the model object, e.g., via `simulate.zicreg` and `extract.zicreg`, as well as base generics like `fitted` and `predict`.
...	Additional arguments to pass on to the chosen optimizer, either `nlm` or `optim`. See 'Examples'.

Value

An S3 `zicreg-class` object, which is a list containing:

- `call` – The original function call
- `obj` – The class of the object
- `coef` – Vector of coefficients, with count, then zi, then dispersion.
- `se` – Vector of asymptotic standard errors
- `grad` – Gradient vector at convergence
- `link.ct` – Name of link used for count portion
- `link.zi` – Name of link used for zero-inflated portion
- `dist` – Name of distribution used for count portion
- `optimizer` – Name of optimization package used in fitting
- `coefmat.ct` – Coefficient matrix for count portion
- `coefmat.zi` – Coefficient matrix for zero-inflated portion
- `convergence` – Convergence code from optimization routine.
- `coefmat.all` – Coefficient matrix for both parts of the model
- `theta` – Coefficient matrix for dispersion, if applicable.
- `covmat` – Asymptotic covariance matrix
- `nobs` – Number of observations
- `aic` – Akaike information
- `bic` – Bayes information
- `loglik` – Log-likelihood at convergence
- `model` – List containing model matrices if keep = TRUE

Author(s)

John Niehaus

References

Lambert, Diane. "Zero-inflated Poisson regression, with an application to defects in manufacturing." Technometrics 34.1 (1992): 1-14.

See Also

[simulate.zicreg](), [extract.zicreg]()

Examples

```
## ZIP example
# Simulate some zip data
n=1000
x = cbind(1, rnorm(n))
z = cbind(1, rbeta(n, 4, 8))
b = c(1, 2.2)
g = c(-1, 1.7)
lam = exp(x %*% b)
psi = plogis(z %*% g)

y = bizicount::rzip(n, lambda = lam, psi=psi)
dat = cbind.data.frame(x = x[,-1], z = z[,-1], y = y)

# estimate zip model using NLM, no data.frame

mod = zic.reg(y ~ x[,-1] | z[,-1])

# same model, with dataframe

mod = zic.reg(y ~ x | z, data = dat)


# estimate zip using NLM, adjust stepmax via ... param

mod = zic.reg(y ~ x[,-1] | z[,-1], stepmax = .5)


# estimate zip using optim

mod = zic.reg(y ~ x[,-1] | z[,-1], optimizer = "optim")


# pass different method, reltol to optim using ... param

mod = zic.reg(y ~ x[,-1] | z[,-1],
        optimizer = "optim",
        method = "Nelder-Mead",
        control = list(reltol = 1e-10)
```

```
        )

# No formula, specify design matrices and offsets.
zic.reg(y=y, X=x, z=z)



## ZINB example
# simulate zinb data

disp = .5
y = bizicount::rzinb(n, psi = psi, size = disp, mu=lam)


# zinb model, use keep = TRUE for post-estimation methods

mod = zic.reg(y ~ x[,-1] | z[,-1], dist = "n", keep = TRUE)

print(mod)
summary(mod)
```

---

zicreg-class   *The zicreg S4 Class*

Description

Note that zicreg objects are, in general, S3. However, this S4 class is defined for compatability with [texreg](). Interaction with zicreg objects should generally use S3 syntax, but the below objects have the same name in both the S3 and S4 objects (but are in a list for S3).

Slots

call The original function call

obj The class of the object

coef Vector of coefficients, with count, then zi, then dispersion.

se Vector of asymptotic standard errors

grad Gradient vector at convergence

link.ct Name of link used for count portion

link.zi Name of link used for zero-inflated portion

dist Name of distribution used for count portion

optimizer Name of optimization package used in fitting

coefmat.ct Coefficient matrix for count portion

coefmat.zi Coefficient matrix for zero-inflated portion

convergence Convergence code from optimization routine.

coefmat.all Coefficient matrix for both parts of the model

theta Coefficient matrix for dispersion, if applicable.

covmat Asymptotic covariance matrix

nobs Number of observations

aic Akaike information

bic Bayes information

loglik Log-likelihood at convergence

model List containing model matrices if keep = TRUE

---

zinb   *The zero-inflated negative binomial (ZINB) distribution*

Description

These functions are used to evaluate the zero-inflated negative binomial distribution's probability mass function (PMF), cumulative distribution function (CDF), and quantile function (inverse CDF), as well as generate random realizations from the ZINB distribution.

Usage

```
dzinb(
  x,
  size,
  psi,
  mu = NULL,
  prob = NULL,
  lower.tail = TRUE,
  log = FALSE,
  recycle = FALSE
)

pzinb(
  q,
  size,
  psi,
  mu = NULL,
  prob = NULL,
  lower.tail = TRUE,
  log.p = FALSE,
  recycle = FALSE
)

qzinb(
  p,
  size,
```

```
  psi,
  mu = NULL,
  prob = NULL,
  lower.tail = TRUE,
  log.p = FALSE,
  recycle = FALSE
)

rzinb(n, size, psi, mu = NULL, prob = NULL, recycle = FALSE)
```

Arguments

x, q	Vector of quantiles at which to evaluate the PMF and CDF, respectively. Should be non-negative integers.
size	The inverse dispersion parameter, or number of successful trials, both for the negative binomial portion of the ZINB mixture distribution. See [nbinom](#).
psi	Vector of zero-inflation probabilities.
mu	Vector of means for the count portion of the zero-inflated negative binomial distribution. Only one of mu or prob should be specified, not both. Should be non-negative. NOTE: This is *not* the mean of the ZINB distribution; it is the mean of the NB component of the mixture distribution. See [nbinom](#).
prob	The probability of success on each trial in the negative binomial portion of the mixture distribution. Only one of mu or prob should be specified, not both. See [nbinom](#).
lower.tail	Logical indicating whether probabilities should be $Pr(X \leq x)$ or $Pr(X > x)$
log, log.p	Logical indicating whether probabilities should be returned on log scale (for dzip and pzip), or are supplied on log-scale (for qzip).
recycle	Logical indicating whether to permit arbitrary recycling of arguments with unequal length. See 'Details' and 'Examples.'
p	Vector of probabilities at which to evaluate the quantile function.
n	Number of realizations to generate from the distribution

Value

dzinb returns the mass function evaluated at x, pzinb returns the CDF evaluated at q, qzinb returns the quantile function evaluated at p, and rzinb returns random realizations with the specified parameters.

Author(s)

John Niehaus

References

Lambert, Diane. "Zero-inflated Poisson regression, with an application to defects in manufacturing." Technometrics 34.1 (1992): 1-14.

Examples

```
# zero-inflated negative binomial examples

# two unique lengthed arguments, one is length 1 though. No error.

dzinb(4, size=.25, mu= c(1,2,3), psi=c(.2, .1, .15))


# two unique lengthed arguments, one of them is not length 1
# error
## Not run:

    dzinb(5, size=c(.25, .3), mu= c(1,2,3), psi=c(.2, .1, .15))


## End(Not run)


# two unique lengthed arguments, one of them is not length 1, set
# recycle = T, no error but can give innacurate results.

dzinb(5, size=c(.25, .3), mu= c(1,2,3), psi=c(.2, .1, .15), recycle=TRUE)
```

---

zip   *The zero-inflated Poisson (ZIP) distribution*

Description

These functions are used to evaluate the zero-inflated Poisson distribution's probability mass function (PMF), cumulative distribution function (CDF), and quantile function (inverse CDF), as well as generate random realizations from the ZIP distribution.

Usage

```
dzip(x, lambda, psi, log = FALSE, recycle = FALSE)

rzip(n, lambda, psi, recycle = FALSE)

pzip(q, lambda, psi, lower.tail = TRUE, log.p = FALSE, recycle = FALSE)

qzip(p, lambda, psi, lower.tail = TRUE, log.p = FALSE, recycle = FALSE)
```

Arguments

x, q   Vector of quantiles at which to evaluate the PMF and CDF, respectively. Should
                be non-negative integers.

lambda	Vector of means for the count portion of the zero-inflated Poisson distribution. Should be non-negative. NOTE: This is *not* the mean of the zero-inflated Poisson distribution; it is the mean of the Poisson component of the mixture distribution. See 'Details.'
psi	Vector of zero-inflation probabilities.
log, log.p	Logical indicating whether probabilities should be returned on log scale (for dzip and pzip), or are supplied on log-scale (for qzip).
recycle	Logical indicating whether to permit arbitrary recycling of arguments with unequal length. See 'Details' and 'Examples.'
n	Number of realizations from the distribution to generate
lower.tail	Logical indicating whether probabilities should be $Pr(X \leq x)$ or $Pr(X > x)$
p	Vector of probabilities at which to evaluate the quantile function.

Details

The zero inflated Poisson distribution is a mixture of a Poisson and a degenerate point-mass at 0. It has the form

$$\psi + (1 - \psi)(\lambda^x e^{-}\lambda)/x!$$

, with mean $(1 - \psi)\lambda$. Thus, the parameter lambda above is the mean of the Poisson distribution that forms part of the zero-inflated distribution, *not* the mean of the ZIP distribution.

recycle – If FALSE (default), all arguments must have identical length, there can be two unique lengths for the arguments, provided that one of those lengths is 1. For example, lambda = c(1,2,3) and psi=.5 is acceptable because there are two unique lengths, and one of them is length 1. However, lambda=c(1,2,3) and psi=c(.5,.2) would fail, as there are two distinct lengths, none of which is 1. If TRUE, no additional checks (beyond those in base R's functions) are made to ensure that the argument vectors have the same length.

Value

dzip returns the mass function evaluated at x, pzip returns the CDF evaluated at q, qzip returns the quantile function evaluated at p, and rzip returns random variates with the specified parameters.

Author(s)

John Niehaus

References

Lambert, Diane. "Zero-inflated Poisson regression, with an application to defects in manufacturing." Technometrics 34.1 (1992): 1-14.

Examples

```
# Unequal lengths, but one of them is length 1, others are same length (3).
# No error.

x = c(1,2,3)
```

```
lambda = c(3,4,5)
psi = .1

dzip(x, lambda, psi)


# unequal lengths, at least one of them is not length 1,
# error

## Not run:

x = c(1,2,3)
lambda = c(3,4)
psi = .1

dzip(x, lambda, psi)


## End(Not run)

# unequal lengths, at least one of them is not length 1.
# but set recycle = T to permit arbitrary recycling.

x = c(1,2,3)
lambda = c(3,4)
psi = .1

dzip(x, lambda, psi, recycle=TRUE)
```

---

zi_test   *He's (2019) test for zero-modification*

Description

This is an implementation of He et al. (2019)'s test for zero-modification (discussed further in Tang & Tang (2019)). This is a test of zero-*modification* instead of *inflation*, because the test is capable of detecting both excessive or lack of zeros, but cannot determine the cause. For example, a mixed data generating process could be generating structural zeros, implying a zero-inflated distribution. However, overdispersion via a negative binomial may also result in excessive zeros. Thus, the test merely determines whether there are excessive (or lacking) zeros, but does not determine the process generating this pattern. That in mind, typical tests in the literature are inappropriate for zero-modified regression models, namely the Vuong, Wald, score, and likelihood ratio tests. See the references below for more information on this claim.

Usage

```
zi_test(model, alternative = "inflated")
```

Arguments

model
: A model object of class [bizicount](#) or [glm](#). If a bizicount model, then at least one margin must be specified as "pois". If a glm model, then the family must be [poisson](#).

alternative
: The alternative hypothesis. One of c("inflated", "deflated", "both"). These correspond to an upper tail, lower tail, or two-tailed test, respectively. Default is "inflated". Partial matching supported.

Details

The test compares the observed proportion of zeros in the data to the expected proportion of zeros under the null hypothesis of a Poisson distribution. This is done using estimating equations to account for the fact that the expected proportion is based on an estimated parameter vector, rather than the true parameter vector. The test statistic is

$\hat{s} = 1/n \sum_i (r_i - \hat{p}_i)$

where $r_i = 1$ if $y_i = 0$, otherwise $r_i = 0$, and $\hat{p} = dpois(0, exp(X\hat{\beta})) = \hat{E}(r_i)$ is the estimated proportion of zeros under the assumption of a Poisson distribution generated with covariates X and parameter vector $\hat{\beta}$.

By the central limit theorem, $\hat{s} \sim AN(0, \sigma_s^2)$. However, estimating $\hat{\sigma}_s$ by a plug-in estimate using $\hat{\beta}$ is inefficient due to $\hat{\beta}$ being an random variable with its own variance. Thus, $\hat{\sigma}$ is estimated via estimating equations in order to account for the variance in $\hat{\beta}$.

See the references below for more discussion and proofs.

Author(s)

John Niehaus

References

He, H., Zhang, H., Ye, P., & Tang, W. (2019). A test of inflated zeros for Poisson regression models. Statistical methods in medical research, 28(4), 1157-1169.

Tang, Y., & Tang, W. (2019). Testing modified zeros for Poisson regression models. Statistical Methods in Medical Research, 28(10-11), 3123-3141.

Examples

```
set.seed(123)
n = 500
u = rpois(n, 3)
y1 = rzip(n, 12, .2) + u
y2 = rpois(n, 8) + u

# Single parameter test, covariates can be added though.
uni1 = glm(y1 ~ 1, family = poisson())
uni2 = glm(y2 ~ 1, family = poisson())

biv = bizicount(y1~1, y2~1, margins = c("pois", "pois"), keep = TRUE)
```

```
zi_test(uni1)
zi_test(uni2)

zi_test(biv)
```

Index