Package: jointCalib (via r-universe)

September 5, 2024

Type Package

Title A Joint Calibration of Totals and Quantiles

Version 0.1.2

Description A small package containing functions to perform a joint
calibration of totals and quantiles. The calibration for totals
is based on Deville and Särndal (1992)
<doi:10.1080/01621459.1992.10475217>, the calibration for
quantiles is based on Harms and Duchesne (2006)
<https://www150.statcan.gc.ca/n1/en/catalogue/12-001-X20060019255>.
The package uses standard calibration via the 'survey',
'sampling' or 'laeken' packages. In addition, entropy balancing
via the 'ebal' package and empirical likelihood based on codes
from Wu (2005)
<https://www150.statcan.gc.ca/n1/pub/12-001-x/2005002/article/9051-eng.pdf>
can be used. See the paper by Beręsewicz and Szymkowiak (2023)
for details <arXiv:2308.13281>. The package also includes
functions to reweight the control group to the treatment
reference distribution and to balance the covariate
distribution using the covariate balancing propensity score via
the 'CBPS' package for binary treatment observational studies.

License GPL-3

Encoding UTF-8

RdMacros mathjaxr

LazyData yes

Depends R (>= 3.5.0)

URL https://github.com/ncn-foreigners/jointCalib,
https://ncn-foreigners.github.io/jointCalib/

BugReports https://github.com/ncn-foreigners/jointCalib/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports laeken, sampling, mathjaxr, survey, MASS, ebal, CBPS

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://ncn-foreigners.r-universe.dev

RemoteUrl https://github.com/ncn-foreigners/jointcalib

RemoteRef HEAD

RemoteSha c8492019a919456268cd258d638f6e10e5e9bf54

Contents

`calib_el` *An internal function for calibration of weights using empirical likelihood method*

Description

`calib_el` performs calibration using empirical likelihood (EL) method. The function is taken from Wu (2005). If algorithm has problem with convergence `constrOptim` is used instead (as in Zhang, Han and Wu (2022)).

In (pseudo) EL the following (pseudo) EL function is maximized

$$\sum_{i \in r} d_i \log(p_i),$$

under the following constraint

$$\sum_{i \in r} p_i = 1,$$

with constraints on quantiles (with notation as in Harms and Duchesne (2006))

$$\sum_{i \in r} p_i (a_i - \alpha/N) = 0,$$

where a_i is created using `joint_calib_create_matrix` function, and possibly means

$$\sum_{i \in r} p_i (x_i - \mu_x) = 0,$$

where μ_x is known population mean of X. For simplicity of notation we assume only one quantile and one mean is known. This can be generalized to multiple quantiles and means.

Usage

```
calib_el(
  X,
  d,
  totals,
  maxit = 50,
  tol = 1e-08,
  eps = .Machine$double.eps,
  att = FALSE,
  ...
)
```

Arguments

`X`	matrix of variables for calibration of quantiles and totals (first column should be intercept),
`d`	initial d-weights for calibration (e.g. design-weights),
`totals`	vector of totals (where 1 element is the population size),
`maxit`	a numeric value giving the maximum number of iterations,
`tol`	the desired accuracy for the iterative procedure,
`eps`	the desired accuracy for computing the Moore-Penrose generalized inverse (see `MASS::ginv()`),
`att`	indicating whether the weights should sum up treatment group (for `joint_calib_att` function),
`...`	arguments passed to stats::optim via stats::constrOptim.

Value

Returns a vector of empirical likelihood g-weights

Author(s)

Maciej Beręsewicz based on Wu (2005) and Zhang, Han and Wu (2022)

References

Wu, C. (2005). Algorithms and R codes for the pseudo empirical likelihood method in survey sampling. Survey Methodology, 31(2), 239 (code is taken from `https://sas.uwaterloo.ca/~cbwu/Rcodes/LagrangeM2.txt`).

Zhang, S., Han, P., and Wu, C. (2023) Calibration Techniques Encompassing Survey Sampling, Missing Data Analysis and Causal Inference. International Statistical Review, 91: 165–192. https://doi.org/10.1111/insr.1251 (code is taken from Supplementary Materials).

Examples

```
## generate data based on Haziza and Lesage (2016)
set.seed(123)
N <- 1000
x <- runif(N, 0, 80)
y <- exp(-0.1 + 0.1*x) + rnorm(N, 0, 300)
p <- rbinom(N, 1, prob = exp(-0.2 - 0.014*x))
totals_known <- c(N=N, x=sum(x))
df <- data.frame(x, y, p)
df_resp <- df[df$p == 1, ]
df_resp$d <- N/nrow(df_resp)
res <- calib_el(X = model.matrix(~x, df_resp),
                d = df_resp$d,
                totals = totals_known)
data.frame(known = totals_known, estimated=colSums(res*df_resp$d*model.matrix(~x, df_resp)))
```

---

control_calib *control parameters*

Description

`control_calib` is function that contains control parameters for `joint_calib_create_matrix`

Usage

```
control_calib(
  interpolation = c("logit", "linear"),
  logit_const = -1000,
  sum_to_sample = FALSE,
  sum_to_one = FALSE,
  survey_sparse = FALSE,
  ebal_constraint_tolerance = 1,
  ebal_print_level = 0,
  el_att = FALSE
)
```

Arguments

`interpolation`	type of interpolation: `logit` or `linear`,
`logit_const`	constant for `logit` interpolation,
`sum_to_sample`	whether weights should sum to sample,
`sum_to_one`	whether weights should sum to one (aka normalized weights),
`survey_sparse`	whether to use sparse matrices via `Matrix` package in `survey::grake()` (currently not supported),

`ebal_constraint_tolerance`
: this is the tolerance level used by ebalance to decide if the moments in the reweighted data are equal to the target moments (see `ebal::ebalance()`),

`ebal_print_level`
: controls the level of printing: 0 (normal printing), 2 (detailed), and 3 (very detailed) (see `ebal::ebalance()`),

`el_att`
: whether weights for control should sum up to treatment size (for `calib_el` function only).

Value

a list with parameters

Author(s)

Maciej Beręsewicz

`joint_calib` *Function for the joint calibration of totals and quantiles*

Description

`joint_calib` allows joint calibration of totals and quantiles. It provides a user-friendly interface that includes the specification of variables in formula notation, a vector of population totals, a list of quantiles, and a variety of backends and methods.

Usage

```
joint_calib(
  formula_totals = NULL,
  formula_quantiles = NULL,
  data = NULL,
  dweights = NULL,
  N = NULL,
  pop_totals = NULL,
  pop_quantiles = NULL,
  subset = NULL,
  backend = c("sampling", "laeken", "survey", "ebal", "base"),
  method = c("raking", "linear", "logit", "sinh", "truncated", "el", "eb"),
  bounds = c(0, 10),
  maxit = 50,
  tol = 1e-08,
  eps = .Machine$double.eps,
  control = control_calib(),
  ...
)
```

Arguments

`formula_totals`	a formula with variables to calibrate the totals,
`formula_quantiles`	a formula with variables for quantile calibration,
`data`	a data.frame with variables,
`dweights`	initial d-weights for calibration (e.g. design weights),
`N`	population size for calibration of quantiles,
`pop_totals`	a named vector of population totals for `formula_totals`. Should be provided exactly as in `survey` package (see `survey::calibrate`),
`pop_quantiles`	a named list of population quantiles for `formula_quantiles` or an `newsvyquantile` class object (from `survey::svyquantile` function),
`subset`	a formula for subset of data,
`backend`	specify an R package to perform the calibration. Only `sampling`, `laeken`, `survey`, `ebal` or `base` are allowed,
`method`	specify method (i.e. distance function) for the calibration. Only `raking`, `linear`, `logit`, `sinh`, `truncated`, `el` (empirical likelihood), `eb` (entropy balancing) are allowed,
`bounds`	a numeric vector of length two giving bounds for the g-weights,
`maxit`	a numeric value representing the maximum number of iterations,
`tol`	the desired accuracy for the iterative procedure (for `sampling`, `laeken`, `ebal`, `el`) or tolerance in matching population total for `survey::grake` (see help for survey::grake)
`eps`	the desired accuracy for computing the Moore-Penrose generalized inverse (see `MASS::ginv()`)
`control`	a list of control parameters (currently only for `joint_calib_create_matrix`)
`...`	arguments passed either to `sampling::calib`, `laeken::calibWeights`, `survey::calibrate` or `optim::constrOptim`

Details

Imports for the function

Value

Returns a list with containing:

- `g` – g-weight that sums up to sample size,
- `Xs` – matrix used for calibration (i.e. Intercept, X and X_q transformed for calibration of quantiles),
- `totals` – a vector of totals (i.e. `N`, `pop_totals` and `pop_quantiles`),
- `method` – selected method,
- `backend` – selected backend.

Author(s)

Maciej Beręsewicz

References

Beręsewicz, M., and Szymkowiak, M. (2023). A note on joint calibration estimators for totals and quantiles Arxiv preprint https://arxiv.org/abs/2308.13281

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

Harms, T. and Duchesne, P. (2006). On calibration estimation for quantiles. Survey Methodology, 32(1), 37.

Wu, C. (2005) Algorithms and R codes for the pseudo empirical likelihood method in survey sampling, Survey Methodology, 31(2), 239.

Zhang, S., Han, P., and Wu, C. (2023) Calibration Techniques Encompassing Survey Sampling, Missing Data Analysis and Causal Inference, International Statistical Review 91, 165–192.

Haziza, D., and Lesage, É. (2016). A discussion of weighting procedures for unit nonresponse. Journal of Official Statistics, 32(1), 129-145.

See Also

`sampling::calib()` – for standard calibration.

`laeken::calibWeights()` – for standard calibration.

`survey::calibrate()` – for standard and more advanced calibration.

`ebal::ebalance()` – for standard entropy balancing.

Examples

```
## generate data based on Haziza and Lesage (2016)
set.seed(123)
N <- 1000
x <- runif(N, 0, 80)
y <- exp(-0.1 + 0.1*x) + rnorm(N, 0, 300)
p <- rbinom(N, 1, prob = exp(-0.2 - 0.014*x))
probs <- seq(0.1, 0.9, 0.1)
quants_known <- list(x=quantile(x, probs))
totals_known <- c(x=sum(x))
df <- data.frame(x, y, p)
df_resp <- df[df$p == 1, ]
df_resp$d <- N/nrow(df_resp)
y_quant_true <- quantile(y, probs)
## standard calibration for comparison
result0 <- sampling::calib(Xs = cbind(1, df_resp$x),
                           d = df_resp$d,
                           total = c(N, totals_known),
                           method = "linear")

y_quant_hat0 <- laeken::weightedQuantile(x = df_resp$y,
                                         probs = probs,
```

```
                                         weights = result0*df_resp$d)
x_quant_hat0 <- laeken::weightedQuantile(x = df_resp$x,
                                         probs = probs,
                                         weights = result0*df_resp$d)

## example 1: calibrate only quantiles (deciles)
result1 <- joint_calib(formula_quantiles = ~x,
                       data = df_resp,
                       dweights = df_resp$d,
                       N = N,
                       pop_quantiles = quants_known,
                       method = "linear",
                       backend = "sampling")
## estimate quantiles
y_quant_hat1 <- laeken::weightedQuantile(x = df_resp$y,
                                         probs = probs,
                                         weights = result1$g*df_resp$d)
x_quant_hat1 <- laeken::weightedQuantile(x = df_resp$x,
                                         probs = probs,
                                         weights = result1$g*df_resp$d)

## compare with known
data.frame(standard = y_quant_hat0, est=y_quant_hat1, true=y_quant_true)

## example 2: calibrate with quantiles (deciles) and totals
result2 <- joint_calib(formula_totals = ~x,
                       formula_quantiles = ~x,
                       data = df_resp,
                       dweights = df_resp$d,
                       N = N,
                       pop_quantiles = quants_known,
                       pop_totals = totals_known,
                       method = "linear",
                       backend = "sampling")
## estimate quantiles
y_quant_hat2 <- laeken::weightedQuantile(x = df_resp$y,
                                         probs = probs,
                                         weights = result2$g*df_resp$d)
x_quant_hat2 <- laeken::weightedQuantile(x = df_resp$x,
                                         probs = probs,
                                         weights = result2$g*df_resp$d)

## compare with known
data.frame(standard = y_quant_hat0, est1=y_quant_hat1,
           est2=y_quant_hat2, true=y_quant_true)

## example 3: calibrate wigh quantiles (deciles) and totals with
## hyperbolic sinus (sinh) and survey package

result3 <- joint_calib(formula_totals = ~x,
                       formula_quantiles = ~x,
                       data = df_resp,
                       dweights = df_resp$d,
```

```
                       N = N,
                       pop_quantiles = quants_known,
                       pop_totals = totals_known,
                       method = "sinh",
                       backend = "survey")

## estimate quantiles
y_quant_hat3 <- laeken::weightedQuantile(x = df_resp$y,
                                         probs = probs,
                                         weights = result3$g*df_resp$d)
x_quant_hat3 <- laeken::weightedQuantile(x = df_resp$x,
                                         probs = probs,
                                         weights = result3$g*df_resp$d)

## example 4: calibrate wigh quantiles (deciles) and totals with ebal package
result4 <- joint_calib(formula_totals = ~x,
                       formula_quantiles = ~x,
                       data = df_resp,
                       dweights = df_resp$d,
                       N = N,
                       pop_quantiles = quants_known,
                       pop_totals = totals_known,
                       method = "eb",
                       backend = "ebal")

## estimate quantiles
y_quant_hat4 <- laeken::weightedQuantile(x = df_resp$y,
                                         probs = probs,
                                         weights = result4$g*df_resp$d)
x_quant_hat4 <- laeken::weightedQuantile(x = df_resp$x,
                                         probs = probs,
                                         weights = result4$g*df_resp$d)

## compare with known
data.frame(standard = y_quant_hat0,
           est1=y_quant_hat1,
           est2=y_quant_hat2,
           est3=y_quant_hat3,
           est4=y_quant_hat4,
           true=y_quant_true)
## compare with known X
data.frame(standard = x_quant_hat0,
           est1=x_quant_hat1,
           est2=x_quant_hat2,
           est3=x_quant_hat3,
           est4=x_quant_hat4,
           true = quants_known$x)
```

joint_calib_att — *Function to balance the covariate distributions of a control and treatment group using* joint_calib

Description

joint_calib_att allows quantile or mean and quantile balancing of the covariate distributions of the control and treatment groups. It provides a user-friendly interface for specifying the variables and quantiles to be balanced. joint_calib_att uses joint_calib function, so the user can apply different methods to find the weights that balance the control and treatment groups. For more details see joint_calib() and Beręsewicz and Szymkowiak (2023) working paper.

Usage

```
joint_calib_att(
  formula_means = NULL,
  formula_quantiles = NULL,
  treatment = NULL,
  data,
  probs = c(0.25, 0.5, 0.75),
  ...
)
```

Arguments

formula_means	a formula with variables to be balanced at means,
formula_quantiles	a formula with variables to be balanced at quantiles,
treatment	a formula with a treatment indicator,
data	a data.frame with variables,
probs	a vector or a named list of quantiles to be balanced (default is c(0.25, 0.5, 0.75)),
...	other parameters passed to joint_calib function.

Value

Returns a list with containing:

- g – g-weight that sums up to treatment group size,
- Xs – matrix used for balancing (i.e. Intercept, X based on formula_means and X_q transformed for balancing of quantiles based on formula_quantiles and probs),
- totals – a vector of treatment reference size (N), means (pop_totals) and order of quantiles (based on formula_quantiles and probs).
- method – selected method,
- backend – selected backend.

Author(s)

Maciej Beręsewicz

References

Beręsewicz, M. and Szymkowiak, M. (2023) A note on joint calibration estimators for totals and quantiles Arxiv preprint https://arxiv.org/abs/2308.13281

Greifer N (2023). WeightIt: Weighting for Covariate Balance in Observational Studies. R package version 0.14.2, https://CRAN.R-project.org/package=WeightIt.

Greifer N (2023). cobalt: Covariate Balance Tables and Plots. R package version 4.5.1, https://CRAN.R-project.org/package=cobalt.

Ho, D., Imai, K., King, G., and Stuart, E. A. (2011). MatchIt: Nonparametric Preprocessing for Parametric Causal Inference. Journal of Statistical Software, 42(8), 1–28. https://doi.org/10.18637/jss.v042.i08

Xu, Y., and Yang, E. (2023). Hierarchically Regularized Entropy Balancing. Political Analysis, 31(3), 457-464. https://doi.org/10.1017/pan.2022.12

Examples

```
## generate data as in the hbal package
set.seed(123)
N <- 1500
X1 <- rnorm(N)
X2 <- rnorm(N)
X3 <- rbinom(N, size = 1, prob = .5)
X1X3 <- X1*X3
D_star <- 0.5*X1 + 0.3*X2 + 0.2*X1*X2 - 0.5*X1*X3 -1
D <- ifelse(D_star > rnorm(N), 1, 0)
y <- 0.5*D + X1 + X2 + X2*X3 + rnorm(N)
dat <- data.frame(D = D, X1 = X1, X2 = X2, X3 = X3, X1X3=X1X3, Y = y)
head(dat)

## Balancing means of X1, X2 and X3 and quartiles (0.25, 0.5, 0.75) of X1 and X2
## sampling::raking is used
results <- joint_calib_att(
formula_means = ~ X1 + X2 + X3,
formula_quantiles = ~ X1 + X2,
treatment = ~ D,
data = dat,
method = "raking"
)

## Results are presented with summary statistics of balance weights (g-weights)
## and information on the accuracy of reproducing reference treatment distributions
results

## An interaction between X1 and X2 is added to means
results2 <- joint_calib_att(
formula_means = ~ X1 + X2 + X3 + X1*X3,
formula_quantiles = ~ X1 + X2,
```

```
treatment = ~ D,
data = dat,
method = "raking"
)

## Results with interaction are presented below
results2

## As noted in the documentation, the probs argument can be a named list of different orders
## In this example, we specify that X1 should be balanced at the mean,
## while X2 should be balanced at Q1 and Q3
results3 <- joint_calib_att(
formula_means = ~ X1 + X2 + X3 + X1*X3,
formula_quantiles = ~ X1 + X2,
treatment = ~ D,
data = dat,
method = "raking",
probs = list(X1 = 0.5, X2 = c(0.25, 0.75))
)

## Results with different orders are presented below
results3

## Finally, we specify an order of quantile for the interaction
results4 <- joint_calib_att(
formula_means = ~ X1 + X2 + X3,
formula_quantiles = ~ X1 + X2 + X1:X3,
treatment = ~ D,
data = dat,
probs = list(X1=0.5, X2 = c(0.25, 0.5), `X1:X3` = 0.75),
method = "raking"
)

## Results with Q3 balancing for interaction are presented below
results4
```

`joint_calib_cbps` *Function to balance the covariate distributions using covariate balancing propensity score* CBPS

Description

`joint_calib_cbps` allows quantile or mean and quantile balancing of the covariate distributions of the control and treatment groups using the covariate balancing propensity score method (Imai & Ratkovic (2014)). `CBPS::CBPS()` and `CBPS::hdCBPS()` are used a backend for estimating the parameters. This function works in a similar way to the `joint_calib_att()` function, i.e. the user can specify variables for the balancing means as well as the quantiles.

Usage

```
joint_calib_cbps(
  formula_means = NULL,
  formula_quantiles = NULL,
  treatment = NULL,
  data,
  probs = c(0.25, 0.5, 0.75),
  control = control_calib(),
  standardize = FALSE,
  method = "exact",
  variable_selection = FALSE,
  target = NULL,
  ...
)
```

Arguments

`formula_means`	a formula with variables to be balanced at means,
`formula_quantiles`	a formula with variables to be balanced at quantiles,
`treatment`	a formula with a treatment indicator,
`data`	a data.frame with variables,
`probs`	a vector or a named list of quantiles to be balanced (default is `c(0.25, 0.5, 0.75)`),
`control`	a control list of parameters for creation of X_q matrix based on `formula_quantiles` and `probs` (see `joint_calib_create_matrix()`),
`standardize`	default is FALSE, which normalizes weights to sum to 1 within each treatment group (passed to `CBPS()` function),
`method`	default is "exact". Choose "over" to fit an over-identified model that combines the propensity score and covariate balancing conditions; choose "exact" to fit a model that only contains the covariate balancing conditions (passed to `CBPS()` function)
`variable_selection`	default is FALSE. Set to TRUE to select high dimension CBPS via `CBPS::hdCBPS()`,
`target`	specify target (y) variable for `hdCBPS` function,
`...`	other parameters passed to `CBPS` or `hdCBPS` functions.

Details

Imports for the function

Value

Returns a `CBPS` or a `list` object as a result of the `hdCBPS` function.

Author(s)

Maciej Beręsewicz

References

Imai, K., and Ratkovic, M. (2014). Covariate balancing propensity score. Journal of the Royal Statistical Society Series B: Statistical Methodology, 76(1), 243-263.

Fong C, Ratkovic M, and Imai K (2022). CBPS: Covariate Balancing Propensity Score. R package version 0.23, https://CRAN.R-project.org/package=CBPS.

Examples

```
## generate data as in the hbal package (see [hbal::hbal()])
set.seed(123)
N <- 1500
X1 <- rnorm(N)
X2 <- rnorm(N)
X3 <- rbinom(N, size = 1, prob = .5)
X1X3 <- X1*X3
D_star <- 0.5*X1 + 0.3*X2 + 0.2*X1*X2 - 0.5*X1*X3 - 1
D <- ifelse(D_star > rnorm(N), 1, 0) # Treatment indicator
y <- 0.5*D + X1 + X2 + X2*X3 + rnorm(N) # Outcome
dat <- data.frame(D = D, X1 = X1, X2 = X2, X3 = X3, X1X3 = X1X3, Y = y)
head(dat)

## Balancing means of X1, X2 and X3 and quartiles (0.25, 0.5, 0.75) of X1 and X2.
result <- joint_calib_cbps(formula_means = ~ X1 + X2 + X3,
                           formula_quantiles = ~ X1 + X2,
                           treatment = ~ D,
                           data = dat)

## CBPS output is presented
result

## calculate ATE by hand
w_1 <- dat$D/fitted(result)
w_1 <- w_1/mean(w_1)
w_0 <- (1-dat$D)/(1-fitted(result))
w_0 <- w_0/mean(w_0)
mean((w_1-w_0)*dat$Y)

## Compare with standard CBPS using only means
result2 <- CBPS::CBPS(D ~ X1 + X2 + X3, data = dat, method = "exact", standardize = FALSE, ATT = 0)

## calculate ATE by hand
w_1a <- dat$D/fitted(result2)
w_1a <- w_1a/mean(w_1a)
w_0a <- (1-dat$D)/(1-fitted(result2))
w_0a <- w_0a/mean(w_0a)
mean((w_1a-w_0a)*dat$Y)
```

joint_calib_create_matrix

An internal function to create an A matrix for calibration of quantiles

Description

`joint_calib_create_matrix` is function that creates an $A = [a_{ij}]$ matrix for calibration of quantiles. Function allows to create matrix using `logistic` interpolation (using `stats::plogis`, default) or `linear` (as in Harms and Duchesne (2006), i.e. slightly modified Heavyside function).

In case of `logistic` interpolation elements of A are created as follows

$$a_{ij} = \frac{1}{(1 + \exp\left(-2l\left(x_{ij} - Q_{x_j,\alpha}\right)\right))N},$$

where x_{ij} is the ith row of the auxiliary variable X_j, N is the population size, $Q_{x_j,\alpha}$ is the known population αth quantile, and l is set to -1000 (by default).

In case of `linear` interpolation elements of A are created as follows

$$a_{ij} = \begin{cases} N^{-1}, & x_{ij} \leqslant L_{x_j,r}\left(Q_{x_j,\alpha}\right), \\ N^{-1}\beta_{x_j,r}\left(Q_{x_j,\alpha}\right), & x_{ij} = U_{x_j,r}\left(Q_{x_j,\alpha}\right), \\ 0, & x_{ij} > U_{x_j,r}\left(Q_{x_j,\alpha}\right), \end{cases}$$

$i = 1, ..., r$, $j = 1, ..., k$, where r is the set of respondents, k is the auxiliary variable index and

$$L_{x_j,r}(t) = \max\left\{\{x_{ij}, i \in s \mid x_{ij} \leqslant t\} \cup \{-\infty\}\right\},$$
$$U_{x_j,r}(t) = \min\left\{\{x_{ij}, i \in s \mid x_{ij} > t\} \cup \{\infty\}\right\},$$
$$\beta_{x_j,r}(t) = \frac{t - L_{x_j,s}(t)}{U_{x_j,s}(t) - L_{x_j,s}(t)},$$

$i = 1, ..., r$, $j = 1, ..., k$, $t \in \mathbb{R}$.

Usage

```
joint_calib_create_matrix(X_q, N, pop_quantiles, control = control_calib())
```

Arguments

`X_q`	matrix of variables for calibration of quantiles,
`N`	population size for calibration of quantiles,
`pop_quantiles`	a vector of population quantiles for `X_q`,
`control`	a control parameter for creation of `X_q` matrix.

Value

Return matrix A

Author(s)

Maciej Beręsewicz

References

Harms, T. and Duchesne, P. (2006). On calibration estimation for quantiles. Survey Methodology, 32(1), 37.

Examples

```
# Create matrix for one variable and 3 quantiles
set.seed(123)
N <- 1000
x <- as.matrix(rnorm(N))
quants <- list(quantile(x, c(0.25,0.5,0.75)))
A <- joint_calib_create_matrix(x, N, quants)
head(A)
colSums(A)

# Create matrix with linear interpolation
A <- joint_calib_create_matrix(x, N, quants, control_calib(interpolation="linear"))
head(A)
colSums(A)

# Create matrix for two variables and different number of quantiles

set.seed(123)
x1 <- rnorm(N)
x2 <- rchisq(N, 1)
x <- cbind(x1, x2)
quants <- list(quantile(x1, 0.5), quantile(x2, c(0.1, 0.75, 0.9)))
B <- joint_calib_create_matrix(x, N, quants)
head(B)
colSums(B)
```

Index