# Package 'mpindex'

July 23, 2025

**Type** Package

**Title** Multidimensional Poverty Index (MPI)

**Version** 0.2.1

**Author** Bhas Abdulsamad [aut, cre, cph] (ORCID:
<https://orcid.org/0009-0002-5891-8124>)

**Maintainer** Bhas Abdulsamad <aeabdulsamad@gmail.com>

**Description** A set of easy-to-use functions for computing the Multidimensional Poverty Index (MPI).

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** dplyr, tidyr, stringr, jsonlite, purrr, tibble, openxlsx

**Suggests** rlang, testthat (>= 3.0.0), lifecycle, knitr, rmarkdown, gt

**Config/testthat/edition** 3

**RoxygenNote** 7.2.3

**BugReports** <https://github.com/yng-me/mpindex/issues>

**URL** <https://github.com/yng-me/mpindex>,
<https://yng-me.github.io/mpindex/>

**VignetteBuilder** knitr

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2024-01-09 09:50:02 UTC

## Contents

---

compute_mpi                      *Compute Multidimensional Poverty Index (MPI)*

---

### Description

This function uses the Alkire-Foster (AF) counting method developed by Sabina Alkire and James
Foster. It requires a deprivation profile created using the ([define_deprivation](#)) fuction containing
all indicators defined in the specification files.

### Usage

```
compute_mpi(
  .data,
  .deprivation_profile,
  ...,
  .mpi_specs = getOption("mpi_specs"),
  .include_deprivation_matrix = TRUE,
  .generate_output = FALSE,
  .formatted_output = TRUE,
  .mpi_output_filename = NULL,
  .include_table_summary = TRUE,
  .include_specs = FALSE
)
```

### Arguments

.data                A tidy data frame where each observation is the unit of analysis defined in
                     [define_mpi_specs](#).

.deprivation_profile
                     list of deprivation profile created using [define_deprivation](#).

...                  Grouping columns (supports tidyselect), e.g. area (country, urbanity, region,
                     province), sex, ethnic group, etc.

.mpi_specs           MPI specifications defined in [define_mpi_specs](#).

.include_deprivation_matrix
                     Whether to include deprivation matrix in the output.

.generate_output
                     Whether to generate an output (Excel file) as side effect.

.formatted_output
                     NOT YET IMPLEMENTED. Whether formatting is to be applied to the output.

.mpi_output_filename
>    Output filename.

.include_table_summary
>    NOT YET IMPLEMENTED. Whether to include summary information in the
>    generated output.

.include_specs    NOT YET IMPLEMENTED. Whether to include MPI specification in the gen-
>    erated output.

### Value

Returns list of objects: index (the MPI), contribution (contribution by dimension), headcount_ratio
(censored and uncensored), and deprivation_matrix (censored and uncensored). If poverty_cutoffs
defined in define_mpi_specs contain more than one (1) value, index and contribution object
will output each cutoff in a separate table.

### References

Alkire-Foster Method
How to Apply the Alkire-Foster Method

### See Also

define_mpi_specs, define_deprivation, save_mpi

### Examples

```
# ----------------------------------
# Load MPI specs from the built-in specs file
specs_file <- system.file("extdata", "global-mpi-specs.csv", package = "mpindex")
mpi_specs <- define_mpi_specs(specs_file, .uid = 'uuid')

# ----------------------------------
# Create an empty list to store deprivation profile for each indicator
deprivation_profile <- list()

deprivation_profile$nutrition <- df_household_roster |>
 define_deprivation(
   .indicator = nutrition,
   .cutoff = undernourished == 1 & age < 70,
   .collapse = TRUE
 )
deprivation_profile$child_mortality <- df_household |>
 define_deprivation(
   .indicator = child_mortality,
   .cutoff = with_child_died == 1
 )
deprivation_profile$year_schooling <- df_household_roster |>
 define_deprivation(
   .indicator = year_schooling,
   .cutoff = completed_6yrs_schooling == 2,
   .collapse = TRUE
```

```
  )
  deprivation_profile$school_attendance <- df_household_roster |>
    define_deprivation(
      .indicator = school_attendance,
      .cutoff = attending_school == 2 & age %in% c(5:24),
      .collapse = TRUE
    )
  deprivation_profile$cooking_fuel <- df_household |>
    define_deprivation(
      .indicator = cooking_fuel,
      .cutoff = cooking_fuel %in% c(4:6, 9)
    )
  deprivation_profile$sanitation <- df_household |>
    define_deprivation(
      .indicator = sanitation,
      .cutoff = toilet > 1
    )
  deprivation_profile$drinking_water <- df_household |>
    define_deprivation(
      .indicator = drinking_water,
      .cutoff = drinking_water == 2
    )
  deprivation_profile$electricity <- df_household |>
    define_deprivation(
      .indicator = electricity,
      .cutoff = electricity == 2
    )
  deprivation_profile$housing <- df_household |>
    define_deprivation(
      .indicator = housing,
      .cutoff = roof %in% c(5, 7, 9) | walls %in% c(5, 8, 9, 99) == 2 | floor %in% c(5, 6, 9)
    )
  deprivation_profile$assets <- df_household |>
    dplyr::mutate_at(dplyr::vars(dplyr::starts_with('asset_')), ~ dplyr::if_else(. > 0, 1L, 0L)) |>
    dplyr::mutate(
      asset_phone = dplyr::if_else(
        (asset_telephone + asset_mobile_phone) > 0,
        1L,
        0L
      )
    ) |>
    dplyr::mutate(
      with_hh_conveniences = (
        asset_tv + asset_phone + asset_computer +
          asset_animal_cart + asset_bicycle +
          asset_motorcycle + asset_refrigerator) > 1,
      with_mobility_assets = (asset_car + asset_truck) > 0
    ) |>
    define_deprivation(
      .indicator = assets,
      .cutoff = !(with_hh_conveniences & with_mobility_assets)
    )
```

```
# ---------------------------------
# Compute the MPI
mpi_result <- df_household |>
  compute_mpi(deprivation_profile)

# ---------------------------------
# You may also save your output into an Excel file
## Not run:
save_mpi(mpi_result, .filename = 'MPI Sample Output')

## End(Not run)
```

---

define_deprivation          *Define deprivation cutoffs*

---

### Description

A deprivation cutoff must be set for each indicator defined in the MPI specifications. This step establishes the first cutoff in the methodology where every person/household (defined as the unit of analysis) can be identified as deprived or non-deprived with respect to each indicator.

For each indicator, 0 will be used to indicate "not deprived", 1 if deprived, and NA if missing or non-response. Additional column containing the product of the value of the indicator obtained and its corresponding weight will also be computed for convenience.

### Usage

```
define_deprivation(
  .data,
  .indicator,
  .cutoff,
  .mpi_specs = getOption("mpi_specs"),
  .collapse = FALSE,
  .set_na_equal_to = 0,
  .collapse_condition = NULL
)
```

### Arguments

| | |
|---|---|
| .data | A data frame or tibble |
| .indicator | Name of indicator defined in MPI specs (must exactly match the specs). |
| .cutoff | A conditional logic that defines the poverty line to determine whether deprived or not. |
| .mpi_specs | MPI specifications defined in define_mpi_specs. |
| .collapse | A boolean indicating whether to collapse the data frame or not. This is useful, for instance, if the original data where the .cutoff argument above applies to an individual person but your unit of analysis in household. |

.set_na_equal_to

> Coerce value from NA to either 0 (not deprived) or 1 (deprived). Default is 0.

.collapse_condition

> NOT YET FULLY IMPLEMENTED. ONLY WORKS WITH DEFAULT. A condition when .collapse is set to TRUE. If NULL, max() will be used as default.

### Value

A data frame of deprivation value for the indicator (.*_unweighted): 0 for "not deprived", 1 for deprived, and NA for missing and non-response; and product of .*_unweighted and its corresponding weight (.*_weighted).

### References

[How to Apply the Alkire-Foster Method](#)

### See Also

[define_mpi_specs](#)

### Examples

```
# Use sample specs file included in the package
specs_file <- system.file(
 "extdata",
 "global-mpi-specs.csv",
 package = "mpindex"
)
specs <- define_mpi_specs(specs_file, .uid = 'uuid')

# Using built-in dataset
df_household |>
  define_deprivation(
    .indicator = drinking_water,
    .cutoff = drinking_water == 2
  )

df_household_roster |>
  define_deprivation(
    .indicator = school_attendance,
    .cutoff = attending_school == 2,
    .collapse = TRUE
  )
```

| define_mpi_specs | *Define MPI specifications: dimensions, indicators, and weights* |
|---|---|

## Description

Use to define MPI dimensions, indicators and its corresponding weights using any of the accessible
file types: `.xlsx` (Excel), `.json`, `.csv`, or `.txt` (TSV). You can also set the poverty cutoff or list of
poverty cutoffs (to achieve gradient list of MPIs) that will be used in the computation of MPI.

## Usage

```
define_mpi_specs(
  .mpi_specs_file = NULL,
  .indicators = NULL,
  .poverty_cutoffs = 1/3,
  .unit_of_analysis = NULL,
  .aggregation = NULL,
  .uid = NULL,
  .source_of_data = NULL,
  .names_separator = ">",
  .save_as_global_options = TRUE
)
```

## Arguments

`.mpi_specs_file`

Accepts `.xlsx` (Excel), `.json`, `.csv`, or `.txt` (TSV) file format. This file should
contain the following columns/variables: `Dimension`, `Indicator`, `Variable`,
`Weight`, and `Description` (optional). See example below.

`.indicators` A data frame of MPI indicators. Useful if prefer define your indicators instead
of using an external file.

`.poverty_cutoffs`

Accepts single value or a vector of poverty cutoffs. This parameter (usually
denoted by k) reflects the minimum level of deprivations or deprivation score an
individual or household must be suffering simultaneously to be considered poor.
See example below.

`.unit_of_analysis`

e.g. `individuals`, `families`, `households`, or `communities`. Default value is
`NULL`.

`.aggregation` Column name in the dataset that defines an aggregation level.

`.uid` Column name containing unique ID of the dataset which defines the lowest level
of disaggregation (usually unit of analysis).

`.source_of_data`

Source of data used in the computation. This will be used in the footnote of the
table when generating an output.

```
.names_separator
```
                            **[Deprecated]** Column separator that defines the hierarchy of the column header.
```
.save_as_global_options
```
                            Whether to save the specs globally. Equivalent to invoking `options()`.

### Value

MPI specifications data frame required in [compute_mpi](#) function. As as side effect, a global option named 'mpi_specs' will be saved for efficiency. See 'getOption('mpi_specs')'.

### See Also

[compute_mpi](#)

### Examples

```
# Use sample specs file included in the package
specs_file <- system.file(
  "extdata",
  "global-mpi-specs.csv",
  package = "mpindex"
)
# To see other sample specs file (with different supported file format)
system.file("extdata", package = "mpindex") |>
  list.files()
```

---

df_household                *Sample dataset of households*

---

### Description

This is a synthetic dataset containing household information primarily used for demonstration purposes on how to use the mpindex package.

### Usage

```
df_household
```

### Format

A tibble with 198 rows and 21 variables:

**uuid** Unique ID

**class** Urbanity: `Rural` or `Urban`

**drinking_water** Acess to drinking water: 1 - improved; 2 - unimproved

**toilet** Service level of toilet or sanitation facility: 1 - basic; 2 - limited; 3 - unimproved; 4 - open defecation

**with_child_died**  With at least one (1) child died in the last five (5) years: 1 - with child died; 2 - without child died

**roof**  Main construction material of the roof: 1 - galvanized iron/aluminum; 2 - concrete/clay tile; 3 - half galvanized iron and half concrete; 4 - wood/bamboo; 5 - cogon/nipa/anahaw; 6 - asbestos; 7 - makeshift/salvaged/improvised materials; 9 - other construction material

**walls**  Main construction material of the outer walls: 1 - concrete/brick/stone; 2 - wood; 3 - half concrete/brick/stone and half wood; 4 - Galvanized iron/aluminum; 5 - bamboo/sawali/cogon/nipa; 6 - asbestos; 7 - glass; 8 - makeshift/salvaged/improvised materials; 9 - none; 10 - concrete hollow blocks; 11 - concrete hollow blocks/wood; 12 - shear walls; 99 - other construction material

**floor**  Main construction material of the floor: 1 - concrete; 2 - wood; 3 - coconut lumber; 4 - bamboo; 5 - earth/sand/mud; 6 - makeshift/salvaged/improvised materials; 9 - other construction material

**electricity**  Access to electricity: 1 - with access to electricity; 2 - without access to electricity

**cooking_fuel**  Fuel use for cooking: 1 - electricity; 2 - kerosene (gaas); 3 - liquified petroleum gas (LPG); 4 - charcoal; 5 - wood; 6 - none; 9 - other cooking fuel such as dung, agricultural crop, or shrubs

**asset_radio**  Number of working radio owned by the household

**asset_tv**  Number of working television owned by the household

**asset_telephone**  Number of working telephone owned by the household

**asset_mobile_phone**  Number of working mobile phone owned by the household

**asset_computer**  Number of working computer owned by the household

**asset_animal_cart**  Number of animal carts owned by the household

**asset_bicycle**  Number of bicycle owned by the household

**asset_motorcycle**  Number of motorcylce owned by the household

**asset_refrigerator**  Number of working refrigerator owned by the household

**asset_car**  Number of car owned by the household

**asset_truck**  Number of trucks owned by the household

## See Also

df_household_roster

## Examples

```
df_household
```

df_household_roster          *Sample dataset of household members*

### Description

This dataset contains a many-to-one relationship with the df_household dataset. Hence, you can apply joins using the uuid.

### Usage

df_household_roster

### Format

A tibble with 905 rows and 8 variables:

**uuid**  Unique ID

**line_number**  Number identifier for each member within the household

**class**  Urbanity: Rural or Urban

**sex**  Sex of the household member

**age**  Age of the household member

**attending_school**  Whether the household member (aged 5-24 years old) is currently attending school: 1 - currently attending; 2 - currently not attending

**completed_6yrs_schooling**  Whether completed at least six (6) years of schooling: 1 - completed; 2 -not completed

**undernourished**  Whether the household member (aged below 70 years old) is undernourished: 1 - undernourished; 2 - not undernourished

### See Also

df_household

### Examples

df_household_roster

---

save_mpi                          *Save MPI output*

---

### Description

Save the MPI output into an Excel file format.

### Usage

```
save_mpi(
  .mpi_output,
  .mpi_specs = getOption("mpi_specs"),
  .filename = NULL,
  .formatted_output = TRUE,
  .include_table_summary = TRUE,
  .include_specs = FALSE
)
```

### Arguments

| | |
|---|---|
| `.mpi_output` | An object derived from [compute_mpi](). |
| `.mpi_specs` | MPI specifications defined in [define_mpi_specs](). |
| `.filename` | Output filename |
| `.formatted_output` | |
| | Whether formatting is to be applied to the output. |
| `.include_table_summary` | |
| | NOT YET IMPLEMENTED. Whether to include summary information in the generated output. |
| `.include_specs` | Whether to include MPI specification in the generated output. |

### Value

Returns the file location of the output generated.

### Examples

```
## Not run:
# It requires an MPI output (list type) in the first argument
save_mpi(mpi_result, .filename = "MPI Sample Output")

## End(Not run)
```

---

use_global_mpi_specs            *Use Global MPI specification*

---

### Description

Use built-in specification file for Global MPI.

### Usage

```
use_global_mpi_specs(...)
```

### Arguments

| | |
|---|---|
| `...` | Accepts all arguments in `define_mpi_specs` |

### Value

Global MPI specs

### Examples

```
use_global_mpi_specs()
```

# Index