Package: mdepriv (via r-universe)

October 12, 2024

Title Synthetic Indicators of Multiple Deprivation

Version 0.0.5

Author Attilio Benini [aut, cre], Aldo Benini [aut]

Maintainer Attilio Benini <attilio.benini@gmx.net>

Description mdepriv is a R-adaptation of a same-named user-written Stata command for computing basic synthetic scores of multiple deprivation.

License GPL-3

Depends R (>= 3.1.0)

Imports stats (>= 3.6.1), wCorr (>= 1.9.1), Weighted.Desc.Stat (>= 1.0)

Suggests dplyr, forcats, ggplot2, graphics, kableExtra, knitr, purrr, rmarkdown, stringr, testthat (>= 2.1.0), tibble, tidyr, utils, visdat

URL https://github.com/a-benini/mdepriv, https://a-benini.github.io/mdepriv

BugReports https://github.com/a-benini/mdepriv/issues

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.0

Repository https://humanitarian-user-group.r-universe.dev

RemoteUrl https://github.com/a-benini/mdepriv

RemoteRef HEAD

RemoteSha 10378986c7705121a010ac89baf706538437c7ea

Contents

`corr_mat` *Correlation Matrix Used in Certain Weighting Schemes*

Description

Returns the item correlation matrix (and when K dimensions have been specified, K matrices) that the function `mdepriv` internally generates

- when the argument `method = "bv"` and the argument `bv_corr_type = "mixed"` or `"pearson"`
- or when the argument `wa` is specified and the argument `wb = "mixed"` or `"pearson"`.

Permits inspection of the correlation structure and can be used for complementary analytic purposes such as factor analysis or biplots.

Usage

```
corr_mat(
  data,
  items,
  sampling_weights = NA,
  corr_type = c("mixed", "pearson"),
  output = c("numeric", "type", "both")
)
```

Arguments

`data`	a `data.frame` or a `matrix` with columns containing variables representing deprivation items measured on the [0,1] range. Optionally a column with user defined sampling weights (s. argument `sampling_weights`) can be included.
`items`	a character string vector or list of such vectors specifying the indicators / items within the argument `data` from which the deprivation scores are derived. If the user inputs a list with more than one vector, the items are grouped in dimensions. By naming the list elements, the user defines the dimensions' names. Else, dimensions are named by default `"Dimension 1"` to `"Dimension K"` where K is the number of dimensions.
`sampling_weights`	a character string corresponding to column heading of a numeric variable within the argument `data` which is specified as sampling weights. By default set to `NA`, which means no specification of sampling weights.

corr_type	a character string selecting the correlation type. Available choices are "mixed" (default) and "pearson". The option "mixed" detects automatically the appropriate correlation type "pearson", "polyserial" or "polychoric" for each pair of items (s. Details).
output	a character string vector selecting the output. Available choices are "numeric" (default), "type" or "both". "numeric" delivers a numeric correlation matrix or list of such matrices. (the latter if the items are grouped in more than one dimension). "type" delivers the correlation type applied to each pair of items in a matrix, respectively list of matrices analog to the numeric output. "both" combines the "numeric" and "type" as a list.

Details

The calculation of the correlation coefficient for a pair of items is based on the function weightedCorr.

When setting the argument corr_type to "mixed" the appropriate correlation type "pearson", "polyserial" or "polychoric" is automatically detected for each pair of items by the following rules:

- "pearson": both items have > 10 distinct values.
- "polyserial": one item has $\leq$ 10, the other > 10 distinct values.
- "polychoric": both items have $\leq$ 10 distinct values.

When the argument corr_type is set to "pearson" this correlation type is forced on all item pairs.

Depending on the correlation type(s) used, the matrix may not be positive semidefinite and therefore not immediately suitable for purposes such as factor analysis. This is more likely to happen when some of the items are binary. The function nearPD will produce the nearest positive definite matrix.

Value

Either a single matrix or a list composed of several matrixes (s. argument "output").

Examples

```
head(simul_data, 3) # data used for demonstration

corr_mat(simul_data, c("y1", "y4", "y5", "y6")) # default output: numeric
corr_mat(simul_data, c("y1", "y4", "y5", "y6"), output = "type")
corr_mat(simul_data, c("y1", "y4", "y5", "y6"), output = "both")

# with sampling weights (3rd argument)
corr_mat(simul_data, c("y1", "y4", "y5", "y6"), "sampl_weights")

# choose correlation type
corr_mat_default <- corr_mat(simul_data, c("y1", "y4", "y5", "y6"))
corr_mat_mixed <- corr_mat(simul_data, c("y1", "y4", "y5", "y6"), corr_type = "mixed")
all.equal(corr_mat_default, corr_mat_mixed) # "mixed is corr_type's default
# force a correlation type on all pairs of items
corr_mat(simul_data, c("y1", "y4", "y5", "y6"), corr_type = "pearson")

# grouping items in dimensions
```

```
corr_mat(simul_data, list(c("y1", "y4", "y5", "y6"), c("y2", "y3", "y7")))
# customized group / dimension labels
corr_mat(simul_data, list("Group A" = c("y1", "y4", "y5", "y6"),
                          "Group B" = c("y2", "y3", "y7")))

# mdepriv output / returns as template for corr_mat arguments
# items grouped as dimensions
dim <- list("Group X" = c("y1", "y4", "y5", "y6"), "Group Z" = c("y2", "y3", "y7"))

# model: betti-verma ("bv"): correlation type = pearson, rhoH = NA (data driven)
bv_pearson <- mdepriv(simul_data, dim, "sampl_weights",
                      method = "bv", bv_corr_type = "pearson", output = "all")
# use model output as arguments
corr_mat(bv_pearson$data, bv_pearson$items, bv_pearson$sampling_weights,
         corr_type = bv_pearson$wb, output = "both")

# model: user defined double weighting
# 1st factor = wa = "equal", 2nd factor = wa = "mixed" (correlation type),
# rhoH = NA (data driven)
eq_mixed <- mdepriv(simul_data, dim, "sampl_weights",
                    wa = "equal", wb = "mixed", output = "all")
# use model output as arguments
corr_mat(eq_mixed$data, eq_mixed$items, eq_mixed$sampling_weights,
         corr_type = eq_mixed$wb, output = "both")

# model: user defined double weighting
# 1st factor = wa = "bv", 2nd factator = wb = "diagonal"
# (all off-diagonal correlations = 0), rhoH = NA (irrelvant)
bv_diagonal <- mdepriv(simul_data, dim, "sampl_weights",
                       wa = "bv", wb = "diagonal", output = "all")
# use model output as arguments
try(
corr_mat(bv_diagonal$data, bv_diagonal$items, bv_diagonal$sampling_weights,
         corr_type = bv_diagonal$wb, output = "both")
)
# triggers an error because:
bv_diagonal$wb
# if corr_type is left as the default or set to a valid option, then ...
corr_mat(bv_diagonal$data, bv_diagonal$items, bv_diagonal$sampling_weights)
# ...  it works
# for the arguments data, items and sampling_weights the ...
# ... corresponding mdepriv outputs are always valid

# plot unique correlation values and their relation to rhoH
items_sel <- c("y1", "y4", "y5", "y6", "y2", "y3", "y7") # a selection of items
corr_mat  <- corr_mat(simul_data, items_sel) # corr_type default: "mixed"
rhoH      <- mdepriv(simul_data, items_sel, method = "bv", output = "rhoH")
             # bv_corr_type default: "mixed"

corr_val       <- unique(sort(corr_mat)) # sorted unique correlation values
dist_rhoH_corr <- abs(corr_val - rhoH) # distance of corr. values to rhoH
bounding_rhoH  <- abs(dist_rhoH_corr - min(dist_rhoH_corr)) < 10^-10
# TRUE if one of the two corr. values bounding rhoH else FALSE
```

```
corr_val_col   <- ifelse(bounding_rhoH, "black", "gray") # colors for corr. values

barplot(corr_val,
  col = corr_val_col,
  border = NA,
  ylim = c(-0.2, 1),
  ylab = "correlation value [-1,+1]",
  main = "sorted unique correlation values and rhoH"
)

abline(h = rhoH, col = "red", lwd = 1.5)
text(0, rhoH + 0.05, paste0("rhoH = ", round(rhoH, 4)), adj = 0, col = "red")

legend("left",
  "correlation values\nbounding largest gap",
  col = "black", pch = 15, pt.cex = 2, bty = "n"
)
```

`mdepriv` *Synthetic Indicators of Multiple Deprivation*

Description

calculates synthetic scores of multiple deprivation from unidimensional indicators and/or basic items of deprivation. The indicators / items all have to be negatively oriented, meaning: higher values are less desirable. The scores are weighted sums of the individual indicators / items taking values on the [0,1] range. Several alternative weighting methods are available, notably Betti & Verma's (1998) double-weighting method.

Usage

```
mdepriv(
  data,
  items,
  sampling_weights = NA,
  method = c("cz", "ds", "bv", "equal"),
  bv_corr_type = c("mixed", "pearson"),
  rhoH = NA,
  wa = NA,
  wb = NA,
  user_def_weights = NA,
  score_i_heading = "score_i",
  output = c("view", "all", "weighting_scheme", "aggregate_deprivation_level",
    "summary_by_dimension", "summary_by_item", "summary_scores", "score_i",
   "sum_sampling_weights", "data", "items", "sampling_weights", "wa", "wb", "rhoH",
    "user_def_weights", "score_i_heading")
)
```

Arguments

data	a data.frame or a matrix with columns containing variables representing deprivation items measured on the [0,1] range. Optionally a column with user defined sampling weights (s. argument sampling_weights) can be included.
items	a character string vector or list of such vectors specifying the indicators / items within the argument data, from which the deprivation scores are derived. By using a list with more than one vector, the items are grouped in dimensions. By naming the list elements, the user defines the dimensions' names. Else, dimensions are named by default "Dimension 1" to "Dimension K" where K is the number of dimensions.
sampling_weights	a character string corresponding to the column heading of a numeric variable within the argument data which is specified as sampling weights. By default set to NA, which means no specification of sampling weights.
method	a character string selecting the weighting scheme. Available choices are "cz" (default), "ds", "bv" and "equal" (s. Details).
bv_corr_type	a character string selecting the correlation type if method = "bv". Available choices are "mixed" (default) and "pearson". The option "mixed" automatically detects the appropriate correlation type "pearson", "polyserial" or "polychoric" for each pair of items (s. Details).
rhoH	numeric. Permits setting rhoH if the argument method is set to "bv", or the argument wb is set to either "mixed" or "pearson". rhoH distributes high and low coefficients of the triangular item correlation table to two factors. By default rhoH is set to NA, which causes its automatic calculation according to Betti & Verma's (1998) suggestion to divide the ordered set of correlation coefficients at the point of their largest gap. Alternatively, the user can set a value for rhoH in the interval [-1, +1]. When rhoH is automatically calculated, the weights of items that overall are more weakly correlated with the other items turn out higher, compared to their weights when the user chooses a rhoH value far from the automatic version. If the user chooses more than one dimension, rhoH is common for all.
wa, wb	two single character strings providing alternative, more flexible ways to select the weighting schemes. Weights are computed as the product of two terms as in the Betti-Verma scheme. wa selects the form of the first factor and is one of "cz", "ds", "bv" or "equal" (s. Details). wb selects the form of the second factor and is one of "mixed", "pearson" or "diagonal", where the latter sets all off-diagonal correlations to zero (s. Details). wa and wb are set both by default to NA, which means no specification. Specify either wa and wb, or method, not both.
user_def_weights	a numeric vector or list of such vectors to pass user-defined weights. To pass these weights correctly to the corresponding items, the structure of the vector, respectively of the list, must match the argument items. By creating a list with more than one vector, the user groups the items in dimensions. The elements of the vector, respectively of each vector within the list, must sum to 1. User-defined names of dimensions are inherited from the argument items. user_def_weights is set by default to NA, which means unspecified.

`score_i_heading`
: a character string (default: `"score_i"`) giving a heading to the `score_i` column in the output `"data"`.

`output`
: a character string vector selecting the output. Available multiple choices are `"view"` (default), `"all"`, `"weighting_scheme"`, `"aggregate_deprivation_level"`, `"summary_by_dimension"`, `"summary_by_item"`, `"summary_scores"`, `"score_i"`, `"sum_sampling_weights"`, `"data"`, `"items"`, `"sampling_weights"`, `"wa"`, `"wb"`, `"rhoH"`, `"user_def_weights"` and `"score_i_heading"` (s. Value).

Details

`mdepriv` is an adaptation for R of a homonymous community-contributed Stata command developed by Pi Alperin & Van Kerm (2009) for computing synthetic scores of multiple deprivation from unidimensional indicators and/or basic items of deprivation. The underlying literature and algebra are not recapitulated here. They are documented in Pi Alperin & Van Kerm (2009). There are minor differences vis-a-vis the predecessor in Stata, pointed out in the `vignette("what_is_different_from_stata")`.

The scores are weighted sums of the individual indicators / items. Both the items and the scores are limited to the [0, 1] range, and the item weights automatically sum to 1. As customary in deprivation research, all indicators / items require negative orientation (i.e., higher values are less desirable). In preparation, any item with values on [0, max], where max > 1, has to be transformed, using a suitable transformation. The choice of transformation function is dictated by substantive concerns. With i indexing the $i-th$ observation and j indexing the $j-th$ item, and all xij >= 0, plausible functions are:

- $yij = xij/(theoreticalmaximumofxj)$, particularly if all xj to be transformed have natural scales.
- $yij = xij/(c * mean(xj))$ in the absence of natural scales or theoretical maxima, with $c > 1$ a constant identical for all xj and chosen large enough so that all $max(yj) <= 1$.
- the asymptotic $yij = (xij/mean(xj))/(1 + (xij/mean(xj)))$, which implies $mean(yj) =$ 0.5 and $max(yj) < 1$ for all yj

and many others.

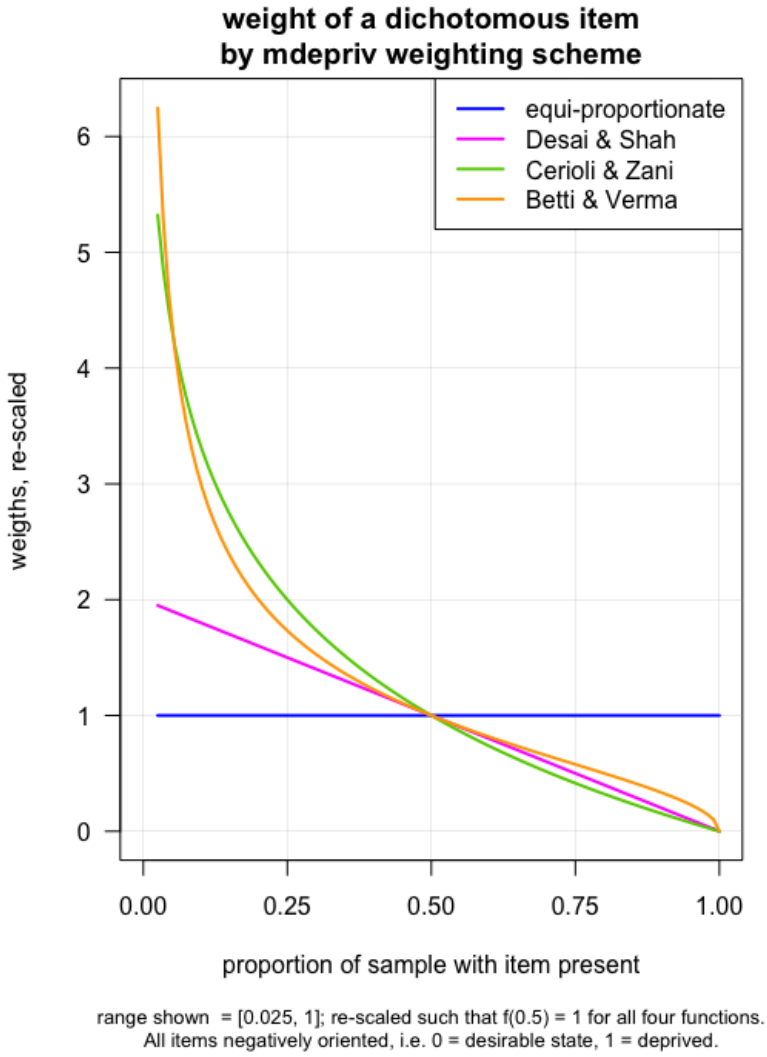
The multiplicative re-scaling (first two examples) preserves the coefficient of variation, an important aspect of item discrimination; in the third example, yj = 0.5 at the point where $xj = mean(xj)$ for all xj. This ensures the same neutral location for all items thus transformed. Dichotomous items must be coded 0/1, where 1 represents the undesirable state.

The transformation of ordinal indicators requires special care. Multiplicative re-scaling, which turns ordinal variables into pseudo-interval level ones in the [0, 1] range, is acceptable if it is unlikely to create significant distortions. Else, a data-driven transformation, such as the ridit, may be appropriate.

The available weighting schemes with the argument `method`, respectively `wa` are:

- `"cz"` for Cerioli & Zani (1990) weighting.
- `"ds"` for Desai & Shah (1988) weighting.
- `"bv"` for Betti & Verma (1998) weighting.
- `"equal"` for equi-proportionate weighting of items.

The differences among the four methods are visualized in this graph, as far as the weighting of dichotomous items is concerned. From $equal$ to $Betti - Verma$, the weighting schemes are increasingly sensitive to items with low prevalence in the sample. Thus, in all schemes except "equal", assets that most households own, and which only the most destitute lack (e.g., minimal cooking equipment), are given higher weights than items that most lack, and which only the better-off may own (e.g., a motorcycle) (Items are negatively coded, e.g., 1 = "household lacks the asset").



For continuous items (e.g., a food insecurity score), higher values denote less desirable states. Item weights are proportionate to (1 - $mean(yj)$) for the Desai-Shah scheme, to log(1 / $mean(yj)$) for Cerioli-Zani, and to the coefficients of variation [i.e., $std.dev(xj)$ / $mean(xj)$] for Betti-Verma.

Differently from the other three methods, Betti-Verma also controls for redundancy among items by lowering the weights of items that are highly correlated with many items.

Formulas and literature references are provided in Pi Alperin & Van Kerm (2009). `"cz"` and `"ds"` are built on the function `w.mean`. Whereas `"bv"` relies for its 1st factor (`wa`) on `w.cv`. `"bv"`'s 2nd

factor (wb) as well as any specification of wb but "diagonal" rely on weightedCorr.

When setting the argument bv_corr_type, respectively wb to "mixed", the appropriate correlation type "pearson", "polyserial" or "polychoric" is automatically detected for each pair of items by the following rules:

- "pearson": both items have > 10 distinct values.
- "polyserial": one item has ≤ 10, the other > 10 distinct values.
- "polychoric": both items have ≤ 10 distinct values.

When bv_corr_type respectively wb is set to "pearson" this correlation type is forced on all item pairs.

Value

a list or a single object according to the argument output. Possible output elements:

- "view" (default) composes a list including "weighting_scheme", "aggregate_deprivation_level", "summary_by_dimension", "summary_by_item" and "summary_scores".
- "all" delivers a list with all possible output elements.
- "weighting_scheme" a character string returning the weighting scheme chosen by the argument method, the arguments wa and wb, or the argument user_def_weights, respectively.
- "aggregate_deprivation_level" a single numeric value in the [0,1] range displaying the aggregate deprivation level.
- "summary_by_dimension" a data.frame containing the variables: Dimension (dimension names), N_Item (number of items per dimension), Index (within-dimension index), Weight (dimension weights), Contri (dimension contribution), Share (dimension relative contribution). If the user did not specify two or more dimensions (groups of items) "summary_by_dimension" is dropped from the output, unless it is explicitly requested as an element of the argument output.
- "summary_by_item" a data.frame containing variables: Dimension (dimension names), Item (item names), Index (item means), Weight (item weights), Contri (item contributions), Share (relative item contributions). The column Dimension is dropped unless at least two dimensions (groups of items) are specified or if "summary_by_dimension" is explicitly requested as an element of the argument output.
- "summary_scores" a data.frame containing these statistics of "score_i": N_Obs. (number of observations), Mean (mean), Std._Dev. (standard deviation), Min (minimum), Max (maximum).
- "score_i" a numeric vector returning the score for each observation.
- "sum_sampling_weights" a numeric value equal to the sum of sampling weights. If the argument sampling_weights is unspecified, NA is returned.
- "data" a data.frame including the argument data as well as a merged column containing the scores (default heading "score_i", which can be altered by the argument score_i_heading).
- "items" a named list of one or more character vectors returning the argument items grouped as dimensions. If no dimensions were specified a list with only one vector is returned. The list can be re-used as a template for the argument "items" in the functions mdepriv and corr_mat without needing to prior unlist.

- "sampling_weights" single character strings returning the specification of the argument sampling_weights, if unspecified NA.
- "wa", "wb" two single character strings giving the weighting scheme for the 1st, respectively the 2nd weighting factor. If the argument user_def_weights is specified, NA's are returned.
- "rhoH" a numeric value giving the effective rhoH. For weighting schemes not relying on rhoH, NA is returned.
- "user_def_weights" a named list of one or more numeric vectors returning the argument user_def_weights grouped as dimensions. The names of the list's elements are identical with those of the output "items". If the argument user_def_weights is unspecified, NA is returned.
- "score_i_heading" single character strings returning the specification of the argumnent score_i_heading.

References

Betti, G. & Verma, V. K. (1998), 'Measuring the degree of poverty in a dynamic and comparative context: a multi-dimensional approach using fuzzy set theory', Working Paper 22, Dipartimento di Metodi Quantitativi, Universit'a di Siena.

Cerioli, A. & Zani, S. (1990), 'A fuzzy approach to the measurement of poverty', in C. Dagum & M. Zenga (eds.), Income and Wealth Distribution, Inequality and Poverty, Springer Verlag, Berlin, 272-284.

Desai, M. & Shah, A. (1988), 'An econometric approach to the measurement of poverty', Oxford Economic Papers, 40(3):505-522.

Pi Alperin, M. N. & Van Kerm, P. (2009), 'mdepriv - Synthetic indicators of multiple deprivation', v2.0 (revised March 2014), CEPS/INSTEAD, Esch/Alzette, Luxembourg. http://medim.ceps.lu/stata/mdepriv_v3.pdf (2019-MM-DD).

See Also

```
vignette("mdepriv_get_started")
```

Examples

```
head(simul_data, 3) # data used for demonstration

# minimum possible specification: data & items:
mdepriv(simul_data, c("y1", "y2", "y3", "y4", "y5", "y6", "y7"))
# group items in dimensions:
mdepriv(simul_data, list(c("y1", "y2", "y3", "y4"), c("y5", "y6", "y7")))
# customized labelling of dimensions:
mdepriv(simul_data, list("Group A" = c("y1", "y2", "y3", "y4"), "Group B" = c("y5", "y6", "y7")))

# available outputs
no_dim_specified <- mdepriv(simul_data, c("y1", "y2", "y3", "y4", "y5", "y6", "y7"), output = "all")
two_dim <- mdepriv(simul_data, list(c("y1", "y2", "y3", "y4"), c("y5", "y6", "y7")), output = "all")
length(no_dim_specified)
length(two_dim)
data.frame(
```

```
  row.names = names(two_dim),
  no_or_1_dim_specified = ifelse(names(two_dim) %in% names(no_dim_specified), "X", ""),
  at_least_2_dim_specified = "X"
)
setdiff(names(two_dim), names(no_dim_specified))
# if no dimensions are specified, "summary_by_dimension" is dropped from the two output wrappers
# (output = "view" (default), output = "all")
# however, even if no dimension is specified "summary_by_dimension" is accessible
mdepriv(simul_data, c("y1", "y2", "y3", "y4", "y5", "y6", "y7"), output = "summary_by_dimension")

# apply sampling weights (3rd argument)
with_s_w <- mdepriv(simul_data, c("y1", "y4", "y5", "y6"), "sampl_weights", output = "all")
without_s_w <- mdepriv(simul_data, c("y1", "y4", "y5", "y6"), output = "all")
# return sum and specification of sampling weights if applied ...
with_s_w[c("sum_sampling_weights", "sampling_weights")]
# if not, NA's are returned:
without_s_w[c("sum_sampling_weights", "sampling_weights")]

# weighting schemes
# the default weighting scheme is "Cerioli & Zani": method = "cz"
mdepriv(simul_data, c("y1", "y2", "y3"), output = "weighting_scheme")

methods <- c("cz", "ds", "bv", "equal") # 4 standard weighting schemes available
sapply(methods,
    function(x) mdepriv(simul_data, c("y1", "y2", "y3"), method = x, output = "weighting_scheme")
      )

# alternative, more flexible ways to select (double) weighting schemes
mdepriv(simul_data, c("y1", "y2", "y3"), wa = "cz", wb = "mixed", output = "weighting_scheme")
# some of the double weighting specification are almost lookalikes of the standard weight methods
method_bv_pearson <- mdepriv(simul_data, c("y1", "y2", "y3"),
                             method = "bv", bv_corr_type = "pearson", output = "all")
method_bv_pearson$weighting_scheme
wa_wb_bv_pearson <- mdepriv(simul_data, c("y1", "y2", "y3"),
                            wa = "bv", wb = "pearson", output = "all")
wa_wb_bv_pearson$weighting_scheme
all.equal(method_bv_pearson[-1], wa_wb_bv_pearson[-1])

# either a fixed or a data driven rhoH is involved in any true double weighting scheme
# (effective single weighting schemes: method: "cs", "ds", "equal" or wb = "diagonal")
items_sel <- c("y1", "y2", "y3", "y4", "y5", "y6", "y7") # a selection of items
# data driven:
mdepriv(simul_data, items_sel, method = "bv", output = "rhoH")
mdepriv(simul_data, items_sel, wa = "cz", wb = "pearson", output = "rhoH")
# fixed:
mdepriv(simul_data, items_sel, method = "bv", rhoH = 0.3, output = "rhoH")
mdepriv(simul_data, items_sel, wa = "cz", wb = "pearson", rhoH = 0.3, output = "rhoH")

# check how weighting settings are applied:
bv <- mdepriv(simul_data, items_sel, method = "bv", output = "all")
bv[c("weighting_scheme", "wa", "wb", "rhoH")]
ds <- mdepriv(simul_data, items_sel, method = "ds", output = "all")
ds[c("weighting_scheme", "wa", "wb", "rhoH")]
```

```
equal_pearson <- mdepriv(simul_data, items_sel,
                         wa = "equal", wb = "pearson", output = "all")
equal_pearson[c("weighting_scheme", "wa", "wb", "rhoH")]
equal_pearson_rhoH_fixed <- mdepriv(simul_data, items_sel,
                                    wa = "equal", wb = "pearson", rhoH = 0.3, output = "all")
equal_pearson_rhoH_fixed[c("weighting_scheme", "wa", "wb", "rhoH")]

# pass expertise-based weights to the items
dim <- list("Group A" = c("y1", "y2", "y3"), "Group B" = c("y4", "y5", "y6"))
# 'expertise weights' structured as dimensions
w_expertise <- list(c(0.5, 0.25, 0.25), c(0.4, 0.45, 0.15))
model_expertise <- mdepriv(simul_data, items = dim,
                           user_def_weights = w_expertise, output = "all")
# check weighting settings ...
model_expertise[c("weighting_scheme", "wa", "wb", "rhoH", "user_def_weights")]
# ... wa, wb and rhoH are not involved, when expertise weights are applied,
# and therefore returned as NA's.
# user-defined names of dimensions are inherited from the argument items.

# use outputs elements
dim <- list(c("y1", "y2", "y3"), c("y4", "y5", "y6", "y7"))
model_1 <- mdepriv(simul_data, items = dim, method = "bv", output = "all")

model_1$summary_by_item
by_item_no_total <- subset(model_1$summary_by_item, Weight != 1)
barplot(Weight ~ Item, data = by_item_no_total)

model_1$summary_scores
hist(model_1$score_i,
  main = 'model: method = "bv", bv_corr_type = "mixed" (default)',
  xlab = "scores"
)

# output data ...
head(model_1$data, 3)
# ... compare to input data ...
head(simul_data, 3)
# ... only the scores have been merged to the (input) data
all.equal(model_1$data[, names(model_1$data) != "score_i"], simul_data)
# scores are twofold accessible
all.equal(model_1$score_i, model_1$data$score_i)

# re-use output of a model as arguments in another model:
dim <- list(c("y1", "y2", "y3"), c("y4", "y5", "y6", "y7"))
model_1 <- mdepriv(simul_data, items = dim, method = "bv", output = "all")
model_2 <- mdepriv(simul_data, items = model_1$items, method = "ds", output = "all")
all.equal(model_1$items, model_2$items)
# how do the scores of the 2 models differ?
plot(model_1$score_i, model_2$score_i,
  xlab = model_1$weighting_scheme, ylab = model_2$weighting_scheme,
  xlim = c(0, 1), ylim = c(0, 1),
  asp = 1, main = "same item grouping"
)
```

```
abline(0, 1, col = "red", lty = 2, lwd = 2)

# accumulating scores from different models in the output data
# this code will throw an error message with a hint on how to handle the re-use ...
# ... of 'data' output as agrument. so run it and read!
try(
model_3 <- mdepriv(model_1$data, items = model_1$items,
                   wa = "cz", wb = "mixed", output = "all")
)
model_3 <- mdepriv(model_1$data, items = model_1$items,
                   wa = "cz", wb = "mixed", output = "all",
                   score_i_heading = "score_i_model_3")
head(model_3$data, 3)

# if gathering scores from iteratered runs is the purpose it's expedient to avoid confusion ...
# ... by naming already the 1st scores column with reference to its model
model_1 <- mdepriv(simul_data, dim, method = "bv",
                   score_i_heading = "score_i_1", output = "all")
model_2 <- mdepriv(model_1$data, model_1$items, method = "ds",
                   score_i_heading = "score_i_2", output = "all")
model_3 <- mdepriv(model_2$data, model_1$items, wa = "cz", wb = "mixed",
                   score_i_heading = "score_i_3", output = "all")
head(model_3$data, 3)
```

MSNA_HC *Data Related to Living Standards in Host Communities Adjacent to the Rohingya Refugee Camps in Bangladesh, 2018*

Description

Extract from the "COX'S BAZAR HOST COMMUNITY (HC) MULTI-SECTOR NEEDS ASSESSMENT (MSNA) - HOUSEHOLD DATA", December 2018. Probability sample of 2855 households in 11 communes ("Unions") in two sub-districts ("Upazillas") in south-eastern Bangladesh. Basis for the calculation of a living standards deprivation index, on which households differ by gender of household heads and commune of residence.

Usage

```
MSNA_HC
```

Format

A data frame with 2855 obs. in 15 variables:

id Household ID: integer

sampl_weights Sampling weights: float, 11 distinct values (1 per Union), range: [0.5694934, 1.396798]

upazilla Upazilla (sub-district): two distinct values: "Teknaf", "Ukhiya": character string / factor

union Union (commune): 11 distinct values. "Baharchhara", "Nhilla", "Sabrang", "Teknaf", "Teknaf Paurashava", and "Whykong" Unions in Teknaf Upazilla, as well as "Haldia Palong", "Jalia Palong", "Palong Khali", "Raja Palong", and "Ratna Palong" Unions in Ukhiya Upazilla: character string / factor

hhh_gender Gender of household head: female or male: character string / factor

ls_1_food Food deprivation indicator, calculated from the original Food Consumption Scores: float, range [0.0669643, 1]

ls_2_livelihood Indicator of less favorable livelihoods combinations: Ordinal, rescaled to: 0.25 0.50 0.75 1.00. Of nine livelihood types that the MSNA observed, five - domestic work, non-agricultural work, fishing, small business, and remittances - are significantly associated with the scores of household food consumption. Across the sample households, the five types appeared in 23 combinations. These were mapped to a scale, with four levels, of increasingly less favorable combinations, reflected in a (nearly linear) decrease in the predicted food consumption scores. Because of that nearly linear effect, this indicator is treated as interval-level variable.

ls_3_shelter Probability of living in a worse shelter than others: Four distinct values: 0.0696, 0.3204, 0.6645, 0.9137. Household dwellings are classified by construction types, which reflect decreasing levels of comfort and value: Pucca (highest, best), Semi-pucca, Kutcha, and Jhuprie (lowest, worst). For lack of finer grading or market value data, it is assumed that within each class, half of the households are slightly worse off than the other half. Thus, over the four classes, for a given household one of the noted four distinct values is the probability of occupying a dwelling worse than the other households in the population, calculated as the sum of proportions of households in types better than its own (if any) plus half of the proportion in its own (this measure is known as the "ridit"; see https://en.wikipedia.org/wiki/Ridit_scoring).

water_1 No year-round access to improved water source: binary / 0 or 1

water_2 Problems encountered when collecting water: binary / 0 or 1

water_3 Walking to and from the water source takes more than 30 minutes: binary / 0 or 1

sanit_1 Trash visible: binary / 0 or 1

sanit_2 Faeces visible: binary / 0 or 1

sanit_3 Stagnant water visible: binary / 0 or 1

sanit_4 Household members defecate outside home: binary / 0 or 1

Details

The continuous indicators ls_1_food, ls_2_livelihood and ls_3_shelter measure deprivation in three living standards (ls) components, each one corresponding to a particular humanitarian sector. The items with prefixes water_ and sanit_ are observed in the water, respectively sanitation sub-sectors of the Water, Sanitation and Hygiene (WASH) sector.

This dataset has been chosen to demonstrate a two-level deprivation model using the function mdepriv, with the Betti-Verma double weighting rule operating at both levels. At the lower level, the seven binary WASH items are aggregated to a continuous WASH deprivation indicator, ls_4_WASH. To equalize subsector contributions, water_ and sanit_ items are grouped in different dimensions. At the higher level, ls_4_WASH is combined with the other three ls-indicators and aggregated to a living standards deprivation index.

In this humanitarian context, there is no basis to consider one or the other water or sanitation problems more or less important on the basis of their different prevalence. Therefore, in aggregating the seven binary water_ and sanit_ items, `mdepriv` takes the arguments `wa = "equal"`. Moreover, `wb = "mixed"` has the effect to reduce weights on more redundant items (`wb = "diagonal"` would be neutral to redundancy; and `wb = "pearson"` would underrate the strength of correlations between binary items). This model returns the deprivation scores in the variable "ls_4_WASH".

At the second level, in the aggregation of the four continuous ls-indicators (ls_1_food, ls_2_livelihood, ls_3_shelter, ls_4_WASH), the default Betti-Verma method is used, by setting `method = "bv"` in the function `mdepriv`. This activates both mechanisms of the double-weighting scheme - rewarding more discriminating indicators with higher weights, and penalizing redundant ones with lower weights.

Source

ISCG (Inter Sector Coordination Group) / REACH / ACAPS-NPM, Cox Bazar, Bangladesh, December 2018, published with the provision: "This tool is made available to all staff and partners of the ISCG, and to the general public as a support tool for strategy and programming in the humanitarian response in Bangladesh and other related purposes only. Extracts from the information from this tool may be reviewed, reproduced or translated for the above-mentioned purposes, but are not for sale or for use in conjunction with commercial purposes." Original food consumptions scores calculated by REACH. Indicators ls_1_food, ls_2_livelihood and ls_3_shelter calculated by Aldo Benini, from a cleaned dataset compiled by Sudeep Shrestha, ACAPS.

`simul_data` *Demonstration Dataset*

Description

A dataset containing 100 simulated observations for the purpose of demonstrating the workings of the functions `mdepriv` and `corr_mat`.

Usage

```
simul_data
```

Format

A data frame with 100 observations and 9 variables:

id ID integer 1, ... , 100: a record identifier

y1 binary data: 0 or 1: deprivation item

y2 binary data: 0 or 1: deprivation item

y3 binary data: 0 or 1: deprivation item

y4 ordinal data, rescaled to the range [0, 1]: 0, 0.2, 0.4, 0.6, 0.8, 1: deprivation item

y5 continuous data: range [0,1]: deprivation item

y6 continuous data: range [0,1]: deprivation item

y7 continuous data: range [0,1]: deprivation item

sampl_weights continuous data: sampling weights

Index