Package 'underdisp'

August 20, 2026

Type Package

Title Diagnostics and Models for Underdispersed Count Data

Version 0.1.0

Description Tools for detecting and modeling underdispersion in count data (conditional variance below the conditional mean), a phenomenon overlooked by the Poisson and negative binomial defaults. Provides a screening diagnostic that benchmarks at-risk dispersion against a zero-truncated Poisson; the continuous parameter binomial (CPB) regression and its zero-truncated variant, with an interpretable observation-specific bound and high-dimensional fixed-effects support; validated bootstrap (for coefficients) and profile-likelihood (for the dispersion parameter) inference; and quantities of interest including predicted probabilities and the implied ceiling. The likelihood is implemented in C++ for speed.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

Imports Rcpp, stats, MASS, VGAM, graphics, methods, numDeriv

LinkingTo Rcpp

Suggests sandwich, pscl, DHARMa, testthat (>= 3.0.0), knitr, rmarkdown, broom, modelsummary, texreg

VignetteBuilder knitr

RoxygenNote 7.3.3

URL https://github.com/bagozzib/underdisp

BugReports https://github.com/bagozzib/underdisp/issues

Config/testthat/edition 3

NeedsCompilation yes

Author Benjamin E. Bagozzi [aut, cre]

Maintainer Benjamin E. Bagozzi <bagozzib@udel.edu>

Repository CRAN

Date/Publication 2026-08-20 16:40:02 UTC

Contents

`alpha_confint` *Profile-likelihood interval for the dispersion parameter alpha*

Description

Profile-likelihood interval for the dispersion parameter alpha

Usage

```
alpha_confint(object, level = 0.95)
```

Arguments

`object`	A "cpb" object.
`level`	Confidence level (default 0.95).

Value

A length-2 numeric vector (`lower`, `upper`) with attributes `alpha` (the point estimate) and `boundary` (`TRUE` if the lower bound is at the feasibility boundary, i.e. strong underdispersion, where the interval is one-sided).

Examples

```
set.seed(7); x <- rnorm(300)
N <- pmax(round(exp(1.5 + 0.4 * x) / 0.5), 1); y <- rbinom(300, N, 0.5)
fit <- cpb(y ~ x, data.frame(y = y, x = x)[y > 0, ], se = "none")
alpha_confint(fit)
```

augment.cpb *Augment data with CPB fitted values (broom method)*

Description

Augment data with CPB fitted values (broom method)

Usage

```
## S3 method for class 'cpb'
augment(x, ...)
```

Arguments

x	A "cpb" object.
...	Unused.

Value

A data frame with fitted values, response residuals, and the implied per-observation ceiling.

compare_dispersion *Compare count-model dispersion across the model family*

Description

Fits the Poisson and negative binomial defaults alongside the two underdispersed workhorses—the soft-tail Conway–Maxwell–Poisson (fit natively, no external dependency) and the hard-ceiling CPB—and returns a comparison on both fit (log-likelihood, AIC, BIC) and calibration (mean logarithmic score and ranked probability score, lower is better, plus the fitted share of zeros against the observed share). Whether the tail is better described by an accelerated decay (COM-Poisson) or a binding ceiling (CPB) is a testable question, not an assumption; the scoring rules are the calibration counterpart to the information criteria. The CPB's ceiling-exceedance share (observations above the implied ceiling) is reported as a falsification check on the hard bound.

Usage

```
compare_dispersion(
  formula,
  data,
  max.support = 500,
  hurdle = FALSE,
  zi = FALSE
)
```

Arguments

formula	A model formula.
data	A data frame.
max.support	Passed to cpb().
hurdle	If TRUE and the data contain zeros, also fit and score a hurdle_cpb().
zi	If TRUE and the data contain zeros, also fit and score a zi_cpb(). The mixture EM is slow on large panels, so it is a separate opt-in from hurdle.

Details

With hurdle = TRUE and zeros present, a hurdle-CPB is added, so a zero-inflated underdispersed process can be compared to the single-equation models on the same footing.

Value

A list with table (a data frame of df, logLik, AIC, BIC, logscore, rps, and zero_fit per model), obs_zero (the observed zero share), nu (the COM-Poisson dispersion, >1 = underdispersion, NA if the COM-Poisson fit failed), alpha (the CPB shape parameter), ceiling_exceedance (share of observations above the CPB ceiling), and cpb_ok (FALSE if the single-equation CPB could not satisfy its feasibility constraint on the data, e.g. under heavy zero-inflation with a wide count range — itself a signal that a hurdle or zero-inflated model is needed).

See Also

ud_screen(), cpb(), score()

Examples

```
set.seed(1); x <- rnorm(120)
N <- pmax(round(exp(1.6 + 0.4 * x) / 0.5), 1); y <- rbinom(120, N, 0.5)
compare_dispersion(y ~ x, data = data.frame(y = y, x = x))$table
```

---

compare_models *Compare fitted underdispersed-count models*

Description

Compares fitted models from this package — cpb, cpb_fe, hurdle_cpb, or zi_cpb, in any combination of fixed effects and robust/clustered standard errors — on information criteria and, where a predicted distribution is available, on proper scores. Unlike zi_test() this is not a hypothesis test: the models need not be nested, so it is the appropriate tool for the non-nested hurdle-versus-mixture choice. Passing an object that is not a fitted model from this package is an error, and models fit on different data raise a warning.

Usage

```
compare_models(...)
```

Arguments

...	Two or more fitted models (cpb, cpb_fe, hurdle_cpb, zi_cpb), optionally named.

Value

A data frame with df, logLik, AIC, BIC, and (where the predicted distribution is available) logscore and rps, one row per model, ordered by AIC.

See Also

zi_test(), compare_dispersion()

Examples

```
set.seed(1); n <- 800; x <- rnorm(n); z <- rnorm(n)
y <- rhurdle_cpb(n, exp(1.2 + 0.5 * x), 0.5, plogis(-0.2 + 0.8 * z))
d <- data.frame(y = y, x = x, z = z)
compare_models(hurdle = hurdle_cpb(y ~ x, data = d, participation = ~ z),
               zi = zi_cpb(y ~ x, data = d, zero = ~ z))
```

---

compois-distribution *Conway–Maxwell–Poisson distribution functions*

Description

Density, distribution, quantile, and random generation for the COM-Poisson with rate lambda and dispersion nu (nu > 1 underdispersed, nu = 1 Poisson, nu < 1 overdispersed). Complements the estimator count_reg(..., family = "compois").

Usage

```
dcompois(x, lambda, nu, log = FALSE)

pcompois(q, lambda, nu, lower.tail = TRUE, log.p = FALSE)

qcompois(p, lambda, nu, lower.tail = TRUE, log.p = FALSE)

rcompois(n, lambda, nu)
```

Arguments

x, q	Vector of quantiles (non-negative integers).
lambda	Rate parameter (scalar or vector, recycled).
nu	Dispersion parameter (scalar).
log, log.p	Return log probabilities.

`lower.tail`	If `TRUE` (default), $P(X \leq x)$.
`p`	Vector of probabilities.
`n`	Number of draws.

Value

`dcompois` a density, `pcompois` a CDF, `qcompois` a quantile, `rcompois` a numeric vector of count draws.

Examples

```
dcompois(0:5, lambda = 3, nu = 1.5)
mean(rcompois(1000, lambda = 3, nu = 1.5))
```

`confint.cpb` *Confidence intervals for a CPB fit*

Description

Coefficient intervals use the cold-multistart bootstrap percentile method (validated to nominal coverage); the interval for `alpha` uses the profile-likelihood method, which is reliable except under strong underdispersion, where `alpha` sits at the feasibility boundary and the interval is one-sided.

Usage

```
## S3 method for class 'cpb'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	A `"cpb"` object fit with `se = "bootstrap"`.
`parm`	Optional subset of parameters (coefficient names and/or `"alpha"`).
`level`	Confidence level (default 0.95).
`...`	Unused.

Value

A matrix of lower/upper bounds.

count_reg *Poisson and negative-binomial count regression (matched CPB baselines)*

Description

Fits a Poisson or negative-binomial regression with a log link, optionally zero-truncated and/or with unit fixed effects, returning an object that `compare_models()`, `score()`, and the **broom** methods treat on the same footing as a `cpb()` or `gec()` fit. This is the matched baseline for the underdispersed models: it exists so a Poisson/NB can be compared to a CPB with identical degrees-of-freedom, log-likelihood, proper-score, and robust-standard-error accounting, rather than reconciled across packages.

Usage

```
count_reg(
  formula,
  data,
  family = c("poisson", "negbin", "compois"),
  truncated = FALSE,
  fe = NULL,
  offset = NULL,
  se = c("analytic", "robust", "cluster", "none"),
  cluster = NULL,
  ...
)
```

Arguments

`formula`	A model formula.
`data`	A data frame.
`family`	`"poisson"`, `"negbin"` (negative binomial), or `"compois"` (Conway–Maxwell–Poisson). For `"compois"` the coefficients are on the log-rate scale ($\log \lambda = x'\beta$; the rate λ is *not* the mean, though `predict(type = "response")` and `fitted()` return the mean), and the shape in `$theta` is the dispersion ν ($\nu > 1$ underdispersed, $\nu = 1$ Poisson, $\nu < 1$ overdispersed).
`truncated`	Logical; if `TRUE`, fit the zero-truncated form (requires all `Y >= 1`), the count analogue of the CPB's zero-truncated default.
`fe`	Optional column name(s) for fixed effects, entered as factor dummies (so the degrees of freedom count each absorbed intercept, matching `cpb_fe()`). Pass a vector of two columns for two-way (e.g. unit and time) fixed effects.
`offset`	Optional offset entered on the linear-predictor (log) scale – the log-mean for `"poisson"`/`"negbin"`, the log-rate ($\log \lambda$) for `"compois"`. A numeric vector or the name of a column in `data`, e.g. $\log(\text{exposure})$ so the model becomes a rate model.

se	Standard errors: "analytic" (inverse information, default), "robust" (heteroskedasticity-consistent sandwich), "cluster" (cluster-robust; needs cluster), or "none".
cluster	Optional cluster identifier (a column name in data or a vector aligned to its rows) for se = "cluster".
...	Unused.

Value

An object of class "count_reg". The component $theta holds the shape parameter: the negative-binomial size for "negbin", the COM-Poisson dispersion ν for "compois", and NA for "poisson".

See Also

cpb(), compare_models(), hurdle_count(), zi_count()

Examples

```
set.seed(1); n <- 400; x <- rnorm(n)
y <- rpois(n, exp(1 + 0.5 * x))
m <- count_reg(y ~ x, data = data.frame(y = y, x = x), family = "poisson")
compare_models(pois = m,
               nb = count_reg(y ~ x, data = data.frame(y = y, x = x), family = "negbin"))
```

---

cpb *Fit a Continuous Parameter Binomial (CPB) regression*

Description

Fits the underdispersed continuous parameter binomial model of King (1989), in which the conditional variance is a fraction of the conditional mean, $\mathrm{Var}(Y \mid x) = \alpha\,\mathrm{E}(Y \mid x)$ with $0 < \alpha < 1$, and each observation has an endogenous ceiling $\lambda_i/(1-\alpha)$. A zero-truncated variant (the default) conditions on $Y \geq 1$, appropriate when underdispersion lives among the positive counts of an otherwise zero-inflated outcome.

Usage

```
cpb(
  formula,
  data,
  truncated = TRUE,
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  offset = NULL,
  alpha.start = 0.5,
  max.support = 500,
  maxit = 20000,
  reltol = 1e-08
)
```

Arguments

formula	A model formula.
data	A data frame.
truncated	Logical; if TRUE (default) fit the zero-truncated CPB (requires all Y >= 1); if FALSE fit the untruncated CPB on Y >= 0. **Note the sibling default differs:** cpb_fe() defaults to truncated = FALSE, because its typical call fits a whole panel including zeros, while cpb()'s typical call fits the positive counts of a zero-inflated outcome. State truncated explicitly when moving a specification between the two.
se	Inference method: "none" (default; fast, no standard errors) or "bootstrap" (cold-multistart pairs/cluster bootstrap; slower).
B	Number of bootstrap resamples (default 500).
cluster	Optional cluster identifier for cluster-robust inference: a column name in data or a vector aligned to its rows. When supplied, the bootstrap resamples whole clusters (a block bootstrap), giving cluster-robust standard errors and intervals; when NULL (default) it resamples observations, giving heteroskedasticity-robust errors. This is the appropriate route to robust inference here because the CPB's parameter-dependent support makes a Hessian-based sandwich unreliable.
offset	Optional offset on the log-mean scale (an exposure): a numeric vector or the name of a column in data, making the model a rate model.
alpha.start	Starting value for the dispersion parameter (default 0.5); the fit also multi-starts over a spread of alpha values.
max.support	Guard on the maximum evaluated support (default 500); fits with an implied ceiling above this are treated as infeasible (alpha near 1).
maxit, reltol	Optimizer controls passed to stats::optim().

Details

The mean is modelled log-linearly, $\lambda_i = \exp(x_i'\beta)$. Because the support depends on the parameters, the log-likelihood is non-smooth at the feasibility boundary and the numerical Hessian is unreliable; inference therefore uses a cold-multistart bootstrap for the coefficients (validated to nominal coverage) and a profile-likelihood interval for α (see confint.cpb()).

Value

An object of class "cpb": a list with coefficients, alpha, bootstrap se.beta/vcov/ci.beta, loglik, loglik.null, fitted.values, ceiling (the observation-specific implied ceiling), the model frame pieces, and the original call.

References

King, G. (1989). Variance specification in event count models. *American Journal of Political Science*, 33(3), 762-784.

See Also

ud_screen(), confint.cpb(), predict.cpb(), implied_ceiling()

Examples

```
set.seed(1)
n <- 300; x <- rnorm(n)
N <- pmax(round(exp(1.6 + 0.5 * x) / 0.5), 1)
y <- rbinom(n, N, 0.5)                # underdispersed (var/mean approx 0.5)
d <- data.frame(y = y, x = x)
fit <- cpb(y ~ x, data = d[d$y > 0, ], se = "none")
summary(fit)
```

`cpb-distribution` *Continuous parameter binomial distribution functions*

Description

Density, distribution function, and quantile function for the continuous parameter binomial (CPB), and its zero-truncated form. The CPB has a hard ceiling at $\lfloor \lambda/(1-\alpha) \rfloor$; probability above it is zero. Complements the simulator `rcpb()`.

Usage

```
dcpb(x, lambda, alpha, truncated = FALSE, log = FALSE)

pcpb(q, lambda, alpha, truncated = FALSE, lower.tail = TRUE, log.p = FALSE)

qcpb(p, lambda, alpha, truncated = FALSE, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`x`, `q`	Vector of quantiles (non-negative integers).
`lambda`	Mean parameter (scalar or vector, recycled).
`alpha`	Shape parameter in (0, 1).
`truncated`	If `TRUE`, use the zero-truncated CPB.
`log`, `log.p`	If `TRUE`, probabilities are given as log.
`lower.tail`	If `TRUE` (default), probabilities are $P(X \le x)$.
`p`	Vector of probabilities.

Value

`dcpb` a density, `pcpb` a distribution function, `qcpb` a quantile.

See Also

`rcpb()`

Examples

```
dcpb(0:5, lambda = 3, alpha = 0.5)
pcpb(3, lambda = 3, alpha = 0.5)
qcpb(0.9, lambda = 3, alpha = 0.5)
```

---

cpb_fe *CPB regression with high-dimensional unit fixed effects*

Description

Fits the CPB with a full set of unit fixed effects by concentrating (profiling) out the unit intercepts. Each unit's intercept is solved by a one-dimensional inner maximization, so the outer optimizer handles only the covariate coefficients and the dispersion parameter. This avoids an explicit unit-dummy design matrix and scales to thousands of units, where dummy-based fitting exhausts memory. The concentrated log-likelihood equals the full-dummy log-likelihood exactly.

Usage

```
cpb_fe(
  formula,
  data,
  fe,
  truncated = FALSE,
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  offset = NULL,
  max.support = 500L,
  inner_it = 30L,
  maxit = 3000L,
  reltol = 1e-07,
  bias_correct = c("none", "jackknife")
)
```

Arguments

formula	A model formula for the covariates only. Do not include the unit factor; supply it via fe. The intercept is absorbed by the fixed effects.
data	A data frame.
fe	Name of the column holding the unit identifier.
truncated	Logical; fit the zero-truncated CPB if TRUE. **Note the sibling default differs:** cpb_fe() defaults to FALSE (whole-panel fits including zeros), while cpb() defaults to TRUE (positives-only fits). State truncated explicitly when moving a specification between the two.
se	Inference for the covariate coefficients: "none" (default) or "bootstrap", a pairs/cluster bootstrap.

B	Bootstrap resamples when se = "bootstrap".
cluster	Cluster for the bootstrap. NULL (default) resamples the fixed-effects units themselves (unit-clustered inference, the natural panel default); a column name or vector resamples those clusters while preserving the unit fixed effects. Duplicated blocks are relabeled so the concentrated likelihood treats them as distinct units.
offset	Optional offset on the log-mean scale (an exposure): a numeric vector or the name of a column in data.
max.support	Guard on the maximum evaluated support.
inner_it	Golden-section iterations for each unit's inner maximization.
maxit, reltol	Outer optimizer controls.
bias_correct	"none" (default) or "jackknife", the split-panel jackknife correction for the 1/T incidental-parameters bias (see Details).

Details

Short panels: the dispersion parameter alpha and the fixed effects are subject to the incidental-parameters bias of nonlinear fixed-effects estimation. The bias in alpha is downward and of order 1/T, where T is the per-unit number of observations. It is small for T at least about 30 and should be treated cautiously for short panels. The covariate coefficients are not materially affected. bias_correct = "jackknife" removes the leading 1/T term by the split-panel (half-panel) jackknife: the model is refit on the first and second temporal halves of every unit's series (rows are split in the order supplied, which should be temporal order) and the corrected estimate is 2 * full - mean(halves). In the package's fixed-effects bias Monte Carlo the alpha bias at T = 6/10/20/40 falls from -0.147/-0.096/-0.056/-0.033 to -0.038/-0.022/-0.016/-0.009. With correction on, coefficients and alpha are the corrected estimates (the maximum-likelihood values are kept in $uncorrected), the unit effects and fitted values are re-concentrated at the corrected parameters, and logLik/AIC continue to refer to the maximum-likelihood fit.

Validity gate: the split-panel identity requires the two half-panels to estimate the same pseudo-true parameter (time-homogeneity; Dhaene & Jochmans 2015, Review of Economic Studies). On trending or time-heterogeneous panels the correction is invalid, so the function REFUSES it – returning the uncorrected maximum-likelihood fit with a warning naming the failed check – whenever the corrected dispersion leaves its feasible space or the two half-panel dispersion estimates disagree beyond what the 1/T bias can explain. A refusal is diagnostic information about the panel, not an error. The gate is deliberately powered over sized: in the package's calibration Monte Carlo it refuses about 9 percent of genuinely time-homogeneous panels (a conservative nuisance; the returned fit is exactly the ordinary maximum-likelihood estimate) while catching 98 percent of dispersion regime changes and 100 percent of smooth unmodeled trends – the cases where an uncaught correction would be silently wrong.

One set of fixed effects is concentrated out. For two-way (e.g. unit and time) fixed effects, add factor(time) to formula (entered as dummies), or use count_reg(), which supports two-way fixed effects natively.

Value

An object of class "cpb_fe" with coefficients, alpha, loglik, fe (the estimated unit intercepts), fitted.values, and the implied per-observation ceiling.

See Also

cpb(), ud_screen()

Examples

```
set.seed(1)
d <- do.call(rbind, lapply(1:60, function(i) {
  x <- rnorm(20); lam <- exp(rnorm(1, 0, 0.5) + 0.5 * x)
  N <- pmax(round(lam / 0.5), 1)
  data.frame(unit = i, x = x, y = rbinom(20, N, 0.5))
}))
fit <- cpb_fe(y ~ x, data = d, fe = "unit")
fit
```

---

cv_score *Cross-validated proper scores for a count model*

Description

K-fold cross-validated log score and RPS: the data are split into k folds, the model is refit on each training set via fitfun, and its predicted distribution is scored on the held-out fold. Because the score is genuinely out of sample it penalizes over-parameterization, so it is the honest criterion for comparing models that differ in the number of parameters (for example a hurdle with per-unit participation fixed effects against a zero-inflated model). Per-observation held-out scores are returned so two models fit on the *same* folds can be compared with a paired, cluster-robust test.

Usage

```
cv_score(fitfun, data, k = 5, kmax = NULL, folds = NULL)
```

Arguments

fitfun	A function of one argument (a training data frame) returning a fitted underdisp model, e.g. function(d) hurdle_cpb(y ~ x, d, participation = ~ z, fe = "unit", se = "none").
data	The full data frame.
k	Number of folds (default 5).
kmax	Highest count to evaluate; if NULL, taken from a fit on the full data (one extra fit).
folds	Optional integer vector of length nrow(data) giving a fixed fold assignment (so two models can be scored on identical folds). If NULL, folds are drawn at random.

Value

An object of class "cv_score": a list with the mean held-out logscore and rps, the per-observation vectors logscore_i/rps_i, and the folds used.

See Also

score()

Examples

```
set.seed(1)
d <- data.frame(y = rcpb(500, lambda = 3, alpha = 0.5))
cv <- cv_score(function(tr) cpb(y ~ 1, tr, truncated = FALSE, se = "none"), d, k = 5)
cv$logscore
```

---

first_difference *King-style first difference for a CPB fit*

Description

The effect on a quantity of interest of moving one covariate `from` one value `to` another, holding the other covariates at their means, with a bootstrap percentile interval (Tomz, Wittenberg & King style, using the model's bootstrap draws).

Usage

```
first_difference(object, ...)

## S3 method for class 'cpb'
first_difference(
  object,
  variable,
  from,
  to,
  quantity = c("mean", "ceiling", "prob"),
  y = NULL,
  level = 0.95,
  ...
)
```

Arguments

`object`	A `"cpb"` object fit with `se = "bootstrap"`.
`...`	Further arguments passed to methods; unknown arguments error.
`variable`	Name of a model-matrix column to vary.
`from, to`	The two values of `variable` to contrast.
`quantity`	`"mean"` (E(Y)), `"ceiling"` (lambda/(1-alpha)), or `"prob"` (P(Y = y)).
`y`	The count value for `quantity = "prob"`.
`level`	Confidence level (default 0.95).

Value

A "ud_fd" data frame – the package-wide first-difference contract (columns component, from, to, diff, lower, upper, method) – with one row for the requested quantity and a bootstrap percentile interval on the difference (method = "bootstrap (stored)"). Every first_difference() method in the package returns this same shape.

Examples

```
set.seed(1); x <- rnorm(400)
N <- pmax(round(exp(1.6 + 0.5 * x) / 0.5), 1); y <- rbinom(400, N, 0.5)
fit <- cpb(y ~ x, data = data.frame(y = y, x = x)[y > 0, ], se = "bootstrap", B = 200)
first_difference(fit, "x", from = -1, to = 1, quantity = "mean")
```

---

first_difference.gec *First differences for the matched count families*

Description

The discrete-change effect on the expected count E(Y) as variable moves from from to to, holding the other covariates at their sample means. For count_reg this is a single number with a delta-method interval; for the two-part models it is decomposed exactly into extensive (participation / non-structural-zero) and intensive (count) channels that sum to the total.

Usage

```
## S3 method for class 'gec'
first_difference(object, variable, from, to, level = 0.95, ...)

## S3 method for class 'gec_fe'
first_difference(object, variable, from, to, level = 0.95, ...)

## S3 method for class 'cpb_fe'
first_difference(object, variable, from, to, level = 0.95, ...)

## S3 method for class 'hurdle_gec'
first_difference(
  object,
  variable,
  from,
  to,
  level = 0.95,
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)

## S3 method for class 'zi_gec'
```

```
first_difference(
  object,
  variable,
  from,
  to,
  level = 0.95,
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)

## S3 method for class 'count_reg'
first_difference(object, variable, from, to, level = 0.95, ...)

## S3 method for class 'hurdle_count'
first_difference(
  object,
  variable,
  from,
  to,
  level = 0.95,
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)

## S3 method for class 'zi_count'
first_difference(
  object,
  variable,
  from,
  to,
  level = 0.95,
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)
```

Arguments

object	A fitted count_reg, gec, gec_fe, cpb_fe, hurdle_count, zi_count, hurdle_gec, or zi_gec model.
variable	Name of the covariate to change (must be in the model).
from, to	The two values of variable.
level	Confidence level for the delta-method intervals (all methods on this page that can compute one).
...	Unused; unknown arguments error.
stage	For the two-part models, which equation(s) the change is applied to. "both" (default) moves the variable wherever it appears; "intensity"/"count" or "participation"/"zero" moves it in only that equation, holding it at the reference in the other. For a variable in only one equation all options coincide; for a variable in both, "both"

gives the total effect and the single-stage options the partial effect through that margin. All component rows are always returned.

Value

A "ud_fd" data frame – the package-wide first-difference contract shared by every `first_difference()` method: columns `component`, `from`, `to`, `diff`, `lower`, `upper`, `method`. Single-equation fits return one row (`component = "mean"`); two-part fits return one row per margin level (binary-stage probability, count-stage mean, marginal mean). `method` records the uncertainty source per row (`"delta"`, a bootstrap label, or `"none"` with `NA` bounds).

---

first_difference.hurdle_cpb

Extensive/intensive first-difference decomposition for a hurdle- or ZI-CPB

Description

The effect of moving `variable` `from` one value `to` another, decomposed into the change in the participation (extensive) margin, the intensity (intensive) margin, and the marginal expected count, holding other covariates at their means. This is the quantity the single-equation defaults cannot separate.

Usage

```
## S3 method for class 'hurdle_cpb'
first_difference(
  object,
  variable,
  from,
  to,
  B = 0,
  level = 0.95,
  data = NULL,
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)

## S3 method for class 'zi_cpb'
first_difference(
  object,
  variable,
  from,
  to,
  B = 0,
  level = 0.95,
  data = NULL,
```

```
  stage = c("both", "participation", "intensity", "zero", "count"),
  ...
)
```

Arguments

object	A "hurdle_cpb" or "zi_cpb" object.
variable	Name of a model-matrix column to vary (in either margin).
from, to	The two values to contrast.
B	Bootstrap replicates for percentile intervals; 0 (default) returns point estimates only. For zi_cpb the bootstrap is slow (each replicate refits the mixture).
level	Confidence level.
data	The original data frame; required when B > 0.
stage	Which equation(s) the covariate moves in: "both" (default), the binary stage only ("participation"/"zero"), or the count stage only ("intensity"/"count"), holding the covariate at its reference in the other equation. All three component rows are always returned.
...	Unused; unknown arguments error.

Value

A "ud_fd" data frame (the package-wide first-difference contract): columns component, from, to, diff, lower, upper, method, one row per margin level.

---

fitted.underdisp *Fitted means for the two-part and fixed-effects classes*

Description

Completes stats::fitted() across the family, returning the same quantity as predict(object, type = "response") on the estimation data: the mean for the single-equation fits and the marginal expected count (zeros included) for the two-part fits. gec_fe inherits fitted.gec.

Usage

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

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

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

## S3 method for class 'hurdle_count'
```

```
fitted(object, ...)

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

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

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

Arguments

`object`	A fitted model from this package.
`...`	Unused.

Value

A numeric vector, one element per estimation observation.

`gec` *Generalized event count (Katz-family) regression*

Description

Fits King's generalized event count model, a Katz-family count regression whose single dispersion parameter `delta` (the variance-to-mean ratio) is estimated freely and spans underdispersion (`delta` < 1, a finite-support member; the continuous parameter binomial is this cell), equidispersion (`delta` = 1, Poisson), and overdispersion (`delta` > 1, the negative binomial). Unlike `cpb()`, which fixes the direction of dispersion to under, `gec()` lets the data choose. The Katz recursion delivers exact first and second moments—the Winkelmann–Signorino–King correction realized directly—and the likelihood is evaluated in C++.

Usage

```
gec(
  formula,
  data,
  truncated = FALSE,
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  offset = NULL,
  max.support = 500,
  maxit = 20000,
  reltol = 1e-08
)
```

Arguments

formula	A model formula.
data	A data frame.
truncated	Logical; if TRUE, fit the zero-truncated GEC (all Y >= 1), the intensity model of hurdle_gec().
se	Coefficient inference: "none" (default; fast) or "bootstrap".
B	Bootstrap resamples when se = "bootstrap".
cluster	Optional cluster identifier (a column name in data or a vector) for a cluster/block bootstrap; see cpb().
offset	Optional offset on the log-mean scale (an exposure): a numeric vector or the name of a column in data.
max.support	Guard on the maximum evaluated support.
maxit, reltol	Optimizer controls.

Value

An object of class "gec" with coefficients, delta (the estimated dispersion), loglik, bootstrap standard errors, and bookkeeping.

See Also

cpb(), compare_dispersion()

Examples

```
set.seed(1); x <- rnorm(400)
y <- rpois(400, exp(1 + 0.5 * x))
gec(y ~ x, data = data.frame(y = y, x = x), se = "none")
```

---

gec-distribution *Generalized event count (Katz) distribution functions*

Description

Density, distribution, quantile, and random generation for the King (1989) generalized event count model with rate lambda (the mean) and dispersion delta (the variance-to-mean ratio: < 1 underdispersed, = 1 Poisson, > 1 overdispersed). Consistent with the estimator gec().

Usage

```
dgec(x, lambda, delta, max.support = 500, log = FALSE)

pgec(q, lambda, delta, max.support = 500, lower.tail = TRUE, log.p = FALSE)

qgec(p, lambda, delta, max.support = 500, lower.tail = TRUE, log.p = FALSE)

rgec(n, lambda, delta, max.support = 500)
```

Arguments

x, q	Vector of quantiles (non-negative integers).
lambda	Rate/mean parameter (scalar or vector, recycled).
delta	Dispersion (variance-to-mean ratio).
max.support	Guard on the evaluated support.
log, log.p	Return log probabilities.
lower.tail	If TRUE (default), $P(X \leq x)$.
p	Vector of probabilities.
n	Number of draws.

Value

dgec a density, pgec a CDF, qgec a quantile, rgec a numeric vector of count draws.

See Also

gec()

Examples

```
dgec(0:5, lambda = 3, delta = 0.7)
var(rgec(2000, lambda = 3, delta = 0.7)) / mean(rgec(2000, lambda = 3, delta = 0.7))
```

---

gec_fe *GEC (Katz-family) regression with high-dimensional unit fixed effects*

Description

Fits gec() with a full set of unit fixed effects, concentrating (profiling) out the unit intercepts by a one-dimensional inner maximization per unit, so the outer optimizer handles only the covariate coefficients and the free dispersion parameter delta. This is the cpb_fe() concentration generalized to the whole Katz family: the panel can be under-, equi-, or overdispersed and the direction is estimated, not presumed.

Usage

```
gec_fe(
  formula,
  data,
  fe,
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  offset = NULL,
  max.support = 500L,
```

```
  inner_it = 30L,
  maxit = 3000L,
  reltol = 1e-07,
  bias_correct = c("none", "jackknife")
)
```

Arguments

formula	A model formula for the covariates only (no unit factor, no intercept; the fixed effects absorb it).
data	A data frame.
fe	Name of the column holding the unit identifier.
se	Inference for the covariate coefficients: "none" (default) or "bootstrap", a pairs/cluster bootstrap over units.
B	Bootstrap resamples when se = "bootstrap".
cluster	Cluster for the bootstrap; NULL (default) resamples the fixed-effects units. See cpb_fe().
offset	Optional offset on the log-mean scale (an exposure): a numeric vector or the name of a column in data.
max.support	Guard on the maximum evaluated support.
inner_it	Golden-section iterations for each unit's inner maximization.
maxit, reltol	Outer optimizer controls.
bias_correct	"none" (default) or "jackknife", the split-panel jackknife correction for the 1/T incidental-parameters bias (see Details).

Details

Limitation: unlike cpb_fe(), gec_fe() has no truncated argument – a concentrated zero-truncated GEC is not currently implemented. For a zero-truncated GEC with fixed effects, enter the unit factor as dummies in gec()'s formula (feasible for moderate unit counts).

Short panels: delta and the fixed effects carry the incidental-parameters bias of nonlinear fixed-effects estimation, of order 1/T; treat delta cautiously below about T = 30. The covariate coefficients are not materially affected. bias_correct = "jackknife" removes the leading 1/T term by the split-panel jackknife exactly as in cpb_fe(): refit on each unit's temporal halves and report 2 * full - mean(halves), with the unit effects and fitted values re-concentrated at the corrected parameters and logLik/AIC kept at the maximum-likelihood fit (uncorrected estimates in $uncorrected). The same time-homogeneity validity gate applies: on panels where the two halves do not estimate a common parameter the correction is REFUSED with a warning and the maximum-likelihood fit is returned (see cpb_fe(), Details).

Value

An object of class c("gec_fe", "gec").

See Also

gec(), cpb_fe()

Examples

```
set.seed(5)
d <- do.call(rbind, lapply(1:30, function(i) {
  x <- rnorm(10); N <- pmax(round(exp(1 + rnorm(1, 0, 0.4) + 0.3 * x) / 0.5), 1)
  data.frame(unit = i, x = x, y = rbinom(10, N, 0.5))
}))
gec_fe(y ~ x, data = d, fe = "unit")   # delta ~ 0.5: underdispersed
```

`glance.cpb` *Glance at a CPB fit (broom method)*

Description

Glance at a CPB fit (broom method)

Usage

```
## S3 method for class 'cpb'
glance(x, ...)

## S3 method for class 'cpb_fe'
glance(x, ...)

## S3 method for class 'hurdle_cpb'
glance(x, ...)

## S3 method for class 'zi_cpb'
glance(x, ...)

## S3 method for class 'gec'
glance(x, ...)

## S3 method for class 'hurdle_gec'
glance(x, ...)

## S3 method for class 'zi_gec'
glance(x, ...)

## S3 method for class 'count_reg'
glance(x, ...)

## S3 method for class 'zi_count'
glance(x, ...)

## S3 method for class 'hurdle_count'
glance(x, ...)
```

Arguments

x	A "cpb" object.
...	Unused.

Value

A one-row data frame of fit statistics.

---

hurdle_count *Hurdle Poisson / negative-binomial regression (matched CPB baseline)*

Description

Fits a participation logit joined to a zero-truncated Poisson or negative-binomial intensity, the count analogue of hurdle_cpb(). Returned as a "hurdle_count" object that compare_models() and score() accept, so a hurdle-CPB and a hurdle-NB can be compared on one footing.

Usage

```
hurdle_count(
  formula,
  data,
  family = c("poisson", "negbin", "compois"),
  participation = NULL,
  fe = NULL,
  part_fe = NULL,
  link = c("logit", "probit", "cloglog"),
  offset = NULL,
  se = c("analytic", "robust", "cluster", "none"),
  cluster = NULL,
  ...
)
```

Arguments

formula	Intensity formula (y ~ x).
data	A data frame.
family	"poisson", "negbin", or "compois" for the intensity; see count_reg() for the COM-Poisson parameterization.
participation	Optional one-sided formula for the participation model; defaults to the intensity right-hand side.
fe, part_fe	Optional fixed-effects column names for the intensity and participation models (entered as factor dummies).

link	Link for the participation model: "logit" (default), "probit", or "cloglog". Positive participation coefficients raise the probability of a *positive count* (participation).
offset	Optional offset for the intensity, on the linear-predictor (log) scale (log-mean for Poisson/NB, log-rate for COM-Poisson): a numeric vector or the name of a column in data.
se, cluster	Standard-error type and optional cluster for the intensity; see count_reg().
...	Unused.

Value

An object of class "hurdle_count".

See Also

hurdle_cpb(), zi_count(), compare_models()

Examples

```
set.seed(1); n <- 300; x <- rnorm(n); z <- rnorm(n)
y <- ifelse(rbinom(n, 1, plogis(0.4 + 0.8 * z)) == 1, rpois(n, exp(1 + 0.3 * x)) + 1L, 0L)
hurdle_count(y ~ x, data.frame(y = y, x = x, z = z), family = "poisson",
             participation = ~ z, se = "none")
```

---

hurdle_cpb *Hurdle continuous parameter binomial regression*

Description

Fits a participation (hurdle) model joined to a zero-truncated CPB for the positive counts. Because the hurdle log-likelihood factorizes, the two parts are fit separately: a logistic regression of participation on all units, and a zero-truncated CPB (cpb()) on the positive counts. The result is the natural model for a bounded, underdispersed participation process — most units at zero, participants carrying a tight count.

Usage

```
hurdle_cpb(
  formula,
  data,
  participation = NULL,
  fe = NULL,
  part_fe = NULL,
  link = c("logit", "probit", "cloglog"),
  offset = NULL,
  cluster = NULL,
  se = c("none", "bootstrap"),
```

```
  B = 500,
  ...
)
```

Arguments

formula	Intensity model formula (y ~ x).
data	A data frame.
participation	Optional one-sided formula (~ z) for the participation (hurdle) model; defaults to the intensity model's right-hand side.
fe	Optional column name for unit fixed effects in the intensity model, absorbed by a concentrated likelihood (cpb_fe()). This is how the within-unit underdispersion is recovered.
part_fe	Optional column name for fixed effects in the participation model (added as factors). Units with no within-unit variation in participation are uninformative for a fixed-effects logit.
link	Link for the participation model: "logit" (default), "probit", or "cloglog". Positive participation coefficients raise the probability of a positive count.
offset	Optional offset on the log-mean scale for the intensity (an exposure): a numeric vector or the name of a column in data.
cluster	Optional cluster identifier (a column name in data or a vector) for cluster-robust bootstrap inference on the intensity coefficients; see cpb(). Cluster-robust errors for the participation logit are available directly via sandwich::vcovCL(fit$participation, cluster = ...).
se	Inference for the intensity coefficients: "none" or "bootstrap".
B	Bootstrap replicates when se = "bootstrap".
...	Passed to cpb() or cpb_fe().

Value

An object of class "hurdle_cpb" with elements participation (a glm), intensity (a cpb or cpb_fe), and bookkeeping.

Examples

```
set.seed(1)
n <- 800; x <- rnorm(n); z <- rnorm(n)
y <- rhurdle_cpb(n, lambda = exp(1.3 + 0.5 * x), alpha = 0.5,
                 p = plogis(-0.2 + 0.8 * z))
fit <- hurdle_cpb(y ~ x, data = data.frame(y = y, x = x, z = z),
                  participation = ~ z)
fit
```

hurdle_gec *Hurdle GEC (Katz-family) regression*

Description

Joins a participation (hurdle) logit to a zero-truncated GEC intensity on the positive counts. Unlike `hurdle_cpb()`, whose intensity is fixed to the underdispersed CPB, the GEC intensity estimates its own dispersion direction.

Usage

```
hurdle_gec(
  formula,
  data,
  participation = NULL,
  part_fe = NULL,
  link = c("logit", "probit", "cloglog"),
  offset = NULL,
  cluster = NULL,
  se = c("none", "bootstrap"),
  B = 500,
  max.support = 500
)
```

Arguments

`formula`	Intensity formula, `y ~ x`.
`data`	A data frame.
`participation`	One-sided formula for the participation logit; defaults to the intensity's right-hand side.
`part_fe`	Optional column name for fixed effects in the participation model (added as factor dummies).
`link`	Link for the participation model: `"logit"` (default), `"probit"`, or `"cloglog"`.
`offset`	Optional offset (log scale) for the intensity: a numeric vector or a column name in `data`.
`cluster`	Optional cluster (column name or vector) for the intensity's cluster bootstrap.
`se`	Intensity inference: `"none"` (default) or `"bootstrap"`.
`B`	Bootstrap resamples.
`max.support`	Guard on the maximum evaluated support.

Value

An object of class `"hurdle_gec"`.

Limitation

Unlike hurdle_cpb(), hurdle_gec() has no fe argument for a concentrated fixed-effects intensity (no concentrated zero-truncated GEC is implemented). For within-unit intensities, enter the unit factor as dummies in formula (feasible for moderate unit counts), or use hurdle_cpb().

See Also

hurdle_cpb(), gec(), zi_gec()

Examples

```
set.seed(3); n <- 300; x <- rnorm(n); z <- rnorm(n)
y <- ifelse(rbinom(n, 1, plogis(0.3 + 0.8 * z)) == 1, rpois(n, exp(1 + 0.3 * x)) + 1L, 0L)
hurdle_gec(y ~ x, data.frame(y = y, x = x, z = z), participation = ~ z)
```

---

implied_ceiling *Implied ceiling with a profile-likelihood interval*

Description

Returns the observation- (or profile-) specific ceiling lambda/(1-alpha), with an interval propagating the profile-likelihood uncertainty in alpha at the fitted mean. (Coefficient uncertainty in lambda is not propagated here; use first_difference() with quantity = "ceiling" for a fully bootstrapped contrast.)

Usage

```
implied_ceiling(object, ...)

## S3 method for class 'cpb'
implied_ceiling(object, newdata = NULL, level = 0.95, ...)
```

Arguments

object	A "cpb" object.
...	Further arguments passed to methods.
newdata	Optional covariate profiles.
level	Confidence level (default 0.95).

Value

A data frame with lambda, ceiling, and lower/upper bounds.

Examples

```
set.seed(6); x <- rnorm(300)
N <- pmax(round(exp(1.5 + 0.4 * x) / 0.5), 1); y <- rbinom(300, N, 0.5)
fit <- cpb(y ~ x, data.frame(y = y, x = x)[y > 0, ], se = "none")
implied_ceiling(fit, newdata = data.frame(x = c(-1, 0, 1)))
```

`irr` *Incidence rate ratios for a CPB fit*

Description

Incidence rate ratios for a CPB fit

Usage

```
irr(object, ...)

## S3 method for class 'cpb'
irr(object, level = 0.95, ...)
```

Arguments

`object`	A `"cpb"` object fit with `se = "bootstrap"`.
`...`	Further arguments passed to methods.
`level`	Confidence level (default 0.95).

Value

A `"ud_irr"` data frame – the package-wide rate-ratio contract (columns `term`, `equation`, `ratio`, `estimate`, `lower`, `upper`, `method`) shared by every `irr()` method; here with bootstrap percentile intervals from the stored draws.

Examples

```
set.seed(8); x <- rnorm(300)
N <- pmax(round(exp(1.5 + 0.4 * x) / 0.5), 1); y <- rbinom(300, N, 0.5)
fit <- cpb(y ~ x, data.frame(y = y, x = x)[y > 0, ], se = "bootstrap", B = 100)
irr(fit)
```

`irr.count_reg` *Rate and odds ratios for the matched count families*

Description

Rate ratios (`exp` of the count/intensity coefficients, column `IRR`) and, for the two-part models, odds ratios of the binary stage (column `OR`), by equation. Element names match the CPB family (`intensity` for the count component; `participation` for the hurdle stage, `zero` for the inflation stage).

Usage

```
## S3 method for class 'count_reg'
irr(object, level = 0.95, ...)

## S3 method for class 'gec'
irr(object, level = 0.95, ...)

## S3 method for class 'gec_fe'
irr(object, level = 0.95, ...)

## S3 method for class 'cpb_fe'
irr(object, level = 0.95, ...)

## S3 method for class 'hurdle_gec'
irr(object, level = 0.95, ...)

## S3 method for class 'zi_gec'
irr(object, level = 0.95, ...)

## S3 method for class 'hurdle_count'
irr(object, level = 0.95, ...)

## S3 method for class 'zi_count'
irr(object, level = 0.95, ...)
```

Arguments

`object`	A fitted model from this package.
`level`	Confidence level.
`...`	Unused.

Value

A `"ud_irr"` data frame – the package-wide ratio contract shared by every `irr()` method: columns `term`, `equation` (`"count"` or `"binary"`), `ratio` (`"IRR"` or `"OR"`), `estimate`, `lower`, `upper`, and

`method` (the interval source; `"none"` with `NA` bounds when no covariance is available). Two-part models stack both equations.

`mundlak` *Mundlak (correlated random effects) device*

Description

Augments a data frame with the unit-level means of the time-varying numeric covariates in `formula`, and returns the augmented formula and data. Fitting any of the package's estimators on the result implements the Mundlak / correlated-random-effects specification: the coefficients on the original covariates recover the within-unit (fixed-effects-consistent) effects, while the coefficients on the unit means capture (and test) the correlation between the covariates and the unit effect. It is a lighter, between-variation-preserving alternative to full fixed effects, and it is model-agnostic — the same device feeds `cpb()`, `gec()`, or `count_reg()`.

Usage

```
mundlak(formula, data, unit, suffix = "_mean")
```

Arguments

`formula`	A model formula.
`data`	A data frame.
`unit`	Column name identifying the panel unit.
`suffix`	Suffix for the added unit-mean columns (default `"_mean"`).

Value

A list with `formula` (the augmented formula), `data` (the augmented data frame), and `added` (the names of the unit-mean columns).

See Also

`count_reg()`, `cpb()`

Examples

```
set.seed(1)
d <- data.frame(y = rpois(200, 3), x = rnorm(200), unit = factor(rep(1:20, each = 10)))
m <- mundlak(y ~ x, d, unit = "unit")
fit <- count_reg(m$formula, m$data, family = "poisson")
```

`peacekeeping` *UN peacekeeping contributions, state-years*

Description

A state-year panel of contributions to United Nations peacekeeping operations, the archetype of a bounded, underdispersed participation count: most state-years contribute to no operation, and the states that contribute carry a tight count held in a narrow band by a turning-over roster. About 42\ hurdle-CPB.

Usage

```
data(peacekeeping)
```

Format

A data frame with 4,448 rows and 9 variables:

iso3 ISO3 country code (the panel unit).

year Calendar year, 1990–2024.

contributions Number of distinct UN peacekeeping operations the state contributes personnel to in that year (the count outcome).

democracy Electoral-democracy index.

lgdppc Log GDP per capita.

lpop Log population.

milper Military personnel, thousands.

majorpower Major-power indicator (0/1).

region World region.

Source

Contribution counts aggregated from the International Peace Institute Providing for Peacekeeping database (https://www.providingforpeacekeeping.org); covariates from V-Dem and the Correlates of War National Material Capabilities data. Counts and public covariates only; prepared for illustration.

Examples

```
data(peacekeeping)
table(peacekeeping$contributions == 0)

fit <- hurdle_cpb(contributions ~ lgdppc + milper, data = peacekeeping,
                  participation = ~ democracy + majorpower, fe = "iso3")
fit
```

`pit_hist` *Non-randomized PIT histogram for a fitted count model*

Description

Draws the non-randomized probability integral transform histogram of Czado, Gneiting, and Held (2009). A well-calibrated model yields a flat histogram at height one (the reference line); a U shape indicates under-dispersion in the predictive distribution and a hump indicates over-dispersion.

Usage

```
pit_hist(
  fit,
  bins = 10,
  main = "PIT histogram",
  xlab = "PIT",
  ylab = "Relative frequency",
  ...
)
```

Arguments

`fit`	A `"cpb"` or `"hurdle_cpb"` object.
`bins`	Number of histogram bins.
`main`, `xlab`, `ylab`	Plot labels.
`...`	Passed to `graphics::barplot()`.

Value

Invisibly, the vector of bin heights (normalized so that a calibrated model gives heights near one).

Examples

```
set.seed(1)
y <- rcpb(400, lambda = 3, alpha = 0.5)
fit <- cpb(y ~ 1, data = data.frame(y = y), truncated = FALSE, se = "none")
pit_hist(fit)
```

predict.count_reg *Predictions from a matched Poisson/NB/COM-Poisson fit*

Description

Predictions from a matched Poisson/NB/COM-Poisson fit

Usage

```
## S3 method for class 'count_reg'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "prob"),
  at = NULL,
  offset = NULL,
  ...
)
```

Arguments

object	A "count_reg" object.
newdata	Optional data frame of covariate profiles.
type	"response" (the mean E(Y)), "link" (the linear predictor), or "prob" (P(Y = at); for a zero-truncated fit this is P(Y = at | Y > 0)).
at	Count value for type = "prob".
offset	Optional offset (log scale) for newdata: a numeric vector or a column name in newdata.
...	Unused.

Value

A numeric vector.

predict.cpb *Predictions from a CPB fit*

Description

Predictions from a CPB fit

Usage

```
## S3 method for class 'cpb'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "ceiling", "prob"),
  at = NULL,
  offset = NULL,
  ...
)
```

Arguments

object	A "cpb" object.
newdata	Optional data frame of new covariate profiles; if omitted, the fitted data are used.
type	One of "response" (the mean lambda), "link" (the linear predictor), "ceiling" (the implied ceiling lambda/(1-alpha)), or "prob" (the probability that Y equals at).
at	For type = "prob", the count value(s) y whose probability is returned (length 1, or one per row of the prediction data).
offset	Optional offset (log scale) for the newdata branch: a numeric vector or a column name in newdata. For newdata = NULL the fit's own offset is used.
...	Unused.

Value

A numeric vector.

Examples

```
set.seed(1); x <- rnorm(300)
N <- pmax(round(exp(1.6 + 0.5 * x) / 0.5), 1); y <- rbinom(300, N, 0.5)
fit <- cpb(y ~ x, data = data.frame(y = y, x = x)[y > 0, ], se = "none")
predict(fit, newdata = data.frame(x = 0), type = "ceiling")
predict(fit, newdata = data.frame(x = c(-1, 1)), type = "prob", at = 5)
```

---

predict.cpb_fe *Predictions from a fixed-effects CPB fit*

Description

Predictions from a fixed-effects CPB fit

Usage

```
## S3 method for class 'cpb_fe'
predict(object, newdata = NULL, type = c("response", "link", "ceiling"), ...)
```

Arguments

object	A "cpb_fe" object.
newdata	Optional covariate profiles. With newdata, predictions use the average unit (the mean fixed effect), since the panel's own unit effects do not apply to new rows.
type	"response" (the mean), "link" (the log mean), or "ceiling".
...	Unused.

Value

A numeric vector.

---

predict.gec *Predictions from a generalized event count fit*

Description

Predictions from a generalized event count fit

Usage

```
## S3 method for class 'gec'
predict(
  object,
  newdata = NULL,
  type = c("response", "link", "prob"),
  at = NULL,
  offset = NULL,
  ...
)
```

Arguments

object	A "gec" object.
newdata	Optional covariate profiles.
type	"response" (the mean), "link" (the log mean), or "prob" (the probability of the count at).
at	Count value(s) for type = "prob".
offset	Optional offset (log scale) for the count component when predicting on newdata; a numeric vector or a column name in newdata.
...	Unused.

Value

A numeric vector.

predict.hurdle_count *Predict from a hurdle Poisson/NB fit*

Description

Predict from a hurdle Poisson/NB fit

Usage

```
## S3 method for class 'hurdle_count'
predict(
  object,
  newdata = NULL,
  type = c("response", "participation", "intensity"),
  offset = NULL,
  ...
)
```

Arguments

object	A "hurdle_count" object.
newdata	Optional data frame of covariate profiles.
type	"response" (marginal E(Y), zeros included), "participation" ($P(Y > 0)$), or "intensity" ($E(Y \mid Y > 0)$).
offset	Optional offset (log scale) for the intensity when predicting on newdata; a numeric vector or a column name in newdata.
...	Unused.

Value

A numeric vector.

predict.hurdle_cpb *Predict from a hurdle-CPB fit*

Description

Predict from a hurdle-CPB fit

Usage

```
## S3 method for class 'hurdle_cpb'
predict(
  object,
  newdata = NULL,
  type = c("response", "participation", "intensity"),
  ...
)
```

Arguments

object	A "hurdle_cpb" object.
newdata	Data frame of covariate profiles (must contain both the intensity and participation covariates).
type	"response" for the marginal expected count E(Y) (zeros included), "participation" for P(Y > 0), or "intensity" for E(Y | Y > 0).
...	Unused.

Value

A numeric vector.

---

predict.hurdle_gec *Predict from a hurdle-GEC fit*

Description

Predict from a hurdle-GEC fit

Usage

```
## S3 method for class 'hurdle_gec'
predict(
  object,
  newdata = NULL,
  type = c("response", "participation", "intensity"),
  ...
)
```

Arguments

object	A "hurdle_gec" object.
newdata	Optional covariate profiles (intensity and participation).
type	"response" (marginal E(Y)), "participation" (P(Y>0)), or "intensity" (E(Y | Y>0)).
...	Unused.

Value

A numeric vector.

`predict.zi_count` *Predict from a zero-inflated Poisson/NB fit*

Description

Predict from a zero-inflated Poisson/NB fit

Usage

```
## S3 method for class 'zi_count'
predict(
  object,
  newdata = NULL,
  type = c("response", "count", "zero", "intensity"),
  offset = NULL,
  ...
)
```

Arguments

`object`	A `"zi_count"` object.
`newdata`	Optional data frame of covariate profiles.
`type`	`"response"` (marginal E(Y)), `"count"` (equivalently `"intensity"`; the count component's mean E(Y) of that component), or `"zero"` (structural-zero probability). `"intensity"` is accepted as a synonym for `"count"` for consistency with `predict.zi_cpb()`.
`offset`	Optional offset (log scale) for the count component when predicting on `newdata`; a numeric vector or a column name in `newdata`.
`...`	Unused.

Value

A numeric vector.

predict.zi_cpb *Predict from a zi_cpb fit*

Description

Predict from a zi_cpb fit

Usage

```
## S3 method for class 'zi_cpb'
predict(object, newdata = NULL, type = c("response", "zero", "intensity"), ...)
```

Arguments

object	A "zi_cpb" object.
newdata	Data frame of covariate profiles.
type	"response" for the marginal expected count E(Y), "zero" for the structural-zero probability, or "intensity" for the CPB mean.
...	Unused.

Value

A numeric vector.

predict.zi_gec *Predict from a zi_gec fit*

Description

Predict from a zi_gec fit

Usage

```
## S3 method for class 'zi_gec'
predict(object, newdata = NULL, type = c("response", "zero", "intensity"), ...)
```

Arguments

object	A "zi_gec" object.
newdata	Optional covariate profiles.
type	"response" (marginal E(Y)), "zero" (structural-zero probability), or "intensity" (the count mean E of the GEC component).
...	Unused.

Value

A numeric vector.

rcpb *Simulate from the continuous parameter binomial*

Description

Draws counts from the continuous parameter binomial (CPB) or its zero-truncated variant.

Usage

```
rcpb(n, lambda, alpha, truncated = FALSE)
```

Arguments

n	Number of draws (recycled against lambda).
lambda	Mean parameter; a scalar or a length-n vector.
alpha	Shape/dispersion parameter in (0, 1).
truncated	If TRUE, draw from the zero-truncated CPB.

Value

An integer vector of counts.

Examples

```
set.seed(1)
table(rcpb(1000, lambda = 3, alpha = 0.5))
```

rhurdle_cpb *Simulate from the hurdle continuous parameter binomial*

Description

Simulate from the hurdle continuous parameter binomial

Usage

```
rhurdle_cpb(n, lambda, alpha, p)
```

Arguments

n	Number of units.
lambda	Intensity mean for participants; scalar or length-n.
alpha	CPB shape parameter in (0, 1).
p	Participation probability; scalar or length-n.

Value

An integer vector: 0 for nonparticipants, a zero-truncated CPB draw otherwise.

Examples

```
set.seed(1)
y <- rhurdle_cpb(1000, lambda = 3, alpha = 0.5, p = 0.6)
mean(y == 0)
```

`rootogram` *Hanging rootogram for a fitted count model*

Description

Draws a Tukey hanging rootogram: bars for the observed frequencies hang from the curve of expected frequencies, both on the square-root scale. Bars that hang below the zero line mark counts the model under-predicts; bars that stop short mark counts it over-predicts.

Usage

```
rootogram(
  fit,
  kmax = NULL,
  main = "Rootogram",
  xlab = "Count",
  ylab = "sqrt(frequency)",
  ...
)
```

Arguments

`fit`	A `"cpb"` or `"hurdle_cpb"` object.
`kmax`	Highest count to display; defaults to the maximum observed count.
`main`, `xlab`, `ylab`	Plot labels.
`...`	Passed to `graphics::plot()`.

Value

Invisibly, a data frame of `count`, `observed`, and `expected` frequencies.

Examples

```
set.seed(1)
y <- rcpb(400, lambda = 3, alpha = 0.5)
fit <- cpb(y ~ 1, data = data.frame(y = y), truncated = FALSE, se = "none")
rootogram(fit)
```

rzicpb *Simulate from the zero-inflated continuous parameter binomial*

Description

Simulate from the zero-inflated continuous parameter binomial

Usage

```
rzicpb(n, lambda, alpha, pi)
```

Arguments

n	Number of units.
lambda	Intensity mean; scalar or length-n.
alpha	CPB shape parameter in (0, 1).
pi	Structural-zero probability; scalar or length-n.

Value

An integer vector: a structural zero with probability pi, otherwise an (untruncated) CPB draw, which may itself be zero.

Examples

```
set.seed(1)
y <- rzicpb(1000, lambda = 3, alpha = 0.5, pi = 0.3)
mean(y == 0)
```

score *Proper scoring rules for a fitted count model*

Description

Computes the mean logarithmic score and the ranked probability score (RPS) of a fitted model's predicted distribution against observed counts. Lower is better for both, and both are proper, so they are the natural way to compare the calibration of competing count models.

Usage

```
score(fit, newdata = NULL, kmax = NULL)
```

Arguments

fit	A fitted underdisp count model (cpb, cpb_fe, hurdle_cpb, zi_cpb, count_reg, hurdle_count, zi_count, gec, gec_fe, hurdle_gec, or zi_gec).
newdata	Optional data frame of held-out observations (including the response). When supplied, the fitted model's predicted distribution is evaluated on these rows; fixed-effect units unseen in training fall back to the mean fixed effect. Offsets are assumed absent on newdata.
kmax	Highest count to evaluate; defaults to the maximum observed count in the fitting data.

Details

By default the scores are computed **in sample** (against the data the model was fit to). Supply newdata to score a fitted model on **held-out** observations, or use cv_score() for a cross-validated score; an in-sample log score equals -logLik/n and does not penalize model complexity, so for comparing models of different size the held-out or cross-validated score is the honest criterion.

Value

A named numeric vector c(logscore, rps).

See Also

cv_score()

Examples

```
set.seed(1)
y <- rcpb(400, lambda = 3, alpha = 0.5)
fit <- cpb(y ~ 1, data = data.frame(y = y), truncated = FALSE, se = "none")
score(fit)
```

---

simulate.underdisp *Simulate responses from a fitted underdisp model*

Description

Draws nsim replicate response vectors from the fitted model, at the estimated parameters and the estimation data, in the format of stats::simulate(). Available for every model class in the package: cpb, cpb_fe, gec, gec_fe (through inheritance), count_reg, hurdle_count, zi_count, hurdle_cpb, hurdle_gec, zi_cpb, and zi_gec. Two-part models first draw the binary stage (participation or structural zero), then the count stage from its own distribution, so the replicates carry the model's full zero structure.

Usage

```
## S3 method for class 'cpb'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'cpb_fe'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'gec'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'count_reg'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'hurdle_cpb'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'hurdle_gec'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'hurdle_count'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'zi_cpb'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'zi_gec'
simulate(object, nsim = 1, seed = NULL, ...)

## S3 method for class 'zi_count'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

`object`	A fitted model from this package.
`nsim`	Number of replicate response vectors.
`seed`	Optional seed, handled as in `stats::simulate()`: the caller's RNG state is restored on exit when a seed is supplied.
`...`	Unused.

Details

The main consumer is simulated-residual diagnostics: `DHARMa::createDHARMa(simulatedResponse = as.matrix(simulate(fit, 250)), observedResponse = y, fittedPredictedResponse = fitted(fit), integerResponse = TRUE)` works for any fit in the family.

Value

A data frame with `nsim` integer columns, one row per observation, with a `"seed"` attribute.

Examples

```
set.seed(1); x <- rnorm(200)
N <- pmax(round(exp(1.4 + 0.4 * x) / 0.5), 1); y <- rbinom(200, N, 0.5)
fit <- cpb(y ~ x, data = data.frame(y = y, x = x)[y > 0, ], se = "none")
sims <- simulate(fit, nsim = 5)
colMeans(sims)
```

---

summary.cpb *Summarize a CPB fit*

Description

Summarize a CPB fit

Usage

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

Arguments

object	A "cpb" object.
...	Unused.

Value

An object of class "summary.cpb" with the coefficient table, the dispersion parameter and its profile-likelihood interval, the implied ceiling, fit statistics, and the likelihood-ratio test against a (zero-truncated) Poisson.

---

tidy.cpb *Tidy a CPB fit (broom method)*

Description

Tidy a CPB fit (broom method)

Usage

```
## S3 method for class 'cpb'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'cpb_fe'
tidy(x, ...)

## S3 method for class 'hurdle_cpb'
tidy(x, ...)

## S3 method for class 'zi_cpb'
tidy(x, ...)

## S3 method for class 'gec'
tidy(x, ...)

## S3 method for class 'hurdle_gec'
tidy(x, ...)

## S3 method for class 'zi_gec'
tidy(x, ...)

## S3 method for class 'count_reg'
tidy(x, ...)

## S3 method for class 'zi_count'
tidy(x, ...)

## S3 method for class 'hurdle_count'
tidy(x, ...)
```

Arguments

x	A "cpb" object.
conf.int	If TRUE, add conf.low/conf.high (requires a fit with bootstrap standard errors).
conf.level	Confidence level for the interval.
...	Unused.

Value

A data frame with one row per coefficient.

ud_screen *Screen a count outcome for underdispersion*

Description

Applies the diagnostic sequence developed in Bagozzi (2026): a *marginal* verdict from the conditional Pearson statistic and a through-origin score regression (which gives the direction of dispersion), a negative-binomial-versus-Poisson test for the overdispersion call, and—most importantly—an *at-risk* verdict on the positive counts benchmarked against a **zero-truncated Poisson**. The last step is what separates genuine underdispersion from the artifact of conditioning on $Y > 0$. When the data are not zero-dominated it also fits the CPB and reports a ceiling-exceedance diagnostic, and it fits a generalized-Poisson soft-tail comparator.

Usage

```
ud_screen(
  formula,
  data,
  run_cpb = TRUE,
  cpb_max_n = 3000,
  run_gp = TRUE,
  run_comp = TRUE,
  comp_max_par = 30,
  comp_max_n = 5000,
  ztp_threshold = c("calibrated", "bootstrap"),
  ztp_boot_B = 199L,
  digits = 3
)
```

Arguments

formula	A model formula.
data	A data frame.
run_cpb	Logical; fit the CPB when the data are not zero-dominated (default TRUE).
cpb_max_n	Skip the CPB fit above this sample size (default 3000).
run_gp	Logical; fit the generalized-Poisson comparator (default TRUE).
run_comp	Logical; fit the native COM-Poisson comparator (default TRUE). Skipped when the mean model carries more than comp_max_par parameters (the COM-Poisson has no concentrated fixed-effects path, so dummy-heavy screens would be slow) or when n exceeds comp_max_n.
comp_max_par, comp_max_n	Parameter and sample-size gates for the COM-Poisson comparator (defaults 30 and 5000).

`ztp_threshold` How to set the at-risk test's underdispersion cutoff. `"calibrated"` (default) uses the simulation-calibrated rule $1 - 2.27/\sqrt{n_+}$, whose constant is an estimated standard deviation fitted to one calibration grid (no fixed effects, and that grid's rate profile); `"bootstrap"` calibrates the cutoff on the data at hand by a parametric bootstrap of the fitted zero-truncated Poisson null (simulate `ztp_boot_B` at-risk panels at the fitted rates, refit, and take the empirical 5\ anti-conservative in two separable regimes: when the mean model's parameter count is a nontrivial share of the positive observations (dummy-heavy fixed-effects screens, where the null's center drifts with the parameter share), and when the fitted rates concentrate at small values (roughly λ below 2, and the more severely the smaller the rates), where the null's spread exceeds the fitted constant even without fixed effects (simulated size roughly 0.07–0.10 against the nominal 0.05). The bootstrap absorbs both departures by construction and is the recommended choice in either regime.

`ztp_boot_B` Number of parametric-bootstrap replicates (default 199).

`digits` Printing precision.

Value

An object of class `"ud_screen"` with `verdict_marginal`, `verdict_atrisk`, the conditional and at-risk (ZTP-benchmarked) Pearson statistics, the NB-vs-Poisson LR test, a log-likelihood comparison, (when fit) the CPB alpha and ceiling-exceedance share, and the over-conditioning guard state: `atrisk_skipped` (`TRUE` when the mean model nearly saturates the positive counts, so the at-risk statistic is not computed and the printout says why), `sat_ratio` (the fitted parameter share of the positives), and `overconditioned` (`TRUE` when that share reaches 0.10, the region where the calibrated threshold is anti-conservative; the printout then flags the verdict as diagnostic rather than probative and recommends `ztp_threshold = "bootstrap"`).

References

King, G. (1989). Variance specification in event count models. *AJPS* 33(3):762-784.

Bagozzi, B. E. (2026). Revisiting underdispersion in political science. Companion manuscript.

See Also

`cpb()`, `compare_dispersion()`

Examples

```
set.seed(1); x <- rnorm(250)
N <- pmax(round(exp(1.4 + 0.4 * x) / 0.6), 1); y <- rbinom(250, N, 0.6)
ud_screen(y ~ x, data = data.frame(y = y, x = x))
```

vcov.cpb_fe *Bootstrap covariance for a fixed-effects CPB fit*

Description

The covariance of the covariate coefficients from the stored pairs/cluster bootstrap (`se = "bootstrap"` at fit time); `NULL` when no bootstrap was run.

Usage

```
## S3 method for class 'cpb_fe'
vcov(object, ...)
```

Arguments

`object`	A `"cpb_fe"` object.
`...`	Unused.

Value

A covariance matrix, or `NULL`.

zi_count *Zero-inflated Poisson / negative-binomial regression (matched CPB baseline)*

Description

Fits a structural-zero mixture with a Poisson or negative-binomial count component, the count analogue of `zi_cpb()`. The count component uses a log link (on the mean for Poisson/NB, on the rate λ for COM-Poisson) and the inflation probability a link set by `link`. Returned as a `"zi_count"` object that `compare_models()` and `score()` accept.

Usage

```
zi_count(
  formula,
  data,
  family = c("poisson", "negbin", "compois"),
  zero = NULL,
  fe = NULL,
  zero_fe = NULL,
  link = c("logit", "probit", "cloglog"),
  offset = NULL,
  se = c("analytic", "robust", "cluster", "none"),
  cluster = NULL,
  ...
)
```

Arguments

formula	Count formula (y ~ x).
data	A data frame.
family	"poisson", "negbin", or "compois" for the count component; see count_reg() for the COM-Poisson parameterization.
zero	Optional one-sided formula for the inflation (structural-zero) model; defaults to the count right-hand side.
fe	Optional fixed-effects column for the count equation (factor dummies).
zero_fe	Optional fixed-effects column for the inflation equation (factor dummies); zero-equation fixed effects are opt-in.
link	Link for the inflation probability: "logit" (default), "probit", or "cloglog". Positive inflation coefficients raise the probability of a *structural zero*.
offset	Optional offset for the count component, on the linear-predictor (log) scale (log-mean for Poisson/NB, log-rate for COM-Poisson): a numeric vector or the name of a column in data.
se, cluster	Standard-error type and optional cluster; see count_reg().
...	Unused.

Value

An object of class "zi_count".

See Also

zi_cpb(), hurdle_count(), compare_models()

Examples

```
set.seed(2); n <- 300; x <- rnorm(n); z <- rnorm(n)
y <- ifelse(rbinom(n, 1, plogis(-0.5 + 0.8 * z)) == 1, 0L, rpois(n, exp(1 + 0.3 * x)))
zi_count(y ~ x, data.frame(y = y, x = x, z = z), family = "poisson",
         zero = ~ z, se = "none")
```

---

zi_cpb *Zero-inflated continuous parameter binomial regression*

Description

Fits the zero-inflated CPB. The model mixes a structural-zero process (a logistic model for the probability pi that a unit is a structural zero) with an untruncated CPB for the count, so a zero can arise either structurally or as a sampling zero from the CPB. Unlike hurdle_cpb(), the likelihood does not factorize, so the parameters are estimated by direct joint maximum likelihood, seeded from separate fits (a zero-truncated CPB on the positives and a logit of the zero indicator) and cross-checked against an expectation-maximization climb of the same observed-data likelihood, keeping the better optimum. An EM-only path (method = "em") is retained for comparison.

Usage

```
zi_cpb(
  formula,
  data,
  zero = NULL,
  fe = NULL,
  zero_fe = NULL,
  method = c("ml", "em"),
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  max.support = 500,
  maxit = 200,
  tol = 1e-06,
  offset = NULL
)
```

Arguments

formula	Intensity (count) model formula (y ~ x).
data	A data frame.
zero	Optional one-sided formula (~ z) for the zero-inflation model; defaults to the intensity model's right-hand side.
fe	Optional column name for unit fixed effects in the intensity (count) component. Under method = "ml" the unit effects enter as plug-in offsets from a first-stage fit; under method = "em" a classification (hard-assignment) step handles the structural zeros.
zero_fe	Optional column name for fixed effects in the zero-inflation equation, entered as factor dummies (opt-in).
method	Estimation method: "ml" (default) is robust direct joint maximum likelihood seeded from separate fits; "em" is expectation- maximization, retained for comparison but prone to a degenerate structural-zero-probability collapse under heavy zero-inflation.
se	Coefficient inference: "none" (default) or "bootstrap".
B	Bootstrap resamples when se = "bootstrap".
cluster	Optional cluster for the bootstrap (a column name or vector); under fixed effects the units are resampled by default. The mixture EM makes the bootstrap costly, so keep B modest.
max.support	Maximum support for the CPB pmf.
maxit	Optimizer iteration budget (scaled internally for the joint maximization; also the EM iteration cap under method = "em").
tol	Relative convergence tolerance on the observed-data log-likelihood.
offset	Optional exposure offset (log scale) for the count component: a numeric vector or the name of a column in data, making the count a rate model. Supported for method = "ml" without fe (the fixed-effects path's plug-in unit effects have no

offset handling yet, and errors loudly). The bootstrap carries the offset through every replicate.

Details

The inflation equation uses a logit link. For a probit or cloglog inflation link, use `zi_count()` (Poisson/NB/COM-Poisson count), whose `link` argument covers the binary stage.

Value

An object of class `"zi_cpb"`.

Examples

```
set.seed(1)
n <- 800; x <- rnorm(n); z <- rnorm(n)
y <- rzicpb(n, lambda = exp(1.3 + 0.5 * x), alpha = 0.5, pi = plogis(-0.5 + 0.8 * z))
zi_cpb(y ~ x, data = data.frame(y = y, x = x, z = z), zero = ~ z)
```

---

zi_gec *Zero-inflated GEC (Katz-family) regression*

Description

A structural-zero mixture with a GEC count component whose dispersion `delta` is estimated freely. Estimated by robust direct maximum likelihood, seeded from separate fits (a GEC on the positive counts for the count parameters and a logit of the zero indicator for the inflation), exactly as in `zi_cpb()`; this avoids the degenerate `pi -> 0` basin that coordinate-wise EM falls into.

Usage

```
zi_gec(
  formula,
  data,
  zero = NULL,
  zero_fe = NULL,
  se = c("none", "bootstrap"),
  B = 500,
  cluster = NULL,
  max.support = 500,
  maxit = 200,
  tol = 1e-06,
  offset = NULL
)
```

Arguments

formula	Count formula, y ~ x.
data	A data frame.
zero	One-sided formula for the structural-zero logit; defaults to the count formula's right-hand side.
zero_fe	Optional column name for fixed effects in the zero-inflation equation, entered as factor dummies (opt-in).
se	Coefficient inference: "none" (default) or "bootstrap".
B	Bootstrap resamples.
cluster	Optional cluster (column name or vector) for the bootstrap.
max.support	Guard on the maximum evaluated support.
maxit, tol	Optimizer controls.
offset	Optional exposure offset (log scale) for the count component: a numeric vector or the name of a column in data; the bootstrap carries it through every replicate.

Value

An object of class "zi_gec".

See Also

zi_cpb(), gec(), hurdle_gec()

Examples

```
set.seed(4); n <- 300; x <- rnorm(n); z <- rnorm(n)
y <- ifelse(rbinom(n, 1, plogis(-0.5 + 0.8 * z)) == 1, 0L, rpois(n, exp(1 + 0.3 * x)))
zi_gec(y ~ x, data.frame(y = y, x = x, z = z), zero = ~ z)
```

---

zi_test *Boundary-corrected test for zero-inflation (CPB vs ZI-CPB)*

Description

A likelihood-ratio test of whether a count needs a structural-zero component. The (untruncated) CPB is nested in the zero-inflated CPB at a structural-zero probability of zero. Because that value lies on the boundary of the parameter space, the LR statistic follows a $\frac{1}{2}\chi^2_0 + \frac{1}{2}\chi^2_1$ mixture (Self and Liang 1987), which halves the naive χ^2_1 p-value. The test uses an intercept-only structural-zero probability, so it is a clean single-parameter boundary test; a covariate-dependent structural-zero model is better compared with information criteria and proper scores via compare_dispersion().

Usage

```
zi_test(object, object2 = NULL, data = NULL, ...)
```

Arguments

`object`	Either a model formula — then `data` is required and the nested pair is fit internally — or a fitted model (`cpb`, `cpb_fe`, or `zi_cpb`).
`object2`	The second argument: the data frame when `object` is a formula (so `zi_test(y ~ x, mydata)` works), or the second fitted model when `object` is a fit. In the model interface exactly one of the two models must be a zero-inflated `zi_cpb` and the other its non-inflated nest (`cpb` or `cpb_fe`); the two may carry any combination of fixed effects and robust/clustered standard errors.
`data`	A data frame; an alternative to passing it as `object2`.
`...`	Passed to `cpb()` and `zi_cpb()` in the formula interface.

Value

An object of class `"zi_test"` with the two log-likelihoods, the LR statistic, and the boundary-corrected p-value.

Examples

```
set.seed(1); x <- rnorm(500)
y <- rzicpb(500, lambda = exp(1.2 + 0.4 * x), alpha = 0.5, pi = 0.3)
zi_test(y ~ x, data = data.frame(y = y, x = x))
```

Index