Package 'GIR'

November 17, 2026

Type Package

Title Gini Index Regression with Bayesian Bootstrap Inference

Version 0.2.1

Description Fits linear Gini Index Regression models using a least-squares series estimator of the conditional mid-distribution function. Provides a two-stage Bayesian bootstrap, centered confidence intervals, bootstrap standard errors, tests of smooth coefficient transformations, conditional Gini predictions and a GIR-specific goodness-of-fit measure. The estimator and two-stage bootstrap follow Bhardwaj and Diaz-Villarejo (2026), ``Gini Index Regression'' (working paper, Universidad Carlos III de Madrid) <https://ddiazvil.com/assets/papers/JMP_GIR_DiazVillarejo.pdf>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports splines, stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author David Diaz-Villarejo [aut, cre]

Maintainer David Diaz-Villarejo <ddiazvil@eco.uc3m.es>

Repository CRAN

Date/Publication 2026-11-17 16:30:02 UTC

Contents

bboot.gir 2
gir 6
GIR-validation 10
midcdf 12
predict.gir 13

print.gir 14
summary.gir_bootstrap 16

Index **18**

`bboot.gir` *Bayesian Bootstrap for Gini Index Regression*

Description

Refit both GIR stages using common normalized exponential multipliers, and obtain centered/basic intervals, bootstrap standard errors, covariance, and tests of smooth coefficient transformations.

Usage

```
bboot.gir(
  object,
  R = 999L,
  level = 0.95,
  ...,
  transform = NULL,
  null = 0,
  alternative = c("two.sided", "less", "greater"),
  joint = FALSE,
  seed = NULL,
  cluster = NULL,
  cluster_order = c("auto", "original", "estimation"),
  bootstrap_weights = NULL,
  normalize_weights = TRUE,
  first_stage_boot = c("refit", "fixed"),
  midcdf_boot = NULL,
  inference = TRUE,
  keep_weights = FALSE,
  keep_midcdf = FALSE,
  failure = c("error", "omit")
)
```

Arguments

`object`	A current `gir` fit, preferably. Alternatively a two-sided formula, with fully named `gir()` arguments in `...`.
`R`	Number of replicates, default 999. Omitted with a supplied multiplier matrix, use its number of columns. `R=1` is allowed only with `inference=FALSE`.
`level`	Confidence level strictly between zero and one. Basic intervals remain two-sided regardless of the test alternative.

`...` Fully named `gir()` arguments only when `object` is a formula. A fitted object already fixes its point-estimation specification, so extra arguments are an error. Bootstrap controls after the dots require exact names, preventing `first_stage` and `midcdf` from partially matching bootstrap controls.

`transform` `NULL` for raw coefficient tests; a numeric contrast vector or matrix; or a deterministic function of named coefficients returning a fixed-length finite numeric vector. Named contrast columns can select a subset, with others zero. Smooth, nondegenerate transformations are intended.

`null` Scalar or vector null values for transformed statistics. Named vectors must match all transformed statistic names.

`alternative` `"two.sided"` for equal-tail marginal tests; `"less"` or `"greater"` for directional alternatives. Marginal p-values are unadjusted for multiplicity.

`joint` Also compute a bootstrap-calibrated common-covariance joint Wald test? Requires two-sided, nonredundant regular restrictions. Singular restriction covariance gives an undefined joint test with warning.

`seed` Optional nonnegative integer local seed. Restore the caller RNG state on exit. `NULL` advances the existing stream. Cannot be combined with supplied multipliers.

`cluster` Optional clustering of the multipliers: a one-sided formula naming retained model or first-stage columns, whose combination defines the clusters, or a vector with one value per estimation row or per row of the original data (see `cluster_order`). One Exp(1) multiplier is drawn per cluster and shared by all its observations in both stages, then normalized as usual. Use it when observations are dependent within groups, for example firms within industries under an industry-level treatment. Cannot be combined with `bootstrap_weights`.

`cluster_order` Row order of a `cluster` vector. `"original"` requires one value per original-data row, in the order at fitting time, and aligns it using `object$row.index`. `"estimation"` requires one value per final estimation row and uses the vector unchanged. The default `"auto"` infers the order from length. If both lengths match, automatic ordering is allowed only when `object$row.index` is the unchanged row sequence; otherwise specify `"original"` or `"estimation"` explicitly. Missing cluster values are checked after alignment. Leave `"auto"` for formulas, whose retained columns are already aligned, or when `cluster = NULL`.

`bootstrap_weights` Positive finite nobs-by-R multipliers, or an nobs-vector for a single draw, in final estimation-row order. They multiply original case weights once.

`normalize_weights` Divide each supplied multiplier column by its mean? For the internal estimator, including relative ridge handling, common rescaling preserves the draws up to floating-point error within usable numerical ranges. `FALSE` uses and stores the supplied values unchanged. External first-stage callbacks must respect weight scaling themselves. Generated weights always use the paper normalization.

`first_stage_boot` `"refit"` re-estimates the first stage for each draw. `"fixed"` explicitly freezes it for known/oracle nuisances; it is not full two-stage inference for an estimated nuisance.

`midcdf_boot` Optional callback `function(object, weights, replicate)` for a point fit with a supplied first stage. Return a finite nobs-vector of refitted mid-CDFs. The callback weights already combine original weights and multipliers. Required to refit an external first stage; disallowed for internal series fits or fixed mode.

`inference` Compute summaries and tests? `FALSE` returns raw draws and diagnostics, enabling a one-draw warp-speed experiment. At least two successes are needed for inference.

`keep_weights, keep_midcdf` Store nobs-by-R multipliers as used (raw when supplied with `normalize_weights = FALSE`) and first-stage fitted draw values, respectively. Defaults `FALSE` to limit memory.

`failure` `"error"` stops on any failed draw. `"omit"` records a whole `NA` coefficient row and warns; no replacement draw is generated. Conditioning on successful draws may distort inference.

Details

For each replicate generate iid Exp(1) values and divide by their full-sample mean. Multiply the same vector into the first-stage Gram/scores and second-stage GIR moments. Cells, outcome ties, original basis allocation and support bounds are held fixed; first-stage numerical handling and clipping rules are inherited. Bases and tie orders are cached within the call, and each draw accumulates the second-stage sums over the distinct regressor rows, as `gir()` does. Original estimation data are retained in the fit and are not re-evaluated from a possibly changed data frame.

With `cluster`, the Exp(1) values are drawn per cluster and repeated within it, a Bayesian cluster bootstrap. When every first-stage cell lies inside a single cluster, the cell fits are invariant to a common multiplier, so the draws reflect between-cluster variation only and the estimated first stage enters through the residual variation across clusters, as in a clustered regression of cell Ginis; the returned `cluster$first_stage_invariant` reports whether this is the case. The paper's theorem covers independent observations; clustered inference rests on the usual many-clusters argument.

A formula such as `cluster = ~ industry + region` defines one cluster per distinct industry-region combination: this is one-way clustering at their intersections, not two-way clustering by industry and region separately. Group membership uses the values in each column, so punctuation in labels cannot merge distinct combinations. Numeric identifiers are distinguished by their stored values, not their printed formatting. Clusters are numbered in order of first occurrence in the estimation rows; changing labels without changing group membership preserves seeded multipliers.

For errors $D_{bj} = \widehat{\theta}_j^{*(b)} - \widehat{\theta}_j$, Algorithm 2 gives

$$[\widehat{\theta}_j - q_{1-\alpha/2}(D_j), \quad \widehat{\theta}_j - q_{\alpha/2}(D_j)].$$

Quantiles use R's default type 7. These are basic confidence intervals, not percentile credible intervals. SEs are column sample standard deviations of successful draws with B-1 denominator, not divided by sqrt(B) or sqrt(n). Bootstrap bias is the draw mean minus the point estimate and does not replace that point estimate.

For a transformed null, compare observed $h(t_0) - null$ with centered draws $h(t^*) - h(t_0)$. Tail estimates use (1+exceedances)/(B+1); two-sided p-values are twice the smaller tail, capped at one. This finite-simulation safeguard does not confer exact finite-sample validity. The joint statistic uses a common sample covariance of transformed draws for observed and centered-bootstrap Wald

quadratic forms, with an upper-tail bootstrap p-value. There is no OLS F calibration, nested bootstrap or replicate-specific studentization. Zero bootstrap variation produces an undefined marginal p-value with warning.

Smooth transformation validity requires nondegenerate derivatives. Joint restriction derivatives must have full row rank at the null; numerical rank checks are not a proof of that condition. Weak bootstrap convergence alone does not imply bootstrap variance moment consistency; additional moment/uniform-integrability conditions are needed for interpreting covariance estimates consistently. The paper's inference theory covers continuous or finite-discrete response regimes, not all mixed laws. Nonuniform case weights, clustered multipliers, survey designs, regularization and external first stages require their own justification.

A supplied mid-CDF vector requires a refitting callback or an explicit choice of fixed mode. Bootstrap inference requires a full-rank second stage. Omitting failed draws produces a warning. Use `R=1, inference=FALSE` as a draw generator, for example in a warp-speed Monte Carlo study that pools centered draws across simulated datasets.

Value

An object of class `gir_bootstrap`, not class `boot`. Components `t0` and `t` contain raw coefficients and the R-by-p coefficient draw matrix, even when a transformation is tested. It also contains the point `fit`, requested/successful counts, success indicators, per-draw diagnostics, RNG metadata, the cluster structure (`cluster`: number of clusters, index and first-stage invariance) when clustering was requested, and optional multiplier/mid-CDF matrices. With inference, `std.error`, `covariance`, `conf.int`, `bias` and `coefficient.table` concern raw coefficients; `test` contains transformed inference and an optional joint test. These summary fields are `NULL` for draw-only output.

References

Bhardwaj, V. and Diaz-Villarejo, D. (2026). Gini Index Regression. Working paper, Universidad Carlos III de Madrid. https://ddiazvil.com/assets/papers/JMP_GIR_DiazVillarejo.pdf.

Cheng, G. (2014, revised preprint). Moment Consistency of the Exchangeably Weighted Bootstrap for Semiparametric M-Estimation. https://arxiv.org/abs/1109.4204.

See Also

`gir()`, `predict.gir()`, `summary.gir_bootstrap()`.

Examples

```
d <- data.frame(y = rep(c(1, 2, 2, 4, 5), 8),
                g = factor(rep(c("a", "b"), each = 20)))
f <- gir(y ~ g, data = d, discrete = "g", singular.ok = FALSE)
bb <- bboot.gir(f, R = 499L, seed = 42L)
summary(bb)
confint(bb)
vcov(bb)
summary(bb, transform = c(gb = 1), null = 0)

# Inspect cluster alignment with a reordered estimation sample.
```

```
cluster_dat <- data.frame(y = c(1, 4, 2, 7, 3, 9),
                          id = c("A", "A", "B", "B", "C", "C"))
cluster_fit <- gir(y ~ 1, data = cluster_dat, subset = c(1, 3, 5, 2, 4, 6),
                   first_stage = ~ id, discrete = "id", minimum_cell_size = 1L)
cluster_boot <- bboot.gir(cluster_fit, cluster = cluster_dat$id,
                          cluster_order = "original", R = 3L, seed = 17L,
                          inference = FALSE)
cluster_boot$cluster$index  # 1 2 3 1 2 3
```

---

gir *Linear Gini Index Regression*

Description

Fit the linear GIR point estimator with a tie-aware least-squares-series conditional mid-distribution first stage and optional GIR goodness of fit.

Usage

```
gir(
  formula,
  data,
  subset,
  weights,
  na.action = stats::na.fail,
  first_stage = NULL,
  continuous = NULL,
  discrete = NULL,
  midcdf = NULL,
  basis = c("legendre", "spline"),
  kappa = NULL,
  basis_constant = 1,
  sieve_size = NULL,
  sieve_size_type = c("auto", "dimension", "knots"),
  sieve_size_reference = c("cell", "full"),
  spline_order = 4L,
  basis_bounds = NULL,
  clip = FALSE,
  first_stage_singular = c("error", "ridge", "drop", "none"),
  first_stage_tol = 1e-12,
  ridge = 1e-10,
  minimum_cell_size = 8L,
  max_basis_columns = 1000L,
  singular.ok = TRUE,
  second_stage_tol = 1e-07,
  GIR = FALSE,
  model = TRUE,
```

```
  x = FALSE,
  y = FALSE,
  contrasts = NULL
)
```

Arguments

formula	Two-sided second-stage formula. An intercept is included unless removed with `0 +` or `-1`. Offsets are not implemented.
data	Data frame containing the needed variables. Required.
subset	Optional row selector evaluated in `data`, before missing-value handling.
weights	Nonnegative finite case weights entering both stages. At least one must be positive in the retained estimation sample, after subsetting and missing-value handling. A scalar is repeated; vectors align to original or selected rows. Missing weights are handled by `na.action` together with the other model variables.
na.action	Missing-value action for variables in either stage and supplied mid-CDFs. Defaults to `stats::na.fail`; `na.omit` or `na.exclude` removes incomplete rows with a warning. Remaining missing values error. If no rows or no positive weights remain, fitting stops with a sample-specific error before the routine removal warning. Custom actions must return a data frame selecting original rows with valid, unique row identifiers. Their removal warning distinguishes missing and complete rows, and their errors are reported as custom-action failures.
first_stage	One-sided formula or character vector selecting raw conditioning columns. `NULL` uses underlying variables from the second-stage right-hand side.
continuous	Character vector of numeric continuous first-stage columns. The full tensor product of their marginal bases is used. Numeric first-stage columns listed in neither `continuous` nor `discrete` are treated as continuous, and a message names them. An undeclared numeric column with at most two distinct values in the retained estimation sample is an error, because a two-point support cannot carry a continuous basis; declare it in `discrete`, or in `continuous` if that is intended.
discrete	Character vector of categorical first-stage columns. Observed joint values form separate cells. Unclassified factors, ordered factors, character and logical columns are automatic discrete variables; numeric binary columns must be declared explicitly.
midcdf	Optional supplied numeric vector or data-column name for conditional mid-distribution values. Bypasses internal first-stage estimation. A scalar, original-row vector, or selected-row vector is aligned before missing-value removal.
basis	`"legendre"` or `"spline"`. The spline sieve uses `splines::bs()` internally; there is no separate B-spline or wavelet option.
kappa	Exponent in the automatic sieve-size rule. `NULL` selects 0.245 for Legendre and 0.300 for splines.
basis_constant	Positive multiplier in `floor(basis_constant * n_reference^kappa)`.
sieve_size	Optional fixed total-dimension target, or scalar/vector interior-knot count according to `sieve_size_type`. A knot vector follows `continuous` ordering.

NULL uses the automatic rule. A marginal basis with more columns than the variable has distinct values in a cell is singular by construction: an error under first_stage_singular = "error", otherwise a warning naming the variable while the chosen policy is applied.

sieve_size_type

"auto" is "dimension" for both bases: the size rule gives the total tensor-product dimension K of the paper, allocated greedily across balanced integer marginal dimensions, never below two columns per Legendre margin or spline_order columns per spline margin. "knots" (splines only) applies the rule, or sieve_size, to the interior-knot count of every continuous margin instead.

sieve_size_reference

Use each cell size ("cell") or full estimation-sample size ("full") in the automatic rule. Regression observations remain cell-specific in either case.

spline_order Order of marginal polynomial splines. Default 4 means cubic, with degree 3. Each marginal basis has $J + s_0$ columns for J interior knots and order s_0.

basis_bounds Bounds for rescaling continuous columns to $[0, 1]$, preferably a named list of two-number vectors. Default uses final estimation-sample minima and maxima. Bounds are common across cells.

clip Clip first-stage values to $[0, 1]$? Default FALSE. Out-of-range values always warn. Clipping changes the estimator; it does not constrain the second-stage linear index.

first_stage_singular

"error", "ridge", "drop", or "none". Error is the conservative default; ridge adds a scaled diagonal correction, drop removes QR-dependent columns, and none attempts the unmodified solve.

first_stage_tol

Positive first-stage numerical rank/reciprocal-condition tolerance.

ridge Relative diagonal regularization constant, default 1e-10, used only with first_stage_singular = "ridge". The initial penalty is ridge * mean(diag(G)) for the weighted cell Gram matrix G, without an absolute floor. If the solve fails, a further 100 times this base penalty is added. Diagnostics report the total applied penalty. The default is a numerical heuristic, not a data-tuned or theoretically optimal choice. Unusable scales, penalties, or adjusted matrices cause an informative error. Ridge changes the first-stage estimator; assess its sensitivity.

minimum_cell_size

Minimum number of positive-weight observations per observed cell. Default 8.

max_basis_columns

Upper limit on constructed tensor-product columns. Cell-size constraints can reduce the requested allocation.

singular.ok Allow second-stage aliasing? TRUE retains an identified subset and marks aliased coefficients NA. FALSE errors. The second stage is not ridge-regularized.

second_stage_tol

Positive second-stage numerical rank tolerance and conditioning warning threshold.

GIR	Compute the GIR-specific goodness-of-fit measure using the same first-stage ranks in unrestricted and constant-only losses? Default FALSE.
model, x, y	Retain public model-frame, design-matrix and response slots. Defaults TRUE, FALSE, FALSE. Independently, an internal resolved estimation snapshot is always retained for bootstrap and prediction; these flags do not anonymize the object.
contrasts	Optional contrasts passed to stats::model.matrix().

Details

gir() computes point estimates for the linear GIR model. Its formula/data interface is patterned after lm() and quantreg::rq(). The conditional mid-CDF is either estimated with least-squares series or supplied by the user.

The first-stage response at threshold t is $1(Y < t) + 0.5\,1(Y = t)$. Sorting and cumulative basis sums calculate all observed-threshold scores exactly, including ties. With case weights a_i, the second stage solves

$$\left\{\sum_i a_i Y_i X_i X_i'\right\}\widehat{\theta} = \sum_i a_i Y_i X_i (2\widehat{M}_i - 1).$$

For splines, equally spaced interior knots are specified explicitly and a stable B-spline representation is used. Under total-dimension sizing, integer products need not equal the target; the minimum spline tensor dimension s_0^d and the minimum Legendre tensor dimension 2^d can exceed a small nominal target, in which case the minimum is used and floored is recorded in the diagnostics. Cell-size capping and QR dropping are reported.

Observations with the same regressor values share a design row. The second-stage sums are accumulated over those groups and the moments are formed from the distinct rows, which is exact (only the order of summation differs). With cell-level regressors, such as industry and year effects in firm data, the retained design and every bootstrap draw are therefore the size of the distinct rows rather than of the sample.

Use summary(fit) for the point fit and summary(bboot.gir(fit)) for bootstrap inference. Fitted values target the linear conditional Gini index. See vignette("GIR-guide", package = "GIR") for examples, the underlying calculations and guidance on numerical settings.

Value

An object with classes gir and gir.object, containing named coefficients, complete coefficients for fitted values, fitted indices and generated-rank residuals, weights, first-stage mid-CDF values, formula metadata, row alignment, and diagnostics. first_stage$diagnostics contains named list-columns marginal_dimensions and marginal_knot_counts, total basis_dimension, and effective_dimension after any column removal, and floored records whether the minimum tensor dimension exceeded the requested size. Knots are NA for Legendre; an intercept-only first stage has empty marginal vectors. The estimation component retains the distinct design rows with the index mapping observations to them, the outcome, weights, first-stage columns and controls for later refits; x = TRUE stores the full design.

References

Bhardwaj, V. and Diaz-Villarejo, D. (2026). Gini Index Regression. Working paper, Universidad Carlos III de Madrid. https://ddiazvil.com/assets/papers/JMP_GIR_DiazVillarejo.pdf.

See Also

bboot.gir(), predict.gir(), midcdf(), GIR-validation for mathematical checks and their scope.

Examples

```
d <- data.frame(y = rep(c(1, 2, 2, 4, 5), 8),
                group = factor(rep(c("a", "b"), each = 20)))
f <- gir(y ~ group, data = d, discrete = "group", GIR = TRUE)
summary(f)
summary(midcdf(f))
f$first_stage$diagnostics
```

GIR-validation *Mathematical Validation of GIR*

Description

The package tests check finite-sample identities and independent numerical implementations of the algorithms in Bhardwaj and Diaz-Villarejo (2026). These checks verify the calculations; they do not measure statistical coverage.

Independent algorithm checks

test-math-algorithms.R checks Algorithms 1 and 2 on deterministic data with two continuous covariates, unequal discrete-cell sizes, zeros and ties. The reference builds closed-form Legendre polynomials or cubic B-splines with an interior knot, takes their tensor products, and fits each threshold response $1(Y < t) + 0.5\,1(Y = t)$ using stats::lm.wfit(). It then fits the second stage with weights proportional to the outcome.

Both the unit-weight estimator and the package's nonuniform case-weight extension are checked. Bootstrap reference fits use the same supplied multipliers in both stages. Comparisons include conditional mid-distribution values, coefficients, fitted values, bootstrap draws, covariance, standard errors and basic confidence intervals with type-7 quantiles. The fixtures distinguish refitting from freezing the first stage, and basic from percentile intervals. No GIR internal basis or fitting helper is used to calculate reference answers.

Exact Gini identity

test-math-gini-identity.R checks the paper's Numerical Equivalence lemma. With mutually exclusive group indicators in both stages, each saturated fitted group value equals

$$\widehat{G}_g = \frac{\sum_{i \in g} \sum_{j \in g} |Y_i - Y_j|}{2 n_g \sum_{i \in g} Y_i}.$$

The observations must be nonnegative and each group must have a positive mean. This is the empirical-distribution Gini: no $n_g/(n_g - 1)$ correction is applied. Tests include unequal group sizes, ties, zeros, constant positive groups, and both indicator and treatment coding. Bootstrap draws are also checked against the weighted pairwise formula.

Invariance

`test-math-invariance.R` checks the paper's Scale Invariance lemma using Y, $0.01Y$ and $100Y$, with covariates and basis held fixed. Conditional mid-distribution values, coefficients, predictions, GIR-R-squared and paired bootstrap inference agree within numerical tolerance. This is outcome rescaling, distinct from the multiplier-rescaling checks in `test-ridge-scaling.R`.

The same tests verify row-order invariance for discrete, Legendre and spline first stages. For bootstrap comparisons, multiplier rows are moved with their observations; reusing a seed alone does not establish that pairing.

Goodness of fit

`test-math-r-squared.R` checks Algorithm 3 using the same conditional ranks in the unrestricted and constant-only losses. It verifies the weighted total/explained/residual variation decomposition and the weighted normal equations. With an intercept and positive benchmark variation, GIR-R-squared is in $[0, 1]$ up to rounding. It cannot decrease under nested expansions of the second stage when the first-stage values, sample and weights are fixed. Re-estimating the first stage changes this comparison.

An intercept-only second stage has GIR-R-squared zero when benchmark variation is positive. Zero benchmark variation instead produces `NA` with a warning. Without an intercept, the constant benchmark need not be nested and GIR-R-squared can be negative. The tests explicitly cover these cases.

Sample and diagnostic checks

`test-empty-samples.R` verifies early errors for empty input, empty subsets, complete removal by missing-value handling, and loss of all positive case weights. It checks custom actions, row alignment and omission metadata, agreement with manually filtered data, and empty prediction vectors and confidence-interval matrices. Empty-fit errors are also checked with `options(warn = 2)` so a preceding warning cannot mask the sample diagnosis.

The point-estimator and bootstrap/prediction tests check the full ordered set of expected warnings, or explicitly require a quiet call, instead of broadly suppressing warnings. Numerical diagnostics remain warnings for users. Run `testthat::test_local(filter = "empty-samples")` to isolate the sample-handling checks; the complete suite includes the warning checks.

Reproducibility and scope

From the package source directory, run `testthat::test_local(filter = "math-")` for these mathematical checks or `testthat::test_local()` for the complete suite. They also run through `R CMD check`. No network, unpublished paper file, additional suggested package or large Monte Carlo experiment is needed.

The mathematical tests are deterministic and use full-rank designs without ridge or clipping. General comparisons use tolerance `1e-10`; the small pairwise identities use `1e-12`. These tolerances allow floating-point differences between QR-based reference fits and the package's moment calculations. They are not accuracy guarantees for arbitrary ill-conditioned inputs.

The inference arithmetic fixtures use 40 fixed multiplier columns and confidence level 0.5, ensuring ten expected draws per tail without a long test run. These settings are for software verification, not a recommendation for applied inference. They do not measure coverage or validate the exponential multiplier distribution.

Passing these checks does not prove asymptotic results, establish nominal confidence-interval coverage, or justify clustered, regularized or survey inference. Case-weight tests verify algebra rather than inferential theory. The existing cluster and ridge tests check those implementation extensions separately. Coverage and bias require a separate simulation design with known population targets and Monte Carlo uncertainty.

References

Bhardwaj, V. and Diaz-Villarejo, D. (2026). Gini Index Regression. Working paper, Universidad Carlos III de Madrid. https://ddiazvil.com/assets/papers/JMP_GIR_DiazVillarejo.pdf.

See Also

gir(), bboot.gir(), predict.gir().

---

midcdf *Extract Conditional Mid-Distribution Estimates*

Description

Extract the named first-stage conditional mid-distribution vector from a GIR fit.

Usage

```
midcdf(object, ...)

## S3 method for class 'gir'
midcdf(object, ...)
```

Arguments

object	A fitted GIR model.
...	Further arguments; currently unused.

Value

A numeric vector in retained estimation-sample order. Standard numeric summary methods give minimum, quartiles, median, mean and maximum.

See Also

gir(), bboot.gir().

predict.gir *Predict the Linear Conditional Gini Index*

Description

Evaluate the GIR linear index on the estimation sample or new covariates, with optional bootstrap standard errors and pointwise basic confidence intervals.

Usage

```
## S3 method for class 'gir'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  bootstrap = NULL,
  R = 999L,
  seed = NULL,
  na.action = stats::na.pass,
  ...
)
```

Arguments

object	A current gir fit.
newdata	Optional data frame of raw second-stage predictors. NULL uses retained estimation rows in the order of fitted(object). No response or first-stage-only variables are required.
se.fit	Return a list with prediction(s) and bootstrap SEs? Default FALSE.
interval	"none" or "confidence". The latter gives pointwise basic bootstrap confidence intervals for the linear Gini index, not individual-outcome prediction intervals.
level	Confidence level strictly between zero and one.
bootstrap	Optional matching gir_bootstrap object. Its raw coefficient draws are reused without refitting or new randomness. A different estimator or sample is rejected.
R, seed	Replicate count and local seed for an automatic bootstrap only. Do not supply them together with an existing bootstrap object.
na.action	For newdata only: default na.pass preserves missing rows as NA; omit removes them; exclude reinserts them; fail stops. Empty newdata, or omission of every prediction row, returns a length-zero point vector or a zero-row, three-column confidence-interval matrix. Exclude restores NA positions, including when every row is missing.
...	Additional bboot.gir() controls for an automatic bootstrap only, including cluster and cluster_order. Cluster vectors refer to the original fitting data or estimation sample, never newdata. These controls do not change gir() settings.

Details

The target is $x'\theta$, not the response Y, its conditional mean, or the first-stage conditional CDF. Under misspecification it is the fitted linear-approximation target. Predictor terms, safe transformation attributes, original contrasts and factor levels are retained; unrecognized factor levels or missing predictor columns error.

Given raw coefficient draws, predict each $x'\widehat{\theta}^{*(b)}$ and take the Algorithm 2 basic quantiles of its centered errors. Prediction SE is $\sqrt{x'\widehat{V}_{boot}x}$, equivalently their sample standard deviation. Do not combine marginal coefficient interval endpoints or add an OLS outcome-noise variance. Uncertainty is pointwise, not simultaneous. Point predictions outside $[0, 1]$ warn and are not clipped; intervals are not clipped either.

A pure point-prediction call uses no bootstrap. Supplied bootstrap draws are preferred for repeated requests; otherwise uncertainty triggers `bboot.gir()`. At least two successful draws are required. Original-sample output contains estimation rows only, not rows omitted from fitting; `row.index` maps them back. `newdata` `na.exclude` is reinserted in the returned objects.

Point predictions from aliased fits are returned only for estimable linear functions, with nonestimable new rows `NA`. Bootstrap prediction uncertainty requires a full-rank coefficient specification in this version. The supplied inference theory and covariance moment qualifications apply.

Value

A named prediction vector when `interval` is `"none"`, or a matrix with columns `fit`, `lwr` and `upr` for confidence intervals. With `se.fit=TRUE` return a list with `fit`, `se.fit`, `level`, `method` and `R`. The result has no `residual.scale` or Student degrees of freedom, since uncertainty is calculated from the bootstrap draws.

See Also

`gir()`, `bboot.gir()`, `stats::predict.lm()`.

Examples

```
d <- data.frame(y = rep(c(1, 2, 2, 4, 5), 8),
                g = factor(rep(c("a", "b"), each = 20)))
f <- gir(y ~ g, data = d, discrete = "g")
predict(f)
nd <- data.frame(g = c("a", "b"))
bb <- bboot.gir(f, R = 499L, seed = 9L)
predict(f, newdata = nd, bootstrap = bb,
        interval = "confidence", se.fit = TRUE)
```

`print.gir` *Summaries and Extraction for GIR Point Fits*

Description

Print and summarize a GIR point fit and extract its model components. Bootstrap inference remains a separate operation.

Usage

```
## S3 method for class 'gir'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

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

## S3 method for class 'summary.gir'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'gir'
coef(object, complete = TRUE, ...)

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

## S3 method for class 'gir'
residuals(object, ...)

## S3 method for class 'gir'
weights(object, ...)

## S3 method for class 'gir'
nobs(object, ...)

## S3 method for class 'gir'
formula(x, ...)

## S3 method for class 'gir'
terms(x, ...)

## S3 method for class 'gir'
model.frame(formula, ...)

## S3 method for class 'gir'
model.matrix(object, ...)
```

Arguments

digits	Print precision.
...	Further method arguments; currently unused except supported print formatting.
object, x, formula	A gir fit, or the summary object for its print method. In the model.frame method

	the argument named `formula` is the fit.
`complete`	Return all coefficients including aliased `NA` values? Default `TRUE`.

Value

`summary` returns a point-estimator summary with a one-column Estimate table, rank and alias information, optional GIR-R2, mid-CDF summary and diagnostics. Standard extraction methods return the indicated components. `fitted` returns the linear conditional Gini index; `residuals` returns centered estimated mid-rank minus that index. `model.frame` requires the public `model` slot, and `model.matrix` requires the public `x` or `model` slot. The internal estimation snapshot is reserved for bootstrap and prediction.

See Also

`gir()`, `midcdf()`, `bboot.gir()`, `predict.gir()`.

---

summary.gir_bootstrap *Summaries and Extraction for GIR Bootstrap Results*

Description

Reanalyze stored raw bootstrap draws at a new level or for smooth coefficient transformations, and extract raw coefficients, covariance or confidence intervals.

Usage

```
## S3 method for class 'gir_bootstrap'
summary(
  object,
  level = object$level,
  transform = object$transform,
  null = object$null,
  alternative = object$alternative,
  joint = object$joint,
  ...
)

## S3 method for class 'gir_bootstrap'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.gir_bootstrap'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'gir_bootstrap'
coef(object, ...)

## S3 method for class 'gir_bootstrap'
```

```
vcov(object, ...)

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

Arguments

`object`, `x`	A `gir_bootstrap` object, or its summary for the summary print method.
`level`, `transform`, `null`, `alternative`, `joint`	As in `bboot.gir()`. Summary defaults inherit the original bootstrap call. A new summary reuses the stored successful raw draws and does not modify the bootstrap object.
`...`	Print methods forward formatting arguments. Summary, covariance and interval methods reject unused arguments; coefficient extraction ignores extras.
`digits`	Print precision.
`parm`	Optional raw coefficient names or positive integer indices for `confint`.

Details

At least two successful draws are needed for inferential summaries. A low expected number of observations in an interval tail warns. Marginal tests are unadjusted and confidence intervals remain two-sided. Standard errors, moment-consistency qualifications and the common-covariance Wald test are described under `bboot.gir()`.

Value

`summary` returns class `summary.gir_bootstrap` with raw coefficient table, SEs, covariance, intervals, bias, transformed test results and diagnostics. `print` invisibly returns its argument. `coef` returns the original raw coefficient estimate, not bootstrap means. `vcov` returns raw coefficient covariance; `confint` returns basic intervals for raw coefficients. Obtain transformed intervals from `summary(...)$test$conf.int`.

See Also

`bboot.gir()`, `predict.gir()`.

Index

bboot.gir, 2
bboot.gir(), *10*, *12*, *14*, *16*, *17*

coef.gir *(*print.gir*)*, 14
coef.gir_bootstrap
*(*summary.gir_bootstrap*)*, 16
confint.gir_bootstrap
*(*summary.gir_bootstrap*)*, 16

fitted.gir *(*print.gir*)*, 14
formula.gir *(*print.gir*)*, 14

gir, 6
gir(), *5*, *12*, *14*, *16*
GIR-validation, 10
gir_bootstrap *(*bboot.gir*)*, 2

midcdf, 12
midcdf(), *10*, *16*
model.frame.gir *(*print.gir*)*, 14
model.matrix.gir *(*print.gir*)*, 14

nobs.gir *(*print.gir*)*, 14

predict.gir, 13
predict.gir(), *5*, *10*, *12*, *16*, *17*
print.gir, 14
print.gir_bootstrap
*(*summary.gir_bootstrap*)*, 16
print.summary.gir *(*print.gir*)*, 14
print.summary.gir_bootstrap
*(*summary.gir_bootstrap*)*, 16

residuals.gir *(*print.gir*)*, 14

stats::lm.wfit(), *10*
stats::predict.lm(), *14*
summary.gir *(*print.gir*)*, 14
summary.gir_bootstrap, 16
summary.gir_bootstrap(), *5*

terms.gir *(*print.gir*)*, 14

vcov.gir_bootstrap
*(*summary.gir_bootstrap*)*, 16

weights.gir *(*print.gir*)*, 14