Package: GECal (via r-universe)

October 28, 2024

Type Package

Title Generalized Entropy Calibration

Version 0.1.5

Description Generalized Entropy Calibration produces calibration weights using generalized entropy as the objective function for optimization. This approach, as implemented in the 'GECal' package, is based on Kwon, Kim, and Qiu (2024) <doi:10.48550/arXiv.2404.01076>. Unlike traditional methods, 'GECal' incorporates design weights into the constraints to maintain design consistency, rather than including them in the objective function itself.

Encoding UTF-8

URL https://github.com/yonghyun-K/GECal

BugReports https://github.com/yonghyun-K/GECal/issues

Depends R (>= 2.10.0)

LazyData true

Imports nleqslv

Suggests sampling

RoxygenNote 7.3.2

License MIT + file LICENSE

Repository https://yonghyun-k.r-universe.dev

RemoteUrl https://github.com/yonghyun-k/gecal

RemoteRef HEAD

RemoteSha 51a752d8fe829c9a67c100d633850b27ef1e9710

Contents

`estimate` *Performing statistical inference after calibration*

Description

`estimate` performs statistical inference after calibration.

Usage

```
estimate(formula, data = NULL, calibration, pimat = NULL)
```

Arguments

`formula`	An object of class "formula" specifying the calibration model.
`data`	An optional data frame containing the variables in the model (specified by `formula`).
`calibration`	An object of class "calibration", generated by `GECalib`.
`pimat`	An optional matrix contatining the joint inclusion probability matrix used for variance estimation.

Value

A list of class `estimation` including the point estimates and its standard error.

References

Kwon, Y., Kim, J., & Qiu, Y. (2024). Debiased calibration estimation using generalized entropy in survey sampling. Arxiv preprint <https://arxiv.org/abs/2404.01076>

Deville, J. C., and Särndal, C. E. (1992). Calibration estimators in survey sampling. Journal of the American statistical Association, 87(418), 376-382.

Examples

```
set.seed(11)
N = 10000
x = data.frame(x1 = rnorm(N, 2, 1), x2= runif(N, 0, 4))
pi = pt((-x[,1] / 2 - x[,2] / 2), 3);
pi = ifelse(pi >.7, .7, pi)

delta = rbinom(N, 1, pi)
Index_S = (delta == 1)
pi_S = pi[Index_S]; d_S = 1 / pi_S
x_S = x[Index_S,,drop = FALSE]
# pimat = diag(d_S^2 - d_S) / N^2 # 1 / pi_i * (1 - 1 / pi_i)

e = rnorm(N, 0, 1)
y = x[,1] + x[,2] + e;
y_S = y[Index_S] # plot(x_S, y_S)
```

```
calibration0 <- GECal::GEcalib(~ 1, dweight = d_S, data = x_S,
                               const = N,
                               entropy = "SL", method = "DS")
GECal::estimate(y_S ~ 1, calibration = calibration0)$estimate # Hajek estimator
# sum(y_S * d_S) * N / sum(d_S)

calibration <- GECal::GEcalib(~ 0, dweight = d_S, data = x_S,
const = numeric(0),
entropy = "SL", method = "DS")
GECal::estimate(y_S ~ 1, calibration = calibration)$estimate # HT estimator

calibration1 <- GECal::GEcalib(~ ., dweight = d_S, data = x_S,
                               const = colSums(cbind(1, x)),
                               entropy = "ET", method = "DS")
GECal::estimate(y_S ~ 1, calibration = calibration1)$estimate

calibration2 <- GECal::GEcalib(~ ., dweight = d_S, data = x_S,
                               const = colSums(cbind(1, x)),
                               entropy = "ET", method = "GEC0")
GECal::estimate(y_S ~ 1, calibration = calibration2)$estimate

calibration3 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
                               const = colSums(cbind(1, x, log(1 / pi))),
                               entropy = "ET", method = "GEC")
GECal::estimate(y_S ~ 1, calibration = calibration3)$estimate

calibration4 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
                               const = colSums(cbind(1, x, NA)),
                               entropy = "ET", method = "GEC")
GECal::estimate(y_S ~ 1, calibration = calibration4)$estimate

calibration5 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
                               const = colSums(cbind(1, x, NA)),
                               entropy = "ET", method = "GEC", K_alpha = "log")
GECal::estimate(y_S ~ 1, calibration = calibration5)$estimate
```

---

g *Debiasing covariate for GECalib*

Description

It returns the debiasing covariate, which is equivalent to the first order derivatie of the generalized entropy G.

Usage

```
g(x, entropy = NULL, del = NULL)
```

Arguments

x	A vector of design weights
entropy	An optional data frame containing the variables in the model (specified by formula).
del	The optional vector for threshold (δ) when entropy == "PH".

Value

A vector of debiasing covariate.

Examples

```
set.seed(11)
N = 10000
x = data.frame(x1 = rnorm(N, 2, 1), x2= runif(N, 0, 4))
pi = pt((-x[,1] / 2 - x[,2] / 2), 3);
pi = ifelse(pi >.7, .7, pi)

g_EL <- g(1 / pi, entropy = 1)
g_ET <- g(1 / pi, entropy = 0)
g_EL <- g(1 / pi, entropy = -1)
```

GEcalib *Generalized Entropy Calibration*

Description

GEcalib computes the calibration weights. Generalized entropy calibration weights maximize the generalized entropy:

$$H(\boldsymbol{\omega}) = -\sum_{i \in A} G(\omega_i),$$

subject to the calibration constraints $\sum_{i \in A} \omega_i \boldsymbol{z}_i = \sum_{i \in U} \boldsymbol{z}_i$, where A denotes the sample index, and U represents the population index. The auxiliary variables, whose population totals are known, are defined as $\boldsymbol{z}_i^T = (\boldsymbol{x}_i^T, g(d_i))$, where g is the first-order derivative of the gerenalized entropy G, and d_i is the design weight for each sampled unit $i \in A$.

Usage

```
GEcalib(
  formula,
  dweight,
  data = NULL,
  const,
  method = c("GEC", "GEC0", "DS"),
  entropy = c("SL", "EL", "ET", "CE", "HD", "PH"),
  weight.scale = 1,
  G.scale = 1,
```

```
  K_alpha = NULL,
  is.total = TRUE,
  del = NULL
)
```

Arguments

formula	An object of class "formula" specifying the calibration model.
dweight	A vector of sampling weights.
data	An optional data frame containing the variables in the model (specified by formula).
const	A vector used in the calibration constraint for population totals(or means).
method	The method to be used in calibration. See "Details" for more information.
entropy	The generalized entropy used in calibration, which can be either a numeric value or a string. If numeric, entropy represents the order of Renyi's entropy, where $G(\omega) = r^{-1}(r+1)^{-1}\omega^{r+1}$ if $r \neq 0, -1$. If a string, valid options include: "SL" (Squared-loss, $r = 1$), "EL" (Empirical Likelihood, $r = -1$), "ET" (Exponential Tilting, $r = 0$), "HD" (Hellinger Distance, $r = -1/2$), "CE" (Cross-Entropy), and "PH" (Pseudo-Huber). See "Summary" for details.
weight.scale	Positive scaling factor for the calibration weights ω_i. Asymptotics justify setting weight.scale to the finite population correction ($fpc = n/N$).
G.scale	Positive scaling factor for the generalized entropy function G. Asymptotics justify setting G.scale to the variance of the error term in a linear super-population model.
K_alpha	The K function used in joint optimization when the const of the debiasing covariate $g(d_i)$ is not available. K_alpha can be NULL, "log", or custom functions. See "Details".
is.total	Logical, TRUE if sum(const[1]) equals the population size.
del	The optional threshold (δ) used when Pseudo-Huber (PH) entropy is selected. del = quantile(dweight, 0.75) if not specified.

Details

The GEcal object returns the calibration weights and necessary information for estimating population totals(or mean).

The terms to the right of the ~ symbol in the formula argument define the calibration constraints. When method == "GEC", the debiasing covariate g(dweight) must be included in the formula. If the population total(mean) of g(dweight) is unavailable, const that corresponds to g(dweight) can be set to NA. In this case, GECalib performs joint optimization over both the calibration weights ω_i and the missing value of const.

The length of the const vector should match the number of columns in the model.matrix generated by formula. Additionally, the condition number of the model.matrix must exceed .Machine$double.eps to ensure its invertibility.

Both weight.scale and G.scale are positive scaling factors used for calibration. Note that weight.scale is not supported when method == "DS".

Let q_i be the scaling factor for the generalized entropy function G, and ϕ_i be the scaling factor for the calibration weights ω_i.

If `method == "GEC"`, `GEcalib` minimizes the negative entropy:

$$\sum_{i \in A} q_i G(\phi_i \omega_i),$$

with respect to $\boldsymbol{\omega}$ subject to the calibration constraints $\sum_{i \in A} \omega_i \boldsymbol{z}_i = \sum_{i \in U} \boldsymbol{z}_i$, where $\boldsymbol{z}_i^T = (\boldsymbol{x}_i^T, q_i \phi_i g(\phi_i d_i))$, A denotes the sample index, and U represents the population index.

If `method == "GEC"`, but an element of `const` corresponding to the debiasing covariate $g(d_i)$ is `NA`, `GEcalib` minimizes the negative adjusted entropy:

$$\sum_{i \in A} q_i G(\phi_i \omega_i) - K(\alpha),$$

with respect to $\boldsymbol{\omega}$ and α subject to the calibration constraints $\sum_{i \in A} \omega_i (\boldsymbol{x}_i^T, q_i \phi_i g(\phi_i d_i)) = \left(\sum_{i \in U} \boldsymbol{x}_i, \alpha\right)$, where the solution $\hat{\alpha}$ is an estimate of population total for $g(d_i)$. Examples of $K(\alpha)$ includes $K(\alpha) = \alpha$ when `K_alpha == NULL`, and

$$K(\alpha) = \left(\sum_{i \in A} d_i g(d_i) + N\right) \log \left| \frac{1}{N} \sum_{i \in A} q_i \phi_i \omega_i g(\phi_i \omega_i) + 1 \right|$$

when `K_alpha == "log"`.

If `method == "GEC0"`, `GEcalib` minimizes the negative adjusted entropy:

$$\sum_{i \in A} q_i G(\phi_i \omega_i) - q_i \phi_i \omega_i g(\phi_i \omega_i)$$

with respect to $\boldsymbol{\omega}$ subject to the calibration constraints $\sum_{i \in A} \omega_i \boldsymbol{x}_i = \sum_{i \in U} \boldsymbol{x}_i$.

If `method == "DS"`, `GEcalib` minimizes the divergence between $\boldsymbol{\omega}$ and $\boldsymbol{d}$:

$$\sum_{i \in A} q_i d_i \tilde{G}(\omega_i / d_i)$$

with respect to $\boldsymbol{\omega}$ subject to the calibration constraints $\sum_{i \in A} \omega_i \boldsymbol{x}_i = \sum_{i \in U} \boldsymbol{x}_i$. When `method == "DS"`, `weight.scale`, the scaling factor for the calibration weights ϕ_i, is not applicable.

Examples of G and $\tilde{G}$ are given in "Summary".

Value

A list of class `calibration` including the calibration weights and data needed for estimation.

Summary

The table below provides a comparison between the **GEC** and **DS** methods.

GEC	**DS**

$$\min_{\boldsymbol{\omega}} \left(-H(\boldsymbol{\omega})\right) = \sum_{i\in A} G(\omega_i) \qquad \min_{\boldsymbol{\omega}} D(\boldsymbol{\omega}, \boldsymbol{d}) = \sum_{i\in A} d_i \tilde{G}(\omega_i/d_i)$$

$$\text{s.t. } \sum_{i\in A} \omega_i (\boldsymbol{x}_i^T, g(d_i)) = \sum_{i\in U} (\boldsymbol{x}_i^T, g(d_i)) \qquad \text{s.t. } \sum_{i\in A} \omega_i \boldsymbol{x}_i^T = \sum_{i\in U} \boldsymbol{x}_i^T$$

$$G(\omega) = \begin{cases} \frac{1}{r(r+1)}\omega^{r+1} & r \neq 0, -1 \\ \omega\log\omega - \omega & r = 0(\text{ET}) \\ -\log\omega & r = -1(\text{EL}) \end{cases} \qquad \tilde{G}(\omega) = \begin{cases} \frac{1}{r(r+1)}\left(\omega^{r+1} - (r+1)\omega + r\right) & r \neq 0, -1 \\ \omega\log\omega - \omega + 1 & r = 0(\text{ET}) \\ -\log\omega + \omega - 1 & r = -1(\text{EL}) \end{cases}$$

If `method == "GEC"`, further examples include

$$G(\omega) = (\omega - 1)\log(\omega - 1) - \omega\log\omega$$

when `entropy == "CE"`, and

$$G(\omega) = \delta^2 \left(1 + (\omega/\delta)^2\right)^{1/2}$$

for a threshold δ when `entropy == "PH"`.

Author(s)

Yonghyun Kwon

References

Kwon, Y., Kim, J., & Qiu, Y. (2024). Debiased calibration estimation using generalized entropy in survey sampling. Arxiv preprint <https://arxiv.org/abs/2404.01076>

Deville, J. C., and Särndal, C. E. (1992). Calibration estimators in survey sampling. Journal of the American statistical Association, 87(418), 376-382.

Examples

```
set.seed(11)
N = 10000
x = data.frame(x1 = rnorm(N, 2, 1), x2= runif(N, 0, 4))
pi = pt((-x[,1] / 2 - x[,2] / 2), 3);
pi = ifelse(pi >.7, .7, pi)

delta = rbinom(N, 1, pi)
Index_S = (delta == 1)
pi_S = pi[Index_S]; d_S = 1 / pi_S
x_S = x[Index_S,]

# Deville & Sarndal(1992)'s calibration using divergence
w1 <- GECal::GEcalib(~ ., dweight = d_S, data = x_S,
                     const = colSums(cbind(1, x)),
                     entropy = "ET", method = "DS")$w

# Generalized entropy calibration without debiasing covariate
w2 <- GECal::GEcalib(~ ., dweight = d_S, data = x_S,
                     const = colSums(cbind(1, x)),
```

```
                    entropy = "ET", method = "GEC0")$w
all.equal(w1, w2)

# Generalized entropy calibration with debiasing covariate
w3 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
                    const = colSums(cbind(1, x, log(1 / pi))),
                    entropy = "ET", method = "GEC")$w

# Generalized entropy calibration with debiasing covariate
# when its population total is unknown
w4 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
                    const = colSums(cbind(1, x, NA)),
                    entropy = "ET", method = "GEC")$w
all.equal(w1, w4)

w5 <- GECal::GEcalib(~ . + g(d_S), dweight = d_S, data = x_S,
const = colSums(cbind(1, x, NA)),
entropy = "ET", method = "GEC", K_alpha = "log")$w
```

`IAdata` *Synthetic pesticides data in Iowa*

Description

A synthetic proprietary pesticide usage survey data in Iowa CRD(Crop Reporting District) collected from GfK Kynetec in 2020.

Format

A data frame with 1197 rows on the following 32 variables:

Corn10, Corn20, Corn30, Corn40, Corn50, Corn60, Corn70 Haversted acres of corn in each CRD

Soybean10, Soybean20, Soybean30, Soybean40, Soybean50, Soybean60, Soybean70, Soybean90 Haversted acres of soybean in each CRD

Alfalfa10, Alfalfa30, Alfalfa40, Alfalfa50, Alfalfa70, Alfalfa80 Haversted acres of alfalfa in each CRD

Pasture10, Pasture20, Pasture30, Pasture40, Pasture50, Pasture60, Pasture70, Pasture80, Pasture90 Acres of pasture in each CRD

d Design weights, or inverse first-order inclusion probabilities of the sample

y Pesticide usage($) which is of an interest.

Details

The original data is contaminated by adding noise and creating missing values and imputation.

Examples

```
data(IAdata)
data(IApimat)

total <- c(
Corn10 = 2093000, Corn20 = 1993600, Corn30 = 1803200, Corn40 = 2084600,
Corn50 = 2056600, Corn60 = 1429400, Corn70 = 2539600,
Soybean10 = 1472980, Soybean20 = 1192860, Soybean30 = 721920,
Soybean40 = 1477680, Soybean50 = 1353600, Soybean60 = 918380,
Soybean70 = 1485200, Soybean90 = 777380, Alfalfa10 = 60590,
Alfalfa30 = 154395, Alfalfa40 = 57816, Alfalfa50 = 150453,
Alfalfa70 = 66065, Alfalfa80 = 240681, Pasture10 = 141947,
Pasture20 = 61476, Pasture30 = 188310, Pasture40 = 213635,
Pasture50 = 160737, Pasture60 = 222214, Pasture70 = 250807,
Pasture80 = 570647, Pasture90 = 232630
)

calibration <- GECal::GEcalib(~ 0, dweight = d, data = IAdata,
                              const = numeric(0),
                              entropy = "EL", method = "DS")
GECal::estimate(y ~ 1, data = IAdata, calibration = calibration, pimat = IApimat)$estimate


calibration <- GECal::GEcalib(~ 0 + . -y -d, dweight = d, data = IAdata,
                              const = total,
                              entropy = "SL", method = "DS")
GECal::estimate(y ~ 1, data = IAdata, calibration = calibration, pimat = IApimat)$estimate

calibration <- GECal::GEcalib(~ 0 + . -y -d, dweight = d, data = IAdata,
                              const = c(total),
                              entropy = "ET", method = "DS")
GECal::estimate(y ~ 1, data = IAdata, calibration = calibration, pimat = IApimat)$estimate


calibration <- GECal::GEcalib(~ 0 + . -y -d + g(d), dweight = d, data = IAdata,
                              const = c(total, NA),
                              entropy = "HD", method = "GEC")
GECal::estimate(y ~ 1, data = IAdata, calibration = calibration, pimat = IApimat)$estimate

calibration <- GECal::GEcalib(~ 0 + . -y -d, dweight = d, data = IAdata,
                              const = total,
                              entropy = "HD", method = "GEC0")
GECal::estimate(y ~ 1, data = IAdata, calibration = calibration, pimat = IApimat)$estimate
```

IApimat *A matrix used for variance estimation in IAdata*

Description

The matrix that is used for variance estimation in IAdata. The sample is collected from a straitified random sampling.

Examples

```
data(IApimat)
```

Index