Package ‘segregation’

December 2, 2023

Type Package

Title Entropy-Based Segregation Indices

Version 1.1.0

Description Computes segregation indices, including the Index of Dissimilarity,
as well as the information-theoretic indices developed by
Theil (1971) <isbn:978-0471858454>, namely
the Mutual Information Index (M) and Theil's Information Index (H).
The M, further described by Mora and Ruiz-Castillo (2011) <doi:10.1111/j.1467-9531.2011.01237.x>
and Frankel and Volij (2011) <doi:10.1016/j.jet.2010.10.008>,
is a measure of segregation that is highly decomposable. The package provides
tools to decompose the index by units and groups (local segregation),
and by within and between terms. The package also provides a method to decompose
differences in segregation as described by Elbers (2021) <doi:10.1177/0049124121986204>.
The package includes standard error estimation by bootstrapping, which also corrects for
small sample bias. The package also contains functions for visualizing segregation patterns.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports data.table, checkmate, Rcpp, RcppProgress,

Encoding UTF-8

LazyData true

Suggests testthat, covr, knitr, rmarkdown, dplyr, ggplot2, scales,
tidycensus, tigris, rrapply, dendextend, patchwork

URL https://elbersb.github.io/segregation/

BugReports https://github.com/elbersb/segregation/issues

RoxygenNote 7.2.3

VignetteBuilder knitr

SystemRequirements C++17

LinkingTo Rcpp, RcppProgress

NeedsCompilation yes

Author Benjamin Elbers [aut, cre] (<https://orcid.org/0000-0001-5392-3448>)

Maintainer Benjamin Elbers <be2239@columbia.edu>

Repository CRAN

Date/Publication 2023-12-02 12:10:01 UTC

R topics documented:

`compress` *Compresses a data matrix based on mutual information (segregation)*

Description

Given a data set that identifies suitable neighbors for merging, this function will merge units iteratively, where in each iteration the neighbors with the smallest reduction in terms of total M will be merged.

Usage

```
compress(
  data,
  group,
  unit,
  weight = NULL,
  neighbors = "local",
  n_neighbors = 50,
  max_iter = Inf
)
```

Arguments

`data`	A data frame.
`group`	A categorical variable contained in `data`. Defines the first dimension over which segregation is computed.
`unit`	A categorical variable contained in `data`. Defines the second dimension over which segregation is computed.
`weight`	Numeric. Only frequency weights are allowed. (Default `NULL`)
`neighbors`	Either a data frame or a character. If data frame, then it needs exactly two columns, where each row identifies a set of "neighbors" that may be merged. If "local", considers the `n_neighbors` closest neighbors in terms of local segregation. If "all", all units are considered as possible neighbors. This may be very time-consuming.
`n_neighbors`	Only relevant if `neighbors` is `"local"`.
`max_iter`	Maximum number of iterations (Default `Inf`)

Value

Returns a data.table.

`dissimilarity` *Calculates Index of Dissimilarity*

Description

Returns the total segregation between `group` and `unit` using the Index of Dissimilarity.

Usage

```
dissimilarity(
  data,
  group,
  unit,
  weight = NULL,
  se = FALSE,
  CI = 0.95,
  n_bootstrap = 100
)
```

Arguments

data A data frame.

group A categorical variable or a vector of variables contained in data. Defines the first dimension over which segregation is computed. The D index only allows two distinct groups.

unit A categorical variable or a vector of variables contained in data. Defines the second dimension over which segregation is computed.

weight Numeric. (Default NULL)

se If TRUE, the segregation estimates are bootstrapped to provide standard errors and to apply bias correction. The bias that is reported has already been applied to the estimates (i.e. the reported estimates are "debiased") (Default FALSE)

CI If se = TRUE, compute the confidence (CI*100) in addition to the bootstrap standard error. This is based on percentiles of the bootstrap distribution, and a valid interpretation relies on a larger number of bootstrap iterations. (Default 0.95)

n_bootstrap Number of bootstrap iterations. (Default 100)

Value

Returns a data.table with one row. The column est contains the Index of Dissimilarity. If se is set to TRUE, an additional column se contains the associated bootstrapped standard errors, an additional column CI contains the estimate confidence interval as a list column, an additional column bias contains the estimated bias, and the column est contains the bias-corrected estimates.

References

Otis Dudley Duncan and Beverly Duncan. 1955. "A Methodological Analysis of Segregation Indexes," American Sociological Review 20(2): 210-217.

Examples

```
# Example where D and H deviate
m1 <- matrix_to_long(matrix(c(100, 60, 40, 0, 0, 40, 60, 100), ncol = 2))
m2 <- matrix_to_long(matrix(c(80, 80, 20, 20, 20, 20, 80, 80), ncol = 2))
dissimilarity(m1, "group", "unit", weight = "n")
dissimilarity(m2, "group", "unit", weight = "n")
```

`dissimilarity_expected` *Calculates expected values when true segregation is zero*

Description

When sample sizes are small, one group has a small proportion, or when there are many units, segregation indices are typically upwardly biased, even when true segregation is zero. This function simulates tables with zero segregation, given the marginals of the dataset, and calculates segregation. If the expected values are large, the interpretation of index scores might have to be adjusted.

Usage

```
dissimilarity_expected(
  data,
  group,
  unit,
  weight = NULL,
  fixed_margins = TRUE,
  n_bootstrap = 100
)
```

Arguments

`data`	A data frame.
`group`	A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.
`unit`	A categorical variable or a vector of variables contained in `data`. Defines the second dimension over which segregation is computed.
`weight`	Numeric. (Default `NULL`)
`fixed_margins`	Should the margins be fixed or simulated? (Default `TRUE`)
`n_bootstrap`	Number of bootstrap iterations. (Default `100`)

Value

A data.table with one row, corresponding to the expected value of the D index when true segregation is zero.

Examples

```
# build a smaller table, with 100 students distributed across
# 10 schools, where one racial group has 10% of the students
small <- data.frame(
    school = c(1:10, 1:10),
    race = c(rep("r1", 10), rep("r2", 10)),
    n = c(rep(1, 10), rep(9, 10))
)
dissimilarity_expected(small, "race", "school", weight = "n")
# with an increase in sample size (n=1000), the values improve
small$n <- small$n * 10
dissimilarity_expected(small, "race", "school", weight = "n")
```

entropy — *Calculates the entropy of a distribution*

Description

Returns the entropy of the distribution defined by `group`.

Usage

```
entropy(data, group, weight = NULL, base = exp(1))
```

Arguments

`data` A data frame.

`group` A categorical variable or a vector of variables contained in `data`.

`weight` Numeric. (Default `NULL`)

`base` Base of the logarithm that is used in the entropy calculation. Defaults to the natural logarithm.

Value

A single number, the entropy.

Examples

```
d <- data.frame(cat = c("A", "B"), n = c(25, 75))
entropy(d, "cat", weight = "n") # => .56
# this is equivalent to -.25*log(.25)-.75*log(.75)

d <- data.frame(cat = c("A", "B"), n = c(50, 50))
# use base 2 for the logarithm, then entropy is maximized at 1
entropy(d, "cat", weight = "n", base = 2) # => 1
```

exposure *Calculates pairwise exposure indices*

Description

Returns the pairwise exposure indices between groups

Usage

```
exposure(data, group, unit, weight = NULL)
```

Arguments

`data` A data frame.

`group` A categorical variable contained in `data`. Defines the first dimension over which segregation is computed.

`unit` A vector of variables contained in `data`. Defines the second dimension over which segregation is computed.

`weight` Numeric. (Default `NULL`)

Value

Returns a data.table with columns "of", "to", and "exposure". Read results as "exposure of group x to group y".

get_crosswalk *Create crosswalk after compression*

Description

After running compress, this function creates a crosswalk table. Usually it is preferred to call merge_units directly.

Usage

```
get_crosswalk(compression, n_units = NULL, percent = NULL, parts = FALSE)
```

Arguments

`compression` A "segcompression" object returned by compress.

`n_units` Determines the number of merges by specifying the number of units to remain in the compressed dataset. Only `n_units` or `percent` must be given. (default: `NULL`)

`percent` Determines the number of merges by specifying the percentage of total segregation information retained in the compressed dataset. Only `n_units` or `percent` must be given. (default: `NULL`)

`parts` (default: FALSE)

Value

Returns a ggplot2 plot.

Returns a data.table.

`ipf` *Adjustment of marginal distributions using iterative proportional fitting*

Description

Adjusts the marginal distributions for `group` and `unit` in `source` to the respective marginal distributions in `target`, using the iterative proportional fitting algorithm (IPF).

Usage

```
ipf(
  source,
  target,
  group,
  unit,
  weight = NULL,
  max_iterations = 100,
  precision = 1e-04
)
```

Arguments

`source`	A "source" data frame. The marginals of this dataset are adjusted to the marginals of `target`.
`target`	A "target" data frame. The function returns a dataset where the marginal distributions of `group` and `unit` categories are approximated by those of `target`.
`group`	A categorical variable or a vector of variables contained in `source` and `target`. Defines the first distribution for adjustment.
`unit`	A categorical variable or a vector of variables contained in `source` and `target`. Defines the second distribution for adjustment.
`weight`	Numeric. (Default `NULL`)
`max_iterations`	Maximum number of iterations used for the IPF algorithm.
`precision`	Convergence criterion for the IPF algorithm. In every iteration, the ratio of the source and target marginals are calculated for every category of `group` and `unit`. The algorithm converges when all ratios are smaller than `1 + precision`.

Details

The algorithm works by scaling the marginal distribution of group in the source data frame towards the marginal distribution of target; then repeating this process for unit. The algorithm then keeps alternating between group and unit until the marginals of the adjusted data frame are within the allowed precision. This results in a dataset that retains the association structure of source while approximating the marginal distribution of target. If the number of unit and group categories is different in source and target, the data frame returns the combination of unit and group categories that occur in both datasets. Zero values are replaced by a small, non-zero number (1e-4). Note that the values returned sum to the observations of the source data frame, not the target data frame. This is different from other IPF implementations, but ensures that the IPF does not change the number of observations.

Value

Returns a data frame that retains the association structure of source while approximating the marginal distributions for group and unit of target. The dataset identifies each combination of group and unit, and categories that only occur in either source or target are dropped. The adjusted frequency of each combination is given by the column n, while n_target and n_source contain the zero-adjusted frequencies in the target and source dataset, respectively.

References

W. E. Deming and F. F. Stephan. 1940. "On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known". Annals of Mathematical Statistics. 11 (4): 427–444.

T. Karmel and M. Maclachlan. 1988. "Occupational Sex Segregation — Increasing or Decreasing?" Economic Record 64: 187-195.

Examples

```
## Not run:
# adjusts the marginals of group and unit categories so that
# schools00 has similar marginals as schools05
adj <- ipf(schools00, schools05, "race", "school", weight = "n")

# check that the new "race" marginals are similar to the target marginals
# (the same could be done for schools)
aggregate(adj$n, list(adj$race), sum)
aggregate(adj$n_target, list(adj$race), sum)

# note that the adjusted dataset contains fewer
# schools than either the source or the target dataset,
# because the marginals are only defined for the overlap
# of schools
length(unique(schools00$school))
length(unique(schools05$school))
length(unique(adj$school))

## End(Not run)
```

isolation *Calculates isolation indices*

Description

Returns isolation index of each group

Usage

```
isolation(data, group, unit, weight = NULL)
```

Arguments

data	A data frame.
group	A categorical variable contained in data. Defines the first dimension over which segregation is computed.
unit	A vector of variables contained in data. Defines the second dimension over which segregation is computed.
weight	Numeric. (Default NULL)

Value

Returns a data.table with group column and isolation index.

matrix_to_long *Turns a contingency table into long format*

Description

Returns a data.table in long form, such that it is suitable for use in mutual_total, etc. Colnames and rownames of the matrix will be respected.

Usage

```
matrix_to_long(
  matrix,
  group = "group",
  unit = "unit",
  weight = "n",
  drop_zero = TRUE
)
```

Arguments

`matrix` A matrix, where the rows represent the units, and the column represent the groups.

`group` Variable name for group. (Default `group`)

`unit` Variable name for unit. (Default `unit`)

`weight` Variable name for frequency weight. (Default `weight`)

`drop_zero` Drop unit-group combinations with zero weight. (Default `TRUE`)

Value

A data.table.

Examples

```
m <- matrix(c(10, 20, 30, 30, 20, 10), nrow = 3)
colnames(m) <- c("Black", "White")
long <- matrix_to_long(m, group = "race", unit = "school")
mutual_total(long, "race", "school", weight = "n")
```

`merge_units` *Creates a compressed dataset*

Description

After running compress, this function creates a dataset where units are merged.

Usage

```
merge_units(compression, n_units = NULL, percent = NULL, parts = FALSE)
```

Arguments

`compression` A "segcompression" object returned by compress.

`n_units` Determines the number of merges by specifying the number of units to remain in the compressed dataset. Only `n_units` or `percent` must be given. (default: `NULL`)

`percent` Determines the number of merges by specifying the percentage of total segregation information retained in the compressed dataset. Only `n_units` or `percent` must be given. (default: `NULL`)

`parts` (default: FALSE)

Value

Returns a data.table.

mutual_difference *Decomposes the difference between two M indices*

Description

Uses one of three methods to decompose the difference between two M indices: (1) "shapley" / "shapley_detailed": a method based on the Shapley decomposition with a few advantages over the Karmel-Maclachlan method (recommended and the default, Deutsch et al. 2006), (2) "km": the method based on Karmel-Maclachlan (1988), (3) "mrc": the method developed by Mora and Ruiz-Castillo (2009). All methods have been extended to account for missing units/groups in either data input.

Usage

```
mutual_difference(
  data1,
  data2,
  group,
  unit,
  weight = NULL,
  method = "shapley",
  se = FALSE,
  CI = 0.95,
  n_bootstrap = 100,
  base = exp(1),
  ...
)
```

Arguments

`data1`	A data frame with same structure as `data2`.
`data2`	A data frame with same structure as `data1`.
`group`	A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.
`unit`	A categorical variable or a vector of variables contained in `data`. Defines the second dimension over which segregation is computed.
`weight`	Numeric. (Default `NULL`)
`method`	Either "shapley" (the default), "km" (Karmel and Maclachlan method), or "mrc" (Mora and Ruiz-Castillo method).
`se`	If `TRUE`, the segregation estimates are bootstrapped to provide standard errors and to apply bias correction. The bias that is reported has already been applied to the estimates (i.e. the reported estimates are "debiased") (Default `FALSE`)
`CI`	If `se = TRUE`, compute the confidence (CI*100) in addition to the bootstrap standard error. This is based on percentiles of the bootstrap distribution, and a valid interpretation relies on a larger number of bootstrap iterations. (Default `0.95`)

`n_bootstrap` Number of bootstrap iterations. (Default `100`)

`base` Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.

`...` Only used for additional arguments when when `method` is set to `shapley` or `km`. See ipf for details.

Details

The Shapley method is an improvement over the Karmel-Maclachlan method (Deutsch et al. 2006). It is based on several margins-adjusted data inputs and yields symmetrical results (i.e. `data1` and `data2` can be switched). When "shapley_detailed" is used, the structural component is further decomposed into the contributions of individuals units.

The Karmel-Maclachlan method (Karmel and Maclachlan 1988) adjusts the margins of `data1` to be similar to the margins of `data2`. This process is not symmetrical.

The Shapley and Karmel-Maclachlan methods are based on iterative proportional fitting (IPF), first introduced by Deming and Stephan (1940). Depending on the size of the dataset, this may take a few seconds (see ipf for details).

The method developed by Mora and Ruiz-Castillo (2009) uses an algebraic approach to estimate the size of the components. This will often yield substantively different results from the Shapley and Karmel-Maclachlan methods. Note that this method is not symmetric in terms of what is defined as `group` and `unit` categories, which may yield contradictory results.

A problem arises when there are `group` and/or `unit` categories in `data1` that are not present in `data2` (or vice versa). All methods estimate the difference only for categories that are present in both datasets, and report additionally the change in M that is induced by these cases as `additions` (present in `data2`, but not in `data1`) and `removals` (present in `data1`, but not in `data2`).

Value

Returns a data.table with columns `stat` and `est`. The data frame contains the following rows defined by `stat`: `M1` contains the M for `data1`. `M2` contains the M for `data2`. `diff` is the difference between `M2` and `M1`. The sum of the five rows following `diff` equal `diff`.

`additions` contains the change in M induces by `unit` and `group` categories present in `data2` but not `data1`, and `removals` the reverse.

All methods return the following three terms: `unit_marginal` is the contribution of unit composition differences. `group_marginal` is the contribution of group composition differences. `structural` is the contribution unexplained by the marginal changes, i.e. the structural difference. Note that the interpretation of these terms depend on the exact method used.

When using "km", one additional row is returned: `interaction` is the contribution of differences in the joint marginal distribution of `unit` and `group`.

When "shapley_detailed" is used, an additional column "unit" is returned, along with six additional rows for each unit that is present in both `data1` and `data2`. The five rows have the following meaning: `p1` (`p2`) is the proportion of the unit in `data1` (`data2`) once non-intersecting units/groups have been removed. The changes in local linkage are given by `ls_diff1` and `ls_diff2`, and their average is given by `ls_diff_mean`. The row named `total` summarizes the contribution of the unit towards structural change using the formula `.5 * p1 * ls_diff1 + .5 * p2 * ls_diff2`. The sum of all "total" components equals structural change.

If se is set to TRUE, an additional column se contains the associated bootstrapped standard errors, an additional column CI contains the estimate confidence interval as a list column, an additional column bias contains the estimated bias, and the column est contains the bias-corrected estimates.

References

W. E. Deming, F. F. Stephan. 1940. "On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known." The Annals of Mathematical Statistics 11(4): 427-444.

T. Karmel and M. Maclachlan. 1988. "Occupational Sex Segregation — Increasing or Decreasing?" Economic Record 64: 187-195.

R. Mora and J. Ruiz-Castillo. 2009. "The Invariance Properties of the Mutual Information Index of Multigroup Segregation." Research on Economic Inequality 17: 33-53.

J. Deutsch, Y. Flückiger, and J. Silber. 2009. "Analyzing Changes in Occupational Segregation: The Case of Switzerland (1970–2000)." Research on Economic Inequality 17: 171–202.

Examples

```
## Not run:
# decompose the difference in school segregation between 2000 and 2005,
# using the Shapley method
mutual_difference(schools00, schools05,
    group = "race", unit = "school",
    weight = "n", method = "shapley", precision = .1
)
# => the structural component is close to zero, thus most change is in the marginals.
# This method gives identical results when we switch the unit and group definitions,
# and when we switch the data inputs.

# the Karmel-Maclachlan method is similar, but only adjust the data in the forward direction...
mutual_difference(schools00, schools05,
    group = "school", unit = "race",
    weight = "n", method = "km", precision = .1
)

# ...this means that the results won't be identical when we switch the data inputs
mutual_difference(schools05, schools00,
    group = "school", unit = "race",
    weight = "n", method = "km", precision = .1
)

# the MRC method indicates a much higher structural change...
mutual_difference(schools00, schools05,
    group = "race", unit = "school",
    weight = "n", method = "mrc"
)

# ...and is not symmetric
mutual_difference(schools00, schools05,
    group = "school", unit = "race",
    weight = "n", method = "mrc"
)

## End(Not run)
```

mutual_expected *Calculates expected values when true segregation is zero*

Description

When sample sizes are small, one group has a small proportion, or when there are many units, segregation indices are typically upwardly biased, even when true segregation is zero. This function simulates tables with zero segregation, given the marginals of the dataset, and calculates segregation. If the expected values are large, the interpretation of index scores might have to be adjusted.

Usage

```
mutual_expected(
  data,
  group,
  unit,
  weight = NULL,
  within = NULL,
  fixed_margins = TRUE,
  n_bootstrap = 100,
  base = exp(1)
)
```

Arguments

`data`	A data frame.
`group`	A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.
`unit`	A categorical variable or a vector of variables contained in `data`. Defines the second dimension over which segregation is computed.
`weight`	Numeric. (Default `NULL`)
`within`	Apply algorithm within each group defined by this variable, and report the weighted average. (Default `NULL`)
`fixed_margins`	Should the margins be fixed or simulated? (Default `TRUE`)
`n_bootstrap`	Number of bootstrap iterations. (Default `100`)
`base`	Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.

Value

A data.table with two rows, corresponding to the expected values of segregation when true segregation is zero.

Examples

```
## Not run:
# the schools00 dataset has a large sample size, so expected segregation is close to zero
mutual_expected(schools00, "race", "school", weight = "n")

# but we can build a smaller table, with 100 students distributed across
# 10 schools, where one racial group has 10% of the students
small <- data.frame(
    school = c(1:10, 1:10),
    race = c(rep("r1", 10), rep("r2", 10)),
    n = c(rep(1, 10), rep(9, 10))
)
mutual_expected(small, "race", "school", weight = "n")
# with an increase in sample size (n=1000), the values improve
small$n <- small$n * 10
mutual_expected(small, "race", "school", weight = "n")

## End(Not run)
```

---

mutual_local *Calculates local segregation scores based on M*

Description

Returns local segregation indices for each category defined by `unit`.

Usage

```
mutual_local(
  data,
  group,
  unit,
  weight = NULL,
  se = FALSE,
  CI = 0.95,
  n_bootstrap = 100,
  base = exp(1),
  wide = FALSE
)
```

Arguments

`data` A data frame.

`group` A categorical variable or a vector of variables contained in `data`. Defines the dimension over which segregation is computed.

`unit` A categorical variable or a vector of variables contained in `data`. Defines the group for which local segregation indices are calculated.

`weight` Numeric. (Default `NULL`)

`se` If `TRUE`, the segregation estimates are bootstrapped to provide standard errors and to apply bias correction. The bias that is reported has already been applied to the estimates (i.e. the reported estimates are "debiased") (Default `FALSE`)

`CI` If `se = TRUE`, compute the confidence (CI*100) in addition to the bootstrap standard error. This is based on percentiles of the bootstrap distribution, and a valid interpretation relies on a larger number of bootstrap iterations. (Default `0.95`)

`n_bootstrap` Number of bootstrap iterations. (Default `100`)

`base` Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.

`wide` Returns a wide dataframe instead of a long dataframe. (Default `FALSE`)

Value

Returns a data.table with two rows for each category defined by `unit`, for a total of `2*(number of units)` rows. The column `est` contains two statistics that are provided for each unit: `ls`, the local segregation score, and `p`, the proportion of the unit from the total number of cases. If `se` is set to `TRUE`, an additional column `se` contains the associated bootstrapped standard errors, an additional column `CI` contains the estimate confidence interval as a list column, an additional column `bias` contains the estimated bias, and the column `est` contains the bias-corrected estimates. If `wide` is set to `TRUE`, returns instead a wide dataframe, with one row for each `unit`, and the associated statistics in separate columns.

References

Henri Theil. 1971. Principles of Econometrics. New York: Wiley.

Ricardo Mora and Javier Ruiz-Castillo. 2011. "Entropy-based Segregation Indices". Sociological Methodology 41(1): 159–194.

Examples

```
# which schools are most segregated?
(localseg <- mutual_local(schools00, "race", "school",
    weight = "n", wide = TRUE
))

sum(localseg$p) # => 1

# the sum of the weighted local segregation scores equals
# total segregation
sum(localseg$ls * localseg$p) # => .425
mutual_total(schools00, "school", "race", weight = "n") # M => .425
```

mutual_total *Calculates the Mutual Information Index M and Theil's Entropy Index H*

Description

Returns the total segregation between `group` and `unit`. If `within` is given, calculates segregation within each `within` category separately, and takes the weighted average. Also see `mutual_within` for detailed within calculations.

Usage

```
mutual_total(
  data,
  group,
  unit,
  within = NULL,
  weight = NULL,
  se = FALSE,
  CI = 0.95,
  n_bootstrap = 100,
  base = exp(1)
)
```

Arguments

`data` A data frame.

`group` A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.

`unit` A categorical variable or a vector of variables contained in `data`. Defines the second dimension over which segregation is computed.

`within` A categorical variable or a vector of variables contained in `data`. The variable(s) should be a superset of either the `unit` or the `group` for the calculation to be meaningful. If provided, segregation is computed within the groups defined by the variable, and then averaged. (Default `NULL`)

`weight` Numeric. (Default `NULL`)

`se` If `TRUE`, the segregation estimates are bootstrapped to provide standard errors and to apply bias correction. The bias that is reported has already been applied to the estimates (i.e. the reported estimates are "debiased") (Default `FALSE`)

`CI` If `se = TRUE`, compute the confidence (CI*100) in addition to the bootstrap standard error. This is based on percentiles of the bootstrap distribution, and a valid interpretation relies on a larger number of bootstrap iterations. (Default `0.95`)

`n_bootstrap` Number of bootstrap iterations. (Default `100`)

`base` Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.

Value

Returns a data.table with two rows. The column `est` contains the Mutual Information Index, M, and Theil's Entropy Index, H. The H is the the M divided by the `group` entropy. If `within` was given, M and H are weighted averages of the within-category segregation scores. If `se` is set to `TRUE`, an additional column `se` contains the associated bootstrapped standard errors, an additional column `CI` contains the estimate confidence interval as a list column, an additional column `bias` contains the estimated bias, and the column `est` contains the bias-corrected estimates.

References

Henri Theil. 1971. Principles of Econometrics. New York: Wiley.

Ricardo Mora and Javier Ruiz-Castillo. 2011. "Entropy-based Segregation Indices". Sociological Methodology 41(1): 159–194.

Examples

```
# calculate school racial segregation
mutual_total(schools00, "school", "race", weight = "n") # M => .425

# note that the definition of groups and units is arbitrary
mutual_total(schools00, "race", "school", weight = "n") # M => .425

# if groups or units are defined by a combination of variables,
# vectors of variable names can be provided -
# here there is no difference, because schools
# are nested within districts
mutual_total(schools00, "race", c("district", "school"),
    weight = "n"
) # M => .424

# estimate standard errors and 95% CI for M and H
## Not run:
mutual_total(schools00, "race", "school",
    weight = "n",
    se = TRUE, n_bootstrap = 1000
)

# estimate segregation within school districts
mutual_total(schools00, "race", "school",
    within = "district", weight = "n"
) # M => .087

# estimate between-district racial segregation
mutual_total(schools00, "race", "district", weight = "n") # M => .338
# note that the sum of within-district and between-district
# segregation equals total school-race segregation;
# here, most segregation is between school districts

## End(Not run)
```

mutual_total_nested *Calculates a nested decomposition of segregation for M and H*

Description

Returns the between-within decomposition defined by the sequence of variables in `unit`.

Usage

```
mutual_total_nested(data, group, unit, weight = NULL, base = exp(1))
```

Arguments

`data`	A data frame.
`group`	A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.
`unit`	A vector of variables contained in `data`. Defines the levels at which the decomposition should be computed.
`weight`	Numeric. (Default `NULL`)
`base`	Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.

Value

Returns a data.table similar to `mutual_total`, but with column `between` and `within` that define the levels of nesting.

Examples

```
mutual_total_nested(schools00, "race", c("state", "district", "school"),
    weight = "n"
)
# This is a simpler way to run the following manually:
# mutual_total(schools00, "race", "state", weight = "n")
# mutual_total(schools00, "race", "district", within = "state", weight = "n")
# mutual_total(schools00, "race", "school", within = c("state", "district"), weight = "n")
```

mutual_within *Calculates detailed within-category segregation scores for M and H*

Description

Calculates the segregation between group and unit within each category defined by within.

Usage

```
mutual_within(
  data,
  group,
  unit,
  within,
  weight = NULL,
  se = FALSE,
  CI = 0.95,
  n_bootstrap = 100,
  base = exp(1),
  wide = FALSE
)
```

Arguments

data	A data frame.
group	A categorical variable or a vector of variables contained in data. Defines the first dimension over which segregation is computed.
unit	A categorical variable or a vector of variables contained in data. Defines the second dimension over which segregation is computed.
within	A categorical variable or a vector of variables contained in data that defines the within-segregation categories.
weight	Numeric. (Default NULL)
se	If TRUE, the segregation estimates are bootstrapped to provide standard errors and to apply bias correction. The bias that is reported has already been applied to the estimates (i.e. the reported estimates are "debiased") (Default FALSE)
CI	If se = TRUE, compute the confidence (CI*100) in addition to the bootstrap standard error. This is based on percentiles of the bootstrap distribution, and a valid interpretation relies on a larger number of bootstrap iterations. (Default 0.95)
n_bootstrap	Number of bootstrap iterations. (Default 100)
base	Base of the logarithm that is used in the calculation. Defaults to the natural logarithm.
wide	Returns a wide dataframe instead of a long dataframe. (Default FALSE)

Value

Returns a data.table with four rows for each category defined by within. The column est contains four statistics that are provided for each unit: M is the within-category M, and p is the proportion of the category. Multiplying M and p gives the contribution of each within-category towards the total M. H is the within-category H, and ent_ratio provides the entropy ratio, defined as EW/E, where EW is the within-category entropy, and E is the overall entropy. Multiplying H, p, and ent_ratio gives the contribution of each within-category towards the total H. If se is set to TRUE, an additional column se contains the associated bootstrapped standard errors, an additional column CI contains the estimate confidence interval as a list column, an additional column bias contains the estimated bias, and the column est contains the bias-corrected estimates. If wide is set to TRUE, returns instead a wide dataframe, with one row for each within category, and the associated statistics in separate columns.

References

Henri Theil. 1971. Principles of Econometrics. New York: Wiley.

Ricardo Mora and Javier Ruiz-Castillo. 2011. "Entropy-based Segregation Indices". Sociological Methodology 41(1): 159–194.

Examples

```
## Not run:
(within <- mutual_within(schools00, "race", "school",
    within = "state",
    weight = "n", wide = TRUE
))
# the M for state "A" is .409
# manual calculation
schools_A <- schools00[schools00$state == "A", ]
mutual_total(schools_A, "race", "school", weight = "n") # M => .409

# to recover the within M and H from the output, multiply
# p * M and p * ent_ratio * H, respectively
sum(within$p * within$M) # => .326
sum(within$p * within$ent_ratio * within$H) # => .321
# compare with:
mutual_total(schools00, "race", "school", within = "state", weight = "n")

## End(Not run)
```

schools00 *Ethnic/racial composition of schools for 2000/2001*

Description

Fake dataset used for examples. Loosely based on data provided by the National Center for Education Statistics, Common Core of Data, with information on U.S. primary schools in three U.S. states. The original data can be downloaded at https://nces.ed.gov/ccd/.

Usage

```
schools00
```

Format

A data frame with 8,142 rows and 5 variables:

state either A, B, or C

district school agency/district ID

school school ID

race either native, asian, hispanic, black, or white

n n of students by school and race

schools05 *Ethnic/racial composition of schools for 2005/2006*

Description

Fake dataset used for examples. Loosely based on data provided by the National Center for Education Statistics, Common Core of Data, with information on U.S. primary schools in three U.S. states. The original data can be downloaded at `https://nces.ed.gov/ccd/`.

Usage

```
schools05
```

Format

A data frame with 8,013 rows and 5 variables:

state either A, B, or C

district school agency/district ID

school school ID

race either native, asian, hispanic, black, or white

n n of students by school and race

`school_ses` *Student-level data including SES status*

Description

Fake dataset used for examples. This is an individual-level dataset of students in schools.

Usage

```
school_ses
```

Format

A data frame with 5,153 rows and 3 variables:

school_id school ID

ethnic_group one of A, B, or C

ses_quintile SES of the student (1 = lowest, 5 = highest)

`scree_plot` *Scree plot for segregation compression*

Description

A plot that allows to visually see the effect of compression on mutual information.

Usage

```
scree_plot(compression, tail = Inf)
```

Arguments

`compression`	A "segcompression" object returned by compress.
`tail`	Return only the last `tail` units (default: `Inf`)

Value

Returns a ggplot2 plot.

segcurve *A visual representation of two-group segregation*

Description

Produces one or several segregation curves, as defined in Duncan and Duncan (1955)

Usage

```
segcurve(data, group, unit, weight = NULL, segment = NULL)
```

Arguments

`data` A data frame.

`group` A categorical variable contained in `data`. Defines the first dimension over which segregation is computed.

`unit` A categorical variable contained in `data`. Defines the second dimension over which segregation is computed.

`weight` Numeric. (Default `NULL`)

`segment` A categorical variable contained in `data`. (Default `NULL`) If given, several segregation curves will be shown, one for each segment.

Value

Returns a ggplot2 object.

segplot *A visual representation of segregation*

Description

Produces a segregation plot.

Usage

```
segplot(
  data,
  group,
  unit,
  weight,
  order = "segregation",
  secondary_plot = NULL,
  reference_distribution = NULL,
  bar_space = 0,
  hline = NULL
)
```

Arguments

`data` A data frame.

`group` A categorical variable or a vector of variables contained in `data`. Defines the first dimension over which segregation is computed.

`unit` A categorical variable or a vector of variables contained in `data`. Defines the second dimension over which segregation is computed.

`weight` Numeric. (Default `NULL`)

`order` A character, either "segregation", "entropy", "majority", or "majority_fixed". Affects the ordering of the units. The horizontal ordering of the groups can be changed by using a factor variable for `group`. The difference between "majority" and "majority_fixed" is that the former will reorder the groups in such a way that the majority group actually comes first. If you want to control the ordering yourself, use "majority_fixed" and specify the `group` variable as a factor variable.

`secondary_plot` If `NULL` (default), no secondary plot is drawn. If "segregation", a secondary plot is drawn that shows adjusted local segregation scores for each unit. If "cumulative", a secondary plot is drawn that shows the cumulative contribution of each unit toward the total H (calculated as the proportion of each unit times the adjusted local segregation of each unit)0.

`reference_distribution` Specifies the reference distribution, given as a two-column data frame, to be plotted on the right. If order is `segregation`, then this reference distribution is also used to compute the local segregation scores.

`bar_space` Specifies space between single units.

`hline` Default `NULL`. If a color is specified, horizontal lines will be drawn where groups are separated.

Value

Returns a ggplot2 or patchwork object.

Index