Package 'iop'

September 3, 2026

Type Package

Title Inflated Ordered Probit and Logit Models

Version 0.1.0

Description Estimation, inference, and quantities of interest for ordered probit and ordered logit models whose outcome contains an inflated category: a single ordered category (bottom, middle, top, or any other) that mixes observations generated by the ordered process with observations generated by a distinct split-population process. Fits the zero-inflated ordered probit of Harris and Zhao (2007) <doi:10.1016/j.jeconom.2007.01.002> and its middle- and top-inflated extensions (Bagozzi and Mukherjee 2012 <doi:10.1093/pan/mps020>; Bagozzi, Hill, Moore and Mukherjee 2015 <doi:10.1177/0022002713520530>; Bagozzi, Joo and Mukherjee 2024 <doi:10.1093/fpa/orae006>), generalized to an arbitrary inflated category and to the logit link, with optional correlated errors for the probit form, plus the standard ordered probit and logit and their partial proportional-odds (non-parallel) variants on the same footing. Provides analytic, robust, and cluster-robust standard errors, survey weights and offsets, model comparison (Vuong, likelihood-ratio, information criteria), regime-specific predicted probabilities and first differences, simulation for residual diagnostics, and tidy/table-package integration. The likelihood, its gradient, and the bivariate-normal probabilities are implemented in C++.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

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

LinkingTo Rcpp

Suggests ordinal, VGAM, mvtnorm, pbivnorm, sandwich, DHARMa, testthat (>= 3.0.0), knitr, rmarkdown, broom, modelsummary, texreg, parallel, AER

VignetteBuilder knitr

URL https://github.com/bagozzib/iop, https://bagozzib.github.io/iop/

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

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Benjamin E. Bagozzi [aut, cre]

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

Repository CRAN

Date/Publication 2026-09-03 11:50:24 UTC

Contents

iop-package *iop: Inflated Ordered Probit and Logit Models*

Description

Ordered probit and logit regression for outcomes with an *inflated* category: a single ordered category (bottom, middle, top, or any other) that mixes observations generated by the ordered process with observations generated by a distinct split-population process that places them in that category regardless of the ordered mechanism. The package fits the zero-inflated ordered probit of Harris and Zhao (2007) and its middle- and top-inflated extensions (Bagozzi and Mukherjee 2012; Bagozzi, Hill, Moore and Mukherjee 2015; Bagozzi, Joo and Mukherjee 2024), generalized to any inflated category and to the logit link, with optional correlated errors for the probit form, alongside the standard ordered probit and logit on the same engine, so that the plain and inflated models share one formula interface, one set of methods, and one set of quantities of interest.

Estimators

All four share the formula interface `y ~ x1 + x2 | z1 + z2` (outcome equation before the `|`, inflation equation after it), survey `weights`, offsets, partial proportional-odds effects (`parallel =`, with `parallel_test()` and `parallel = "auto"`), unit random intercepts (`re =`), unit fixed effects (`fe =`, with the split-panel jackknife `fe_correction = "jackknife"`), and analytic, robust, or cluster-robust standard errors (`se =`, `cluster =`).

`oprobit()`, `ologit()` The standard ordered probit and logit.

`iop()` The inflated ordered probit for any single inflated category (`inflate = "bottom"`, `"middle"`, `"top"`, or a category label), with `correlated = TRUE` for the ZiOPC / MiOPC / TiOPC models and `split = "category"` for the category-specific split equations of the generalised GZiOP / GMiOP (Brown, Harris and Spencer 2020).

`iol()` The inflated ordered logit counterpart.

Standard errors: analytic, robust, cluster-robust, or nonparametric bootstrap (`se = "bootstrap"`, with percentile intervals from `confint.iord()`).

Methods (one set for every model)

`summary.iord()`, `print()`, `coef()`, `vcov.iord()` (natural or internal scale), `confint.iord()` (`rho` on the $\tanh^{-1}$ scale), `logLik()`, `nobs()`, `fitted()`, `residuals.iord()`, `predict.iord()` (category probabilities, modal class, ordered-stage probabilities, regime and inflation probabilities, posterior probability of being an inflated case, linear predictors, all with optional delta-method `se.fit`), `ranef()` for random-intercept fits, `simulate.iord()` for simulated-residual diagnostics, and broom (`tidy.iord()`, `glance.iord()`, `augment.iord()`) and texreg (`extract()`) integration so `modelsummary::modelsummary()` and `texreg::screenreg()` work out of the box.

Quantities of interest

`first_difference()` (by category and regime, by stage, at a profile or averaged over the data, delta-method or simulation intervals), `ame()` (average marginal effects), both with `decompose =`

TRUE for the two components of the inflated-category probability (predict(type = "zeros"), Harris and Zhao's two types of zeros), and plot.iop_fd() / plot.iop_ame() for both.

Model comparison

vuong() (raw, AIC-, BIC-corrected), lr_test() for nested pairs, inflation_test() (inflated vs plain ordered, refit internally, with a parametric-bootstrap likelihood-ratio test), split_test() (common vs category-specific split equations, LM and LR), compare_models(), parallel_test() for the parallel-regression assumption, and classification() (confusion table, Brier and ranked probability scores, precision/recall).

Panels

Random intercepts by adaptive Gauss–Hermite quadrature (re =, re_inflation =, nAGQ =), unit fixed effects (fe =, fe_inflation =, fe_correction = "jackknife", time =), the mundlak() correlated-random-effects device, and cluster-robust standard errors.

Data and simulation

bp (political violence, zero-inflated), pta (escape-flexibility provisions, top-inflated), repression (repression of nonviolent campaigns, top-inflated); riop() draws data from the model's own data-generating process; diord(), piord(), qiord(), riord() are the probability mass, cumulative probability, quantile, and random-draw functions of the (inflated) ordered response.

Vignettes

vignette("iop") (getting started), vignette("quantities") (predicted probabilities, first differences, marginal effects, tables, diagnostics), vignette("panels") (random intercepts, fixed effects, Mundlak, the jackknife, and the package's Monte Carlo), and vignette("model") (the likelihood, identification, multi-start estimation, boundary cases, and validation).

Author(s)

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

Authors:

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

References

Harris, M.N. and Zhao, X. (2007). A zero-inflated ordered probit model, with an application to modelling tobacco consumption. *Journal of Econometrics*, 141, 1073-1099.

Bagozzi, B.E. and Mukherjee, B. (2012). A mixture model for middle category inflation in ordered survey responses. *Political Analysis*, 20, 369-386.

Bagozzi, B.E., Hill, D.W., Moore, W.H. and Mukherjee, B. (2015). Modeling two types of peace: The zero-inflated ordered probit (ZiOP) model in conflict research. *Journal of Conflict Resolution*, 59, 728-752.

Bagozzi, B.E., Joo, M.M. and Mukherjee, B. (2024). Top-category inflation in ordered international relations outcomes. *Foreign Policy Analysis*, 20, orae006.

See Also

Useful links: the source repository https://github.com/bagozzib/iop; report bugs at https://github.com/bagozzib/iop/issues.

Examples

```
data(bp)
m <- iop(violence ~ loggdppc + parliament + disaster | loggdppc + parliament + disaster,
         data = bp, inflate = "bottom")
m
```

---

ame *Average marginal effects*

Description

Average (over the estimation data) effect of each covariate on every category probability and, for inflated models, on the probability of the ordered regime (one per split equation under a category-specific split). `ame()` has no profile mode by design – it always averages over the estimation data (or over `newdata`); for the effect at a covariate profile use `first_difference()` with `newdata =`. Numeric covariates get a derivative (central difference averaged over observations); 0/1 covariates get the discrete change 0 -> 1; factors get each level against the base level. By default a covariate is moved in every equation in which it appears (the total effect); `stage` restricts the move to the outcome or the inflation equation, as in `first_difference()`. Intervals are by the delta method.

Usage

```
ame(
  object,
  vars = NULL,
  level = 0.95,
  ci = c("delta", "none"),
  eps = 1e-04,
  decompose = FALSE,
  stage = c("both", "outcome", "inflation")
)
```

Arguments

`object`	An `"iord"` object.
`vars`	Covariates to include (default: all covariates in either equation, excluding unit identifiers).
`level`	Confidence level.
`ci`	`"delta"` (default) or `"none"`.
`eps`	Relative step for numeric derivatives (times the covariate's SD).

decompose	For inflated models, also report the effects on the two components of the inflated-category probability – through the inflation process and through the ordered stage (Harris and Zhao 2007; see `predict(type = "zeros")`).
stage	For inflated models: move each covariate in `"both"` equations (default, the total effect), in the `"outcome"` equation only, or in the `"inflation"` equation only. A covariate absent from the selected equation has a zero effect.

Value

A data frame of class `c("iop_ame", "data.frame")`: `variable`, `contrast`, `component`, `estimate`, `lower`, `upper`, `method`.

See Also

`first_difference()`, `plot.iop_fd()`, `predict.iord()`

Other quantities of interest: `first_difference()`, `plot.iop_fd()`, `predict.iord()`

Examples

```
set.seed(2)
d <- riop(600, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom")
ame(m)
ame(m, vars = "x1")
ame(m, vars = "z1", decompose = TRUE)
ame(m, vars = "z1", stage = "inflation")          # through the split equation only
```

`augment.iord` *Augment data with fitted classes and regime probabilities (broom method)*

Description

Augment data with fitted classes and regime probabilities (broom method)

Usage

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

Arguments

x	An `"iord"` object.
...	Unused.

Value

The estimation data with `.fitted` (the modal category), `.prob_<level>` columns, and for inflated models `.regime` (fitted $P(s = 1)$).

See Also

`predict.iord()` for the full set of prediction types.

Other broom methods: `glance.iord()`, `tidy.iord()`

---

bp *Political violence, 1976–1996 (Besley and Persson 2009)*

Description

Country-year data on political violence used by Bagozzi, Hill, Moore and Mukherjee (2015) to introduce the zero-inflated ordered probit in conflict research. The outcome is ordered – no violence, repression, civil war – and its bottom category mixes countries that are structurally at peace with countries that are at risk but happened not to experience violence that year: a zero-inflated ordered outcome. Rows are the complete cases of the published specification.

Usage

```
data(bp)
```

Format

A data frame with 1984 rows and 8 variables:

country Country name.

year Year.

violence Ordered factor: `none` < `repression` < `civil war`.

loggdppc Log real GDP per capita.

parliament 1 if a parliamentary democracy.

disaster Number of natural disasters in the year.

major_oil 1 if a major oil exporter.

major_primary 1 if a major primary-commodity exporter.

Provenance and terms

Taken from the public replication archive of the cited article and redistributed here, with the variables renamed and recoded as documented in `data-raw/make_data.R`, so that the published results can be reproduced; the archive states the original terms of use.

Source

Besley, T. and Persson, T. (2009). Repression or civil war? *American Economic Review: Papers and Proceedings*, 99, 292-297 (replication data); as analyzed in Bagozzi, B.E., Hill, D.W., Moore, W.H. and Mukherjee, B. (2015). Modeling two types of peace: The zero-inflated ordered probit (ZiOP) model in conflict research. *Journal of Conflict Resolution*, 59, 728-752.

See Also

iop(); vignette("iop") and vignette("quantities") analyze these data.

Other datasets: pta, repression

Examples

```
data(bp)
table(bp$violence)
m <- iop(violence ~ loggdppc + parliament + disaster + major_oil + major_primary |
           loggdppc + parliament + disaster + major_oil + major_primary,
         data = bp, inflate = "bottom")
summary(m)
```

---

classification *Classification table and accuracy scores*

Description

Summarizes how well a fitted model's predicted probabilities reproduce the observed categories: the classification (confusion) table of observed versus modal predicted categories, the share of correct classifications, the Brier score and the ranked probability score (both strictly proper scoring rules, smaller is better, 0 for perfect probabilistic prediction), and for every category the precision, the recall (hit rate), and the adjusted noise-to-signal ratio of Kaminsky and Reinhart (1999). The same measures are used by Dale and Sirchenko (2021) to compare ordered and inflated ordered fits; because they are computed from predicted probabilities they apply identically to every model in the package and to new data.

Usage

```
classification(object, newdata = NULL, weights = NULL)
```

Arguments

object	An "iord" object.
newdata	Optional data frame with the covariates and the response, to score out of sample; default is the estimation data.
weights	Optional prior weights for newdata (the fit's weights are used for the estimation data).

Details

The modal-class accuracy measures (share correctly classified, precision, recall) can look poor for a minority category even when the model is well specified: the modal category of a probability vector is rarely a minority category, so its recall is often zero; the proper scoring rules (Brier, ranked probability, log score) are the better summaries of fit. The Brier score is $\frac{1}{n}\sum_i\sum_j(P_{ij}-I_{ij})^2$ and the ranked probability score $\frac{1}{n}\sum_i\sum_j(Q_{ij}-D_{ij})^2$ with Q and D the cumulative predicted

probabilities and the cumulative indicator; both are weighted by the prior weights when present. Precision is TP/(TP + FP), recall TP/(TP + FN), and the adjusted noise-to-signal ratio {FP/(FP + TN)}/{TP/(TP + FN)}, all from the modal-category classification.

Value

An object of class `"iop_classification"`: a list with `table` (observed in rows, predicted in columns; with weights, sums of weights per cell, so that it agrees with the weighted scores), `correct` (share correctly classified), `brier`, `rps` (ranked probability score), `by_category` (precision, recall, noise-to-signal, and the observed share per category), `n`, and `loglik` (the per-observation mean log score, another proper scoring rule).

References

Brier, G.W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78, 1-3. Epstein, E.S. (1969). A scoring system for probability forecasts of ranked categories. *Journal of Applied Meteorology*, 8, 985-987. Kaminsky, G.L. and Reinhart, C.M. (1999). The twin crises: The causes of banking and balance-of-payments problems. *American Economic Review*, 89, 473-500. Dale, D. and Sirchenko, A. (2021). Estimation of nested and zero-inflated ordered probit models. *Stata Journal*, 21, 3-38.

See Also

`predict.iord()`, `compare_models()`, `vuong()`

Other model comparison: `compare_models()`, `inflation_test()`, `lr_test()`, `parallel_test()`, `split_test()`, `vuong()`

Examples

```
data(bp)
m_op <- oprobit(violence ~ loggdppc + parliament + disaster, data = bp)
m_zi <- iop(violence ~ loggdppc + parliament + disaster | loggdppc + parliament + disaster,
            data = bp, inflate = "bottom")
classification(m_op)
classification(m_zi)
c(op = classification(m_op)$brier, ziop = classification(m_zi)$brier)
```

---

compare_models *Side-by-side fit statistics for ordered / inflated ordered models*

Description

Side-by-side fit statistics for ordered / inflated ordered models

Usage

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

Arguments

...	Named "iord" fits.

Value

A data frame with one row per model: model label, log-likelihood, degrees of freedom, AIC, BIC, N, and the inflated category (if any), sorted as supplied.

See Also

vuong(), lr_test(), glance.iord()

Other model comparison: classification(), inflation_test(), lr_test(), parallel_test(), split_test(), vuong()

Examples

```
set.seed(6)
d <- riop(500, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
compare_models(op = oprobit(y ~ x1 + x2, d), ziop = iop(y ~ x1 + x2 | z1, d, inflate = "bottom"))
```

---

confint.iord *Confidence intervals for an ordered / inflated ordered fit*

Description

Wald intervals from the fitted covariance matrix (the interval for rho is formed on the $\tanh^{-1}$ scale and transformed back, so it respects the (-1, 1) bounds), or – for a fit with se = "bootstrap" – percentile intervals from the bootstrap replicates.

Usage

```
## S3 method for class 'iord'
confint(object, parm, level = 0.95, type = c("wald", "percentile"), ...)
```

Arguments

object	An "iord" object fit with standard errors.
parm	Optional subset of coefficient names.
level	Confidence level (default 0.95).
type	"wald" (default) or "percentile" (requires se = "bootstrap"; quantiles of the replicate estimates on the natural scale).
...	Unused.

Value

A matrix of lower/upper bounds.

See Also

vcov.iord(), summary.iord(), tidy.iord() (conf.int = TRUE).

Other inference methods: summary.iord(), vcov.iord()

Examples

```
data(pta)
m <- iop(flexibility ~ depth + democracy + gdp | gdp + democracy, data = pta, inflate = "top")
confint(m)
confint(m, parm = c("depth", "infl_democracy"), level = 0.9)

mc <- iop(flexibility ~ depth + democracy + gdp | gdp + democracy, data = pta,
          inflate = "top", correlated = TRUE)
confint(mc, parm = "rho")                      # formed on the atanh scale
```

---

first_difference *First differences in predicted probabilities*

Description

The change in each category probability $P(y = j)$ – and, for inflated models, in the ordered-regime probability $P(s = 1)$ – when one covariate moves from from to to, holding the others at a profile (default: weighted means of numeric covariates and modal levels of factors) or averaging over every observation's own covariates (average = TRUE). For inflated models the covariate can be moved in both equations (the total effect), in the outcome equation only, or in the inflation equation only; a covariate absent from an equation simply has no effect there.

Usage

```
first_difference(
  object,
  var,
  from,
  to,
  newdata = NULL,
  average = FALSE,
  stage = c("both", "outcome", "inflation"),
  ci = c("delta", "sim", "none"),
  level = 0.95,
  R = 1000,
  decompose = FALSE,
  ...
)
```

Arguments

object	An "iord" object.
var	Name of the covariate to move.
from, to	Its two values (numbers, or level labels for a factor).
newdata	Optional one-row data frame giving the profile of the other covariates; default is the typical profile described above. For a fixed-effects fit the default profile sits at the modal unit (ties go to the first level); average = TRUE evaluates every unit at its own fixed effect, which is usually the more natural summary there. For a random-intercept fit the probabilities are marginal over the intercept.
average	If TRUE, average the first difference over the estimation data (each observation keeps its own other covariates); newdata is then ignored.
stage	For inflated models: "both" (default), "outcome", or "inflation".
ci	"delta" (default), "sim", or "none".
level	Confidence level.
R	Number of simulation draws when ci = "sim".
decompose	For inflated models, also report the change in the two components of the inflated-category probability – through the inflation process and through the ordered stage (see predict(type = "zeros")), the decomposition of Harris and Zhao (2007).
...	Unused; unknown arguments are an error.

Details

Intervals are by the delta method (analytic gradient of the probabilities with respect to the parameters, from the fitted covariance) or by simulation (R draws of the parameters from their asymptotic normal distribution, percentile bounds; Krinsky and Robb 1986; King, Tomz and Wittenberg 2000).

Value

A data frame of class c("iop_fd", "data.frame") with one row per category (and one for P(ordered regime) in inflated models – one per non-inflated category under a category-specific split – plus the two components of the inflated-category probability when decompose = TRUE): component, from, to, diff, lower, upper, method.

References

King, G., Tomz, M. and Wittenberg, J. (2000). Making the most of statistical analyses: improving interpretation and presentation. *American Journal of Political Science*, 44, 347-361. Harris, M.N. and Zhao, X. (2007). A zero-inflated ordered probit model, with an application to modelling tobacco consumption. *Journal of Econometrics*, 141, 1073-1099.

See Also

predict.iord(), ame() for average marginal effects of every covariate, plot.iop_fd().

Other quantities of interest: ame(), plot.iop_fd(), predict.iord()

Examples

```
set.seed(7)
d <- riop(600, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom")
first_difference(m, "x1", from = -1, to = 1)
first_difference(m, "z1", from = -1, to = 1, stage = "inflation", average = TRUE)
first_difference(m, "z1", from = -1, to = 1, decompose = TRUE)    # the two types of zeros
plot(first_difference(m, "x1", from = -1, to = 1))
```

`glance.iord` *Glance at an ordered / inflated ordered fit (broom method)*

Description

Glance at an ordered / inflated ordered fit (broom method)

Usage

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

Arguments

`x`	An `"iord"` object.
`...`	Unused.

Value

A one-row data frame: `logLik`, `AIC`, `BIC`, `df`, `nobs`, for inflated models `inflate` (the category) and `share_inflated` (its observed share), `rho` for correlated fits, and the fit-quality flags `converged`, `boundary`, and `ill_conditioned` (see `?iop`, section "Diagnostics and fixed thresholds"), so that tables built with **modelsummary** or by filtering `glance()` output carry them.

See Also

`compare_models()` for several fits side by side.

Other broom methods: `augment.iord()`, `tidy.iord()`

`inflated` *Inflated ordered probit and inflated ordered logit regression*

Description

Fits an ordered probit (`iop()`) or ordered logit (`iol()`) model in which one ordered category is *inflated*: it collects observations generated by the ordered process together with observations generated by a distinct split-population process that places them in that category regardless of the ordered mechanism. The zero-inflated ordered probit of Harris and Zhao (2007), the middle-inflated ordered probit of Bagozzi and Mukherjee (2012), and the top-inflated ordered probit of Bagozzi, Joo and Mukherjee (2024) are the cases `inflate = "bottom"`, `"middle"`, and `"top"`; any single category may be named. `iop()` optionally estimates the correlation between the two latent equations' errors (`correlated = TRUE`, the ZiOPC / MiOPC / TiOPC models). Both accept unit random intercepts (`re`), in the outcome equation and optionally also in the inflation equation.

Usage

```
iop(
  formula,
  data,
  inflate,
  correlated = FALSE,
  split = c("common", "category"),
  parallel = TRUE,
  re = NULL,
  re_inflation = FALSE,
  nAGQ = 15,
  fe = NULL,
  fe_inflation = FALSE,
  fe_correction = c("none", "jackknife"),
  time = NULL,
  weights = NULL,
  offset = NULL,
  offset_inflation = NULL,
  se = c("analytic", "robust", "cluster", "bootstrap", "none"),
  cluster = NULL,
  nboot = 200,
  cores = 1,
  start = NULL,
  maxit = 1000,
  reltol = 1e-10
)

iol(
  formula,
  data,
  inflate,
```

```
  split = c("common", "category"),
  parallel = TRUE,
  re = NULL,
  re_inflation = FALSE,
  nAGQ = 15,
  fe = NULL,
  fe_inflation = FALSE,
  fe_correction = c("none", "jackknife"),
  time = NULL,
  weights = NULL,
  offset = NULL,
  offset_inflation = NULL,
  se = c("analytic", "robust", "cluster", "bootstrap", "none"),
  cluster = NULL,
  nboot = 200,
  cores = 1,
  start = NULL,
  maxit = 1000,
  reltol = 1e-10
)
```

Arguments

formula	A model formula `y ~ x1 + x2`. The response may be an ordered factor, a factor (level order taken as the ordinal order), or a numeric/integer vector (sorted unique values define the order). The outcome equation has no intercept (the cutpoints absorb it), so an explicit `0 +` / `- 1` is ignored with a message and factors are coded as with an intercept; a set of columns that sums to a constant is rejected as rank deficient.
data	A data frame.
inflate	The inflated category: `"bottom"`, `"middle"` (odd number of categories only), `"top"`, or one of the response's categories (a level label or, for a numeric response, a value).
correlated	Logical (`iop()` only): estimate the correlation between the inflation- and outcome-equation errors.
split	`"common"` (default): one split equation shared by every category (the ZiOP / MiOP / TiOP); `"category"`: a separate split equation for every non-inflated category (the generalised GZiOP / GMiOP of Brown, Harris and Spencer 2020; see Details).
parallel	`TRUE` (default; all effects parallel), `FALSE` (all effects category-specific), a one-sided formula naming the terms that are held parallel, e.g. `parallel = ~ . - x2`, or `"auto"`: starting from the all-parallel fit, relax the term whose parallel restriction has the smallest likelihood-ratio p-value while that p-value is below 0.05 (a forward version of Stata's `gologit2, autofit`; the steps are stored in `object$autofit`). See also `parallel_test()`.
re	Optional column name in `data` identifying units that receive a random intercept in the outcome equation.

`re_inflation` Logical: with `re`, also give the inflation equation an (independent) unit random intercept.

`nAGQ` Number of adaptive Gauss–Hermite quadrature nodes per random intercept (default 15).

`fe` Optional column name in `data` identifying units that receive a fixed effect (a dummy per unit, reference level omitted) in the outcome equation. Units whose response is constant at an extreme category have no finite fixed effect and are dropped with a message; a covariate that does not vary within units is collinear with the dummies and is refused by name (keep it with `re =` or `mundlak()` instead). Maximum-likelihood unit dummies carry incidental-parameters bias when units have few observations (Greene 2004); the fit warns when the median unit has fewer than 10. In the package's Monte Carlo (`system.file("mc", package = "iop")`: ordered probit, 100 units, a covariate correlated with a N(0,1) unit effect, 100 replications) the bias of that covariate's coefficient is +26 / +11 / +5 / +2.5 percent at T = 4 / 8 / 16 / 32 with unit dummies, +21 / +11 / +6 / +3 percent with a random intercept, +12 percent throughout when pooled, and within 0.3 percent at every T with the `mundlak()` device – the recommended route for short panels; reserve `fe =` for long ones. Computationally the dummies enter the parameter vector one per unit, and the exact-Hessian Newton polish and the covariance cost grow roughly with the square of the number of parameters (about 0.5 s at 20 units, 3 s at 100, 18 s at 300 for a plain ordered probit on 4,000 rows; the multistart of an inflated model multiplies this); for panels with many hundreds of units prefer `re =` or `mundlak()`.

`fe_inflation` Logical: with `fe`, also give the inflation equation unit dummies; units never or always in the inflated category are then dropped. A unit whose inflated-category observations are all absorbed by the ordered stage still has an inflation dummy at the boundary (regime probability 1); the fit reports that dummy's standard error as `NA` (`$se_na`) and flags the conditioning (`$ill_conditioned`); like the outcome-equation dummies, the inflation dummies are exempt from the quasi-separation signatures.

`fe_correction` With `fe`, `"jackknife"` applies the split-panel jackknife of Dhaene and Jochmans (2015): the model is refit on the two half-panels of every unit (first and second half of the observations ordered by `time`, or by row order if `time` is missing) and the common parameters are bias-corrected as `2 * full - (half1 + half2) / 2`, removing the leading incidental-parameters bias; the uncorrected estimates are kept in `object$coefficients_uncorrected`, the half-panel estimates and log-likelihoods in `object$jackknife`, and the full-sample covariance is reported: Dhaene and Jochmans (2015, Section 2) show that the split-panel jackknife removes the leading bias without changing the first-order asymptotic variance, so the full-sample covariance is the asymptotically valid one for the corrected estimator (its finite-sample variance is somewhat larger, which `se = "bootstrap"` cannot assess under `fe`; treat the intervals as approximate). Units with a single observation do not enter the half-panels. The correction is only as good as the half-panel fits: when a covariate is identified mainly by a within-unit trend, or the half-panels are very short, the half-panel estimates are noisy and the corrected estimate inherits that noise – inspect `object$jackknife$half_coefficients` before reporting it.

`time` Optional column name ordering the observations within units (used by `fe_correction = "jackknife"`).

`weights` Optional weights: a column name in `data` or a numeric vector aligned to its rows. They enter as frequency weights: a fit with weight 2 equals a fit on duplicated rows, in the estimates, the model-based standard errors, the information criteria (whose `n` is the weight total), the Vuong statistics, and the averaged quantities of interest. Survey (sampling, probability) weights are not frequency weights: with them the default model-based standard errors are not the design-based ones, so use `se = "robust"` (the pseudo-maximum-likelihood sandwich) or `cluster =` for the primary sampling units.

`offset` Optional offset on the latent scale of the outcome equation: a column name in `data` or a numeric vector.

`offset_inflation` Optional offset on the latent scale of the inflation equation: a column name in `data` or a numeric vector.

`se` Standard errors: `"analytic"` (inverse observed information, default), `"robust"` (heteroskedasticity-consistent sandwich over observations, or over units for random-intercept fits), `"cluster"` (cluster-robust; needs `cluster`), `"bootstrap"` (non-parametric bootstrap over the estimation rows – or over the clusters when `cluster` is given, or over the units of a random-intercept fit – with `nboot` refits from the full-sample estimate; the replicate estimates are kept in `object$boot` and `confint.iord()` can then report percentile intervals), or `"none"`. The bootstrap is not available with `fe`. Replicates whose split equation (or `rho`) runs to a boundary are counted in `object$boot$n_boundary` and trigger a warning: their estimates inflate the bootstrap standard errors of the affected block, for which the percentile intervals are the more robust summary. For `"bootstrap"`, a seed set before the call (`set.seed()`) makes the resamples reproducible; runs with `cores = 1` and `cores > 1` use different random streams.

`cluster` Optional cluster identifier (a column name in `data` or a vector aligned to its rows); supplying it selects `se = "cluster"` unless `se` is given explicitly (with `se = "bootstrap"` it defines the resampling blocks). With a random intercept, clusters must nest the units.

`nboot` Number of bootstrap refits for `se = "bootstrap"` (default 200). Each refit costs about one fit of the model (roughly a third of a second per refit for the bundled `bp` ZiOP), so budget a minute or two at the default on data of that size, and use `cores`.

`cores` Number of parallel workers for the bootstrap refits (a PSOCK cluster via the **parallel** package; default 1).

`start` Optional starting values on the internal parameter scale (see `object$theta`); rarely needed.

`maxit, reltol` Controls for the BFGS stage of the optimizer, passed to `stats::optim()`; the exact-Hessian Newton polish that follows runs up to 25 further iterations regardless of `maxit` (so `maxit` bounds the quasi-Newton phase, not the total). Results are insensitive to `maxit` above a few hundred; `maxit = 0` is reported as non-converged.

Details

Two latent equations are estimated jointly. The inflation (split) equation $s_i^* = z_i'\gamma + u_i$ assigns unit i to the ordered regime ($s_i = 1$, with probability $P(s_i = 1) = F(z_i'\gamma)$) or to the inflated regime ($s_i = 0$), in which the outcome is the inflated category k with certainty. The outcome equation is the ordered model of `oprobit()`/`ologit()` with cell probabilities π_{ij}. Hence

$$P(y_i = j) = F(z_i'\gamma)\,\pi_{ij} + 1\{j = k\}\,[1 - F(z_i'\gamma)].$$

Note the sign convention, which follows Harris and Zhao (2007) and the political-science literature: **positive inflation coefficients raise the probability of the ordered (non-inflated) regime**; `predict(type = "inflated")` returns $1 - F(z_i'\gamma)$.

With `correlated = TRUE` (probit only) (u_i, ϵ_i) are bivariate standard normal with correlation ρ, and the regime-1 cell probabilities become rectangle probabilities of the bivariate normal, $P(s_i = 1, y_i \le j) = \Phi_2(z_i'\gamma, \tau_j - x_i'\beta; -\rho)$, evaluated by a deterministic Gauss–Legendre algorithm (Drezner and Wesolowsky 1990; Genz 2004) so the likelihood is smooth for the optimizer. ρ is estimated on the $\tanh^{-1}$ scale and reported on the natural scale; its confidence interval is transformed from the former.

The inflation equation is specified after a `|` in the formula, `y ~ x1 + x2 | z1 + z2`, and includes an intercept. If the `|` part is omitted the outcome covariates are reused, which identifies the model by functional form alone; an exclusion restriction (a covariate in one equation only) is advisable and a message says so.

The likelihood of an inflated mixture can have several local maxima – a "soft" split with modest inflation coefficients and a "sharp" split with steep ones can both be stationary points – so every inflated fit is a multi-start: the inflation equation is started from a binary model of membership in the non-inflated category at several slope scales and signs, from flat high-regime values, and from the plain ordered baseline, and the best optimum is kept. Starts are staged: each gets a short quasi-Newton run, and the two best are run to convergence and polished with exact-Hessian Newton steps. `object$start_logliks` records the log-likelihood reached from every start (short-run values for the non-finalists); a spread of several units across starts is the signature of multimodality, and `start =` lets you add your own. Every inflated fit also fits its plain ordered counterpart first (whose log-likelihood is concave) and checks that the inflated fit is not below it: the plain model is the limit of the inflated one as the inflation intercept tends to infinity, so the inflated log-likelihood can never be lower at a true optimum. Fits whose inflation equation degenerates – every regime probability at 1 (the fit has collapsed to the plain ordered model) or a quasi-separated split with a few units at 0 and the rest at 1 (the coefficients are not finite) – whose `rho` sits at ± 1, or whose random-intercept SD is zero are flagged (`object$boundary`) with a warning that names the case; such data contain no identifiable inflation (or correlation) process, or an over-rich split equation.

Random intercepts (`re`) enter the outcome equation; with `re_inflation = TRUE` an independent random intercept also enters the inflation equation, integrated by two-dimensional adaptive Gauss–Hermite quadrature (`nAGQ^2` nodes per unit).

Value

An object of class `c("iop", "iord")` or `c("iol", "iord")`, as for `oprobit()`, with additional components: `inflate` (the inflated category label), `k` (its 0-based index), `regime` (the fitted $P(s_i = 1)$), `rho` (when `correlated = TRUE`), `sigma_u` / `sigma_v` (random-intercept SDs), `loglik_uninflated`

(the plain ordered baseline), boundary, and coefficients named infl_<term> for the inflation equation. The coefficient vector is ordered outcome terms, cutpoints ("a|b"), inflation terms, rho, sigma_u, sigma_v.

Diagnostics and fixed thresholds

Every fit carries programmatic diagnostics that summary() and print() also report. object$boundary (with object$boundary_messages) is TRUE when (i) a split equation is degenerate – every fitted regime probability within 1e-6 of 0 or 1 (collapse to the plain ordered model, or a perfect split); (ii) a split equation is quasi-separated: a standardized split slope (coefficient times the covariate's SD) above 10 or a split intercept above 40 in absolute value, i.e. the split acts as a step function and its maximum-likelihood estimate may not be finite (legitimately sharp splits in the package's applications stay below 6 and 21), or a split standard error above 50 on the standardized scale or below 1e-6 times its coefficient; the same signatures (standardized coefficient above 10, or standardized standard error above 50) flag a quasi-separated outcome equation, unit dummies excepted; (iii) rho beyond 0.985 in absolute value; (iv) a random-intercept SD below 1e-3; (v) the ordered stage supplies less than 1e-6 of the inflated category's fitted probability mass, or less than 1e-2 of it while the adjacent cutpoint's standard error exceeds 50 – the inflation process absorbs every observation in that category (a hurdle-type split), so the adjacent cutpoint is at $-\infty$ (bottom) or $+\infty$ (top), or the two cutpoints bracketing a middle category coincide; the finite value the optimizer stops at is an artifact and that cutpoint's standard error is reported as NA. Degenerate split- or outcome-equation standard errors are left visible (they are the evidence of the problem and the warning names them); only a cutpoint at $\pm\infty$ is reported as NA, because the finite value the optimizer stops at carries no information. The flags also travel with glance() and the **texreg** GOF block. object$converged is FALSE when the Newton decrement at the optimum exceeds 1e-6 and the gradient (per observation) exceeds 1e-6 or the Hessian is indefinite; object$max_grad is the largest gradient component at the optimum in the optimizer's standardized parameterization (the units the convergence rule uses; the raw gradient scales with the covariates' units). converged is a statement about the optimizer – the returned point is a stationary point of the likelihood – while the boundary flags are statements about that point; a quasi-separated fit can be converged in this sense, so check both. object$flat_hessian marks a converged optimum with a near-flat direction (relative eigenvalue of the final scaled Hessian below 1e-10, weak identification; every ill-conditioned fit is flat, a flat fit whose information matrix still inverts has computable but large standard errors); object$ill_conditioned marks an information matrix with reciprocal condition number below 1e-12 (near-collinear covariates or a near-flat split), computed in the optimizer's scaled parameterization – every covariate column standardized – so that the units of the covariates do not affect it (the Hessian itself is differenced in that parameterization, so the standard errors, like the condition number, are invariant to a change of units); object$se_na lists parameters whose delta-method variance is not positive (reported as NA, never as 0). These thresholds are fixed by design (they are not tuning parameters) and were chosen from the package's applications and stress tests so that every legitimate fit in them passes and every constructed boundary case is caught; see vignette("model").

Category-specific split equations. With split = "category" every non-inflated category j gets its own split equation $s^*_{ij} = z_i'\gamma_j + u_{ij}$ (and, with correlated = TRUE, its own ρ_j): a unit whose ordered outcome would be j is "tempered" into the inflated category with probability $1 - F(z_i'\gamma_j)$, so

$$P(y_i = j) = F(z_i'\gamma_j)\,\pi_{ij}\ (j \neq k), \qquad P(y_i = k) = 1 - \sum_{j \neq k} F(z_i'\gamma_j)\,\pi_{ij}.$$

This is the generalised zero-/middle-inflated ordered probit (GZiOP, GMiOP, and their correlated versions) of Brown, Harris and Spencer (2020), which nests the common-split model when all γ_j (and ρ_j) are equal; `split_test()` tests that restriction by a score (LM) test from the common-split fit and by the likelihood-ratio test. The fit starts from its common-split counterpart (kept in `object$loglik_common`) and checks that it is not below it. Identification now rests on $J - 1$ split equations, so exclusion restrictions matter more; coefficients are named `infl_<term>:<category>` and `rho:<category>`, and the regime quantities (`predict(type = "regime")`, `$regime`) become one column per non-inflated category. Not available with random intercepts or `fe_inflation`.

References

Harris, M.N. and Zhao, X. (2007). A zero-inflated ordered probit model, with an application to modelling tobacco consumption. *Journal of Econometrics*, 141, 1073-1099.

Bagozzi, B.E. and Mukherjee, B. (2012). A mixture model for middle category inflation in ordered survey responses. *Political Analysis*, 20, 369-386.

Bagozzi, B.E., Hill, D.W., Moore, W.H. and Mukherjee, B. (2015). Modeling two types of peace: The zero-inflated ordered probit (ZiOP) model in conflict research. *Journal of Conflict Resolution*, 59, 728-752.

Bagozzi, B.E., Joo, M.M. and Mukherjee, B. (2024). Top-category inflation in ordered international relations outcomes. *Foreign Policy Analysis*, 20, orae006.

Brooks, R., Harris, M.N. and Spencer, C. (2012). Inflated ordered outcomes. *Economics Letters*, 117, 683-686.

Brown, S., Harris, M.N. and Spencer, C. (2020). Modelling category inflation with multiple inflation processes: Estimation, specification, and testing. *Oxford Bulletin of Economics and Statistics*, 82, 1342-1361.

Dale, D. and Sirchenko, A. (2021). Estimation of nested and zero-inflated ordered probit models. *Stata Journal*, 21, 3-38.

Drezner, Z. and Wesolowsky, G.O. (1990). On the computation of the bivariate normal integral. *Journal of Statistical Computation and Simulation*, 35, 101-107. Genz, A. (2004). Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Statistics and Computing*, 14, 251-260.

See Also

`oprobit()`, `ologit()`, `riop()`, `predict.iord()`, `first_difference()`, `ame()`, `vuong()`, `inflation_test()`, `ranef()`; `vignette("iop")`, `vignette("quantities")`, `vignette("model")`.

Other estimators: `ordered`

Examples

```
set.seed(2)
d <- riop(n = 800, beta = c(0.8, -0.5), gamma = c(0.4, 1), tau = c(-0.6, 0.7),
          inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, data = d, inflate = "bottom")
summary(m)
head(predict(m, type = "prob"))
head(predict(m, type = "inflated"))         # P(inflated regime)
```

```
m0 <- oprobit(y ~ x1 + x2, data = d)
vuong(m, m0)                                # inflated vs plain ordered probit

mc <- iop(y ~ x1 + x2 | z1, data = d, inflate = "bottom", correlated = TRUE)
summary(mc)
```

`inflation_test` *Test for category inflation*

Description

Compares an inflated ordered fit with the plain ordered model of the same link on the same design (refit internally), reporting the Vuong test with its AIC and BIC corrections, the information criteria, and – with `boot > 0` – a parametric-bootstrap likelihood-ratio test.

Usage

```
inflation_test(object, boot = 0, cores = 1)
```

Arguments

`object`	An `"iop"` or `"iol"` fit.
`boot`	Number of parametric-bootstrap replications for the likelihood-ratio test (0, the default, skips it). Each replication refits both models, so the cost is `boot` inflated fits and grows with the sample size (a few minutes for `bp` with `boot = 199` on one core; tens of minutes for samples of several thousand rows): use `cores =` to parallelize. A seed set before the call (`set.seed()`) makes the replications reproducible; sequential and parallel runs use different random streams. The simulated data sets are drawn independently from the fitted ordered model; a `cluster =` structure of the fit is not reproduced in the reference distribution.
`cores`	Number of parallel workers for the bootstrap refits.

Details

The plain ordered model is the limit of the inflated one as the inflation intercept tends to infinity, a point on the boundary of the parameter space, so the classical chi-squared reference for the likelihood-ratio statistic does not apply (Andrews 2001). The applied literature has used the Vuong (1989) test for this comparison (Harris and Zhao 2007; Bagozzi et al. 2015), which is what `vuong` reports; Wilson (2015) and Dale and Sirchenko (2021) object that the two models are nested (the Vuong z-test is derived for non-nested or overlapping models), and Dale and Sirchenko's Monte Carlo finds the information criteria the most reliable selectors. The parametric bootstrap sidesteps the boundary problem directly: `boot` data sets are simulated from the fitted plain ordered model, both models are refit on each, and the p-value is the share of bootstrap likelihood-ratio statistics at least as large as the observed one (with the +1 correction). It is exact up to simulation error under the null and costs `boot` inflated refits; 199 is a reasonable default for a reported test. Not available for random-intercept fits.

Value

A list of class "inflation_test" with the Vuong table (vuong), the log-likelihoods and information criteria of both models (fit), the inflated-category share, and – with boot > 0 – lr (the observed statistic, the bootstrap p-value, the number of replications, and the bootstrap statistics).

References

Andrews, D.W.K. (2001). Testing when a parameter is on the boundary of the maintained hypothesis. *Econometrica*, 69, 683-734. Wilson, P. (2015). The misuse of the Vuong test for non-nested models to test for zero-inflation. *Economics Letters*, 127, 51-53. Dale, D. and Sirchenko, A. (2021). Estimation of nested and zero-inflated ordered probit models. *Stata Journal*, 21, 3-38.

See Also

vuong(), lr_test(), split_test()

Other model comparison: classification(), compare_models(), lr_test(), parallel_test(), split_test(), vuong()

Examples

```
set.seed(5)
d <- riop(700, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "top")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "top")
inflation_test(m)

inflation_test(m, boot = 99)       # parametric-bootstrap likelihood-ratio test
```

---

iord-distribution *Distribution functions of the (inflated) ordered response*

Description

Probability mass (diord), cumulative probability (piord), quantile (qiord), and random generation (riord) for the ordered outcome of the models in this package, given the outcome linear predictor eta, the cutpoints tau, and – for inflated models – the inflation linear predictor(s) a, the inflated category k, and the error correlation rho; or taken from a fitted model (object, optionally at newdata). Categories are indexed 0, ..., J - 1 (the order of the fitted levels).

Usage

```
diord(
  x,
  eta = NULL,
  tau = NULL,
  a = NULL,
  k = NULL,
```

```
  link = c("probit", "logit"),
  rho = 0,
  object = NULL,
  newdata = NULL,
  log = FALSE
)

piord(
  q,
  eta = NULL,
  tau = NULL,
  a = NULL,
  k = NULL,
  link = c("probit", "logit"),
  rho = 0,
  object = NULL,
  newdata = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)

qiord(
  p,
  eta = NULL,
  tau = NULL,
  a = NULL,
  k = NULL,
  link = c("probit", "logit"),
  rho = 0,
  object = NULL,
  newdata = NULL,
  lower.tail = TRUE,
  log.p = FALSE
)

riord(
  n,
  eta = NULL,
  tau = NULL,
  a = NULL,
  k = NULL,
  link = c("probit", "logit"),
  rho = 0,
  object = NULL,
  newdata = NULL
)
```

Arguments

x, q	Vector of categories (0-based indices, or level labels when object is given).
eta	Outcome linear predictor $x'\beta$ (numeric vector).
tau	Cutpoints, strictly increasing (length J - 1).
a	Inflation linear predictor $z'\gamma$: a vector (common split) or an n x (J - 1) matrix (category-specific split); NULL for the plain ordered model.
k	The inflated category, a 0-based index (or a level label when object is given); NULL for the plain ordered model.
link	"probit" or "logit".
rho	Error correlation (probit only; a vector with one entry per split equation under a category-specific split).
object	Optional fitted "iord" object from which eta, tau, a, k, link, and rho are taken (fixed-effects fits only).
newdata	Optional data frame of covariate values for object (default: the estimation data).
log, log.p, lower.tail	As in the base distribution functions.
p	Vector of probabilities.
n	Number of draws (one per element of eta; a scalar n with a single eta gives n draws).

Details

With a = NULL the distribution is the plain ordered probit or logit, $P(y = j) = F(\tau_j - \eta) - F(\tau_{j-1} - \eta)$. With a vector a it is the inflated model of iop() / iol(), $P(y = j) = F(a)\,\pi_j + 1\{j = k\}[1 - F(a)]$ (with the bivariate normal rectangle probabilities when rho != 0), and with an n x (J - 1) matrix a it is the category-specific split model (split = "category"), one column per non-inflated category in order. qiord() returns the smallest category whose cumulative probability reaches p; riord() draws categories by inversion. The functions are vectorized over eta (recycling x, q, p against it); riop() remains the data-generating simulator that also draws covariates.

Value

diord: probabilities P(y = x); piord: P(y <= q); qiord: categories (0-based, or labels when object is given); riord: drawn categories. For object-based calls the length is the number of rows of newdata (or of the estimation data), recycling x/q/p.

See Also

riop() – not the low-level sampler: it simulates whole data sets (covariates included) from a chosen data-generating process, whereas riord() draws the response given linear predictors; predict.iord(), simulate.iord()

Other simulation and diagnostics: residuals.iord(), riop(), simulate.iord()

Examples

```
## plain ordered probit, one observation
diord(0:2, eta = 0.3, tau = c(-0.5, 0.8))
piord(1, eta = 0.3, tau = c(-0.5, 0.8))
## zero-inflated ordered probit: a split probability of F(0.4) = 0.66
diord(0:2, eta = 0.3, tau = c(-0.5, 0.8), a = 0.4, k = 0)
diord(0:2, eta = 0.3, tau = c(-0.5, 0.8), a = 0.4, k = 0, rho = -0.5)
qiord(c(0.1, 0.5, 0.9), eta = 0.3, tau = c(-0.5, 0.8), a = 0.4, k = 0)
table(riord(1000, eta = 0.3, tau = c(-0.5, 0.8), a = 0.4, k = 0))
## from a fitted model
data(bp)
m <- iop(violence ~ loggdppc + parliament + disaster | loggdppc + parliament, data = bp,
         inflate = "bottom")
head(diord("none", object = m))                    # P(y = none) for each observation
head(piord("repression", object = m))              # P(y <= repression)
qiord(0.5, object = m, newdata = bp[1:5, ])        # median category at five profiles
```

---

lr_test *Likelihood-ratio test for nested ordered / inflated ordered fits*

Description

For pairs in which the restricted model is an interior special case of the full one: a parallel vs a (partially) non-parallel fit of the same model, or an uncorrelated inflated probit vs its correlated version ($\rho = 0$). Not appropriate for an ordered vs an inflated ordered comparison, where the restriction sits at infinity; use `vuong()` / `inflation_test()` there.

Usage

```
lr_test(restricted, full)
```

Arguments

`restricted`, `full`
: Two `"iord"` objects on the same data; `full` must have the larger log-likelihood and more parameters.

Value

A data frame with the statistic, degrees of freedom, and p-value, of class `"lr_test"`.

See Also

`vuong()`, `compare_models()`, `parallel_test()`

Other model comparison: `classification()`, `compare_models()`, `inflation_test()`, `parallel_test()`, `split_test()`, `vuong()`

Examples

```
set.seed(4)
d <- riop(500, beta = c(0.8, -0.5), tau = c(-0.6, 0.7))
lr_test(ologit(y ~ x1 + x2, d), ologit(y ~ x1 + x2, d, parallel = FALSE))
```

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

Description

Augments a data frame with the unit-level means of the time-varying numeric covariates in `formula` (both equations of a two-part `y ~ x | z` formula), and returns the augmented formula and data. Fitting any estimator in the package on the result implements the Mundlak / correlated-random-effects specification: the coefficients on the original covariates recover the within-unit effects, while the coefficients on the unit means capture (and test) the correlation between the covariates and the unit effect. For ordered and inflated ordered models this is the recommended panel device: it avoids the incidental-parameters bias of unit dummies in short panels while preserving between-unit variation, and it pairs naturally with cluster-robust standard errors (`cluster = unit`) or a random intercept (`re = unit`).

Usage

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

Arguments

`formula`	A model formula, possibly with a `| z` inflation part.
`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, keeping the `|` structure), `data` (the augmented data frame), and `added` (the names of the unit-mean columns).

References

Mundlak, Y. (1978). On the pooling of time series and cross section data. *Econometrica*, 46, 69-85.
Wooldridge, J.M. (2010). *Econometric Analysis of Cross Section and Panel Data*, 2nd ed.

See Also

`iop()`, `oprobit()`; the `re`, `fe`, and `cluster` arguments of the estimators; `ranef()`.

Other panel tools: `ranef()`

Examples

```
set.seed(1)
d <- riop(600, beta = c(0.8, -0.5), tau = c(-0.5, 0.7), gamma = c(0.4, 1), inflate = "bottom")
d$unit <- rep(1:30, each = 20)
m <- mundlak(y ~ x1 + x2 | z1, d, unit = "unit")
m$formula
fit <- iop(m$formula, m$data, inflate = "bottom", cluster = "unit")
coef(fit)[m$added]                          # unit-mean coefficients
```

`ordered` *Ordered probit and ordered logit regression*

Description

Fits the standard ordered probit (`oprobit()`) or ordered logit (`ologit()`) model by maximum likelihood, with optional partial proportional-odds (non-parallel, "generalized ordered") effects, a unit random intercept, survey weights, offsets, and analytic, robust, or cluster-robust standard errors. These are the matched baselines for the inflated models `iop()` and `iol()`: the same engine, the same cutpoint conventions, and the same methods, so comparisons across the four are like for like.

Usage

```
oprobit(
  formula,
  data,
  parallel = TRUE,
  re = NULL,
  nAGQ = 15,
  fe = NULL,
  fe_correction = c("none", "jackknife"),
  time = NULL,
  weights = NULL,
  offset = NULL,
  se = c("analytic", "robust", "cluster", "bootstrap", "none"),
  cluster = NULL,
  nboot = 200,
  cores = 1,
  start = NULL,
  maxit = 1000,
  reltol = 1e-10
)

ologit(
  formula,
  data,
  parallel = TRUE,
  re = NULL,
```

```
  nAGQ = 15,
  fe = NULL,
  fe_correction = c("none", "jackknife"),
  time = NULL,
  weights = NULL,
  offset = NULL,
  se = c("analytic", "robust", "cluster", "bootstrap", "none"),
  cluster = NULL,
  nboot = 200,
  cores = 1,
  start = NULL,
  maxit = 1000,
  reltol = 1e-10
)
```

Arguments

formula	A model formula `y ~ x1 + x2`. The response may be an ordered factor, a factor (level order taken as the ordinal order), or a numeric/integer vector (sorted unique values define the order). The outcome equation has no intercept (the cutpoints absorb it), so an explicit `0 +` / `- 1` is ignored with a message and factors are coded as with an intercept; a set of columns that sums to a constant is rejected as rank deficient.
data	A data frame.
parallel	`TRUE` (default; all effects parallel), `FALSE` (all effects category-specific), a one-sided formula naming the terms that are held parallel, e.g. `parallel = ~ . - x2`, or `"auto"`: starting from the all-parallel fit, relax the term whose parallel restriction has the smallest likelihood-ratio p-value while that p-value is below 0.05 (a forward version of Stata's `gologit2, autofit`; the steps are stored in `object$autofit`). See also `parallel_test()`.
re	Optional column name in `data` identifying units that receive a random intercept in the outcome equation.
nAGQ	Number of adaptive Gauss–Hermite quadrature nodes per random intercept (default 15).
fe	Optional column name in `data` identifying units that receive a fixed effect (a dummy per unit, reference level omitted) in the outcome equation. Units whose response is constant at an extreme category have no finite fixed effect and are dropped with a message; a covariate that does not vary within units is collinear with the dummies and is refused by name (keep it with `re =` or `mundlak()` instead). Maximum-likelihood unit dummies carry incidental-parameters bias when units have few observations (Greene 2004); the fit warns when the median unit has fewer than 10. In the package's Monte Carlo (`system.file("mc", package = "iop")`: ordered probit, 100 units, a covariate correlated with a N(0,1) unit effect, 100 replications) the bias of that covariate's coefficient is +26 / +11 / +5 / +2.5 percent at T = 4 / 8 / 16 / 32 with unit dummies, +21 / +11 / +6 / +3 percent with a random intercept, +12 percent throughout when pooled, and within 0.3 percent at every T with the `mundlak()` device – the recommended route for

short panels; reserve `fe =` for long ones. Computationally the dummies enter the parameter vector one per unit, and the exact-Hessian Newton polish and the covariance cost grow roughly with the square of the number of parameters (about 0.5 s at 20 units, 3 s at 100, 18 s at 300 for a plain ordered probit on 4,000 rows; the multistart of an inflated model multiplies this); for panels with many hundreds of units prefer `re =` or `mundlak()`.

`fe_correction` With `fe`, `"jackknife"` applies the split-panel jackknife of Dhaene and Jochmans (2015): the model is refit on the two half-panels of every unit (first and second half of the observations ordered by `time`, or by row order if `time` is missing) and the common parameters are bias-corrected as `2 * full - (half1 + half2) / 2`, removing the leading incidental-parameters bias; the uncorrected estimates are kept in `object$coefficients_uncorrected`, the half-panel estimates and log-likelihoods in `object$jackknife`, and the full-sample covariance is reported: Dhaene and Jochmans (2015, Section 2) show that the split-panel jackknife removes the leading bias without changing the first-order asymptotic variance, so the full-sample covariance is the asymptotically valid one for the corrected estimator (its finite-sample variance is somewhat larger, which `se = "bootstrap"` cannot assess under `fe`; treat the intervals as approximate). Units with a single observation do not enter the half-panels. The correction is only as good as the half-panel fits: when a covariate is identified mainly by a within-unit trend, or the half-panels are very short, the half-panel estimates are noisy and the corrected estimate inherits that noise – inspect `object$jackknife$half_coefficients` before reporting it.

`time` Optional column name ordering the observations within units (used by `fe_correction = "jackknife"`).

`weights` Optional weights: a column name in `data` or a numeric vector aligned to its rows. They enter as frequency weights: a fit with weight 2 equals a fit on duplicated rows, in the estimates, the model-based standard errors, the information criteria (whose `n` is the weight total), the Vuong statistics, and the averaged quantities of interest. Survey (sampling, probability) weights are not frequency weights: with them the default model-based standard errors are not the design-based ones, so use `se = "robust"` (the pseudo-maximum-likelihood sandwich) or `cluster =` for the primary sampling units.

`offset` Optional offset on the latent scale of the outcome equation: a column name in `data` or a numeric vector.

`se` Standard errors: `"analytic"` (inverse observed information, default), `"robust"` (heteroskedasticity-consistent sandwich over observations, or over units for random-intercept fits), `"cluster"` (cluster-robust; needs `cluster`), `"bootstrap"` (nonparametric bootstrap over the estimation rows – or over the clusters when `cluster` is given, or over the units of a random-intercept fit – with `nboot` refits from the full-sample estimate; the replicate estimates are kept in `object$boot` and `confint.iord()` can then report percentile intervals), or `"none"`. The bootstrap is not available with `fe`. Replicates whose split equation (or `rho`) runs to a boundary are counted in `object$boot$n_boundary` and trigger a warning: their estimates inflate the bootstrap standard errors of the affected block, for which the percentile intervals are the more robust summary. For `"bootstrap"`, a seed set before the call (`set.seed()`) makes the resamples reproducible; runs with `cores = 1` and `cores > 1` use different random streams.

`cluster` Optional cluster identifier (a column name in `data` or a vector aligned to its rows); supplying it selects `se = "cluster"` unless `se` is given explicitly (with `se = "bootstrap"` it defines the resampling blocks). With a random intercept, clusters must nest the units.

`nboot` Number of bootstrap refits for `se = "bootstrap"` (default 200). Each refit costs about one fit of the model (roughly a third of a second per refit for the bundled `bp` ZiOP), so budget a minute or two at the default on data of that size, and use `cores`.

`cores` Number of parallel workers for the bootstrap refits (a PSOCK cluster via the **parallel** package; default 1).

`start` Optional starting values on the internal parameter scale (see `object$theta`); rarely needed.

`maxit`, `reltol` Controls for the BFGS stage of the optimizer, passed to `stats::optim()`; the exact-Hessian Newton polish that follows runs up to 25 further iterations regardless of `maxit` (so `maxit` bounds the quasi-Newton phase, not the total). Results are insensitive to `maxit` above a few hundred; `maxit = 0` is reported as non-converged.

Details

The latent-variable model is $y_i^* = x_i'\beta + \epsilon_i$ with $y_i = j$ when $\tau_{j-1} < y_i^* \le \tau_j$ ($\tau_0 = -\infty$, $\tau_J = \infty$), so $P(y_i \le j) = F(\tau_j - x_i'\beta)$, with F the standard normal (probit) or logistic (logit) distribution function. The outcome equation has no intercept (it is absorbed by the cutpoints), matching `MASS::polr()`.

With `parallel = FALSE`, or a one-sided formula naming the terms held parallel (all others relaxed; `~ . - x2` relaxes only `x2`), the relaxed terms get a separate coefficient per cutpoint, $P(y_i \le j) = F(\tau_j - x_i'\beta - \tilde{x}_i'\beta_j)$, the partial proportional-odds / generalized ordered model of Peterson and Harrell (1990) and Williams (2006). Such fits can imply negative cell probabilities when the category-specific curves cross; the fit warns when that happens at the optimum.

With `re = "unit"` a random intercept $u_g \sim N(0, \sigma_u^2)$ for the units of that column enters the latent equation, $y_{ig}^* = x_{ig}'\beta + u_g + \epsilon_{ig}$, and is integrated out by adaptive Gauss–Hermite quadrature with `nAGQ` nodes per unit (the mode and curvature of each unit's integrand re-center and re-scale the nodes), as in `ordinal::clmm()`. Reported probabilities and first differences are then population-averaged (marginal over the random intercept); conditional probabilities at $u = 0$ and the empirical-Bayes unit effects are available through `predict.iord()` and `ranef()`.

For panels, `mundlak()` adds unit means of the covariates (correlated random effects), which pairs naturally with `re =` or with `cluster =`.

Value

An object of class `c("oprobit", "iord")` or `c("ologit", "iord")`: a list with `coefficients` (outcome coefficients, then the cutpoints named `"a|b"`, then `sigma_u` for a random-intercept fit), `vcov`, `loglik`, `fitted.values` (the n x J matrix of category probabilities), `cutpoints`, `converged`, `boundary`, and the design pieces used by the methods. See `summary.iord()`, `predict.iord()`, `first_difference()`, `compare_models()`.

Missing values, refits, and new data

Rows with a missing value in any variable of either equation (or in `weights`, offsets, `cluster`, `re`, `fe`) are dropped before fitting – there is no `na.action` argument – and `object$data`, `fitted()`, `residuals()`, and `predict()` without `newdata` refer to the retained rows (so they are not padded to the original row positions as with `na.exclude`). The stored call makes `update.iord()` work as usual (`update(fit, . ~ . + x3)`, `update(fit, data = subset)`; for the two-part formulas `update(fit, . ~ . + x3 | .)` changes the outcome equation and `update(fit, . ~ . | . + z2)` the inflation equation). In `predict.iord()`, `newdata` must contain every covariate of both equations; extra columns are ignored, unseen factor levels are an error, and rows with missing covariates give `NA` predictions.

References

McKelvey, R.D. and Zavoina, W. (1975). A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology*, 4, 103-120.

Peterson, B. and Harrell, F.E. (1990). Partial proportional odds models for ordinal response variables. *Applied Statistics*, 39, 205-217.

Williams, R. (2006). Generalized ordered logit/partial proportional odds models for ordinal dependent variables. *Stata Journal*, 6, 58-82.

Greene, W. (2004). The behaviour of the maximum likelihood estimator of limited dependent variable models in the presence of fixed effects. *Econometrics Journal*, 7, 98-119.

Dhaene, G. and Jochmans, K. (2015). Split-panel jackknife estimation of fixed-effect models. *Review of Economic Studies*, 82, 991-1030.

Dale, D. and Sirchenko, A. (2021). Estimation of nested and zero-inflated ordered probit models. *Stata Journal*, 21, 3-38.

See Also

`iop()`, `iol()`, `predict.iord()`, `first_difference()`, `vuong()`, `parallel_test()`, `mundlak()`, `ranef()`; `vignette("iop")`, `vignette("panels")`.

Other estimators: `inflated`

Examples

```
set.seed(1)
n <- 500; x <- rnorm(n); z <- rbinom(n, 1, 0.5)
ystar <- 0.8 * x - 0.5 * z + rnorm(n)
y <- cut(ystar, c(-Inf, -0.7, 0.5, Inf), labels = c("low", "mid", "high"), ordered_result = TRUE)
d <- data.frame(y, x, z)
m1 <- oprobit(y ~ x + z, data = d)
summary(m1)
m2 <- ologit(y ~ x + z, data = d, se = "robust")
compare_models(probit = m1, logit = m2)
head(predict(m1, type = "prob"))
```

`parallel_test` *Likelihood-ratio tests of the parallel-regression assumption*

Description

For every outcome-equation term currently held parallel, fits the model with that term's effect made category-specific and reports the likelihood-ratio test of the parallel restriction (the likelihood-based counterpart of the Brant test), together with the omnibus test that relaxes every term at once. Works for all four models; in inflated models the inflation equation is unchanged.

Usage

```
parallel_test(object)
```

Arguments

`object`	An `"iord"` object.

Value

A data frame of class `"parallel_test"`: one row per term plus an `all terms` row, with `LR`, `df`, `p.value`, and the log-likelihood of the relaxed fit.

See Also

`lr_test()`, the `parallel` argument of `oprobit()`

Other model comparison: `classification()`, `compare_models()`, `inflation_test()`, `lr_test()`, `split_test()`, `vuong()`

Examples

```
set.seed(1)
d <- riop(700, beta = c(0.8, -0.5), tau = c(-0.6, 0.4, 1.3))
parallel_test(ologit(y ~ x1 + x2, d))
## automatic relaxation: the steps are stored on the fit
m_auto <- ologit(y ~ x1 + x2, d, parallel = "auto")
m_auto$autofit$relaxed
```

`plot.iop_fd` *Plot first differences or average marginal effects*

Description

A dot-and-whisker display of the estimates and their intervals, one point per component (category / regime), grouped by variable for `ame()` tables.

Usage

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

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

Arguments

`x`	An object from `first_difference()` or `ame()`.
`...`	Passed to `graphics::plot()` (e.g. `main`).

Value

`x`, invisibly.

See Also

`first_difference()`, `ame()`

Other quantities of interest: `ame()`, `first_difference()`, `predict.iord()`

Examples

```
set.seed(3)
d <- riop(500, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom")
plot(first_difference(m, "x1", -1, 1))
plot(ame(m))
```

predict.iord *Predictions from an ordered / inflated ordered fit*

Description

Predictions from an ordered / inflated ordered fit

Usage

```
## S3 method for class 'iord'
predict(
  object,
  newdata = NULL,
 type = c("prob", "prob_conditional", "class", "prob_outcome", "regime", "inflated",
    "posterior", "zeros", "mean", "cumulative", "link", "link_inflation"),
  offset = NULL,
  offset_inflation = NULL,
  se.fit = FALSE,
  ...
)
```

Arguments

object	An "iord" object.
newdata	Optional data frame of covariate profiles; if omitted, the estimation data are used. Every covariate of both equations must be present (a missing one is an error); extra columns are ignored; a factor level not seen at estimation is an error; rows with missing covariate values give NA predictions.
type	What to return:

- "prob" (default): the n x J matrix of category probabilities $P(y = j)$ (for random-intercept fits: marginal over the random intercept, i.e. population-averaged);
- "prob_conditional": for random-intercept fits, the category probabilities at $u = 0$ (a median unit); identical to "prob" otherwise;
- "class": the modal category (a factor with the response levels);
- "prob_outcome": the ordered-stage probabilities $P(y = j \mid s = 1)$ (equal to "prob" for uninflated models);
- "regime": $P(s = 1)$, the probability of the ordered (non-inflated) regime (1 for uninflated models);
- "inflated": $P(s = 0) = 1 - P(s = 1)$;
- "posterior": the posterior probability that an observation in the inflated category is an inflated case, $P(s = 0 \mid y = k) = (1 - P(s = 1))/P(y = k)$, and 0 for observations in other categories (needs the response: available for the estimation data, or when newdata contains it); under a category-specific split, the probability that the observation was tempered into k from another category;

- "zeros": the two components of $P(y = k)$ – an n x 2 matrix with columns inflation (through the inflation process: $1 - P(s = 1)$ under a common split, or the tempered mass under a category-specific split) and ordered (through the ordered stage: $P(s = 1, y^* = k)$), which sum to $P(y = k)$ – Harris and Zhao's (2007) two types of zeros;
- "mean": the expected category index $\sum_j j\, P(y = j)$ on the 0, ..., J-1 scale;
- "cumulative": the n x (J-1) matrix of $P(y \leq j)$;
- "link": the outcome-equation linear predictor $x'\beta$ (a matrix with one column per cutpoint when any term is non-parallel);
- "link_inflation": the inflation-equation linear predictor $z'\gamma$ (one column per split equation under a category-specific split).

Under a category-specific split (split = "category"), "regime" and "inflated" return an n x (J-1) matrix with one column per non-inflated category ($P(s_j = 1)$ for a unit whose ordered outcome would be j).

offset, offset_inflation
: Optional offsets for newdata (column names in newdata or numeric vectors); the fitted offsets are used when newdata is NULL.

se.fit
: If TRUE (probability types only, including "zeros", "mean", and "cumulative"), also return delta-method standard errors: a list with fit and se.fit of the same shape.

...
: Unused.

Value

A numeric vector, matrix, or factor as described under type; with se.fit = TRUE, a list fit / se.fit.

See Also

first_difference() and ame() for changes in these probabilities; fitted() (the "prob" matrix at the estimation data); ranef() for random-intercept fits.

Other quantities of interest: ame(), first_difference(), plot.iop_fd()

Examples

```
data(bp)
m <- iop(violence ~ loggdppc + parliament + disaster | loggdppc + parliament + disaster,
         data = bp, inflate = "bottom")
head(predict(m), 3)                                  # P(y = j)
head(predict(m, type = "class"), 3)                  # modal category
head(predict(m, type = "prob_outcome"), 3)           # P(y = j | ordered regime)
summary(predict(m, type = "inflated"))               # P(structurally peaceful)
## posterior probability that an observed "none" is a structural zero
summary(predict(m, type = "posterior")[bp$violence == "none"])
## covariate profiles, with delta-method standard errors
nd <- data.frame(loggdppc = c(7, 9), parliament = 0, disaster = 0)
predict(m, newdata = nd, se.fit = TRUE)
predict(m, newdata = nd, type = "regime")
```

pta *Escape-flexibility provisions in preferential trade agreements*

Description

Agreement-level data from Baccini, Dur and Elsig (2015) on the number of flexibility provisions (safeguards, suspension of tariff cuts, anti-dumping and countervailing duties) in 559 preferential trade agreements, 1945–2009. The top category (all four provisions) holds 46 percent of agreements and mixes step-by-step insurance-seeking members with members adopting maximum flexibility at once for protectionist reasons: the top-inflated ordered outcome analyzed by Bagozzi, Joo and Mukherjee (2024, Table 1). Rows are the complete cases of that specification.

Usage

```
data(pta)
```

Format

A data frame with 559 rows and 11 variables:

agreement Agreement name.

year Year of signature.

flexibility Number of escape-flexibility provisions, 0–4 (the ordered outcome).

depth Depth of trade-liberalization commitments (index).

gdp Log GDP of the member states.

gdppc Log GDP per capita.

trade Log imports plus exports.

democracy 1 if the least democratic member has a Polity IV score above 5.

gattwto 1 if all members are GATT/WTO members.

members Number of member states.

democratization 1 if at least one member democratized over the previous ten years.

Provenance and terms

Taken from the public replication archive of the cited article and redistributed here, with the variables renamed and recoded as documented in `data-raw/make_data.R`, so that the published results can be reproduced; the archive states the original terms of use.

Source

Baccini, L., Dur, A. and Elsig, M. (2015). The politics of trade agreement design: Revisiting the depth-flexibility nexus. *International Studies Quarterly*, 59, 765-775 (replication data); as analyzed in Bagozzi, B.E., Joo, M.M. and Mukherjee, B. (2024). Top-category inflation in ordered international relations outcomes. *Foreign Policy Analysis*, 20, orae006 (replication archive, Harvard Dataverse).

See Also

iop(); vignette("iop") and vignette("model") analyze these data.

Other datasets: bp, repression

Examples

```
data(pta)
table(pta$flexibility)
m <- iop(flexibility ~ depth * democracy + gdp + gdppc + trade + gattwto + members +
           democratization | gdp + gdppc + democracy + democratization,
         data = pta, inflate = "top")
summary(m)
```

---

ranef *Empirical-Bayes unit effects of a random-intercept fit*

Description

Posterior modes and posterior standard deviations of the unit random intercept(s), from the adaptive quadrature at the fitted parameters.

Usage

```
ranef(object, ...)

## S3 method for class 'iord'
ranef(object, ...)
```

Arguments

object	An "iord" object fit with re.
...	Unused.

Value

A data frame with one row per unit: unit, u (outcome-equation intercept), u_sd, and for re_inflation = TRUE also v, v_sd.

See Also

The re, re_inflation, and nAGQ arguments of oprobit() and iop(); predict.iord() for marginal and conditional probabilities.

Other panel tools: mundlak()

Examples

```
set.seed(3)
d <- riop(300, beta = c(0.8, -0.5), tau = c(-0.5, 0.7))
d$unit <- rep(1:15, each = 20)
m <- oprobit(y ~ x1 + x2, d, re = "unit", nAGQ = 7)
head(ranef(m))
## marginal (population-averaged) vs conditional (u = 0) probabilities
head(cbind(predict(m)[, 1], predict(m, type = "prob_conditional")[, 1]), 3)
```

`repression` *State repression of nonviolent campaigns*

Description

Campaign-year data from Girod, Stewart and Walters (2018) on the intensity of state repression against nonviolent anti-government campaigns (0 = none to 3 = extreme). The top category holds 77 percent of observations and mixes targeted repression of campaign activity with indiscriminate repression of non-campaign actors: the top-inflated ordered outcome analyzed in the appendix of Bagozzi, Joo and Mukherjee (2024). Rows are campaign-years with nonviolent campaign activity and complete cases of that specification.

Usage

```
data(repression)
```

Format

A data frame with 367 rows and 8 variables:

campaign Campaign name.

country Country.

year Year.

repression Repression intensity, 0–3 (the ordered outcome).

negxpol Authoritarianism (negative Polity score).

oilrent Lagged log oil rents per capita.

dom_media Domestic media salience of the campaign.

civil_war 1 if an internal armed conflict is ongoing.

Provenance and terms

Taken from the public replication archive of the cited article and redistributed here, with the variables renamed and recoded as documented in `data-raw/make_data.R`, so that the published results can be reproduced; the archive states the original terms of use.

Source

Girod, D.M., Stewart, M.A. and Walters, M.R. (2018). Mass protests and the resource curse: The politics of demobilization in rentier autocracies. *Conflict Management and Peace Science*, 35, 503-522 (replication data); as analyzed in Bagozzi, B.E., Joo, M.M. and Mukherjee, B. (2024), *Foreign Policy Analysis*, 20, orae006, appendix Table A.5.

See Also

`iop()`; `vignette("iop")` analyzes these data.

Other datasets: `bp`, `pta`

Examples

```
data(repression)
table(repression$repression)
m <- iop(repression ~ negxpol * oilrent | dom_media + civil_war,
         data = repression, inflate = "top")
summary(m)
```

---

residuals.iord *Residuals for ordered / inflated ordered fits*

Description

Ordinal outcomes have no canonical residual; two are offered. `"response"` is the observed category index minus its expected index under the fitted probabilities (on the 0, 1, ..., J-1 scale). `"pearson"` divides that by the fitted standard deviation of the index. For model checking prefer the simulation route: `simulate()` feeds `DHARMa::createDHARMa()`; see `simulate.iord()`.

Usage

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

Arguments

`object`	An `"iord"` object.
`type`	`"response"` or `"pearson"`.
`...`	Unused.

Value

A numeric vector.

See Also

simulate.iord() for simulated-residual diagnostics with **DHARMa**; fitted() for the fitted category probabilities.

Other simulation and diagnostics: iord-distribution, riop(), simulate.iord()

Examples

```
data(bp)
m <- oprobit(violence ~ loggdppc + parliament + disaster, data = bp)
r <- residuals(m, type = "pearson")
summary(r)
## the simulation route, preferred for model checking
s <- simulate(m, nsim = 5)
dim(s)
```

---

riop *Simulate data from an (inflated) ordered probit or logit process*

Description

Draws a data frame from the data-generating process of oprobit(), ologit(), iop(), or iol(): an ordered latent equation $y^* = x'\beta + \epsilon$ cut at tau, and, when inflate is given, a split equation $s^* = z'\gamma + u$ that sends units with $s^* \leq 0$ to the inflated category. Errors are standard normal (link = "probit") or standard logistic ("logit"); with rho != 0 (probit only) (u, ϵ) are bivariate normal with that correlation.

Usage

```
riop(
  n,
  beta,
  tau,
  gamma = NULL,
  inflate = NULL,
  rho = 0,
  link = c("probit", "logit"),
  X = NULL,
  Z = NULL,
  labels = NULL
)
```

Arguments

n	Number of observations (ignored when X is supplied).
beta	Outcome coefficients (length p).
tau	Cutpoints, strictly increasing (length J - 1).

gamma	Inflation coefficients, intercept first (length 1 + q); NULL for an uninflated process. A matrix with 1 + q rows and J - 1 columns gives every non-inflated category its own split equation (one column per non-inflated category, in category order): the generalised (category-specific split) process of Brown, Harris and Spencer (2020) fitted by split = "category".
inflate	The inflated category: "bottom", "middle", "top", or a 0-based category index; NULL for an uninflated process.
rho	Error correlation (probit with inflation only); a vector of length J - 1 (one per split equation) when gamma is a matrix.
link	"probit" or "logit".
X, Z	Optional covariate matrices (without intercept for X; without intercept for Z, the intercept is added).
labels	Optional category labels (length J); default 0:(J-1).

Details

Covariates are drawn as: x1, ..., x<p> standard normal, z1, ..., z<q> standard normal (the inflation equation also gets an intercept, the first element of gamma), unless X / Z matrices are supplied.

Value

A data frame with the response y (an ordered factor when labels is given, otherwise an integer 0..J-1), the covariates, and two attributes: "regime" (the latent s indicator; 1 = ordered regime) and "truth" (the parameter list).

See Also

iop(), iol(), simulate.iord(); riord() is the low-level response sampler given linear predictors (the d/p/q/r family), whereas riop() draws whole data sets including the covariates.

Other simulation and diagnostics: iord-distribution, residuals.iord(), simulate.iord()

Examples

```
d <- riop(500, beta = c(1, -0.5), tau = c(-0.5, 0.8), gamma = c(0.3, 1), inflate = "bottom")
table(d$y)
mean(attr(d, "regime") == 0)         # share of inflated-regime units
## a middle-inflated logit process with labelled categories
d2 <- riop(500, beta = c(0.8, -0.5), tau = c(-0.8, 0.8), gamma = c(0.2, 1),
          inflate = "middle", link = "logit", labels = c("disagree", "neutral", "agree"))
table(d2$y)
```

`simulate.iord` *Simulate responses from a fitted ordered / inflated ordered model*

Description

Draws `nsim` replicate response vectors from the fitted category probabilities at the estimation data, in the format of `stats::simulate()`. The main consumer is simulated-residual diagnostics:

```
sims <- simulate(fit, nsim = 250)
DHARMa::createDHARMa(simulatedResponse = as.matrix(sims),
                     observedResponse  = fit$y,
                fittedPredictedResponse = as.numeric(fitted(fit) %*% (0:(fit$J - 1))),
                     integerResponse = TRUE)
```

Responses are returned as integers 0..J-1 (the internal category index; `fit$levels` maps them to labels), which is what DHARMa expects.

Usage

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

Arguments

`object`	An `"iord"` object.
`nsim`	Number of replicate response vectors.
`seed`	Optional seed, handled as in `stats::simulate()`.
`...`	Unused.

Value

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

See Also

`residuals.iord()`, `riop()` for drawing from a chosen data-generating process, `predict.iord()`.

Other simulation and diagnostics: `iord-distribution`, `residuals.iord()`, `riop()`

Examples

```
set.seed(1)
d <- riop(300, beta = c(0.8, -0.4), tau = c(-0.5, 0.6), gamma = c(0.5, 0.8), inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, data = d, inflate = "bottom")
s <- simulate(m, nsim = 3)
table(s$sim_1)
```

split_test *Test of a common versus category-specific split equation*

Description

Tests the restriction that the inflation (split) equation is the same for every non-inflated category – the ZiOP / MiOP / TiOP – against the category-specific split of Brown, Harris and Spencer (2020) (`split = "category"`): a Lagrange-multiplier (score) test computed from the common-split fit alone, and the likelihood-ratio test from the category-specific refit (or from the stored common-split counterpart when `object` was fit with `split = "category"`). Under the null both statistics are chi-squared with `(J - 2)` times the number of split coefficients degrees of freedom (plus `J - 2` for the correlations when `correlated = TRUE`).

Usage

```
split_test(object, lr = TRUE, information = c("opg", "hessian"))
```

Arguments

`object`	An `"iop"` or `"iol"` fit with `split = "common"` (the default) or `split = "category"`.
`lr`	Also compute the likelihood-ratio test (refits the category-specific model when `object` has a common split).
`information`	Information-matrix estimate for the LM statistic: `"opg"` (outer product of gradients, as in Brown, Harris and Spencer; default) or `"hessian"` (observed information; `NA` when it is not positive definite at the restricted estimates).

Details

The LM statistic is $s' I^{-1} s$, with s the score of the category-specific model evaluated at the common-split estimates (every split equation set equal to the common one) and I an estimate of the information matrix at that point. It needs no refit. `information = "opg"` (default) uses the outer product of the per-observation scores, as Brown, Harris and Spencer (2020) do; `"hessian"` uses the observed information (the negative Hessian of the category-specific log-likelihood at the restricted estimates). Under the null both are correctly sized (in a 120-replication simulation at n = 1500 both reject 4.2 percent of the time at the 5 percent level, as does the LR) and the Hessian form tracks the LR more closely. But the LM is a *local* test: it evaluates the information where the restriction holds, and when the restriction is strongly violated neither estimate is representative there – on the bp application the OPG statistic is 267 where the LR is 57 (outer-product information is known to over-reject; Davidson and MacKinnon 1983), and the observed information is not even positive definite at the restricted estimates (the Hessian-form statistic is then reported as `NA` with a note). The LR test refits the category-specific model, which is nested in the restricted one as an interior restriction, so the classical reference applies (unlike the ordered-versus-inflated comparison of `inflation_test()`). Treat the LM as a screening statistic and report the LR.

Value

A data frame of class "split_test" with one row per test (LM, and LR when requested): statistic, df, p.value, and the log-likelihoods of the two models on the LR row; the information attribute records the LM variant and note any caveat.

References

Davidson, R. and MacKinnon, J.G. (1983). Small sample properties of alternative forms of the Lagrange multiplier test. *Economics Letters*, 12, 269-275.

Brown, S., Harris, M.N. and Spencer, C. (2020). Modelling category inflation with multiple inflation processes: Estimation, specification, and testing. *Oxford Bulletin of Economics and Statistics*, 82, 1342-1361.

See Also

iop() (split), inflation_test(), lr_test()

Other model comparison: classification(), compare_models(), inflation_test(), lr_test(), parallel_test(), vuong()

Examples

```
set.seed(8)
G <- cbind(c(0.3, 1), c(0.3, 1))            # equal split equations: the common model holds
d <- riop(800, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = G, inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom")
split_test(m)

G2 <- cbind(c(0.3, 1.2), c(1.0, 0.2))            # different split equations
d2 <- riop(800, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = G2, inflate = "bottom")
split_test(iop(y ~ x1 + x2 | z1, d2, inflate = "bottom"))
```

---

summary.iord *Summarize an ordered / inflated ordered fit*

Description

The printed summary shows the outcome equation, the cutpoints, and – for inflated models – the inflation equation (with the observed share of the inflated category and the mean fitted probability of the ordered regime) and the error correlation, followed by the random-intercept and fixed-effect summaries where relevant and the fit statistics. Unit fixed-effect dummies are counted but not printed (see coef()).

Usage

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

Arguments

object	An "iord" object.
...	Unused.

Value

An object of class "summary.iord" holding the coefficient table (coefficients: estimate, standard error, z, p) with a parallel block vector naming each row's block ("outcome", "cutpoint", "inflation", "rho", "random", "fixed effect"), plus the fit statistics (loglik, aic, bic, df, n, se.type, converged).

See Also

confint.iord(), vcov.iord(), tidy.iord() for a data-frame version of the table, compare_models().

Other inference methods: confint.iord(), vcov.iord()

Examples

```
data(bp)
m <- iop(violence ~ loggdppc + parliament + disaster | loggdppc + parliament + disaster,
         data = bp, inflate = "bottom")
s <- summary(m)
s
s$coefficients[s$block == "inflation", ]
```

---

tidy.iord *Tidy an ordered / inflated ordered fit (broom method)*

Description

Tidy an ordered / inflated ordered fit (broom method)

Usage

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

Arguments

x	An "iord" object.
conf.int	If TRUE, add conf.low/conf.high (see confint.iord()).
conf.level	Confidence level for the interval.
...	Unused.

Value

A data frame with one row per parameter: `term`, `estimate`, `std.error`, `statistic`, `p.value`, and `component` (one of `"outcome"`, `"cutpoint"`, `"inflation"`, `"rho"`); inflation terms keep their `infl_` prefix so both equations' terms stay distinct in tables.

See Also

`summary.iord()`; `modelsummary::modelsummary()` consumes these methods, and `texreg::screenreg()` uses the package's `extract()` method; both carry the fit's `converged`/`boundary`/`ill_conditioned` flags.

Other broom methods: `augment.iord()`, `glance.iord()`

Examples

```
if (requireNamespace("broom", quietly = TRUE)) {
  data(bp)
 m <- iop(violence ~ loggdppc + disaster | loggdppc + disaster, data = bp, inflate = "bottom")
  broom::tidy(m, conf.int = TRUE)
  broom::glance(m)
  head(broom::augment(m))
}
```

`update.iord` *Update and refit an iord model*

Description

`update()` modifies the stored call and refits. For the two-part formulas of `iop()` and `iol()` the `formula.` argument may itself be two-part: `. ~ . + x3 | .` adds `x3` to the outcome equation and keeps the inflation equation, `. ~ . | . + z2` changes only the inflation equation, and a one-part `. ~ . + x3` changes the outcome equation and keeps the inflation equation (`stats::update.formula()` alone would fold the `|` into the outcome part). Other arguments replace or add to the stored call as in `stats::update()`.

Usage

```
## S3 method for class 'iord'
update(object, formula., ..., evaluate = TRUE)
```

Arguments

`object`	a fitted `"iord"` model.
`formula.`	changes to the formula, as in `stats::update.formula()`, optionally two-part (see Description).
`...`	further arguments to the estimator, replacing those in the stored call (`data = `, `se = `, `inflate = `, ...).
`evaluate`	if `FALSE`, return the modified call unevaluated.

Value

the refitted model (or the modified call). As with every `update()` method, the refit evaluates the stored call in the calling environment, so the data object named in the original call must exist there; a fit restored with `readRDS()` in a fresh session can be updated only once that object is available again (`predict()`, `summary()`, and the table methods need nothing beyond the fit itself).

Examples

```
data(bp)
m <- oprobit(violence ~ loggdppc + parliament, data = bp)
m2 <- update(m, . ~ . + disaster)
m3 <- update(m, se = "robust")
## two-part formulas: change one equation at a time
mi <- iop(violence ~ loggdppc | parliament, data = bp, inflate = "bottom")
mi2 <- update(mi, . ~ . + disaster | .)     # outcome equation only
update(mi, . ~ . | . + disaster, evaluate = FALSE)   # the call that would refit
```

---

vcov.iord *Covariance matrix of an ordered / inflated ordered fit*

Description

Covariance matrix of an ordered / inflated ordered fit

Usage

```
## S3 method for class 'iord'
vcov(object, scale = c("natural", "internal"), ...)
```

Arguments

`object`	An `"iord"` object.
`scale`	`"natural"` (default) returns the covariance of the reported coefficients (outcome coefficients, cutpoints, inflation coefficients, `rho`); `"internal"` returns the covariance on the optimizer's scale (cutpoint increments, `atanh(rho)`), which the delta-method quantities of interest use.
`...`	Unused.

Value

A symmetric matrix with dimnames.

See Also

`confint.iord()`, `summary.iord()`, the `se` and `cluster` arguments of `oprobit()` and `iop()`.

Other inference methods: `confint.iord()`, `summary.iord()`

Examples

```
set.seed(1)
d <- riop(500, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
m <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom")
round(sqrt(diag(vcov(m))), 3)                  # standard errors on the natural scale
colnames(vcov(m, scale = "internal"))         # the optimizer's parameterization
m_cl <- iop(y ~ x1 + x2 | z1, d, inflate = "bottom", se = "robust")
round(sqrt(diag(vcov(m_cl))), 3)               # sandwich standard errors
```

---

vuong *Vuong test for non-nested (or boundary-nested) model comparison*

Description

The Vuong (1989) test compares two models fit to the same observations by the mean and dispersion of the per-observation log-likelihood differences. The inflated ordered models reduce to the plain ordered model only in the limit $z'\gamma \to \infty$ (every unit in the ordered regime), a point outside the parameter space, so the likelihood-ratio test has no standard distribution there and the Vuong test is the comparison used in this literature (Harris and Zhao 2007; Bagozzi et al. 2015). The raw statistic and the AIC- and BIC-corrected versions (which penalize the model with more parameters) are reported, as in `pscl::vuong()`.

Usage

```
vuong(m1, m2)
```

Arguments

`m1`, `m2`	Two fitted `"iord"` objects on the same data (same response and observations).

Value

An object of class `"vuong"`: a data frame with one row per correction (`raw`, `AIC`, `BIC`) giving the statistic, the one-sided p-value that `m1` is closer to the truth, the one-sided p-value that `m2` is, and the two-sided p-value; positive statistics favor `m1`.

References

Vuong, Q.H. (1989). Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57, 307-333.

See Also

`inflation_test()`, `lr_test()`, `compare_models()`

Other model comparison: `classification()`, `compare_models()`, `inflation_test()`, `lr_test()`, `parallel_test()`, `split_test()`

Examples

```
set.seed(3)
d <- riop(600, beta = c(0.8, -0.5), tau = c(-0.6, 0.7), gamma = c(0.3, 1), inflate = "bottom")
vuong(iop(y ~ x1 + x2 | z1, d, inflate = "bottom"), oprobit(y ~ x1 + x2, d))
```

Index