Package ‘ddecompose’

May 7, 2024

Type Package

Title Detailed Distributional Decomposition

Version 1.0.0

Maintainer Samuel Meier <samuel.meier+ddecompose@immerda.ch>

Description Implements the Oaxaca-Blinder decomposition method and generalizations of it that decompose differences in distributional statistics beyond the mean.
The function ob_decompose() decomposes differences in the mean outcome between two groups into one part explained by different covariates (composition effect) and into another part due to differences in the way covariates are linked to the outcome variable (structure effect). The function further divides the two effects into the contribution of each covariate and allows for weighted doubly robust decompositions. For distributional statistics beyond the mean, the function performs the recentered influence function (RIF) decomposition proposed by Firpo, Fortin, and Lemieux (2018).
The function dfl_decompose() divides differences in distributional statistics into an composition effect and a structure effect using inverse probability weighting as introduced by DiNardo, Fortin, and Lemieux (1996). The function also allows to sequentially decompose the composition effect into the contribution of single covariates.
References:
Firpo, Sergio, Nicole M. Fortin, and Thomas Lemieux. (2018) <doi:10.3390/econometrics6020028>. ``Decomposing Wage Distributions Using Recentered Influence Function Regressions.''
Fortin, Nicole M., Thomas Lemieux, and Sergio Firpo. (2011) <doi:10.3386/w16045>. ``Decomposition Methods in Economics.''
DiNardo, John, Nicole M. Fortin, and Thomas Lemieux. (1996) <doi:10.2307/2171954>. ``Labor Market Institutions and the Distribution of Wages, 1973-1992: A Semiparametric Approach.''
Oaxaca, Ronald. (1973) <doi:10.2307/2525981>. ``Male-Female Wage Differentials in Urban Labor Markets.''
Blinder, Alan S. (1973) <doi:10.2307/144855>. ``Wage Discrimination: Reduced Form and Structural Estimates.''

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends ggplot2, R (>= 2.10)

Imports rifreg, Formula, Hmisc, parallel, pbapply, sandwich, stats, ranger, fastglm, methods

Suggests testthat (>= 3.0.0), tidyr, dplyr

Config/testthat/edition 3

NeedsCompilation no

Author David Gallusser [aut], Samuel Meier [aut, cre]

Repository CRAN

Date/Publication 2024-05-07 15:00:02 UTC

R topics documented:

aggregate_terms *Aggregate decomposition terms*

Description

The function aggregates decomposition terms and calculates their covariance matrix based on detailed decomposition results.

Usage

```
aggregate_terms(
  x,
  aggregate_factors = TRUE,
  custom_aggregation = NULL,
  reweighting
)
```

Arguments

`x` an object of class "ob_decompose", usually , a result of a call to [ob_decompose()].

`aggregate_factors` boolean, if ‘TRUE‘ (default) terms associated with detailed factor levels are aggregated to a single term for every factor variable.

`custom_aggregation` list specifying the aggregation of detailed decomposition terms. The parameter ‘custom_aggregation‘ overrides the parameter ‘aggregate_factors‘. If ‘NULL‘ (default), then either all detailed terms or all terms associated with a single variable are returned.

`reweighting` boolean, if ‘TRUE‘ the decompostion in ‘object‘ contains reweighting (i.e. specification and reweighting error)

Value

The function returns an updated object of class "ob_decompose" containing the aggregated decomposition terms.

bootstrap_estimate_ob_decompose *Bootstrapping the OB decomposition*

Description

The function resamples observations and restimates the OB decomposition with the new sample.

Usage

```
bootstrap_estimate_ob_decompose(
  formula_decomposition,
  formula_reweighting,
  data_used,
  group,
  reference_0,
  normalize_factors,
  reweighting,
  reweighting_method,
  trimming,
  trimming_threshold,
  rifreg,
  rifreg_statistic,
  rifreg_probs,
  custom_rif_function,
  na.action,
  cluster = NULL,
  ...
)
```

Arguments

`formula_decomposition`
: `formula` object that contains the formula for the decomposition

`formula_reweighting`
: `formula` object that contains the formula for the reweighting in case of a reweighted decompostion

`data_used`
: `data.frame` with data used for estimation (including weight and group variable)

`group`
: name of the a binary variable (numeric or factor) identifying the two groups that will be compared. The group identified by the lower ranked value in ‘group‘ (i.e., 0 in the case of a dummy variable or the first level of factor variable) is defined as group 0.

`reference_0`
: boolean: indicating if group 0 is the reference group and if its coefficients are used to compute the counterfactual mean.

`normalize_factors`
: boolean: If ‘TRUE‘, then factor variables are normalized as proposed by Gardeazabal/Ugidos (2004)

`reweighting`
: boolean: if ‘TRUE‘, then the decomposition is performed with with respect to reweighted reference group.

`reweighting_method`
: specifies the method fit and predict conditional probabilities used to derive the reweighting factor. Currently, `"logit"`, `"fastglm"`, and `"random_forest"` are available.

`trimming`
: boolean: If `TRUE`, observations with dominant reweighting factor values are trimmend according to rule of Huber, Lechner, and Wunsch (2013). Per default, trimming is set to `FALSE`.

trimming_threshold
: numeric: threshold defining the maximal accepted relative weight of the reweighting factor value (i.e., inverse probability weight) of a single observation. If NULL, the threshold is set to $sqrt(N)/N$, where N is the number of observations in the reference group.

rifreg
: boolean: if 'TRUE', then RIF decomposition is performed

rifreg_statistic
: string containing the distributional statistic for which to compute the RIF.

rifreg_probs
: a vector of length 1 or more with probabilities of quantiles.

custom_rif_function
: the RIF function to compute the RIF of the custom distributional statistic.

na.action
: generic function that defines how NAs in the data should be handled.

cluster
: numeric vector of same length as dep_var indicating the clustering of observations. If cluster = NULL (default), no clustering is a assumend and bootstrap procedure resamples individual observations. Otherwise bootstrap procedure resamples clusters.

...
: additional parameters passed to custom_rif_function

---

dfl_decompose *DFL reweighting decomposition*

Description

dfl_decompose divides between-group differences in distributional statistics of an outcome variable into a structure effect and a composition effect. Following DiNardo, Fortin, and Lemieux (1996), the procedure reweights the sample distribution of a reference group such that the group's covariates distribution matches the covariates distribution of a comparison group.

The function derives counterfactual distributions with inverse probability weigthing. Reweighting factors are estimate by modelling the probability of belonging to the comparison group conditional on covariates.

The function allows detailed decompositions of the composition effect by sequentially reweighting (conditional) covariate distributions. Standard errors can be bootstrapped.

Usage

```
dfl_decompose(
  formula,
  data,
  weights,
  group,
  na.action = na.exclude,
  reference_0 = TRUE,
  subtract_1_from_0 = FALSE,
  right_to_left = TRUE,
```

```
  method = "logit",
  estimate_statistics = TRUE,
  statistics = c("quantiles", "mean", "variance", "gini", "iq_range_p90_p10",
    "iq_range_p90_p50", "iq_range_p50_p10"),
  probs = c(1:9)/10,
  custom_statistic_function = NULL,
  trimming = FALSE,
  trimming_threshold = NULL,
  return_model = TRUE,
  estimate_normalized_difference = TRUE,
  bootstrap = FALSE,
  bootstrap_iterations = 100,
  bootstrap_robust = FALSE,
  cores = 1,
  ...
)
```

Arguments

`formula` a `formula` object with an outcome variable Y on the left-hand side and the covariates X on the right-hand side. For sequential decompositions, the sequence of covariates X are distinguished by the `|` operator. Covariates are used to estimate the conditional probabilities for the reweighting factors.

`data` a `data.frame` containing all variables and observations of both groups.

`weights` name of the observation weights variable or vector of observation weights.

`group` name of a binary variable (numeric or factor) identifying the two groups for which the differences are to be decomposed. The group identified by the lower ranked value in `group` (i.e., 0 in the case of a dummy variable or the first level of factor variable) is defined as group 0. Per default, group 0 is the reference group (see `reference_0`).

`na.action` a function to filter missing data (default `na.exclude`).

`reference_0` boolean: if `TRUE` (default), then the group 0 – i.e., the group identified by the lower ranked value in `group` – will be defined as reference group. The reference group will be reweighted to match the covariates distribution of the sample of the comparison group.

`subtract_1_from_0` boolean: By default (‘FALSE‘), the distributional statistic of group 0 is subtracted from the one of group 1 to compute the overall difference. Setting ‘subtract_1_from_0‘ to ‘TRUE‘ merely changes the sign of the decomposition results.

`right_to_left` determines the direction of a sequential decomposition. If `TRUE` (default), the sequential decomposition starts right and reweights first the reference group using only the variables entered last into the `formula` sequence. Sequentially, the other variables are added. Otherwise, the decomposition start left and using all variables entered into `formula` object from the start, sequentially removing variables.

method
: specifies the method to fit and predict conditional probabilities used to derive the reweighting factor. At the moment, "logit", "fastglm", and "random_forest" are available.

estimate_statistics
: boolean: if TRUE (default), then distributional statistics are estimated and the decomposition is performed. If FALSE, the function only returns the fitted inverse propensity weights.

statistics
: a character vector that defines the distributional statistics for which the decomposition is performed. Per default, c("quantiles", "mean", "variance", "gini", "iq_range_p90_p10", "iq_range_p90_p50", "iq_range_p50_p10") are estimated and decomposed. Also implemented are ‘c("iq_ratio_p90_p10", "iq_ratio_p90_p50", "iq_ratio_p50_p10")‘. Note: The function calculates the Gini coefficient for the untransformed variable (i.e., exp(log(Y))), if the logarithm of a variable Y is set as outcome variable in formula).

probs
: a vector of length 1 or more with the probabilities of the quantiles to be estimated with default c(1:9)/10.

custom_statistic_function
: a function estimating a custom distributional statistic that will be decomposed (NULL by default). Every custom_statistic_function needs the parameters dep_var (vector of the outcome variable) and weights (vector with observation weights); additional arguments are not allowed or need to be 'hardcoded'. See examples for further details.

trimming
: boolean: If TRUE, observations with dominant reweighting factor values are trimmed according to rule of Huber, Lechner, and Wunsch (2013). Per default, trimming is set to FALSE.

trimming_threshold
: numeric: threshold defining the maximal accepted relative weight of the reweighting factor value (i.e., inverse probability weight) of a single observation. If NULL, the threshold is set to $sqrt(N)/N$, where N is the number of observations in the reference group.

return_model
: boolean: If TRUE (default), the object(s) of the model fit(s) used to predict the conditional probabilities for the reweighting factor(s) are returned.

estimate_normalized_difference
: boolean: If TRUE (default), the normalized differences between the covariate means of the comparison group and the reweighted reference group are calculated.

bootstrap
: boolean: If FALSE (default), then the estimation is not boostrapped and no standard errors are calculated.

bootstrap_iterations
: positive integer with default 100 indicating the number of bootstrap iterations to be executed.

bootstrap_robust
: boolean: if FALSE (default), then bootstrapped standard errores are estimated as the standard deviations of the bootstrapp estimates. Otherwise, the function uses the bootstrap interquartile range rescaled by the interquantile range of the standard distribution to estimate standard errors.

`cores` positive integer with default `1` indicating the number of cores to use when computing bootstrap standard errors.

`...` other parameters passed to the function estimating the conditional probabilities.

Details

The observed difference to be decomposed equals the difference between the values of the distributional statistic of `group` 1 and `group` 0, respectively:

$$\Delta_O = \nu_1 - \nu_0,$$

where $\nu_t = \nu(F_g)$ denotes the statistics of the outcome distribution F_g of group g. Group 0 is identified by the lower ranked value of the `group` variable.

If `reference_0=TRUE`, then group 0 is the reference group and its observations are reweighted such that they match the covariates distribution of group 1, the comparison group. The counterfactual combines the covariates distribution $F_1(x)$ of group 1 with the conditional outcome distribution $F_0(y|x)$ of group 0 and is derived by reweighting group 0

$$F_C(y) = \int F_0(y|x)dF_1(x) = \int F_0(y|x)\Psi(x)dF_0(x),$$

where $\Psi(x)$ is the reweighting factor, i.e., the inverse probabilities of belonging to the comparison group conditional on covariates x.

The distributional statistic of the counterfactual distribution, $\nu_C = \nu(F_C)$, allows to decompose the observed difference into a (wage) structure effect ($\Delta_S = \nu_1 - \nu_C$) and a composition effect ($\Delta_C = \nu_C - \nu_0$).

If `reference_0=FALSE`, then the counterfactual is derived by combining the covariates distribution of group 0 with the conditional outcome distribution of group 1 and, thus, reweighting group 1

$$F_C(y) = \int F_1(y|x)dF_0(x) = \int F_1(y|x)\Psi(x)dF_1(x).$$

The composition effect becomes $\Delta_C = \nu_1 - \nu_C$ and the structure effect $\Delta_S = \nu_C - \nu_0$, respectively.

The covariates are defined in `formula`. The reweighting factor is estimated in the pooled sample with observations from both groups. `method = "logit"` uses a logit model to fit the conditional probabilities. `method = "fastglm"` also fits a logit model but with a faster algorithm from **fastglm**. `method = "random_forest"` uses the **Ranger** implementation of the random forests classifier.

The counterfactual statistics are then estimated with the observed data of the reference group and the fitted reweighting factors.

`formula` allows to specify interaction terms in the conditional probability models. If you are interested in an aggregate decomposition, then all covariates have to be entered at once, e.g., `Y ~ X + Z`.

The procedure allows for sequential decomposition of the composition effect. In this case, more than one reweighting factor based on different sets of covariates are estimated.

If you are interested in a sequential decomposition, the decomposition sequence has to be distinguished by the `|` operator in the `formula` object. For instance, `Y ~ X | Z` would decompose the aggregate composition effect into the contribution of covariate(s) X and the one of covariate(s) Z, respectively.

In this two-fold sequential decomposition, we have the detailed composition effects

$$\Delta_{C_X} = \nu_1 - \nu_{CX},$$

and

$$\Delta_{C_Z} = \nu_{CX} - \nu_C,$$

which sum up to the aggregate composition effect Δ_C. ν_C is defined as above. It captures the contribution of all covariates (i.e., X and Z). In contrast, ν_{CX} corresponds to the statistic of the counterfactual distribution isolating the contribution of covariate(s) X in contrast to the one of covariate(s) Z.

If `right_to_left=TRUE`, then the counterfactual is defined as

$$F_{CX}(y) = \int\int F_0(y|x,z)dF_0(x|z)dF_1(z),$$

where $F_1(x|z)$ is the conditional distribution of X given Z of group 1 and $F_0(z)$ the distribution of Z. If `right_to_left=FALSE`, we have

$$F_{CX}(y) = \int\int F_0(y|x,z)dF_1(x|z)dF_0(z).$$

Note that it is possible to specify the detailed models in every part of `formula`. This is useful if you want to estimate in every step a fully saturated model, e.g., `Y ~ X * Z | Z`. If not further specified, the variables are additively included in the model used to derived the aggregate reweighting factor.

The detailed decomposition terms are path-dependent. The results depend on the sequence the covariates enter the decomposition (e.g, `Y ~ X | Z` yields different detailed decomposition terms than `Y ~ Z | X`) . Even for the same sequence, the results differ depending on the 'direction' of the decomposition. In the example above using `right_to_left=TRUE`, the contribution of Z is evaluated using the conditional distribution of X given Z from group 0. If we use `right_to_left=FALSE` instead, the same contribution is evaluated using the conditional distribution from group 1.

Per default, the distributional statistics for which the between group differences are decomposed are quantiles, the mean, the variance, the Gini coefficient and the interquantile range between the 9th and the 1st decile, the 9th decile and the median, and the median and the first decile, respectively. The interquantile ratios between the same quantiles are implemented, as well.

The quantiles can be specified by `probs` that sets the corresponding probabilities of the quantiles of interest. For other distributional statistics, please use `custom_statistic_function`

The function bootstraps standard errors and derives a bootstrapped Kolmogorov-Smirnov distribution to construct uniform confindence bands. The Kolmogorov-Smirnov distribution is estimated as in Chen et al. (2017).

Value

an object of class `dfl_decompose` containing a data.frame with the decomposition results for the quantiles and for the other distributional statistics, respectively, a data.frame with the estimated reweighting factor for every observation, a data.frame with sample quantiles of the reweighting factors and a list with standard errors for the decomposition terms, the quantiles of the reweighting factor, the bootstrapped Kolmogorov-Smirnov distribution to construct uniform confidence bands for quantiles, as well as a list with the normalized differences between the covariate means of the comparison group and the reweighted reference group.

References

Chen, Mingli, Victor Chernozhukov, Iván Fernández-Val, and Blaise Melly. 2017. "Counterfactual: An R Package for Counterfactual Analysis." *The R Journal* 9(1): 370-384.

DiNardo, John, Nicole M. Fortin, and Thomas Lemieux. 1996. "Labor Market Institutions and the Distribution of Wages, 1973-1992: A Semiparametric Approach." *Econometrica*, 64(5), 1001-1044.

Firpo, Sergio P., Nicole M. Fortin, and Thomas Lemieux. 2018. "Decomposing Wage Distributions Using Recentered Influence Function Regressions." *Econometrics* 6(2), 28.

Fortin, Nicole M., Thomas Lemieux, and Sergio Firpo. 2011. "Decomposition methods in economics." In Orley Ashenfelter and David Card, eds., *Handbook of Labor Economics*. Vol. 4. Elsevier, 1-102.

Firpo, Sergio P., and Cristine Pinto. 2016. "Identification and Estimation of Distributional Impacts of Interventions Using Changes in Inequality Measures." *Journal of Applied Econometrics*, 31(3), 457-486.

Huber, Martin, Michael Lechner, and Conny Wunsch. 2013. "The performance of estimators based on the propensity score." *Journal of Econometrics*, 175(1), 1-21.

Examples

```
## Example from handbook chapter of Fortin, Lemieux, and Firpo (2011: 67)
## with a sample of the original data


data("men8305")

flf_model <- log(wage) ~ union * (education + experience) + education * experience

# Reweighting sample from 1983-85
flf_male_inequality <- dfl_decompose(flf_model,
  data = men8305,
  weights = weights,
  group = year
)

# Summarize results
summary(flf_male_inequality)

# Plot decomposition of quantile differences
plot(flf_male_inequality)

# Use alternative reference group (i.e., reweight sample from 2003-05)
flf_male_inequality_reference_0305 <- dfl_decompose(flf_model,
  data = men8305,
  weights = weights,
```

```
  group = year,
  reference_0 = FALSE
)
summary(flf_male_inequality_reference_0305)

# Bootstrap standard errors (using smaller sample for the sake of illustration)

set.seed(123)
flf_male_inequality_boot <- dfl_decompose(flf_model,
  data = men8305[1:1000, ],
  weights = weights,
  group = year,
  bootstrap = TRUE,
  bootstrap_iterations = 100,
  cores = 1
)

# Get standard errors and confidence intervals
summary(flf_male_inequality_boot)

# Plot quantile differences with pointwise confidence intervals
plot(flf_male_inequality_boot)

# Plot quantile differences with uniform confidence intervals
plot(flf_male_inequality_boot, uniform_bands = TRUE)



## Sequential decomposition

# Here we distinguish the contribution of education and experience
# from the contribution of unionization conditional on education and experience.


model_sequential <- log(wage) ~ union * (education + experience) +
  education * experience |
  education * experience

# First variant:
# Contribution of union is evaluated using composition of
# education and experience from 2003-2005 (group 1)

male_inequality_sequential <- dfl_decompose(model_sequential,
  data = men8305,
  weights = weights,
  group = year
)

# Summarize results
summary(male_inequality_sequential)

# Second variant:
# Contribution of union is evaluated using composition of
```

```
# education and experience from 1983-1985 (group 0)

male_inequality_sequential_2 <- dfl_decompose(model_sequential,
  data = men8305,
  weights = weights,
  group = year,
  right_to_left = FALSE
)

# Summarize results
summary(male_inequality_sequential_2)

# The domposition effects associated with (conditional) unionization for deciles
cbind(
  male_inequality_sequential$decomposition_quantiles$prob,
  male_inequality_sequential$decomposition_quantiles$`Comp. eff. X1|X2`,
  male_inequality_sequential_2$decomposition_quantiles$`Comp. eff. X1|X2`
)


## Trim observations with weak common support
## (i.e. observations with relative factor weights > \sqrt(N)/N)

set.seed(123)
data_weak_common_support <- data.frame(
  d = factor(c(
    c("A", "A", rep("B", 98)),
    c(rep("A", 90), rep("B", 10))
  )),
  group = rep(c(0, 1), each = 100)
)
data_weak_common_support$y <- ifelse(data_weak_common_support$d == "A", 1, 2) +
  data_weak_common_support$group +
  rnorm(200, 0, 0.5)

decompose_results_trimmed <- dfl_decompose(y ~ d,
  data_weak_common_support,
  group = group,
  trimming = TRUE
)

identical(
  decompose_results_trimmed$trimmed_observations,
  which(data_weak_common_support$d == "A")
)



## Pass a custom statistic function to decompose income share of top 10%

top_share <- function(dep_var,
                      weights,
                      top_percent = 0.1) {
```

```
  threshold <- Hmisc::wtd.quantile(dep_var, weights = weights, probs = 1 - top_percent)
  share <- sum(weights[which(dep_var > threshold)] *
    dep_var[which(dep_var > threshold)]) /
    sum(weights * dep_var)
  return(share)
}

flf_male_inequality_custom_stat <- dfl_decompose(flf_model,
  data = men8305,
  weights = weights,
  group = year,
  custom_statistic_function = top_share
)
summary(flf_male_inequality_custom_stat)
```

`dfl_decompose_bootstrap` *Bootstrapping the DFL reweighting decomposition*

Description

The function resamples observations and restimates the DFL decomposition with the new sample.

Usage

```
dfl_decompose_bootstrap(
  formula,
  dep_var,
  data_used,
  weights,
  group_variable,
  reference_group,
  estimate_statistics,
  statistics,
  probs,
  custom_statistic_function,
  right_to_left,
  trimming,
  trimming_threshold,
  ...
)
```

Arguments

`formula`	`formula` object
`dep_var`	dependent variable
`data_used`	`data.frame` with data used for estimation

`weights` weights variable

`group_variable` group variable

`reference_group` reference_group to be reweighted

`estimate_statistics` boolean: if `TRUE` (default), then distributional statistics are estimated and the decomposition is performed. If `FALSE`, the function only returns the fitted inverse propensity weights.

`statistics` a character vector that defines the distributional statistics for which the decomposition is performed.

`probs` a vector of length 1 or more with the probabilities of the quantiles to be estimated.

`custom_statistic_function` a function estimating a custom distributional statistic that will be decomposed.

`right_to_left` determines the direction of a sequential decomposition.

`trimming` boolean: If `TRUE`, observations with dominant reweighting factor values are trimmed according to rule of Huber, Lechner, and Wunsch (2013).

`trimming_threshold` numeric: threshold defining the maximal accepted relative weight of the reweighting factor value (i.e., inverse probability weight) of a single observation. If `NULL`, the threshold is set to $sqrt(N)/N$, where N is the number of observations in the reference group.

`...` other parameters passed to the function estimating the conditional probabilities.

`dfl_decompose_estimate` *Estimate the DFL reweighting decomposition*

Description

This function performs the DFL decomposition. It derives the reweighting factors, estimates the distributional statistics and calculates the decomposition terms.

Usage

```
dfl_decompose_estimate(
  formula,
  dep_var,
  data_used,
  weights,
  group_variable,
  reference_group,
  method,
  estimate_statistics,
```

```
    statistics,
    probs,
    custom_statistic_function,
    right_to_left,
    trimming,
    trimming_threshold,
    return_model,
    estimate_normalized_difference,
    ...
)
```

Arguments

- `formula` — `formula` object
- `dep_var` — dependent variable
- `data_used` — `data.frame` with data used for estimation
- `weights` — weights variable
- `group_variable` — group variable
- `reference_group` — reference_group to be reweighted
- `method` — method used to estimate conditional probabilities
- `estimate_statistics` — boolean: if `TRUE` (default), then distributional statistics are estimated and the decomposition is performed. If `FALSE`, the function only returns the fitted inverse propensity weights.
- `statistics` — a character vector that defines the distributional statistics for which the decomposition is performed.
- `probs` — a vector of length 1 or more with the probabilities of the quantiles to be estimated.
- `custom_statistic_function` — a function estimating a custom distributional statistic that will be decomposed.
- `right_to_left` — determines the direction of a sequential decomposition.
- `trimming` — boolean: If `TRUE`, observations with dominant reweighting factor values are trimmed according to rule of Huber, Lechner, and Wunsch (2013).
- `trimming_threshold` — numeric: threshold defining the maximal accepted relative weight of the reweighting factor value (i.e., inverse probability weight) of a single observation. If `NULL`, the threshold is set to $sqrt(N)/N$, where N is the number of observations in the reference group.
- `return_model` — boolean: If `TRUE` (default), the object(s) of the model fit(s) used to predict the conditional probabilities for the reweighting factor(s) are returned.
- `estimate_normalized_difference` — boolean: If `TRUE` (default), the normalized differences between the covariate means of the comparison group and the reweighted reference group are calculated.
- `...` — other parameters passed to the function estimating the conditional probabilities.

`estimate_iq_range` *Interquantile range*

Description

Interquantile range

Usage

```
estimate_iq_range(dep_var, weights, probs = c(0.1, 0.9))
```

Arguments

`dep_var`	numeric vector of outcome variable
`weights`	numeric vector of weights
`probs`	a vector with probabilities whose range defines the interquantile range

Value

a numeric value indicating the (weighted) interquantile range

`estimate_iq_ratio` *Interquantile ratio*

Description

Interquantile ratio

Usage

```
estimate_iq_ratio(dep_var, weights, probs = c(0.1, 0.9))
```

Arguments

`dep_var`	numeric vector of outcome variable
`weights`	numeric vector of weights
`probs`	a vector with probabilities whose range defines the interquantile range

Value

a numeric value indicating the (weighted) interquantile ratio

estimate_ob_decompose *Estimate OB decomposition*

Description

The function performs the linear Oaxaca-Blinder decomposition.

Usage

```
estimate_ob_decompose(
  formula,
  data_used,
  reference_0,
  normalize_factors,
  compute_analytical_se,
  return_model_fit,
  reweighting,
  rifreg,
  rifreg_statistic,
  rifreg_probs,
  custom_rif_function,
  na.action,
  vcov,
  ...
)
```

Arguments

`formula` — `formula` object

`data_used` — `data.frame` with data used for estimation (including weight and group variable)

`reference_0` — boolean: indicating if group 0 is the reference group and if its coefficients are used to compute the counterfactual mean.

`normalize_factors` — boolean: If ‘TRUE‘, then factor variables are normalized as proposed by Gardeazabal/Ugidos (2004)

`compute_analytical_se` — boolean: If ‘TRUE‘, then analytical standard errors for decomposition terms are calculated (assuming independence between groups).

`return_model_fit` — boolean: If ‘TRUE‘, then model objects are returned.

`reweighting` — boolean: if ‘TRUE‘, then the decomposition is performed with with respect to reweighted reference group.

`rifreg` — boolean: if ‘TRUE‘, then RIF decomposition is performed

`rifreg_statistic` — string containing the distributional statistic for which to compute the RIF.

rifreg_probs	a vector of length 1 or more with probabilities of quantiles.
custom_rif_function	the RIF function to compute the RIF of the custom distributional statistic.
na.action	generic function that defines how NAs in the data should be handled.
vcov	unction estimating covariance matrix of regression coefficients if compute_analytical_se == TRUE
...	additional parameters passed to custom_rif_function

---

fit_and_predict_probabilities

Predict conditional probabilities

Description

This function fits a binary choice model and predicts probabilities for every observations.

Usage

```
fit_and_predict_probabilities(
  formula,
  data_used,
  weights,
  method = "logit",
  return_model = FALSE,
  newdata = NULL,
  ...
)
```

Arguments

formula	formula object specifying the conditional probability model.
data_used	data.frame with data.
weights	weights variable
method	method to estimate conditional probabilities
return_model	boolean: If FALSE (default), the object of the model fit used to predict the conditional probabilities for the reweighting factor are not returned.
newdata	data.frame with data to be used for predictions.
...	other parameters passed to the estimation function.

get_distributional_statistics

Estimate distributional statistics

Description

Estimate weighted distributional statistics for the reference or the counterfactual group.

Usage

```
get_distributional_statistics(
  dep_var,
  weights,
  group_variable,
  group,
  statistics,
  custom_statistic_function = NULL,
  probs = 1:9/10,
  log_transformed
)
```

Arguments

`dep_var`	vector of outcome variable
`weights`	vector of observations weights
`group_variable`	vector of group assignment
`group`	identifier of group for which distributional statistics are calculated
`statistics`	vector of statistics to be calculated
`custom_statistic_function`	a custom statistic function to be evaluated
`probs`	probabilities of quantiles to be calculated
`log_transformed`	indicator if outcome variable is log transformed

get_normalized_difference

Get normalized differences

Description

The function calculates normalized differences between covariate means of comparison group and reweighted reference group.

Usage

```
get_normalized_difference(
  formula,
  data_used,
  weights,
  psi,
  group_variable,
  reference_group
)
```

Arguments

formula	model formula used to calulate the conditional probabilities of the reweighting factor
data_used	`data.frame` with the observation for the estimation of the reweighting factor
weights	vector with observations weights
psi	vector with the estimated reweighting factor
group_variable	variable with group identifier
reference_group	identifier of (reweighted) reference group

References

Imbens, Guido W. and Jeffrey M. Wooldridge. 2009. Recent developments in the econometrics of program evaluation. Journal of Economic Literature 47, no. 1: 5-86.

GU_normalization — *Gardeazabal and Ugidos normalization of factor variables*

Description

The function performs the normalization of the factor variables proposed by Gardeazabal and Ugidos (2004, GU) to estimate detailed decompositions that do not depend on the chosen reference levels of the factor variables.

Usage

```
GU_normalization(formula, data, weights, group)
```

Arguments

formula	an object of class "formula". See lm for further details.
data	a data frame containing the variables in the model.
weights	numeric vector of non-negative observation weights, hence of same length as `dep_var`.
group	name of the a binary variable (numeric or factor) identifying the two groups that will be compared.

Value

a list containing the adjusted formula, adjusted data, adjusted coefficient names, and the normalized regressors for prediction and the

References

Gardeazabal, Javier, and Arantza Ugidos. 2004. "More on identification in detailed wage decompositions." *Review of Economics and Statistics* 86(4): 1034-1036.

Examples

```
data("men8305")
mod1 <- log(wage) ~ union + married + nonwhite + education + experience
normalized_data <- GU_normalization(
  formula = mod1,
  data = men8305,
  weights = weights,
  group = year
)
```

`GU_normalization_get_coefficients`

Get coefficients for GU normalization

Description

This function constructs sums the coefficients of each factor variable to construct a additional coefficients for their originally left-out reference levels and adds them to the estimated coefficients vector.

Usage

```
GU_normalization_get_coefficients(coef_names, est_coef)
```

Arguments

`coef_names`	list with coefficients of every factor variable that need to be adjusted
`est_coef`	vector of estimated coefficients

GU_normalization_get_vcov
Get covariance matrix for GU normalization

Description

This function adjusts the covariance matrix for the additional coefficients of the originally left-out reference levels of all factor variable.

Usage

```
GU_normalization_get_vcov(coef_names, Cov_beta)
```

Arguments

coef_names	list with coefficients of every factor variable that need to be adjusted
Cov_beta	estimated covariance matrix of the regression coefficients

---

GU_normalization_sum_coefficients
Sum coefficients for GU normalization

Description

This function sums the coefficients of a single factor variable to construct an additional coefficient for the left-out reference level.

Usage

```
GU_normalization_sum_coefficients(coef_names, est_coef)
```

Arguments

coef_names	names of the dummy coefficients of a factor variable
est_coef	estimated coefficient vector

`GU_normalization_sum_vcov` *Sum covariance matrix for GU normalization*

Description

This function adjusts the covariance matrix for the additional coefficient of the originally left-out reference level of a single factor variable.

Usage

```
GU_normalization_sum_vcov(coef_names, Cov_beta)
```

Arguments

`coef_names`	names of the dummy coefficients of a factor variable
`Cov_beta`	estimated covariance matrix of the regression coefficients

---

`men8305` *Sample of male wage data from the CPS 1983-1985 and 2003-2005*

Description

A sample of the the Merged Outgoing Rotation Group of the Current Population Survey from 1983 to 1985 and 2003 to 2005, respectively, used as example by Fortin, Lemieux & Firpo (2011) in their handbook chapter. The data set contains a selection of 8 variables and a sample of 40,347 observations of male workers (i.e., a tenth of the origninal data set).

Usage

```
men8305
```

Format

A data frame with 40,347 rows and 8 variables.

wage Hourly wage in US dollars at constant prices

union Union status indicator

education Factor variable with 6 education levels: high-school graduates (reference), elementary, high-school dropouts , some college, college graduates, post college graduates

experience Factor variable with 9 potential experience levels, each of five years gap, 20 to 24 years as reference level)

married Married indicator

nonwhite Non-white indicator

year Indicator distinguishing pooled observations from the 1983 to 1985 period and those from 2003 to 2005

weights CPS sample weights

Source

Fortin, Nicole M., Thomas Lemieux, and Firpo Segio. 2011. "Decomposition Methods in Economics." In Orley Ashenfelter and David Card, eds., Handbook of Labor Economics, Volume 4a., Chapter 1, 1-102.

`nlys00` *Sample of NLSY79 wage data from 2000*

Description

Sample of National Longitudinal Survey (NLSY) 79 containig wage data from the year 2000 of workers who were aged 35 to 43 in that year. The data is from O'Neill and O'Neill (2006) and is used as an illustration of the Oxaca-Blinder mean decomposition in Firpo, Fortin, and Lemieuex (2011). The data contains 2655 male and 2654 female observations, respectively.

Usage

```
nlys00
```

Format

A data frame with 5,396 rows and 15 variables.

female Female indicator

wage Hourly wage in US dollars

age Age in years

central_city Central city indicator

msa Metropolitan statistical area (MSA) indicator

region Factor variable disinguishing 4 large regions

black Black indicator

hispanic Hispanic indicator

education Factor variable indicating highest attained education

afqt Percentile score of armed force qualification test (AFTQ) devided by 10

family_responsibility Family responsibility indicator

years_worked_civilian Years worked in cilivian labor force

years_worked_military Years worked in military

part_time Share of years worked in part-time

industry Factor variable identifying 4 industries

Source

Fortin, Nicole M., Thomas Lemieux, and Firpo Segio. 2011. "Decomposition Methods in Economics." In Orley Ashenfelter and David Card, eds., Handbook of Labor Economics, Volume 4a., Chapter 1, 1-102.

ob_decompose *Oaxaca-Blinder decomposition*

Description

ob_decompose implements the Oaxaca-Blinder decomposition that divides differences in the mean outcome between two groups into one part explained by different covariate means (composition effect) and into another part due to differences in linear regression coefficients linking covariates to the outcome variable (structure effect).

The function allows for 'doubly robust' decompositions where the sample of one group is reweighted such that it matches the covariates distribution of the other group before the regression coefficients are estimated.

For distributional statistics beyond the mean, the function performs the RIF regression decomposition proposed by Firpo, Fortin, and Lemieux (2018).

Usage

```
ob_decompose(
  formula,
  data,
  group,
  weights = NULL,
  reweighting = FALSE,
  normalize_factors = FALSE,
  reference_0 = TRUE,
  subtract_1_from_0 = FALSE,
  reweighting_method = "logit",
  trimming = FALSE,
  trimming_threshold = NULL,
  rifreg_statistic = NULL,
  rifreg_probs = c(1:9)/10,
  custom_rif_function = NULL,
  na.action = na.omit,
  bootstrap = FALSE,
  bootstrap_iterations = 100,
  bootstrap_robust = FALSE,
  cluster = NULL,
  cores = 1,
  vcov = stats::vcov,
  ...
)
```

Arguments

formula a formula object with an outcome variable Y on the left-hand side and the covariates X on the right-hand side. If reweighting = TRUE, the same covariates are used to estimate the conditional probabilities for the reweighting factor. A different model for estimating the conditional probabilities can be defined after a | operator on the right-hand side.

`data` a data frame containing the variables in the model.

`group` name of the a binary variable (numeric or factor) identifying the two groups that will be compared. The group identified by the lower ranked value in ‘group‘ (i.e., 0 in the case of a dummy variable or the first level of factor variable) is defined as group 0.

`weights` numeric vector of non-negative observation weights, hence of same length as `dep_var`. The default (`NULL`) is equivalent to `weights = rep(1, length(dep_var))`. If no weights are used, make sure you do not define this parameter (e.g. with `weights = NULL`).

`reweighting` boolean: if ‘TRUE‘, then the decomposition is performed with with respect to reweighted reference group yielding either a ’doubly robust’ Oaxaca-Blinder decomposition or a reweighted RIF decomposition.

`normalize_factors`
boolean: If ‘TRUE‘, then factor variables are normalized as proposed by Gardeazabal/Ugidos (2004) and results are not dependent on the factor’s reference group. Per default (`normalize_factors = FALSE`) and factors are not normalized.

`reference_0` boolean: if ‘TRUE‘ (default), then the group 0 – i.e., the group identified by the lower ranked value in ‘group‘ – will be defined as reference group. The reference group will be reweighted to match the covariates distribution of the counterfactual sample. By default, the composition effect is computed as `(X1 - X0) * b0` and the structure effect as `X1 * (b1 - b0)`. Putting `reference_0 = FALSE` changes the reference structure. Hence, the composition effect is computed as `(X1 - X0) * b1` and the structure effect as `X0 * (b1 - b0)`.

`subtract_1_from_0`
boolean: By default (‘FALSE‘), X0 is subtracted from X1 and beta0 from beta1 (X1b1 - X0b0) to compute the overall difference. Setting ‘subtract_1_from_0‘ to ‘TRUE‘ merely changes the sign of the decomposition results. This means the composition effect is computed as `(X0 - X1) * b1` and the structure effect as `X0 * (b0 - b1)`.

`reweighting_method`
specifies the method fit and predict conditional probabilities used to derive the reweighting factor. Currently, `"logit"`, `"fastglm"`, and `"random_forest"` are available.

`trimming` boolean: If `TRUE`, observations with dominant reweighting factor values are trimmend according to rule of Huber, Lechner, and Wunsch (2013). Per default, trimming is set to `FALSE`.

`trimming_threshold`
numeric: threshold defining the maximal accepted relative weight of the reweighting factor value (i.e., inverse probability weight) of a single observation. If `NULL`, the threshold is set to $sqrt(N)/N$, where N is the number of observations in the reference group.

`rifreg_statistic`
string containing the distributional statistic for which to compute the RIF. If ‘NULL‘ (default), no RIF regression decomposition is computed. If an available

statistic is selected, ‘ob_decompose‘ estimates a RIF regression decomposition. The ‘rifreg_statistic‘ can be one of "quantiles", "mean", "variance", "gini", "interquantile_range", "interquantile_ratio", or "custom". If "custom" is selected, a `custom_rif_function` needs to be provided.

`rifreg_probs` a vector of length 1 or more with probabilities of quantiles. Each quantile is indicated with a value between 0 and 1. Default is `c(1:9)/10`. If `statistic = "quantiles"`, a single RIF regression for every quantile in `probs` is estimated. An interquantile ratio (range) is defined by the ratio (difference) between the `max(probs)`-quantile and the `min(probs)`-quantile.

`custom_rif_function`
the RIF function to compute the RIF of the custom distributional statistic. Default is NULL. Only needs to be provided if `statistic = "custom"`. Every custom_rif_function needs the parameters `dep_var`, `weights` and `probs`. If they are not needed, they must be set to NULL in the function definition (e.g. `probs = NULL`). A custom function must return a data frame containing at least a "rif" and "weights" column. See `examples` for further details.

`na.action` generic function that defines how NAs in the data should be handled. Default is `na.omit`, leading to exclusion of observations that contain one or more missings. See na.action for further details.

`bootstrap` boolean: If ‘FALSE‘ (default), then no bootstrapped standard errors are calculated and, in the case of a standard Oaxaca-Blinder decomposition, analytical standard errors are estimated (assuming independence between groups).

`bootstrap_iterations`
positive integer indicating the number of bootstrap iterations to execute. Only required if `bootstrap = TRUE`.

`bootstrap_robust`
boolean: if ‘FALSE‘ (default), then bootstrapped standard errors are estimated as the standard deviations of the bootstrapp estimates. Otherwise, the function uses the bootstrap interquartile range rescaled by the interquantile range of the standard distribution to estimate standard errors.

`cluster` numeric vector of same length as `dep_var` indicating the clustering of observations. If `cluster = NULL` (default), no clustering is a assumend and bootstrap procedure resamples individual observations. Otherwise bootstrap procedure resamples clusters.

`cores` positive integer indicating the number of cores to use when computing bootstrap standard errors. Only required if `bootstrap = TRUE`.

`vcov` function estimating covariance matrix of regression coefficients if standard errors are not bootstrapped (i.e., `bootstrap = FALSE`). By default, vcov is used assuming homoscedastic errors.

`...` additional parameters passed to the custom_rif_function. Apart from dep_var, weights and probs they must have a different name than the the ones in rifreg. For instance, if you want to pass a parameter statistic to the custom_rif_function, name it custom_statistic. Additional parameters can also be passed to the density function used to estimate the RIF of quantiles.

Details

`ob_decompose()` contains for four different decomposition methods of observed group differences.

1. The original Oaxaca-Blinder decomposition (default) 2. A 'doubly robust Oaxaca-Blinder decomposition (`reweighting=TRUE`) 3. A RIF Regression decomposition. (e.g., `rifreg_statistic="quantiles"`) 4. A reweighted RIF regression decomposition. (`reweighting=TRUE` and `rifreg_statistic="quantiles"`)

The doubly robust OB decomposition is a robust and path independent alternative for detailed decompositions at the mean. is to combine reweighting with the linear Oaxaca-Blinder method (see Fortin et al., 2011: 48-51). This approach has the valuable side effect of accounting for potential errors introduced by an incomplete inverse probability weighting and the linear model specification, respectively.

A path independent method that goes beyond the mean is the RIF decomposition of Firpo, Fortin, and Lemieux (2018). The approach approximates the expected value of the 'recentered influence function' (RIF) of the distributional statistic (e.g., quantile, variance, or Gini coefficient) of an outcome variable conditional on covariates with linear regressions. RIF regression coefficients can be consistent estimates of the marginal effect of a small change in the expected value of a covariate to the distributional statistics of an outcome variable (see documentation of the companion package `rifreg`). Thus, they can be used to decompose between-group difference in distributional statistics. Firpo et al. (2018) combine the RIF regressions again with the reweighting estimator to avoid specification errors.

Value

an object of class `ob_decompose` containing a data.frame with the decomposition results for the quantiles and for the other distributional statistics, respectively, a data.frame with the estimated reweighting factor for every observation, a data.frame with sample quantiles of the reweighting factors and a list with standard errors for the decomposition terms, the quantiles of the reweighting factor, the bootstrapped Kolmogorov-Smirnov distribution to construct uniform confidence bands for quantiles, as well as a list with the normalized differences between the covariate means of the comparison group and the reweighted reference group.

A list object of class 'ob_decompose' containing the following components:

- 'ob_decompose': A list containing the decomposition results, covariance matrix, model fits and more detailed result information. - 'group_variable_name': A string indicating the name of the group variable. - 'group_variable_levels': A string indicating the levels of the group variable. - 'reference_group': A string indicating the which level of the group variable was used as reference group. - 'reweighting_estimates': A list containing the reweighting estimates if `reweighting=TRUE`, else (`NA`) - 'input_parameters': A list of input parameters used for the estimation.

References

Firpo, Sergio, Nicole M. Fortin, and Thomas Lemieux. 2018. "Decomposing Wage Distributions Using Recentered Influence Function Regressions." *Econometrics*, 6(2):28.

Fortin, Nicole, Thomas Lemieux, and Sergio Firpo. 2011. "Decomposition methods in economics." In Orley Ashenfelter and David Card, eds., *Handbook of labor economics*. Vol. 4. Elsevier, 1-102.

Gardeazabal, Javier, and Arantza Ugidos. 2004. "More on identification in detailed wage decompositions." *Review of Economics and Statistics*, 86(4): 1034-1036.

Examples

```
## Oaxaca-Blinder decomposition of gender wage gap
## with NLYS79 data like in Fortin, Lemieux, & Firpo (2011: 41)

data("nlys00")

mod1 <- log(wage) ~ age + central_city + msa + region + black +
  hispanic + education + afqt + family_responsibility + years_worked_civilian +
  years_worked_military + part_time + industry

# Using female coefficients (reference_0 = TRUE) to estimate counterfactual mean
decompose_female_as_reference <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reference_0 = TRUE
)
decompose_female_as_reference

# Using male coefficients (reference_0 = FALSE)
decompose_male_as_reference <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reference_0 = FALSE
)
decompose_male_as_reference

# Replicate first and third column in Table 3 in Fortin, Lemieux, & Firpo (2011: 41)
# Define aggregation of decomposition terms
custom_aggregation <- list(
  `Age, race, region, etc.` = c(
    "age",
    "blackyes",
    "hispanicyes",
    "regionNorth-central",
    "regionSouth",
    "regionWest",
    "central_cityyes",
    "msayes"
  ),
  `Education` = c(
    "education<10 yrs",
    "educationHS grad (diploma)",
    "educationHS grad (GED)",
    "educationSome college",
    "educationBA or equiv. degree",
    "educationMA or equiv. degree",
    "educationPh.D or prof. degree"
  ),
  `AFTQ` = "afqt",
```

```
  `L.T. withdrawal due to family` = "family_responsibility",
  `Life-time work experience` = c(
    "years_worked_civilian",
    "years_worked_military",
    "part_time"
  ),
  `Industrial sectors` = c(
    "industryManufacturing",
    "industryEducation, Health, Public Admin.",
    "industryOther services"
  )
)

# First column
summary(decompose_male_as_reference, custom_aggregation = custom_aggregation)

# Third column
summary(decompose_female_as_reference, custom_aggregation = custom_aggregation)

## Compare bootstrapped standard errors...
decompose_female_as_reference_bs <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  bootstrap = TRUE,
  bootstrap_iterations = 100
)
summary(decompose_female_as_reference_bs, custom_aggregation = custom_aggregation)

# ... to analytical standard errors (assuming independence between groups and
# homoscedasticity)
decompose_female_as_reference <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reference_0 = TRUE
)
summary(decompose_female_as_reference, custom_aggregation = custom_aggregation)

# Return standard errors for all detailed terms
summary(decompose_female_as_reference, aggregate_factors = FALSE)


## 'Doubly robust' Oaxaca-Blinder decomposition of gender wage gap
mod2 <- log(wage) ~ age + central_city + msa + region + black +
  hispanic + education + afqt + family_responsibility + years_worked_civilian +
 years_worked_military + part_time + industry | age + (central_city + msa) * region + (black +
  hispanic) * (education + afqt) + family_responsibility * (years_worked_civilian +
  years_worked_military) + part_time * industry
decompose_male_as_reference_robust <- ob_decompose(
  formula = mod2,
  data = nlys00,
  group = female,
```

```
  reference_0 = FALSE,
  reweighting = TRUE
)

# ... using random forests instead of logit to estimate weights
decompose_male_as_reference_robust_rf <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reference_0 = FALSE,
  reweighting = TRUE,
  method = "random_forest"
)


# Reweighted RIF Regression Decomposition
data("men8305")

model_rifreg <- log(wage) ~  union + education + experience |
  union * (education + experience) + education * experience

# Variance
variance_decomposition <- ob_decompose(
  formula = model_rifreg,
  data = men8305,
  group = year,
  reweighting = TRUE,
  rifreg_statistic = "variance"
)

# Deciles
deciles_decomposition <- ob_decompose(
  formula = model_rifreg,
  data = men8305,
  group = year,
  reweighting = TRUE,
  rifreg_statistic = "quantiles",
  rifreg_probs = c(1:9) / 10
)

# plot(deciles_decomposition)

# RIF regression decomposition with custom function

# custom function
custom_variance_function <- function(dep_var, weights, probs = NULL) {
  weighted_mean <- weighted.mean(x = dep_var, w = weights)
  rif <- (dep_var - weighted_mean)^2
  rif <- data.frame(rif, weights)
  names(rif) <- c("rif_variance", "weights")
  return(rif)
}
```

```
custom_decomposition <-
  ob_decompose(
   formula = model_rifreg,
   data = men8305,
   group = year,
   reweighting = TRUE,
   rifreg_statistic = "custom",
   custom_rif_function = custom_variance_function
 )
```

`ob_decompose_calculate_terms`
Calculate OB decomposition terms

Description

The function calculates the decomposition terms of the linear Oaxaca-Blinder decomposition based on the estimated OLS coefficients and the respective `model.matrix`.

Usage

```
ob_decompose_calculate_terms(
  beta0,
  beta1,
  X0,
  X1,
  weights0,
  weights1,
  reference_0
)
```

Arguments

`beta0`	vector of estimated coefficients of group 0
`beta1`	vector of estimated coefficients of group 1
`X0`	`model.matrix` of group 0
`X1`	`model.matrix` of group 1
`weights0`	vector of observation weights of group 0
`weights1`	vector of observation weights of group 1
`reference_0`	boolean: indicating if group 0 is the reference group and if its coefficients are used to compute the counterfactual mean.

ob_decompose_calculate_vcov

Calculate covariance matrix for OB decomposition terms

Description

The function calculate the covariance matrix for the decomposition terms of the linear Oaxaca-Blinder decomposition assuming independence between groups.

Usage

```
ob_decompose_calculate_vcov(
  beta0,
  beta1,
  X0,
  X1,
  weights0,
  weights1,
  Cov_beta0,
  Cov_beta1,
  reference_0
)
```

Arguments

beta0	vector of estimated coefficients of group 0
beta1	vector of estimated coefficients of group 1
X0	model.matrix of group 0
X1	model.matrix of group 1
weights0	vector of observation weights of group 0
weights1	vector of observation weights of group 1
Cov_beta0	estimated covariance matrix of coefficients of group 0
Cov_beta1	estimated covariance matrix of coefficients of group 1
reference_0	boolean: indicating if group 0 is the reference group and if its coefficients are used to compute the counterfactual mean.

References

Jann, Ben, 2005. "Standard errors for the Blinder-Oaxaca decomposition." *3rd German Stata Users' Group Meeting 2005*. Available from [https://boris.unibe.ch/69506/1/oaxaca_se_handout.pdf](https://boris.unibe.ch/

plot.dfl_decompose — *Plot decomposition terms for quantiles*

Description

The function plots decomposition terms for quantiles estimated with `dfl_decompose` over the unit interval.

Usage

```
## S3 method for class 'dfl_decompose'
plot(
  x,
  ...,
  confidence_bands = TRUE,
  confidence_level = 0.95,
  uniform_bands = FALSE
)
```

Arguments

`x` — an object of class "dfl_decompose", usually, a result of a call to [dfl_decompose()] with [statistics = "quantiles"].

`...` — other parameters to be passed through to plot function.

`confidence_bands` — If ‘TRUE‘ (default) and if standard errors have been bootstrapped, confidence bands are plotted.

`confidence_level` — numeric value between 0 and 1 (default = 0.95) that defines the confidence interval plotted as a ribbon and defined as `qnorm((1-confidence_level)/2) *` standard error.

`uniform_bands` — If ‘FALSE‘ (default), pointwise confidence bands are computed. Otherwise, uniform bands are constructed based on the bootstrapped Kolmogrov-Smirnov statistic (see summary.dfl_decompose).

Value

a ggplot illustrating the decomposition terms for quantiles.

Examples

```
data("men8305")
flf_model <- log(wage) ~ union * (education + experience) + education * experience
flf_male_inequality <- dfl_decompose(flf_model,
  data = men8305,
  weights = weights,
  group = year
```

```
)
plot(flf_male_inequality)
```

`plot.ob_decompose` *Plot decomposition terms for quantiles*

Description

The function plots decomposition terms for quantiles estimtated with `ob_decompose` over the unit interval.

Usage

```
## S3 method for class 'ob_decompose'
plot(
  x,
  ...,
  detailed_effects = TRUE,
  aggregate_factors = TRUE,
  custom_aggregation = NULL,
  confidence_bands = FALSE,
  confidence_level = 0.95
)
```

Arguments

`x`
: an object of class "ob_decompose", usually, a result of a call to [ob_decompose()] with [statistics = "quantiles"].

`...`
: other parameters to be passed through to plot function.

`detailed_effects`
: If 'TRUE' (default), then the detailed effects are plotted. Otherwise only the total (aggregate) effects are plotted.

`aggregate_factors`
: boolean, if 'TRUE' (default) terms associated with detailed factor levels are aggregated to a single term for every factor variable.

`custom_aggregation`
: list specifying the aggregation of detailed decomposition terms. The parameter 'custom_aggregation' overrides the parameter 'aggregate_factors'. If 'NULL' (default), then either all detailed terms or all terms associated with a single variable are returned.

`confidence_bands`
: If 'TRUE' and if standard errors have been bootstrapped, confidence bands are plotted.

`confidence_level`
: numeric value between 0 and 1 (default = 0.95) that defines the confidence interval plotted as a ribbon and defined as `qnorm((1-confidence_level)/2)` * standard error.

Value

a ggplot illustrating the decomposition terms for quantiles.

Examples

```
data("nlys00")

mod1 <- log(wage) ~ age + central_city + msa + region + black +
  hispanic + education + afqt + family_responsibility + years_worked_civilian +
  years_worked_military + part_time + industry

# plotting RIF regression decomposition of deciles

decompose_rifreg_deciles <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reweighting = TRUE,
  rifreg_statistic = "quantiles",
  bootstrap = TRUE,
  bootstrap_iterations = 50,
  reference_0 = FALSE
)

plot(decompose_rifreg_deciles)

plot(decompose_rifreg_deciles,
  confidence_bands = TRUE
)


# plotting Oaxaca-Blinder decomposition

decompose_ob_mean <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reweighting = TRUE,
  bootstrap = FALSE,
  reference_0 = FALSE
)

plot(decompose_ob_mean)
plot(decompose_ob_mean, detailed_effects = FALSE)


# With custom aggregation

custom_aggregation <- list(
  `Age, race, region, etc.` = c(
    "age",
    "blackyes",
```

```
      "hispanicyes",
      "regionNorth-central",
      "regionSouth",
      "regionWest",
      "central_cityyes",
      "msayes"
    ),
    `Education` = c(
      "education<10 yrs",
      "educationHS grad (diploma)",
      "educationHS grad (GED)",
      "educationSome college",
      "educationBA or equiv. degree",
      "educationMA or equiv. degree",
      "educationPh.D or prof. degree"
    ),
    `AFTQ` = "afqt",
    `L.T. withdrawal due to family` = "family_responsibility",
    `Life-time work experience` = c(
      "years_worked_civilian",
      "years_worked_military",
      "part_time"
    ),
    `Industrial sectors` = c(
      "industryManufacturing",
      "industryEducation, Health, Public Admin.",
      "industryOther services"
    )
  )

plot(decompose_ob_mean, custom_aggregation = custom_aggregation)
```

`print.dfl_decompose` *print method for class "dfl_decompose"*

Description

print method for class "dfl_decompose"

Usage

```
## S3 method for class 'dfl_decompose'
print(x, ...)
```

Arguments

`x`	an object of class "dfl_decompose", usually , a result of a call to [dfl_decompose()].
`...`	other parameters to be passed through to printing functions.

Value

The function `print.dfl_decompose()` displays the decompositions terms saved in `x`.

`print.ob_decompose` *print method for class "ob_decompose"*

Description

print method for class "ob_decompose"

Usage

```
## S3 method for class 'ob_decompose'
print(x, ...)
```

Arguments

`x`	an object of class "ob_decompose", usually , a result of a call to [ob_decompose()].
`...`	other parameters to be passed through to printing functions.

Value

The function `print.ob_decompose()` displays the decompositions terms saved in `x`.

`select_observations_to_be_trimmed` *Select observations with little common support to be trimmed*

Description

This function implements the trimming rule proposed by Huber, Lechner, and Wunsch (2014). Observations above the trimming threshold are trimmed in the reference group and in the comparison group. Per default, the timming is set to sqrt(N)/N, where N is the number of observation in the reweighted reference group. The function returns vector index of observation to be trimmed.

Usage

```
select_observations_to_be_trimmed(
  reweighting_factor,
  group_variable,
  group,
  trimming_threshold = NULL
)
```

Arguments

`reweighting_factor`
: Estimated reweigting factor

`group_variable`
: Variable identifying the reference and comparison group, respectively.

`group`
: Identifier of reference group

`trimming_threshold`
: threshold defining the maximal accepted relative weight of a reweighting factor/observation. If ‘NULL‘, the threshold is set to ‘sqrt(N)/N‘, where N is the number of observations in the reference group.

---

summary.dfl_decompose *summary method for class "dfl_decompose"*

Description

summary method for class "dfl_decompose"

Usage

```
## S3 method for class 'dfl_decompose'
summary(object, ..., confidence_level = 0.95, digits = 4)
```

Arguments

`object`
: an object of class "dfl_decompose", a result of a call to [dfl_decompose()].

`...`
: other parameters to be passed through to printing functions.

`confidence_level`
: numeric value between 0 and 1 (default = 0.95) that defines the confidence level of the printed confidence intervals.

`digits`
: number of digits to be printed.

Details

If standard errors were bootstrapped, standard errors and confidence bands are given. Pointwise confidences bands are defined as `qnorm((1-confidence_level)/2)` * standard error. Uniform bands are constructed by multiplying the standard error with `confidence_level`-quantile of the bootstrapped Kolmogorov-Smirnov statistic as in Chen et al. (2017).

Value

The function `summary.dfl_decompose()` displays the decompositions terms save in `object`. The function further returns a list with the displayed decomposition terms and, if standard errors were bootstrapped, the corresponding standard errors and confindence bands.

References

Chen, Mingli, Victor Chernozhukov, Iván Fernández-Val, and Blaise Melly. 2017. "Counterfactual: An R Package for Counterfactual Analysis." *The R Journal* 9(1): 370-384.

---

summary.ob_decompose *summary method for class "ob_decompose"*

Description

Apart from displaying the (detailed) decomposition results with standard errors, `summary.ob_decompose()` allows to customize the aggregation of the detailed decomposition terms.

Usage

```
## S3 method for class 'ob_decompose'
summary(
  object,
  ...,
  aggregate_factors = TRUE,
  custom_aggregation = NULL,
  confidence_level = 0.95
)
```

Arguments

`object` an object of class "ob_decompose", usually , a result of a call to [ob_decompose()].

`...` other parameters to be passed through to summary function.

`aggregate_factors`
: boolean, if 'TRUE' (default) terms associated with detailed factor levels are aggregated to a single term for every factor variable.

`custom_aggregation`
: list specifying the aggregation of detailed decomposition terms. The parameter 'custom_aggregation' overrides the parameter 'aggregate_factors'. If 'NULL' (default), then either all detailed terms or all terms associated with a single variable are returned.

`confidence_level`
: numeric value between 0 and 1 (default = 0.95) that defines the printed confidence interval.

Value

The function `summary.ob_decompose()` summarizes the decompositions terms saved in `object`.

Examples

```
data("nlys00")
mod1 <- log(wage) ~ age + education + years_worked_civilian +
  years_worked_military + part_time + industry

decompose_results <- ob_decompose(
  formula = mod1,
  data = nlys00,
  group = female,
  reference_0 = TRUE
)

# Print standard errors
summary(decompose_results)

# Aggregate decomposition terms associated with factor levels
summary(decompose_results, aggregate_factors = TRUE)

# custom aggregation of decomposition terms
custom_aggregation <-
  list(
    `Age` = c("age"),
    `Education` = c(
      "education<10 yrs",
      "educationHS grad (diploma)",
      "educationHS grad (GED)",
      "educationSome college",
      "educationBA or equiv. degree",
      "educationMA or equiv. degree",
      "educationPh.D or prof. degree"
    ),
    `Life-time work experience` = c(
      "years_worked_civilian",
      "years_worked_military",
      "part_time"
    ),
    `Industrial sectors` = c(
      "industryManufacturing",
      "industryEducation, Health, Public Admin.",
      "industryOther services"
    )
  )
summary(decompose_results, custom_aggregation = custom_aggregation)
```

Index