Package: seine (via r-universe)

October 27, 2026

Title Semiparametric Ecological Inference

Version 0.1.2

Description Efficient and user-friendly routines for modern ecological inference. Implements the methods described in McCartan & Kuriwaki (2025+) <doi:10.48550/arXiv.2509.20194>, which generalize ecological regression as introduced by Goodman (1953) <doi:10.2307/2088121>. Includes routines for preprocessing, synthetic data generation, double/debiased machine learning (DML) estimation, partial identification bounds, and sensitivity analysis.

Depends R (>= 3.5.0)

Imports rlang, cli, tidyselect, tibble, hardhat, quadprog, graphics, utils, stats

Suggests bases, knitr, rmarkdown, testthat (>= 3.0.0), xml2

LinkingTo cpp11, cpp11armadillo, testthat

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://corymccartan.com/seine/, https://github.com/CoryMcCartan/seine

BugReports https://github.com/CoryMcCartan/seine/issues

VignetteBuilder knitr

Config/testthat/edition 3

Config/build/compilation-database true

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

NeedsCompilation yes

Author Cory McCartan [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6251-669X>), Shiro Kuriwaki [aut, cph] (ORCID: <https://orcid.org/0000-0002-5687-2647>)

Maintainer Cory McCartan <mccartan@psu.edu>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-08 16:32:45 UTC

RemoteUrl https://github.com/cran/seine

RemoteRef HEAD

RemoteSha 4c87fb47ad75967e75ff009dc3103f12e9f3f1f4

Contents

ei_bench . . . 2
ei_bounds . . . 4
ei_est . . . 7
ei_est_local . . . 10
ei_local_cov . . . 13
ei_proportions . . . 14
ei_ridge . . . 15
ei_ridge_impl . . . 19
ei_riesz . . . 20
ei_sens . . . 22
ei_sens_rv . . . 24
ei_spec . . . 25
ei_synthetic . . . 27
ei_test_car . . . 29
ei_wgt . . . 31
ei_wrap_model . . . 32
elec_1968 . . . 33
plot.ei_sens . . . 35
plot.ei_spec . . . 36
ridge-methods . . . 37
weights.ei_riesz . . . 38

Index **39**

`ei_bench` *Benchmark sensitivity parameters from observed covariates*

Description

Produces a table of benchmark values for `c_outcome` and `c_predictor` in `ei_sens()` for each covariate, following the methodology of Chernozhukov et al. (2024).

Usage

```
ei_bench(spec, subset = NULL, contrast = NULL)
```

Arguments

spec	An ei_spec object.
subset	<data-masking> An optional indexing vector describing the subset of units over which to calculate estimates.
contrast	If provided, a list containing entries predictor and outcome, each containing a contrast vector. If only one of predictor or outcome is provided, the contrast will be calculated for all levels of the other variable. For example list(predictor = c(1, -1, 0)) will calculate the difference in each outcome between the first and second predictor groups; list(outcome = c(1, -1)) will calculate the difference between the two outcomes for each predictor group; and list(predictor = c(1, -1, 0), outcome = c(1, -1)) will calculate the difference in differences.

Value

A data frame of benchmark covariate R-squared values.

References

Chernozhukov, V., Cinelli, C., Newey, W., Sharma, A., & Syrgkanis, V. (2024). *Long story short: Omitted variable bias in causal machine learning* (No. w30302). National Bureau of Economic Research.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_ind_wal,
               total = pres_total, covariates = c(educ_elem, pop_urban, farm))
ei_bench(spec)

# with preprocessed covariates
spec = ei_spec(
    data = elec_1968,
    predictors = vap_white:vap_other,
    outcome = pres_ind_wal,
    total = pres_total,
    covariates = c(educ_elem, pop_urban, farm),
    preproc = ~ model.matrix(~ .^2 - 1, data = .x)
)
ei_bench(spec)
ei_bench(spec, subset = pop_urban > 0.5)

# with contrasts
spec = ei_spec(elec_1968, vap_white:vap_other, pres_rep_nix:pres_ind_wal,
               total = pres_total, covariates = c(educ_elem, pop_urban, farm))
ei_bench(spec, contrast = list(predictor = c(1, -1, 0)))
ei_bench(spec, contrast = list(outcome = c(1, -1)))
```

ei_bounds *Compute partial identification bounds on local ecological quantities*

Description

For each observation, computes the minimum and maximum value of each local estimand that is consistent with the accounting identity and the bounds on the outcome. Optionally aggregate the computed bounds across units.

Usage

```
ei_bounds(
  x,
  ...,
  total,
  contrast = NULL,
  bounds = c(0, 1),
  sum_one = NULL,
  subset = NULL,
  global = FALSE
)

## S3 method for class 'ei_spec'
ei_bounds(
  x,
  total,
  contrast = NULL,
  bounds = c(0, 1),
  sum_one = NULL,
  subset = NULL,
  global = FALSE,
  ...
)

## S3 method for class 'formula'
ei_bounds(
  formula,
  data,
  total,
  contrast = NULL,
  bounds = c(0, 1),
  sum_one = NULL,
  subset = NULL,
  global = FALSE,
  ...
)
```

```
## S3 method for class 'data.frame'
ei_bounds(
  x,
  y,
  total,
  contrast = NULL,
  bounds = c(0, 1),
  sum_one = NULL,
  subset = NULL,
  global = FALSE,
  ...
)

## S3 method for class 'matrix'
ei_bounds(
  x,
  y,
  total,
  contrast = NULL,
  bounds = c(0, 1),
  sum_one = NULL,
  subset = NULL,
  global = FALSE,
  ...
)

## Default S3 method:
ei_bounds(x, ...)

## S3 method for class 'ei_bounds'
as.array(x, ...)
```

Arguments

x	An object of class ei_bounds
...	Additional arguments (ignored)
total	<data-masking> A variable containing the total number of observations in each aggregate unit. For example, the column containing the total number of voters. Required for computing weights unless x is an ei_spec() object.
contrast	If provided, a list containing entries predictor and outcome, each containing a contrast vector. If only one of predictor or outcome is provided, the contrast will be calculated for all levels of the other variable. For example list(predictor = c(1, -1, 0)) will calculate the difference in each outcome between the first and second predictor groups; list(outcome = c(1, -1)) will calculate the difference between the two outcomes for each predictor group; and list(predictor = c(1, -1, 0), outcome = c(1, -1)) will calculate the difference in differences.

bounds — A vector `c(min, max)` of bounds for the outcome. If `bounds = NULL`, they will be inferred from the outcome variable: if it is contained within $[0, 1]$, for instance, then the bounds will be `c(0, 1)`. The default `bounds = FALSE` uses an unbounded outcome.

sum_one — If `TRUE`, the outcome variables are constrained to sum to one within each predictor group. Can only apply when bounds are enforced and there is more than one outcome variable. If `NULL`, infers `sum_one = TRUE` when the bounds are `c(0, 1)` and the outcome variables sum to 1.

subset — <data-masking> An optional indexing vector describing the subset of units over which to calculate estimates.

global — If `TRUE`, aggregate the bounds across units to produce bounds on the global estimands.

formula — A formula such as `y ~ x0 + x1` specifying the outcome `y` and the predictors of interest `x`. The predictors should form a partition, that is, `x0 + x1 = 1` for each observation. Users can be include more than two predictors as well, e.g. `pct_white + pct_black + pct_hisp + pct_other`. If there are just two predictors, it is acceptable to only include one in the formula; the other will be formed as 1 minus the provided predictor.

data — When a **formula** is used, `data` is a **data frame** containing both the predictors and the outcome.

y — When `x` is a **data frame** or **matrix**, `y` is the outcome specified as:

- A **data frame** with numeric columns.
- A **matrix**
- A numeric **vector**.

When the outcome is a proportion, you can use `ei_proportions()` to assist in preparing it.

Value

A data frame with bounds. The `.row` column in the output corresponds to the observation index in the input. The `min` and `max` columns contain the minimum and maximum values for each local estimand. The `weight` column contains the product of the predictor variable and total for each observation, where applicable. Taking a weighted average of the bounds against this column will produce global bounds. It has class `ei_bounds`.

Methods (by generic)

- `as.array(ei_bounds)`: Format bounds as an array with dimensions `<rows>*<predictors>*<outcomes>*2`. Does not work if the object has been sorted.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs,
               total = pres_total, covariates = c(state, pop_urban, farm))
```

```
ei_bounds(spec, bounds = c(0, 1))
ei_bounds(spec, bounds = c(0, 1), global = TRUE)

# Infer bounds
ei_bounds(pres_ind_wal ~ vap_white, data = elec_1968, total = pres_total, bounds = NULL)

# With contrast
ei_bounds(
    spec,
    bounds = c(0, 1),
    contrast = list(predictor = c(1, -1, 0), outcome = c(1, -1, 0, 0))
)

# manually aggregate min/max
# easier with dplyr:
# summarize(across(min:max, ~ weighted.mean(.x, weight)), .by=c(predictor, outcome))
grp_units = split(ei_bounds(spec, bounds = c(0, 1)), ~ predictor + outcome)
do.call(rbind, lapply(grp_units, function(b) {
    tibble::tibble(
        predictor = b$predictor[1],
        outcome = b$outcome[1],
        min = weighted.mean(b$min, b$weight),
        max = weighted.mean(b$max, b$weight)
    )
}))
```

`ei_est` *Estimate ecological quantities*

Description

Produces estimates of overall conditional means from a fitted ecological inference model or Riesz representer. If both a regression model and a Riesz representer are provided, a debiased machine learning (DML) estimate is produced.

Usage

```
ei_est(
  regr = NULL,
  riesz = NULL,
  data,
  total,
  subset = NULL,
  contrast = NULL,
  outcome = NULL,
  conf_level = 0.95,
  use_student = TRUE
)
```

```
## S3 method for class 'ei_est'
as.matrix(x, which = "estimate", ...)

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

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

Arguments

regr	A fitted regression model, from ei_ridge(), or another kind of regression model wrapped with ei_wrap_model(). If riesz is not provided and regr is an ei_riesz() object, then riesz will be set to the value of regr and regr will be set to NULL. This is so users can call this function as ei_est(<riesz>, data = <data>).
riesz	A fitted Riesz representer, from ei_riesz(), or a matrix of Riesz weights
data	The data frame, matrix, or ei_spec object that was used to fit the regression or Riesz representer.
total	<tidy-select> A variable containing the total number of observations in each aggregate unit. For example, the column containing the total number of voters. Required if data is not an ei_spec() object and riesz is not provided.
subset	<data-masking> An optional indexing vector describing the subset of units over which to calculate estimates.
contrast	If provided, a list containing entries predictor and outcome, each containing a contrast vector. If only one of predictor or outcome is provided, the contrast will be calculated for all levels of the other variable. For example list(predictor = c(1, -1, 0)) will calculate the difference in each outcome between the first and second predictor groups; list(outcome = c(1, -1)) will calculate the difference between the two outcomes for each predictor group; and list(predictor = c(1, -1, 0), outcome = c(1, -1)) will calculate the difference in differences.
outcome	<data-masking> A vector or matrix of outcome variables. Only required if both riesz is provided alone (without regr) and data is not an ei_spec object.
conf_level	A numeric specifying the level for confidence intervals. If FALSE, no confidence intervals are calculated. Standard errors are always returned.
use_student	If TRUE, use construct confidence intervals from a Student-*t* distribution, which may improve coverage properties in small samples.
x, object	An object of class ei_est
which	Which column of ei_est to convert to a matrix. For example, pass which="std.error" to return standard errors instead of estimates. Partial matching supported.
...	Additional arguments (ignored)

Value

A data frame with estimates. It has class `ei_est`, supporting several methods, and two additional attributes: `vcov`, containing the estimated covariance matrix for the estimates, and `n`, containing the number of aggregate units used in estimation (the number of rows in `data`).

Methods (by generic)

- `as.matrix(ei_est)`: Format estimates, standard errors, or other columns as a matrix.
- `vcov(ei_est)`: Extract full covariance matrix of estimates
- `nobs(ei_est)`: Extract number of units covered by estimates

References

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs,
               total = pres_total, covariates = c(state, pop_urban, farm))

m = ei_ridge(spec)
rr = ei_riesz(spec, penalty = m$penalty)

ei_est(regr = m, data = spec, conf_level = 0.95) # Plug-in estimate
ei_est(riesz = rr, data = spec) # Weighted (Riesz) estimate
est = ei_est(regr = m, riesz = rr, data = spec) # Double/debiased ML estimate
# Working with the output
as.matrix(est)
as.matrix(est, "std.error")
vcov(est)[1:4, 1:4]

# Contrasts
ei_est(regr = m, riesz = rr, data = spec, contrast = list(predictor = c(1, -1, 0)))
ei_est(regr = m, riesz = rr, data = spec,
       contrast = list(predictor = c(-1, 1, 0), outcome = c(1, -1, 0, 0)))

# Subsetting
est = ei_est(m, rr, data = spec, subset = (state == "Alabama"))
as.matrix(est)
nobs(est)
```

`ei_est_local` *Produce local ecological estimates*

Description

Projects predictions from a fitted regression model onto the accounting constraint using a provided residual covariance matrix. This ensures that each set of local estimates satisfies the accounting identity. Local estimates may be truncated to variable bounds.

Usage

```
ei_est_local(
  regr,
  data,
  total,
  b_cov,
  contrast = NULL,
  bounds = regr$blueprint$bounds,
  sum_one = NULL,
  subset = NULL,
  sample = FALSE,
  conf_level = 0.95,
  regr_var = TRUE,
  unimodal = TRUE,
  gaussian = FALSE
)

## S3 method for class 'ei_est_local'
as.array(x, ...)
```

Arguments

`regr`	A fitted regression model, from `ei_ridge()`, or another kind of regression model wrapped with `ei_wrap_model()`.
`data`	The data frame, matrix, or ei_spec object that was used to fit the regression.
`total`	<`tidy-select`> A variable containing the total number of observations in each aggregate unit. For example, the column containing the total number of voters. Required if `data` is not an `ei_spec()` object.
`b_cov`	A covariance matrix to use in projecting the local estimates `b` onto the accounting constraint, such as one estimated with `ei_local_cov()`. When `b_cov` is a single number, it is used to specify the pairwise correlation between the local estimates across predictors, with the same correlation structure as the regression residuals assumed for `b` within each predictor. For example, set `b_cov=0` to assume no correlation in `b` across predictors, or set `b_cov=1` to assume perfect correlation in `b` across predictors, corresponding to a (local) neighborhood

model. (Sometimes, a value of 1 leads to numerical difficulties, as in the example below; in this case, try a value like 0.95 or so instead.) When there are multiple outcome variables and `b_cov` is a matrix with entries for each predictor, it will be applied identically to each outcome. Alternatively, a matrix with entries for each predictor-outcome combination may be provided, with entries in the order (Y1|X1, Y1|X2, ..., Y2|X1, Y2|X2, ...).

`contrast` If provided, a list containing entries `predictor` and `outcome`, each containing a contrast vector. If only one of `predictor` or `outcome` is provided, the contrast will be calculated for all levels of the other variable. For example `list(predictor = c(1, -1, 0))` will calculate the difference in each outcome between the first and second predictor groups; `list(outcome = c(1, -1))` will calculate the difference between the two outcomes for each predictor group; and `list(predictor = c(1, -1, 0), outcome = c(1, -1))` will calculate the difference in differences.

`bounds` A vector `c(min, max)` of bounds for the outcome, to which the local estimates will be truncated. In general, truncation will lead to violations of the accounting identity. If `bounds = NULL`, they will be inferred from the outcome variable: if it is contained within $[0, 1]$, for instance, then the bounds will be `c(0, 1)`. Setting `bounds = FALSE` forces unbounded estimates. The default uses the `bounds` attribute of `regr`, if available, or infers from the outcome variable otherwise.

`sum_one` If `TRUE`, the outcome variables are constrained to sum to one. Can only apply when `bounds` are enforced and there is more than one outcome variable. If `NULL`, infers `sum_one = TRUE` when the bounds are `c(0, 1)` the outcome variables sum to 1.

`subset` <data-masking> An optional indexing vector describing the subset of units over which to calculate estimates.

`sample` If `TRUE`, draws a sample for each unit from `b_cov` (plus regression prediction uncertainty if `regr_var = TRUE`), adds it to the point estimate, and projects. This produces a single posterior draw of the local estimates rather than point estimates, and disables confidence intervals.

`conf_level` A numeric specifying the level for confidence intervals. If `FALSE`, no confidence intervals are calculated. For `regr` arguments from `ei_wrap_model()`, confidence intervals will not incorporate uncertainty in the prediction itself, just the residual. This will trigger a warning periodically.

`regr_var` If `TRUE`, incorporate uncertainty from the regression model when calculating confidence intervals. Only applies when `regr` is fitted with `ei_ridge()`, and requires that function be called with `vcov = TRUE`.

`unimodal` If `TRUE`, assume a unimodal residual distribution. Reduces width of confidence intervals by a factor of 2/3.

`gaussian` If `TRUE`, use Gaussian quantiles for confidence intervals, rather than Chebyshev's inequality. This will produce narrower confidence intervals, but they will not be guaranteed to have the nominal coverage. Overrides the `unimodal` argument.

`x` An object of class `ei_est_local`

`...` Additional arguments (ignored)

Details

Local estimates are produced jointly across outcome variables. When bounds are applied, unless `sum_one = TRUE`, the estimates for each observation may not satisfy logical constraints, including the accounting identity.

Projections are done obliquely in accordance with `b_cov` via quadratic programming. Occasionally, the quadratic program may be infeasible due to the specific data, features of `b_cov`, or numerical errors. Various relaxations of the accounting identity and `b_cov` are attempted in these cases; indices where relaxations of `b_cov` were used are stored in the `proj_relax` attribute of the output, and indices of infeasible projections are stored in the `proj_misses` attribute.

Value

A data frame with estimates. The `.row` column in the output corresponds to the observation index in the input. The `weight` column contains the product of the predictor variable and total for each observation. Taking a weighted average of the estimate against this column will produce a global estimate. It has class `ei_est_local`.

Methods (by generic)

- `as.array(ei_est_local)`: Format estimates an array with dimensions `<rows>*<predictors>*<outcomes>`. Does not work if the object has been sorted.

References

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs,
               total = pres_total, covariates = c(state, pop_urban, farm))

m = ei_ridge(spec)

ei_est_local(m, spec, b_cov = 0, bounds = c(0, 1), sum_one = TRUE, conf_level = 0.99)

b_cov = ei_local_cov(m, spec)
e_orth = ei_est_local(m, spec, b_cov = 0, bounds = c(0, 1), sum_one = TRUE)
e_nbhd = ei_est_local(m, spec, b_cov = 0.95, bounds = c(0, 1), sum_one = TRUE)
e_rcov = ei_est_local(m, spec, b_cov = b_cov, bounds = c(0, 1), sum_one = TRUE)
# average interval width
c(
    e_orth = mean(e_orth$conf.high - e_orth$conf.low),
    e_nbhd = mean(e_nbhd$conf.high - e_nbhd$conf.low),
    e_rcov = mean(e_rcov$conf.high - e_rcov$conf.low)
)
```

`ei_local_cov` *Estimate the residual covariance of the local estimands*

Description

Under a slightly stronger coarsening at random assumption (applying to second moments), *and* an assumption of homoskedasticity in the covariates, this function estimates the covariance matrix of the local estimands $\beta_{gj} = \mathbb{E}[Y|X_j = 1, Z = z_g, G = g]$ around their local mean. See the reference for more detail.

Usage

```
ei_local_cov(regr, data, subset = NULL, prior_obs = 10)
```

Arguments

`regr`	A fitted regression model, from `ei_ridge()`, or another kind of regression model wrapped with `ei_wrap_model()`.
`data`	The data frame, matrix, or ei_spec object that was used to fit the regression or Riesz representer.
`subset`	<data-masking> An optional indexing vector describing the subset of units over which to calculate estimates.
`prior_obs`	The effective sample size of the inverse-Wishart conjugate prior, which shrinks the estimate towards the covariance of the regression residuals. Smaller values mean less shrinkage.

Details

Homoskedasticity in the covariates implies that the variance of the residuals depends linearly on the entries of $\bar{X}\bar{X}^\top$. This function fits an auto-tuned ridge regression of the empirical second moments of the residuals on these predictors, and uses the polarization identity discussed in the references to estimate the covariance for each local estimand. When the estimated covariance is not positive semidefinite, it is projected onto the cone of positive semidefinite matrices. A small amount of shrinkage is applied towards a naive estimator (the covariance of the regression residuals) under an inverse-Wishart conjugate prior, whose effective sample size is given by `prior_obs`.

Value

A covariance matrix. The variables are ordered by predictor within outcome, e.g. (Y1|X1, Y1|X2, ..., Y2|X1, Y2|X2, ...).

References

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

`ei_proportions` *Convert counts to proportions*

Description

Divides counts in specified columns by a specified total or by their sum, possibly storing the result in a new column. Also checks for out-of-bounds and missing values, and can create a column containing the remainder amount so that all proportions sum to 1.

Usage

```
ei_proportions(data, ..., .total = ".total", .other = ".other", clamp = 0.001)
```

Arguments

`data`	A data frame.
`...`	<tidy-select> Columns to convert to proportions. If one column is completely missing, it will be imputed as 1 minus the total.
`.total`	<tidy-select> Column to use as the total. If this column does not exist, it will be created as the sum of the selected columns. Supports renaming syntax.
`.other`	<tidy-select> Column to store the remainder, so that the selected columns plus `.other` sum to 1. If the selected columns do sum to 1 after normalization, this argument will not be used; otherwise it will be created or overwritten. The calculation of `.other` is performed *after* clamping (see below).
`clamp`	Proportions that are `clamp` below 0 or above 1 will be rounded to 0 and 1, respectively. Values outside `clamp` will throw an error. Set `clamp=0` to disable or `clamp=Inf` to allow for out-of-bounds proportions (not recommended).

Value

A modified data frame. Unselected columns are unmodified.

Examples

```
data(elec_1968)
# Make a data frame with counts
d_unnorm = with(head(elec_1968, 10), data.frame(
  vap = vap,
  vap_white = vap * vap_white,
  vap_black = vap * vap_black,
  vap_other = vap * vap_other
))

ei_proportions(d_unnorm, vap_white:vap_black, .total=vap) # `.other` column created
ei_proportions(d_unnorm, vap_white:vap_other) # no total provided
# renaming allowed
ei_proportions(d_unnorm, white=vap_white, black=vap_black,
               .total=c(total=vap), .other="vap_other")
```

ei_ridge *Fit an ecological inference regression model*

Description

Fits a penalized regression model for ecological inference, allowing for overall and unit-level estimates of conditional means using `ei_est()`.

Usage

```
ei_ridge(
  x,
  ...,
  weights,
  bounds = FALSE,
  sum_one = FALSE,
  penalty = NULL,
  scale = TRUE,
  vcov = TRUE
)

## S3 method for class 'formula'
ei_ridge(
  formula,
  data,
  weights,
  bounds = FALSE,
  sum_one = FALSE,
  penalty = NULL,
  scale = TRUE,
  vcov = TRUE,
  ...
)

## S3 method for class 'ei_spec'
ei_ridge(
  x,
  weights,
  bounds = FALSE,
  sum_one = FALSE,
  penalty = NULL,
  scale = TRUE,
  vcov = TRUE,
  ...
)

## S3 method for class 'data.frame'
```

```
ei_ridge(
  x,
  y,
  z,
  weights,
  bounds = FALSE,
  sum_one = FALSE,
  penalty = NULL,
  scale = TRUE,
  vcov = TRUE,
  ...
)

## S3 method for class 'matrix'
ei_ridge(
  x,
  y,
  z,
  weights,
  bounds = FALSE,
  sum_one = FALSE,
  penalty = NULL,
  scale = TRUE,
  vcov = TRUE,
  ...
)

## Default S3 method:
ei_ridge(x, ...)
```

Arguments

x	Depending on the context: • A **data frame** of predictors. • A **matrix** of predictors. • An ei_spec object containing the outcome, predictor, and covariates. Predictors must be proportions that sum to 1 across rows. You can use `ei_proportions()` to assist in preparing predictor variables. Covariates in an ei_spec object are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.
...	Not currently used, but required for extensibility.
weights	<data-masking> A vector of unit weights for estimation. These may be the same or different from the total number of observations in each aggregate unit (see the `total` argument to `ei_spec()`). See the discussion below under 'Weights' for choosing this parameter. The default, uniform weights, makes a slightly stronger-than-necessary assumption about the relationship between the unit totals and the unknown data.

bounds
: A vector `c(min, max)` of bounds for the outcome. If `bounds = NULL`, they will be inferred from the outcome variable: if it is contained within $[0, 1]$, for instance, then the bounds will be `c(0, 1)`. The default `bounds = FALSE` uses an unbounded outcome.

sum_one
: If `TRUE`, the outcome variables are constrained to sum to one. Can only apply when `bounds` are enforced and there is more than one outcome variable. If `NULL`, infers `sum_one = TRUE` when the bounds are `c(0, 1)` the outcome variables sum to 1.

penalty
: The ridge penalty (a non-negative scalar). Set to `NULL` to automatically determine the penalty which minimizes mean-square error, via an efficient leave-one-out cross validation procedure. The ridge regression solution is

$$\hat{\beta} = (X^\top X + \lambda I)^{-1} X^\top y,$$

where λ is the value of `penalty`. One can equivalently think of the penalty as imposing a $\mathcal{N}(0, \sigma^2/\lambda^2)$ prior on the β. Keep in mind when choosing `penalty` manually that covariates in `z` are scaled to have mean zero and unit variance before fitting.

scale
: If `TRUE`, scale covariates `z` to have unit variance.

vcov
: If `TRUE`, calculate and return the a scaled covariance matrix of the estimated coefficients. When `bounds` are provided, the (scaled) covariance matrix for the unbounded estimate is returned as a conservative approximation. The covariance matrix is "scaled" because it does not include the residual variance. For the covariance for a particular outcome variable, multiply the returned `$vcov_u` by `sigma2` for that outcome.

formula
: A formula such as `y ~ x0 + x1 | z` specifying the outcome `y` regressed on the predictors of interest `x` and any covariates `z`. The predictors should form a partition, that is, `x0 + x1 = 1` for each observation. Users can be include more than two predictors as well, e.g. `pct_white + pct_black + pct_hisp + pct_other`. If there are just two predictors, it is acceptable to only include one in the formula; the other will be formed as 1 minus the provided predictor. Include additional covariates separated by a vertical bar `|`. These covariates are strongly recommended for reliable ecological inference. Covariates are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.

data
: When a **formula** is used, `data` is a **data frame** containing both the predictors and the outcome.

y
: When `x` is a **data frame** or **matrix**, `y` is the outcome specified as:

  - A **data frame** with numeric columns.
  - A **matrix**
  - A numeric **vector**.

  When the outcome is a proportion, you can use `ei_proportions()` to assist in preparing it.

z
: When `x` is a **data frame** or **matrix**, `w` are any covariates, specified as:

  - A **data frame** with numeric columns.
  - A **matrix**

  These are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.

Details

The regression is calculated using the singular value decomposition, which allows for efficient re-calculation under different `penalty` values as part of leave-one-out cross-validation. When `bounds` are provided, the regression is calculated via quadratic programming, as there is no closed-form solution. The unbounded regression is run to select the `penalty` automatically in this case, if it is not provided. Estimation is still efficient, though somewhat slower than in the unbounded case. The covariance matrix of the estimates is not available when bounds are applied. Note that supplying `bounds` does not guarantee those bounds will be respected by `ei_est()`, and in general it is not necessarily recommended to use `bounds` in `ei_ridge()` any time the outcomes are bounded. Bounds can be enforced on local estimates in `ei_est_local()`.

Value

An `ei_ridge` object, which supports various ridge-methods.

Weights

The weakest identification result for ecological inference makes no assumption about the number of observations per aggregate unit (the totals). It requires, however, weighting the estimation steps according to the totals. This may reduce efficiency when the totals are variable and a slightly stronger condition holds.

Specifically, if the totals are conditionally mean-independent of the missing data (the aggregation-unit level means of the outcome within each predictor level), given covariates, then it is appropriate to use uniform weights in estimation, or any fixed set of weights.

In general, estimation efficiency is improved when units with larger variance in the outcome receive less weight. Various built-in options are provided by the helper functions in `ei_wgt()`.

References

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs,
               total = pres_total, covariates = c(pop_urban, farm))
ei_ridge(spec)

ei_ridge(pres_dem_hum + pres_rep_nix + pres_ind_wal + pres_abs ~
      vap_white + vap_black + vap_other | pop_urban + farm, data = elec_1968)

# bounds inferred
all.equal(
  fitted(ei_ridge(spec, bounds = NULL)),
  fitted(ei_ridge(spec, bounds = 0:1))
)
```

```
# bounds enforced
min(fitted(ei_ridge(spec)))
min(fitted(ei_ridge(spec, bounds = 0:1)))
```

`ei_ridge_impl` *Low-level implementations of* `ei_ridge()` *and* `ei_riesz()`

Description

No checks are performed on the inputs. Use of `ei_ridge()` and `ei_riesz()` is strongly recommended unless many regressions must be fit, e.g., within a tight loop. Only works for a single outcome, i.e., `y` must be a vector, not a matrix.

Usage

```
ei_ridge_impl(
  x,
  y,
  z,
  weights = rep(1, nrow(x)),
  bounds = c(-Inf, Inf),
  sum_one = NULL,
  penalty = NULL,
  vcov = TRUE
)

ei_riesz_impl(x, z, total, weights = rep(1, nrow(x)), penalty)
```

Arguments

`x`	A matrix of predictors
`y`	A vector of outcomes
`z`	A matrix of covariates
`weights`	A vector of estimation weights
`bounds`	A vector `c(min, max)` of bounds for the outcome.
`sum_one`	If `TRUE`, the outcome variables are constrained to sum to one. Can only apply when `bounds` are enforced and there is more than one outcome variable. If `NULL`, infers `sum_one = TRUE` when the bounds are `c(0, 1)` the outcome variables sum to 1.
`penalty`	The ridge penalty (a non-negative scalar), which must be specified for `ei_riesz_impl()` but can be automatically estimated with `ei_ridge_impl()` by providing `penalty=NULL`.
`vcov`	If `TRUE`, calculate and return the a scaled covariance matrix of the estimated coefficients. When `bounds` are provided, the (scaled) covariance matrix for the unbounded estimate is returned as a conservative approximation. The covariance matrix is "scaled" because it does not include the residual variance. For the covariance for a particular outcome variable, multiply the returned `$vcov_u` by `sigma2` for that outcome.

`total` A vector of total observations per unit.

Value

A list with model components.

`ei_riesz` *Estimate Riesz representer for ecological inference*

Description

Fits a penalized Riesz regression for ecological inference, allowing for overall estimates of conditional means using `ei_est()`.

Usage

```
ei_riesz(x, ..., weights, penalty, scale = TRUE)

## S3 method for class 'formula'
ei_riesz(formula, data, total, weights, penalty, scale = TRUE, ...)

## S3 method for class 'ei_spec'
ei_riesz(x, weights, penalty, scale = TRUE, ...)

## S3 method for class 'data.frame'
ei_riesz(x, z, total, weights, penalty, scale = TRUE, ...)

## S3 method for class 'matrix'
ei_riesz(x, z, total, weights, penalty, scale = TRUE, ...)

## Default S3 method:
ei_riesz(x, ...)
```

Arguments

`x` Depending on the context:

- A **data frame** of predictors.
- A **matrix** of predictors.
- An ei_spec object containing the outcome, predictor, and covariates.

Predictors must be proportions that sum to 1 across rows. You can use `ei_proportions()` to assist in preparing predictor variables. Covariates in an ei_spec object are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.

`...` Not currently used, but required for extensibility.

`weights` <data-masking> A vector of unit weights for estimation. These may be the same or different from the total number of observations in each aggregate unit (see the `total` argument to `ei_spec()`). See the discussion below under 'Weights' for choosing this parameter. The default, uniform weights, makes a slightly stronger-than-necessary assumption about the relationship between the unit totals and the unknown data.

`penalty` The ridge penalty (a non-negative scalar), which must be specified. Recommended value is the same penalty used in `ei_ridge()`, which is stored in the `penalty` entry of the fitted model object.

`scale` If `TRUE`, scale covariates `z` to have unit variance.

`formula` A formula such as `y ~ x0 + x1 | z` specifying the outcome `y` regressed on the predictors of interest `x` and any covariates `z`. The predictors should form a partition, that is, `x0 + x1 = 1` for each observation. Users can be include more than two predictors as well, e.g. `pct_white + pct_black + pct_hisp + pct_other`. If there are just two predictors, it is acceptable to only include one in the formula; the other will be formed as 1 minus the provided predictor. Include additional covariates separated by a vertical bar `|`. These covariates are strongly recommended for reliable ecological inference. Covariates are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.

`data` When a **formula** is used, `data` is a **data frame** containing both the predictors and the outcome.

`total` <data-masking> A variable containing the total number of observations in each aggregate unit. For example, the column containing the total number of voters. Required by default.

`z` When `x` is a **data frame** or **matrix**, `w` are any covariates, specified as:

- A **data frame** with numeric columns.
- A **matrix**

These are shifted to have mean zero. If `scale=TRUE` (the default), they are also scaled to have unit variance.

Details

The regression is calculated using the singular value decomposition.

Value

An `ei_riesz` object.

Weights

The weakest identification result for ecological inference makes no assumption about the number of observations per aggregate unit (the totals). It requires, however, weighting the estimation steps according to the totals. This may reduce efficiency when the totals are variable and a slightly stronger condition holds.

Specifically, if the totals are conditionally mean-independent of the missing data (the aggregation-unit level means of the outcome within each predictor level), given covariates, then it is appropriate to use uniform weights in estimation, or any fixed set of weights.

In general, estimation efficiency is improved when units with larger variance in the outcome receive less weight. Various built-in options are provided by the helper functions in `ei_wgt()`.

References

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

Examples

```
data(elec_1968)

# Recommended: get ridge penalty from ei_ridge()
spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs,
               total = pres_total, covariates = c(pop_urban, farm))
m = ei_ridge(spec)

ei_riesz(spec, penalty = m$penalty)

rr = ei_riesz(~ vap_white + vap_black + vap_other | pop_urban + farm,
              data = elec_1968, total = pres_total, penalty = m$penalty)
summary(rr)

# Examine the weights and check they have mean 1
head(weights(rr, group = "vap_black"))
colMeans(weights(rr))

mean(elec_1968$pres_ind_wal * weights(rr, "vap_white"))
```

`ei_sens` *Conduct a sensitivity analysis for estimated ecological quantities*

Description

Relates confounding of an omitted variable with predictor or outcome to bias in ecological estimates, using the nonparametric sensitivity analysis of Chernozhukov et al. (2024).

Usage

```
ei_sens(
  est,
  c_outcome = seq(0, 1, 0.01)^2,
  c_predictor = seq(0, 1, 0.01)^2,
  bias_bound = NULL,
  confounding = 1,
  expand_ci = TRUE
)
```

Arguments

`est`	A set of estimates from `ei_est()` using both regression and Riesz representer.
`c_outcome`	The (nonparametric) partial R^2 of the omitted variables with the outcome variables. Must be between 0 and 1. Can be a vector, in which case all combinations of values with `c_predictor` are used.
`c_predictor`	How much variation latent variables create in the Riesz representer, i.e. $1 - R^2$ of the true Riesz representer on the estimated one without the omitted variable. Must be between 0 and 1. Can be a vector, in which case all combinations of values with `c_outcome` are used.
`bias_bound`	If provided, overrides `c_predictor` and finds values of `c_predictor` that correspond to (the absolute value of) the provided amount of bias.
`confounding`	The confounding parameter (ρ), which must be between 0 and 1 (the adversarial worst-case).
`expand_ci`	If `TRUE` and confidence intervals are present in `est`, expand the width of the intervals in each direction by the calculated bias bound.

Details

The parameter `c_predictor` equals $1 - R^2_{\alpha \sim \alpha_s}$, where α is the true Riesz representer and α_s is the Riesz representer with the observed covariates. The RR can be equivalently expressed as

$$\alpha(n, \bar{x}_j, z, a) = u(n, \bar{x}_j) \partial_{\bar{x}_j} \log f(\bar{x}_j \mid z, a),$$

where $u(n, \bar{x}_j)$ is the weighting term, A is the unobserved confounder, and f is the conditional density. The corresponding `c_predictor` is then

$$1 - R^2_{\alpha \sim \alpha_s} = 1 - \frac{\mathbb{E}[u(N, \bar{X}_j)(\partial_{\bar{x}_j} \log f(\bar{x}_j \mid Z))^2]}{\mathbb{E}[u(N, \bar{X}_j)(\partial_{\bar{x}_j} \log f(\bar{x}_j \mid Z, A))^2]}.$$

The bounds here are plug-in estimates and do not incorporate sampling uncertainty. As such, they may fail to cover the true value in finite samples, even under large enough sensitivity parameters; see Section 5 of Chernozhukov et al. (2024).

Value

A data frame of the same format as `est`, but with additional columns: `c_outcome` and `c_predictor`, matching all combinations of those arguments, and `bias_bound`, containing the bound on the amount of bias. The data frame has additional class `ei_sens`, which supports a `plot.ei_sens()` method.

References

Chernozhukov, V., Cinelli, C., Newey, W., Sharma, A., & Syrgkanis, V. (2024). *Long story short: Omitted variable bias in causal machine learning* (No. w30302). National Bureau of Economic Research.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_ind_wal,
               total = pres_total, covariates = c(state, pop_urban, farm))
m = ei_ridge(spec)
rr = ei_riesz(spec, penalty = m$penalty)
est = ei_est(m, rr, spec)

ei_sens(est, c_outcome=0.2)

# How much variation would the regression residual need to explain of
# Riesz representer to cause bias of 0.4?
ei_sens(est, c_outcome=1, bias_bound=0.4)

# Update confidence intervals and extract as matrix
est = ei_est(m, rr, spec, conf_level=0.95)
sens = ei_sens(est, c_outcome=0.5, c_predictor=0.2)
as.matrix(sens, "conf.high")

# Works for contrasts as well
est = ei_est(m, rr, spec, contrast = list(predictor=c(1, -1, 0)))
ei_sens(est, c_outcome=0.5, c_predictor=0.5)
```

`ei_sens_rv` *Robustness values for ecological inference*

Description

The robustness value is the minimum bound for both `c_outcome` and `c_predictor` in `ei_sens()` such that the bias bound is a certain value. For example, if the provided bias bound is 0.5, meaning a bias of magnitude 0.5 would be considered problematic, then the robustness value is the minimum amount of confounding of outcome and predictor (more specifically, the Riesz representer) that would lead to bias of magnitude 0.5.

Usage

```
ei_sens_rv(est, bias_bound, confounding = 1)
```

Arguments

`est`	A set of estimates from `ei_est()` using both regression and Riesz representer.
`bias_bound`	<`data-masking`> A bias bound: an amount of bias which is considered substantial. Evaluated in the context of `est`, so that one can to refer to `std.error` and `estimate` as needed.
`confounding`	The confounding parameter (ρ), which must be between 0 and 1 (the adversarial worst-case).

Value

A data frame of the same format as `est`, but with a new `rv` column containing the robustness values.

References

Chernozhukov, V., Cinelli, C., Newey, W., Sharma, A., & Syrgkanis, V. (2024). *Long story short: Omitted variable bias in causal machine learning* (No. w30302). National Bureau of Economic Research.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_ind_wal,
               total = pres_total, covariates = c(state, pop_urban, farm))
m = ei_ridge(spec)
rr = ei_riesz(spec, penalty = m$penalty)
est = ei_est(m, rr, spec)

ei_sens_rv(est, 0.1) # how much confounding for bias of 0.1
ei_sens_rv(est, 2 * std.error) # how much confounding for bias of 2 SE

# How much confounding to equalize all estimates (no polarization)
y_avg = weighted.mean(elec_1968$pres_ind_wal, elec_1968$pres_total)
ei_sens_rv(est, estimate - y_avg)

# Extract as matrix
as.matrix(ei_sens_rv(est, 0.2), "rv")

# Works for contrasts as well
est = ei_est(m, rr, spec, contrast = list(predictor=c(1, -1, 0)))
ei_sens_rv(est, estimate) # how much to eliminate disparity
```

`ei_spec` *Specify an ecological inference problem*

Description

Uses tidy-select syntax to specify outcomes, predictors, and covariates. The result of this function can be passed directly into `ei_ridge()` or `ei_riesz()`, or plotted with `plot()`.

Usage

```
ei_spec(
  data,
  predictors,
  outcome,
  total,
```

```
  covariates = NULL,
  preproc = NULL,
  strip = FALSE
)

## S3 method for class 'ei_spec'
weights(object, normalize = TRUE, ...)
```

Arguments

data	A data frame.
predictors	<tidy-select> Predictor variables. This is the x variable in ecological regression that is of primary interest. For example, the columns containing the percentage of each racial group.
outcome	<tidy-select> Outcome variables. This is the y variable in ecological regression that is of primary interest. For example, the columns containing the percentage of votes for each party.
total	<data-masking> A variable containing the total number of observations in each aggregate unit. For example, the column containing the total number of voters. Required by default.
covariates	<tidy-select> Covariates.
preproc	An optional function which takes in a data frame of covariates and returns a modeling-ready numeric matrix of covariates. Useful to apply any preprocessing, such as a basis transformation, as part of the estimation process. Passed to rlang::as_function(), and so supports purrr-style lambda functions. This function is called once when forming the ei_spec object so that the same processing is applied during all estimation steps. The function may also be re-used by other functions, like ei_bench() and ei_test_car(). The default is a call to model.matrix() with the formula ~ 0 + ., i.e., all covariates and no predictors.
strip	Whether to strip common prefixes from column names within each group. For example, columns named vap_white, vap_black, and vap_hisp would be renamed white, black and other in the model and output.
object	An ei_spec object.
normalize	If TRUE, normalize the totals to have mean 1.
...	Additional arguments (ignored).

Details

The function is lightweight and does not perform any checking of the arguments, bounds, sum constraints, etc. All of these checks are performed by functions that use ei_spec objects.

Value

An ei_spec object, which is a data frame with additional attributes recording predictors, outcomes, total, and covariates.

Methods (by generic)

- `weights(ei_spec)`: Extract the totals from a specification

Examples

```
data(elec_1968)
ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs, pres_total)

# basis expansion
if (requireNamespace("bases", quietly = TRUE)) {
    spec = ei_spec(
        data = elec_1968,
        predictors = vap_white:vap_other,
        outcome = pres_dem_hum:pres_abs,
        total = pres_total,
        covariates = c(pop_city:pop_rural, farm:educ_coll, starts_with("inc_")),
        preproc = ~ bases::b_bart(.x, trees = 500)
    )
}
```

---

ei_synthetic *Generate synthetic ecological data*

Description

Samples data from a following truncated Normal ecological model. The data can be generated completely at random, or can be generated conditional on provided predictors `x` and/or covariates `z`.

Usage

```
ei_synthetic(
  n,
  p = 0,
  n_x = 2,
  x = n_x:1,
  z = 0.25 * exp(-(seq_len(p) - 1)/2),
  r2_xz = 0.5,
  r2_bz = 0.5,
  b_loc = NULL,
  b_cov = NULL
)
```

Arguments

n	The number of rows (geographies) to generate. Defaults to the number of rows in x or z, if they are a matrix or data frame.
p	The number of covariates. Defaults to the number of columns in z, if it is a matrix or data frame, or the length of z, if it is a vector of singular values.
n_x	The number of predictor variables. Defaults to the number of columns in x, if it is a matrix or data frame, or the length of x, if it is a vector of mean parameters for the softmax-transformed Normal distribution.
x	Either a matrix or data frame containing the predictor percentages in each row, or a vector containing Dirichlet parameters to use in sampling predictor percentages.
z	A matrix or data frame containing geography-level covariates, or a vector of values to form a Toeplitz covariance matrix for the random covariates.
r2_xz	The approximate R^2 of the covariates z and predictors x. See the model specification for details. If either r2_xz or r2_bz are zero, then there is no confounding, and an unadjusted Goodman regression will estimate the global parameters correctly.
r2_bz	The approximate R^2 of the covariates z and unit-level parameters b. See the model specification for details. If either r2_xz or r2_bz are zero, then there is no confounding, and an unadjusted Goodman regression will estimate the global parameters correctly.
b_loc	The center of the distribution of geography-level parameters. Defaults to a linearly spaced sequence across groups from 0.5 to 0.9. Because of the truncation, this will not exactly be the mean of the geography-level parameters.
b_cov	The residual covariance matrix for geography-level parameters. Defaults to `0.02 * (1 + diag(n_x))`.

Details

This function samples data from the following truncated Normal ecological model:

$$\begin{pmatrix} x_i \\ z_i \end{pmatrix} \overset{\text{iid}}{\sim} \mathcal{N}_{[0,1]^{n_x} \times \mathbb{R}^p} \left(\begin{pmatrix} \mu_x \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_x & \Gamma \\ \Gamma & T \end{pmatrix} \right)$$

$$\eta = z_i^\top \Lambda + \mathtt{b_{loc}}$$

$$b_i \overset{\text{iid}}{\sim} \mathcal{N}_{[0,1]^{n_x}} (\eta, \mathtt{B_{cov}})$$

$$y_i = b_i^\top x_i,$$

where μ_x and Σ_x are the mean and covariance of the Normal approximation to a Dirichlet distribution with parameters supplied by the x argument below, and Γ, T, and Γ are matrices sampled to have certain properties, as described below. The subscripts on $\mathcal{N}$ indicate truncation; i.e., both the predictors x and the unit-level parameters b are truncated to the *n_x*-dimensional hypercube.

The matrix T is a symmetric Toeplitz matrix with diagonals provided by the z argument. Generally, a decreasing set of nonnegative values will be sufficient for a positive definite T.

The matrices Γ and Λ are initially filled with independent samples from a standard Normal distribution. Γ is then projected so that its rows sum to zero, preserving the sum-to-1 requirement on x, and so that its columns are scaled to produce the correct R^2 value matching r2_xz. The matrix Λ is likewise scaled to produce the correct R^2 value matching r2_bz. Due to the truncation in the sampling of x and b, the in-sample R^2 values will generally be slightly smaller than the provided arguments.

Aspects of the model can be replaced with data provided to the function. If x or z is provided as a matrix or data frame, then the other value is sampled from its marginal distribution. If both are provided, then the first row of the model is skipped.

Value

An ei_spec object with additional attributes:

- b_loc and b_cov
- Lambda with the coefficients of z
- eta, the linear predictor for b
- est_true, the mean of the geography-level parameters, formatted similarly to the output from ei_est()
- r2_xz_act and r2_bz_act, containing the actual (sample) R^2 values for x and z, and b and z, respectively.

Examples

```
ei_synthetic(n = 10)

ei_synthetic(n = 10, p = 2, n_x = 3)

# Manual hyperparameters: x2 dominant and z1, z2 very correlated
ei_synthetic(n = 10, x = c(1, 95, 4), z = c(10, 9.999))

# Condition on provided x but not z
data(elec_1968)
ei_synthetic(
    x = cbind(elec_1968$pop_white, 1 - elec_1968$pop_white),
    p = 5,
    b_loc = c(0.3, 0.9),
    b_cov = matrix(c(0.02, 0.016, 0.016, 0.2), nrow=2)
)
```

---

ei_test_car *Test the coarsening at random (CAR) assumption*

Description

The CAR assumption, which is required for accurate estimates with `ei_est()`, implies that the conditional expectation function (CEF) of the aggregate outcomes will take a certain partially linear form. This function tests that implication by comparing a fully nonparametric estimate of the CEF to one in the partially linear form implied by CAR, and comparing their goodness-of-fit. It is **very important** that `preproc` be used in the `spec` to perform a rich basis transformation of the covariates and predictors; a missing `preproc` will lead to a warning.

Usage

```
ei_test_car(
  spec,
  weights,
  iter = 1000,
  undersmooth = 1.5,
  use_chisq = nrow(spec) >= 2000,
  use_hc = FALSE
)
```

Arguments

`spec`	An `ei_spec` object created with `ei_spec()`. The object should use the `preproc` argument to specify a rich basis expansion of the covariates and predictors.
`weights`	<`data-masking`> A vector of unit weights for estimation. These may be the same or different from the total number of observations in each aggregate unit (see the `total` argument to `ei_spec()`). See the discussion below under 'Weights' for choosing this parameter. The default, uniform weights, makes a slightly stronger-than-necessary assumption about the relationship between the unit totals and the unknown data.
`iter`	The number of permutations to use when estimating the null distribution, including the original identity permutation. Ignored when `use_chisq = TRUE`.
`undersmooth`	A value to divide the estimated ridge penalty by when partialling out the partially linear component of the model. A larger value will smooth the partially linear component less, which may improve Type I error control in finite samples at the cost of power.
`use_chisq`	If `TRUE`, use the asymptotic chi-squared distribution for the Wald test statistic instead of conducting a permutation test. Only appropriate for larger sample sizes (Helwig 2022 recommends at least 200 when a single predictor is used).
`use_hc`	If `TRUE`, use a heteroskedasticity-consistent covariance estimate. More computationally intensive, but may make a difference in small samples or when there is substantial heteroskedasticity.

Details

The test is a Kennedy-Cade (1996) style permutation test on a Wald statistic for the coefficients not included in the "reduced" model that would be fit by `ei_ridge()`. The test statistic is asymptotically chi-squared under the null and may be anti-conservative in small samples, especially when the dimensionality of the basis expansion is large.

Value

A tibble with one row per outcome variable and columns describing the test results. The `p.value` column contains the p-values for the test. P-values are not adjusted by default; pass them to `stats::p.adjust()` if desired.

References

Helwig, N. E. (2022). Robust Permutation Tests for Penalized Splines. *Stats*, 5(3), 916-933.

Kennedy, P. E., & Cade, B. S. (1996). Randomization tests for multiple regression. *Communications in Statistics-Simulation and Computation*, 25(4), 923-936.

McCartan, C., & Kuriwaki, S. (2025+). Identification and semiparametric estimation of conditional means from aggregate data. Working paper arXiv:2509.20194.

Examples

```
data(elec_1968)

# basis expansion: poly() with degree=2 not recommended in practice
preproc = if (requireNamespace("bases", quietly = TRUE)) {
    ~ bases::b_bart(.x, trees = 100)
} else {
    ~ poly(as.matrix(.x), degree=2, simple=TRUE)
}

spec = ei_spec(
    data = elec_1968,
    predictors = vap_white:vap_other,
    outcome = pres_dem_hum:pres_abs,
    total = pres_total,
    covariates = c(pop_city:pop_rural, farm:educ_coll, starts_with("inc_")),
    preproc = preproc
)

ei_test_car(spec, iter=20) # use a larger number in practice
```

---

ei_wgt *Estimation weighting models*

Description

Several built-in helper functions to generate estimation weights from a vector of unit totals, or an existing `ei_spec()` object.

Usage

```
ei_wgt_unif(x)

ei_wgt_prop(x)

ei_wgt_sqrt(x)
```

Arguments

x	A numeric vector of unit totals, or an existing ei_spec() object.

Value

A numeric vector of estimation weights with the same number of observations as x. These will have mean 1.

Functions

- ei_wgt_unif(): Uniform weights across units with any population. Appropriate if the unit-level variance is constant, i.e., homoskedastic.
- ei_wgt_prop(): Weights proportional to the totals. Appropriate if the unit-level variance is inversely proportional to the number of observations.
- ei_wgt_sqrt(): Weights proportional to the square root of the totals. Appropriate if the unit-level variance is inversely proportional to the square root of the number of observations.

Examples

```
data(elec_1968)

ei_wgt_unif(head(elec_1968$pres_total))

spec = ei_spec(head(elec_1968), predictors = vap_white:vap_other,
               outcome = pres_ind_wal, total = pres_total)
ei_wgt_prop(spec)
ei_wgt_sqrt(spec)
```

---

ei_wrap_model *Wrap another predictive model for use in* ei_est

---

Description

Stores additional data and attributes on a generic model class so that it can be used as the regr argument to ei_est(). Given the wide variety of model classes, there is no guarantee this function will work. However, most model classes supporting a fitted() and predict() method will work as long as there is no transformation of the predictor variables as part of the model formula or fitting.

Usage

```
ei_wrap_model(x, data, predictors = NULL, outcome = NULL, ...)
```

Arguments

x	A model object, supporting `fitted()` and `predict()` generics.
data	A data frame or matrix containing the data used to fit the model, or an `ei_spec()` object (recommended). If the latter, then the `predictors` and `outcome` arguments are ignored and need not be provided.
predictors	<tidy-select> Predictor variables. This is the `x` variable in ecological regression that is of primary interest. For example, the columns containing the percentage of each racial group.
outcome	<tidy-select> Outcome variables. This is the `y` variable in ecological regression that is of primary interest. For example, the columns containing the percentage of votes for each party.
...	Additional arguments passed to the `predict()` method.

Value

An `ei_wrapped` object, which has the information required to use the provided `x` with `ei_est()`.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_ind_wal, pres_total,
               covariates = c(pop_urban, farm))

# Note: this is not a model recommended for valid ecological inference!
m = suppressWarnings(
    glm(pres_ind_wal ~ 0 + vap_white + vap_black + vap_other + pop_urban + farm,
        data = spec, family = "binomial")
)
m_wrap = ei_wrap_model(m, spec, type = "response")
print(m_wrap)

ei_est(m_wrap, data = spec) # notice all estimates nonnegative
```

---

elec_1968 *1968 U.S. presidential election data*

Description

County-level dataset containing election results and demographics from the 1968 U.S. presidential election in the states of Virginia, North Carolina, South Carolina, Georgia, Florida, Alabama, Mississippi, Louisiana, Arkansas, Tennessee, and Texas.

Usage

```
elec_1968
```

Format

A data frame with 1,143 rows and 41 variables:

`fips` County FIPS code

`state` State name

`abbr` State abbreviation

`region` Census region

`division` Census division

`county` County name

`pop` Total population at the 1970 census

`pop_city`, `pop_urban`, `pop_rural` Proportion of the population in city, urban and rural areas

`pop_white`, `pop_black`, `pop_aian`, `pop_asian`, `pop_hisp` Proportion of the population in each racial group. The first four columns sum to 1, but `pop_hisp` should be considered separately.

`vap` Voting-age population at the 1970 census

`vap_white`, `vap_black`, `vap_other` Proportion of the voting-age population in each racial group

`farm`, `nonfarm` Proportion of the population in farm and non-farm households

`educ_elem`, `educ_hsch`, `educ_coll` Proportion of the population with elementary, high school, and college education

`cvap` Citizen voting-age population at the 1970 census

`cvap_white`, `cvap_black`, `cvap_other` Proportion of the citizen voting-age population in each racial group

`inc_00_03k`, `inc_03_08k`, `inc_08_25k`, `inc_25_99k` Proportion of the population in households with income in each bracket. Median household income in 1970 was $9,870

`pres_dem_hum`, `pres_rep_nix`, `pres_ind_wal`, `pres_abs`, `pres_oth` Proportion of votes for Humphrey, Nixon, and Wallace, and other candidates. `pres_abs` is calculated as one minus the Humphrey, Nixon, and Wallace vote shares.

`pres_total` Total number of votes cast for president.

`pres_turn` Proportion of the voting-age population that turned out to vote.

`pres_turn_icpsr` Proportion of the voting-age population that turned out to vote using ICPSR data in the denominator.

`adj` A FIPS-indexed adjacency list capturing the geographic (rook) adjacency between counties. To convert to a 1-indexed adjacency list, run `lapply(elec_1968$adj, function(x) match(x, elec_1968$fips))`.

Source

Census data: Steven Manson, Jonathan Schroeder, David Van Riper, Katherine Knowles, Tracy Kugler, Finn Roberts, and Steven Ruggles. IPUMS National Historical Geographic Information System: Version 19.0. 1970 Decennial Census. Minneapolis, MN: IPUMS. 2024. doi:10.18128/D050.V19.0

Presidential election data: Clubb, Jerome M., Flanigan, William H., and Zingale, Nancy H. Electoral Data for Counties in the United States: Presidential and Congressional Races, 1840-1972. Inter-university Consortium for Political and Social Research (distributor), 2006-11-13. doi:10.3886/ICPSR08611.v1

`plot.ei_sens` *Bias contour plot for ecological inference estimates*

Description

Displays bias bound as a function of `c_outcome` and `c_predictor` in `ei_sens()` on a contour plot. Bounds on the outcome, and standard errors of the point estimate, can be overlaid as contours on the plot to aid in interpretation. Benchmarked values of `c_outcome` and `c_predictor` based on the observed covariates can also be overlaid.

Usage

```
## S3 method for class 'ei_sens'
plot(
  x,
  y = NULL,
  predictor = NULL,
  bounds = NULL,
  bench = NULL,
  plot_se = 1:3,
  contour_exp = -2:-1,
  ...,
  lwd = 1,
  pch = 8,
  cex = 1
)
```

Arguments

`x`	An ei_sens object
`y`	An outcome variable, as a character vector. Defaults to first.
`predictor`	A predictor variable to plot, as a character vector. Defaults to first.
`bounds`	A vector `c(min, max)` of bounds for the outcome, which will affect the contours which are plotted. If `bounds = NULL` (the default), they will be inferred where possible. Setting `bounds = FALSE` turns off these labels.

bench	A data frame of benchmark values, from ei_bench(), to plot.
plot_se	A vector of multiples of the standard error to plot as contours.
contour_exp	Powers of 10 for which to plot contours of the bias bound.
...	Additional arguments passed on to contour()
lwd	Scaling factor for the contour line widths
pch	The point type (see points()) for the benchmark values, if provided
cex	Scaling factor for the benchmark points and labels, if provided

Value

None, called for side effects (plotting).

References

Chernozhukov, V., Cinelli, C., Newey, W., Sharma, A., & Syrgkanis, V. (2024). *Long story short: Omitted variable bias in causal machine learning* (No. w30302). National Bureau of Economic Research.

Examples

```
data(elec_1968)

spec = ei_spec(elec_1968, vap_white:vap_other, pres_ind_wal,
               total = pres_total, covariates = c(state, pop_urban, farm))
m = ei_ridge(spec)
rr = ei_riesz(spec, penalty = m$penalty)
est = ei_est(m, rr, spec)
sens = ei_sens(est)

plot(sens)

plot(sens, bench = ei_bench(spec), plot_se=NULL)
```

---

plot.ei_spec *Plot an EI specification*

Description

Useful for quickly visualizing scatterplots of outcome versus predictor variables.

Usage

```
## S3 method for class 'ei_spec'
plot(x, ..., pch = 16, cex = 0.2)
```

Arguments

x	An ei_spec object.
...	Additional arguments passed to pairs().
pch, cex	As in plot()

Value

None, called for side effects (plotting).

Examples

```
data(elec_1968)
spec = ei_spec(elec_1968, vap_white:vap_other, pres_dem_hum:pres_abs, pres_total)
plot(spec)
```

---

ridge-methods *Methods for* ei_ridge *models*

Description

Models fitted with ei_ridge() support various generic methods.

Usage

```
## S3 method for class 'ei_ridge'
predict(object, new_data, type = "numeric", ...)

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

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

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

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

## S3 method for class 'ei_ridge'
weights(object, normalize = TRUE, ...)
```

Arguments

`object`	A fitted ei_ridge model
`new_data`	A data frame, matrix, or ei_spec of new predictors.
`type`	The type of predictions to generate; only `"numeric"` is supported.
`...`	Additional arguments (ignored)
`normalize`	If `TRUE`, normalize the weights to have mean 1.

Value

See individual methods.

Functions

- `predict(ei_ridge)`: Predict from an `ei_ridge` model.
- `fitted(ei_ridge)`: Extract fitted values.
- `residuals(ei_ridge)`: Extract residuals.
- `vcov(ei_ridge)`: Extract unscaled covariance of coefficient estimates. Covariance estimate is not currently heteroskedasticity-robust. Multiply by `sigma2` from the fitted model to get the covariance matrix for a particular outcome variable.
- `summary(ei_ridge)`: Summarize the model's fitted values and R^2.
- `weights(ei_ridge)`: Extract estimation weights from a fitted model.

`weights.ei_riesz` *Extract Riesz representer weights*

Description

Extracts a single set of Riesz representer weights from an `ei_riesz` object, for a selected group.

Usage

```
## S3 method for class 'ei_riesz'
weights(object, group = TRUE, loo = FALSE, ...)
```

Arguments

`object`	An `ei_riesz()` object.
`group`	The group for which to extract the weights, as a numeric index or a character column name. The special (default) value `TRUE` will return a matrix of weights, with each column corresponding to a group.
`loo`	If `TRUE`, return the leave-one-out weights
`...`	Additional arguments (ignored)

Value

A numeric vector of weights

Index

∗ **datasets**
elec_1968, 33

as.array.ei_bounds *(*ei_bounds*)*, 4
as.array.ei_est_local *(*ei_est_local*)*, 10
as.matrix.ei_est *(*ei_est*)*, 7

contour(), *36*

ei_bench, 2
ei_bench(), *26*, *36*
ei_bounds, 4
ei_est, 7
ei_est(), *15*, *18*, *20*, *23, 24*, *29, 30*, *32, 33*
ei_est_local, 10
ei_est_local(), *18*
ei_local_cov, 13
ei_local_cov(), *10*
ei_proportions, 14
ei_proportions(), *6*, *16, 17*, *20*
ei_ridge, 15, *37, 38*
ei_ridge(), *8*, *10, 11*, *13*, *18, 19*, *21*, *25*, *30*, *37*
ei_ridge_impl, 19
ei_ridge_impl(), *19*
ei_riesz, 20
ei_riesz(), *8*, *19*, *25*, *38*
ei_riesz_impl *(*ei_ridge_impl*)*, 19
ei_riesz_impl(), *19*
ei_sens, 22, *35*
ei_sens(), *2*, *24*, *35*
ei_sens_rv, 24
ei_spec, *3*, *8*, *10*, *13*, *16*, *20*, 25, *26*, *37, 38*
ei_spec(), *5*, *8*, *10*, *16*, *21*, *30–33*
ei_synthetic, 27
ei_test_car, 29
ei_test_car(), *26*
ei_wgt, 31
ei_wgt(), *18*, *22*
ei_wgt_prop *(*ei_wgt*)*, 31
ei_wgt_sqrt *(*ei_wgt*)*, 31
ei_wgt_unif *(*ei_wgt*)*, 31
ei_wrap_model, 32
ei_wrap_model(), *8*, *10, 11*, *13*
elec_1968, 33

fitted(), *32, 33*
fitted.ei_ridge *(*ridge-methods*)*, 37

nobs.ei_est *(*ei_est*)*, 7

pairs(), *37*
plot(), *25*, *37*
plot.ei_sens, 35
plot.ei_sens(), *23*
plot.ei_spec, 36
points(), *36*
predict(), *32, 33*
predict.ei_ridge *(*ridge-methods*)*, 37

residuals.ei_ridge *(*ridge-methods*)*, 37
ridge-methods, *18*, 37
rlang::as_function(), *26*

stats::p.adjust(), *31*
summary.ei_ridge *(*ridge-methods*)*, 37

vcov.ei_est *(*ei_est*)*, 7
vcov.ei_ridge *(*ridge-methods*)*, 37

weights.ei_ridge *(*ridge-methods*)*, 37
weights.ei_riesz, 38
weights.ei_spec *(*ei_spec*)*, 25