# Package 'myrror'

May 9, 2026

**Title** Compare Two Data Frames and Summarize Differences

**Version** 0.1.2

**Description** Tools for systematic comparison of data frames, offering functionality to identify, quantify, and extract differences. Provides functions with user-friendly and interactive console output for immediate analysis, while also offering options to export differences as structured data frames that can be easily integrated into existing workflows.

**License** MIT + file LICENSE

**URL** https://pip-technical-team.github.io/myrror/, https://github.com/PIP-Technical-Team/myrror

**Depends** R (>= 4.3.0)

**Imports** cli (>= 3.6.2), collapse, data.table (>= 1.15.4), digest, joyn (>= 0.3.0), rlang, utils

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), withr

**VignetteBuilder** knitr

**Config/Needs/website** rmarkdown, tidyverse, gapminder, DT

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**BugReports** https://github.com/PIP-Technical-Team/myrror/issues

**NeedsCompilation** no

**Author** Giorgia Cecchinato [aut],
R.Andres Castaneda [aut, cre],
Rossana Tatulli [aut],
Global Poverty and Inequality Data Team World Bank [cph]

**Maintainer** R.Andres Castaneda <acastanedaa@worldbank.org>

**Repository** CRAN

**Date/Publication** 2026-01-06 11:20:02 UTC

# Contents



---

`compare_type` *Function to compare type of variables of matched data frames.*

---

## Description

This function compares the types of the columns in the two data frames.

## Usage

```
compare_type(
  dfx = NULL,
  dfy = NULL,
  myrror_object = NULL,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  output = c("full", "simple", "silent"),
```

```
  interactive = getOption("myrror.interactive"),
  verbose = getOption("myrror.verbose")
)
```

## Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| myrror_object | myrror object from create_myrror_object |
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| output | character: one of "full" (returns a myrror_object), "simple" (returns a dataframe), "silent" (invisible object returned). |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |
| verbose | logical: If TRUE additional information will be displayed. |

## Value

Depending on output parameter:

- "full": myrror object with compare_type slot containing a data.table of column class comparisons
- "simple": data.table with columns: variable, class_x, class_y, same_class
- "silent": invisibly returns myrror object (same as "full")

Returns NULL if no differences are found and output = "simple".

## Examples

```
# 1. Standard report, myrror_object output:
compare_type(survey_data, survey_data_2, by=c('country', 'year'))

# 2. Simple output, data.table output:
compare_type(survey_data, survey_data_2, by=c('country', 'year'),
             output = 'simple')

# 3. Toggle interactvity:
compare_type(survey_data, survey_data_2, by=c('country', 'year'),
             interactive = FALSE)

# 4. Different keys (see also ?myrror):
compare_type(survey_data, survey_data_2_cap,
             by.x = c('country', 'year'), by.y = c('COUNTRY', 'YEAR'))

# 5. Using existing myrror object created by myrror():
myrror(survey_data, survey_data_2, by=c('country', 'year'))
compare_type()
```

compare_values *Function to compare values of matched data frames.*

## Description

Function to compare values of matched data frames.

## Usage

```
compare_values(
  dfx = NULL,
  dfy = NULL,
  myrror_object = NULL,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  output = c("full", "simple", "silent"),
  interactive = getOption("myrror.interactive"),
  verbose = getOption("myrror.verbose"),
  tolerance = getOption("myrror.tolerance")
)
```

## Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| myrror_object | myrror object from create_myrror_object |
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| output | character: one of "full" (returns a myrror_object), "simple" (returns a dataframe), "silent" (invisible object returned). |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |
| verbose | logical: If TRUE additional information will be displayed. |
| tolerance | numeric, default to 1e-7. |

## Value

Depending on output parameter:

- "full": myrror object with compare_values slot containing a summary tibble of value differences
- "simple": tibble with columns: variable, change_in_value, na_to_value, value_to_na (counts)
- "silent": invisibly returns myrror object (same as "full")

Returns NULL if no differences are found and output = "simple".

## Examples

```
# 1. Standard report, myrror_object output:
compare_values(survey_data, survey_data_2, by=c('country', 'year'))

# 2. Simple output, list of data.tables output:
compare_values(survey_data, survey_data_2, by=c('country', 'year'),
               output = 'simple')

# 3. Toggle tolerance:
compare_values(survey_data, survey_data_2, by=c('country', 'year'),
               tolerance = 1e-5)

# 4. Toggle interactvity:
compare_values(survey_data, survey_data_2, by=c('country', 'year'),
               interactive = FALSE)

# 5. Different keys (see also ?myrror):
compare_values(survey_data, survey_data_2_cap,
               by.x = c('country', 'year'), by.y = c('COUNTRY', 'YEAR'))

# 6. Using existing myrror object created by myrror():
myrror(survey_data, survey_data_2, by=c('country', 'year'))
compare_values()
```

---

# create_myrror_object *Creates a myrror object for comparing two data frames*

---

## Description

This function constructs a myrror object by comparing two data frames. It handles the preparation, validation, and joining of datasets, identifies matching and non-matching observations, and performs column pairing for comparison. The function supports various join types (1:1, 1:m, m:1) and provides detailed reports on the comparison results.

## Usage

```
create_myrror_object(
  dfx,
  dfy,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  factor_to_char = TRUE,
  verbose = getOption("myrror.verbose"),
  interactive = getOption("myrror.interactive")
)
```

### Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| factor_to_char | TRUE or FALSE, default to TRUE. |
| verbose | logical: If TRUE additional information will be displayed. |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |

### Value

An object of class "myrror" containing comparison results, dataset information, and various reports on matching/non-matching observations.

### Examples

```
# convert rownames of mtcars to a column
mtcars2 <- mtcars
mtcars2$car_name <- rownames(mtcars2)
rownames(mtcars2) <- NULL
# modify mtcars2 slightly by remove one row and changing one value
mtcars3 <- mtcars2[-1, ]
mtcars3$mpg[1] <- mtcars3$mpg[1] + 1

mo <- create_myrror_object(mtcars2, mtcars3, by = "car_name")
mo
```

---

## extract_diff_rows *Extract Different Rows Function to extract missing or new rows from comparing two data frames.*

---

### Description

Extract Different Rows Function to extract missing or new rows from comparing two data frames.

### Usage

```
extract_diff_rows(
  dfx = NULL,
  dfy = NULL,
  myrror_object = NULL,
  by = NULL,
  by.x = NULL,
```

```
  by.y = NULL,
  output = c("simple", "full", "silent"),
  tolerance = 0.0000001,
  verbose = getOption("myrror.verbose"),
  interactive = getOption("myrror.interactive")
)
```

## Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| myrror_object | myrror object from create_myrror_object |
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| output | character: one of "full", "simple", "silent". |
| tolerance | numeric, default to 1e-7. |
| verbose | logical: If TRUE additional information will be displayed. |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |

## Value

Depending on output parameter:

- "full": myrror object with extract_diff_rows slot containing a data.table of non-matching rows
- "simple": data.table with columns: df (indicating 'dfx' or 'dfy'), keys, and all other columns. Contains rows that exist in only one dataset
- "silent": invisibly returns myrror object (same as "full")

Returns NULL if no row differences are found and output = "simple".

## Examples

```
# 1. Standard report, after running myrror() or compare_values():
myrror(survey_data, survey_data_2, by=c('country', 'year'))
extract_diff_rows()

# 2. Standard report, with new data:
extract_diff_rows(survey_data, survey_data_2, by=c('country', 'year'))


# 3. Toggle tolerance:
extract_diff_rows(survey_data, survey_data_2, by=c('country', 'year'),
                    tolerance = 1e-5)
```

`extract_diff_table` *Extract Different Values in Table Format Function to extract rows with different values between two data frames.*

## Description

Extract Different Values in Table Format Function to extract rows with different values between two data frames.

## Usage

```
extract_diff_table(
  dfx = NULL,
  dfy = NULL,
  myrror_object = NULL,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  output = c("simple", "full", "silent"),
  tolerance = 0.0000001,
  verbose = getOption("myrror.verbose"),
  interactive = getOption("myrror.interactive")
)
```

## Arguments

| | |
|---|---|
| `dfx` | a non-empty data.frame. |
| `dfy` | a non-empty data.frame. |
| `myrror_object` | myrror object from create_myrror_object |
| `by` | character, key to be used for dfx and dfy. |
| `by.x` | character, key to be used for dfx. |
| `by.y` | character, key to be used for dfy. |
| `output` | character: one of "full", "simple", "silent". |
| `tolerance` | numeric, default to 1e-7. |
| `verbose` | logical: If `TRUE` additional information will be displayed. |
| `interactive` | logical: If `TRUE`, print S3 method for myrror objects displays by chunks. If `FALSE`, everything will be printed at once. |

## Value

Depending on `output` parameter:

- `"full"`: myrror object with `extract_diff_values` slot containing a list with `diff_list` and `diff_table`

- "simple": data.table with all observations where at least one value differs. Contains columns: diff, variable, indexes, keys, and all compared variables with .x/.y suffixes
- "silent": invisibly returns myrror object (same as "full")

Returns NULL if no differences are found and output = "simple".

## Examples

```
# 1. Standard report, after running myrror() or compare_values():
myrror(survey_data, survey_data_2, by=c('country', 'year'))
extract_diff_table()

# 2. Standard report, with new data:
extract_diff_table(survey_data, survey_data_2, by=c('country', 'year'))

# 3. Toggle tolerance:
extract_diff_table(survey_data, survey_data_2, by=c('country', 'year'),
                   tolerance = 1e-5)
```

---

# extract_diff_values *Extract Different Values Function to extract rows with different values between two data frames.*

---

## Description

Extract Different Values Function to extract rows with different values between two data frames.

## Usage

```
extract_diff_values(
  dfx = NULL,
  dfy = NULL,
  myrror_object = NULL,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  output = c("simple", "full", "silent"),
  tolerance = 0.0000001,
  verbose = getOption("myrror.verbose"),
  interactive = getOption("myrror.interactive")
)
```

## Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| myrror_object | myrror object from create_myrror_object |

| | |
|---|---|
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| output | character: one of "full", "simple", "silent". |
| tolerance | numeric, default to 1e-7. |
| verbose | logical: If TRUE additional information will be displayed. |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |

## Value

Depending on output parameter:

- "full": myrror object with extract_diff_values slot containing a list with diff_list and diff_table
- "simple": named list of data.tables, one per variable with differences. Each table contains columns: diff, indexes, keys, variable.x, variable.y
- "silent": invisibly returns myrror object (same as "full")

Returns NULL if no differences are found and output = "simple".

## Examples

```
# 1. Standard report, after running myrror() or compare_values():
myrror(survey_data, survey_data_2, by=c('country', 'year'))
extract_diff_values()

# 2. Standard report, with new data:
extract_diff_values(survey_data, survey_data_2, by=c('country', 'year'))

# 3. Toggle tolerance:
extract_diff_values(survey_data, survey_data_2, by=c('country', 'year'),
                    tolerance = 1e-5)
```

---

# iris_var1 *Iris Dataset Variation 1*

---

## Description

This dataset variation includes:

- Additional rows by duplicating the first 5 rows.
- A new column Petal.Area calculated from Petal.Length and Petal.Width.
- Introduction of NA values in Sepal.Length.

### Usage

```
iris_var1
```

### Format

A `data.frame` with 155 rows and 6 variables:

**Sepal.Length** Numeric, Sepal length in cm, with some NA values.

**Sepal.Width** Numeric, Sepal width in cm.

**Petal.Length** Numeric, Petal length in cm.

**Petal.Width** Numeric, Petal width in cm.

**Species** Factor with levels: "setosa","versicolor","virginica".

**Petal.Area** Numeric, calculated as `Petal.Length * Petal.Width`.

### Details

It tests the handling of extended datasets, new calculated fields, and missing values.

### Source

Modified `iris` dataset.

---

## `iris_var2` *Iris Dataset Variation 2*

---

### Description

This dataset variation includes:

- NaN values in the `Sepal.Width` column.
- Random adjustments to `Sepal.Length` to create a range of different values.
- A shuffled order of rows to test comparison without reliance on row order.

### Usage

```
iris_var2
```

### Format

A data frame with 150 rows and 5 variables, row order shuffled:

**Sepal.Length** Numeric, Sepal length in cm, modified by adding random values.

**Sepal.Width** Numeric, Sepal width in cm, with some NaN values.

**Petal.Length** Numeric, Petal length in cm.

**Petal.Width** Numeric, Petal width in cm.

**Species** Factor with levels: "setosa","versicolor","virginica".

## Details

It is designed to test the handling of NaN values, comparison of numeric differences, and insensitivity to row order.

## Source

Modified iris dataset.

---

# iris_var3 *Iris Dataset Variation 3*

---

## Description

This dataset variation includes:

- Column name changes (e.g., Sepal.Length to SL).
- Conversion of numeric to character type for the Sepal.Length (now SL) column.
- An NA value introduced into the SL column.

## Usage

```
iris_var3
```

## Format

A data.frame with 150 rows and 5 variables:

**SL** Character, originally numeric, with one NA value.

**SW** Numeric, Sepal width in cm.

**PL** Numeric, Petal length in cm.

**PW** Numeric, Petal width in cm.

**Species** Factor with levels: "setosa","versicolor","virginica".

## Details

This variation tests the package's ability to correctly identify and handle column renaming, type conversion, and missing values.

## Source

Modified iris dataset.

`iris_var4` *Iris Dataset Variation 4*

## Description

This dataset variation includes:

- Uppercase transformation of `Species` factor levels.
- Duplicated rows (first 10 rows repeated).
- An altered scale for `Petal.Width` (values multiplied by 10).

## Usage

```
iris_var4
```

## Format

A `data.frame` with 160 rows and 5 variables:

**Sepal.Length** Numeric, length in cm.

**Sepal.Width** Numeric, width in cm.

**Petal.Length** Numeric, length in cm.

**Petal.Width** Numeric, width in cm, values scaled by a factor of 10.

**Species** Factor with levels modified to uppercase: "SETOSA", "VERSICOLOR", "VIRGINICA".

## Details

Designed to test handling of categorical variable level modifications, duplicate rows, and numeric scale adjustments.

## Source

Modified `iris` dataset.

## iris_var5 *Iris Dataset Variation 5*

### Description

This dataset variation includes:

- Column with different type: `Sepal.Length` (character).
- Column with different values: `Sepal.Length` (1 modified value).

### Usage

```
iris_var5
```

### Format

A `data.frame` with 160 rows and 5 variables:

**Sepal.Length** Charactert, length in cm.
**Sepal.Width** Numeric, width in cm.
**Petal.Length** Numeric, length in cm.
**Petal.Width** Numeric, Petal width in cm.
**Species** Factor with levels: "setosa","versicolor","virginica".

### Source

Modified `iris` dataset.

## iris_var6 *Iris Dataset Variation 6*

### Description

Iris Dataset Variation 6

### Usage

```
iris_var6
```

### Format

A `data.frame` with 146 rows and 5 variables:

**Sepal.Length** Numeric, Sepal length in cm.
**Sepal.Width** Numeric, Sepal width in cm.
**Petal.Length** Numeric, Petal length in cm.
**Petal.Width** Numeric, Petal width in cm.
**Species** Factor with levels: "setosa","versicolor","virginica".

### Source

Modified `iris` dataset.

---

`iris_var7` *Iris Dataset Variation 7*

---

### Description

Iris Dataset Variation 7

### Usage

```
iris_var7
```

### Format

A `data.frame` with 146 rows and 5 variables:

**Sepal.Length** Numeric, Sepal length in cm.

**Sepal.Width** Numeric, Sepal width in cm.

**Petal.Length** Numeric, Petal length in cm.

**Petal.Width** Numeric, Petal width in cm.

**Species** Factor with levels: "setosa","versicolor","virginica".

### Source

Modified `iris` dataset.

---

`myrror` *myrror: Compare Data Frames*

---

### Description

`myrror` provides tools for comparing data frames, identifying differences, and extracting summary tables or lists of differences.

## Usage

```
myrror(
  dfx,
  dfy,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  compare_type = TRUE,
  compare_values = TRUE,
  extract_diff_values = TRUE,
  factor_to_char = TRUE,
  interactive = getOption("myrror.interactive"),
  verbose = getOption("myrror.verbose"),
  tolerance = getOption("myrror.tolerance")
)
```

## Arguments

| | |
|---|---|
| dfx | a non-empty data.frame. |
| dfy | a non-empty data.frame. |
| by | character, key to be used for dfx and dfy. |
| by.x | character, key to be used for dfx. |
| by.y | character, key to be used for dfy. |
| compare_type | TRUE or FALSE, default to TRUE. |
| compare_values | TRUE or FALSE, default to TRUE. |
| extract_diff_values | TRUE or FALSE, default to TRUE. |
| factor_to_char | TRUE or FALSE, default to TRUE. |
| interactive | logical: If TRUE, print S3 method for myrror objects displays by chunks. If FALSE, everything will be printed at once. |
| verbose | logical: If TRUE, print messages. |
| tolerance | numeric, default to 1e-7. |

## Value

Object of class "myrror" containing:

- name_dfx, name_dfy: Names of input data frames
- prepared_dfx, prepared_dfy: Prepared versions of input data frames
- set_by.x, set_by.y: Keys used for comparison
- datasets_report: Characteristics of input datasets (rows, columns)
- match_type: Type of join relationship ("1:1", "1:m", "m:1")
- merged_data_report: Information about matched and unmatched data
- pairs: Column pairing information

- `compare_type`: Results from type comparison (if enabled)
- `compare_values`: Results from value comparison (if enabled)
- `extract_diff_values`: Extracted differences (if enabled)
- `interactive`: Whether interactive mode is enabled

Returns `NULL` invisibly if the two datasets are identical.

## Author(s)

**Maintainer**: R.Andres Castaneda <acastanedaa@worldbank.org>

Authors:

- Giorgia Cecchinato <gcecchinato@worldbank.org>
- Rossana Tatulli <rtatulli@worldbank.org>

Other contributors:

- Global Poverty and Inequality Data Team World Bank [copyright holder]

## References

https://pip-technical-team.github.io/myrror/

## See Also

Useful links:

- https://pip-technical-team.github.io/myrror/
- https://github.com/PIP-Technical-Team/myrror
- Report bugs at https://github.com/PIP-Technical-Team/myrror/issues

## Examples

```
# 1. Specifying by, by.x or by.y:
myrror(survey_data, survey_data_2, by=c('country', 'year'))

## These are equivalent:
myrror(survey_data, survey_data_2_cap, by.x=c('country', 'year'), by.y = c('COUNTRY', 'YEAR'))
myrror(survey_data, survey_data_2_cap, by=c('country' = 'COUNTRY', 'year' = 'YEAR'))

# 2. Turn off interactivity:
myrror(survey_data, survey_data_2, by=c('country', 'year'), interactive = FALSE)

# 3. Turn off factor_to_char (it will treat factors as factors):
myrror(survey_data, survey_data_2, by=c('country', 'year'), factor_to_char = FALSE)

# 4. Turn off compare_type:
myrror(survey_data, survey_data_2, by=c('country', 'year'), compare_type = FALSE)
## Same can be done for compare_values and extract_diff_values.
```

```
# 5. Set tolerance:
myrror(survey_data, survey_data_2, by=c('country', 'year'), tolerance = 1e-5)
```

---

`print.myrror` *Print method for Myrror object.*

---

## Description

Print method for Myrror object.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | an object of class 'myrror_object' |
| `...` | additional arguments |

## Value

Invisibly returns the myrror object `x`. Called for side effects (printing comparison report to console).

## Examples

```
# Create example datasets
dfx <- data.frame(id = 1:5,
                  name = c("A", "B", "C", "D", "E"),
                  value = c(10, 20, 30, 40, 50))

dfy <- data.frame(id = 1:6,
                  name = c("A", "B", "C", "D", "E", "F"),
                  value = c(10, 20, 35, 40, 50, 60))

# Create a myrror object
library(myrror)
m <- myrror(dfx, dfy, by.x = "id", by.y = "id")

# Print the myrror object (happens automatically)
m

# Create object with different print settings

# With interactive mode disabled
m2 <- myrror(dfx, dfy, by.x = "id", by.y = "id", interactive = FALSE)
print(m2)
```

`survey_data` *Survey Data A country-year level dataset with 15 rows and 6 variables. 2 countries, 4 years, and 4 additional variables.*

## Description

Survey Data A country-year level dataset with 15 rows and 6 variables. 2 countries, 4 years, and 4 additional variables.

## Usage

```
survey_data
```

## Format

A `data.table` with 16 rows and 4 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010, 2011, 2012, 2013.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

@source Simulated data.

`survey_data_1m` *Survey Data 1:m Variation 1*

## Description

Survey Data 1:m Variation 1

## Usage

```
survey_data_1m
```

## Format

A `data.table` with 36 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

## Source

Variation of `survey_data` with non-unique ids and a 1:m relationship between ids and values.

---

`survey_data_1m_2` *Survey Data 1:m Variation 2*

---

## Description

Survey Data 1:m Variation 2

## Usage

```
survey_data_1m_2
```

## Format

A `data.table` with 36 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

## Source

Variation of `survey_data` with non-unique ids and a 1:m relationship between ids and values.

---

`survey_data_2` *Survey Data Variation 2*

---

## Description

Survey Data Variation 2

## Usage

```
survey_data_2
```

## Format

A `data.table` with 15 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010, 2011, 2012, 2013.

**variable1** Numeric.

**variable2** Numeric. Modified variable values.

**variable3** Numeric.

**variable4** Numeric.

## Source

Simulated data.

---

`survey_data_2_cap` *Survey Data Variation 2 with Cap Keys*

---

## Description

Survey Data Variation 2 with Cap Keys

## Usage

```
survey_data_2_cap
```

## Format

A `data.table` with 15 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010, 2011, 2012, 2013.

**variable1** Numeric.

**variable2** Numeric. Modified variable values.

**variable3** Numeric.

**variable4** Numeric.

## Source

Simulated data.

## survey_data_3 *Survey Data Variation 3*

### Description

Survey Data Variation 3

### Usage

```
survey_data_3
```

### Format

A `data.table` with 15 rows and 6 variables:

**country** Factor with levels: "A", "B".
**year** Numeric, with values: 2010-2017.
**variable1** Character. Modified variable class.
**variable2** Numeric.
**variable3** Numeric.
**variable4** Numeric.

### Source

Simulated data.

## survey_data_4 *Survey Data Variation 4*

### Description

Survey Data Variation 4

### Usage

```
survey_data_4
```

### Format

A `data.table` with 12 rows (4 missing) and 6 variables:

**country** Factor with levels: "A", "B".
**year** Numeric, with values: 2010-2017.
**variable1** Numeric.
**variable2** Numeric.
**variable3** Numeric.
**variable4** Numeric.

## Source

Simulated data.

---

`survey_data_5` *Survey Data Variation 5*

---

## Description

Survey Data Variation 5

## Usage

```
survey_data_5
```

## Format

A `data.table` with 15 rows and 4 variables (2 missing):

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

## Source

Simulated data.

---

`survey_data_6` *Survey Data Variation 6*

---

## Description

Survey Data Variation 6

## Usage

```
survey_data_6
```

## Format

A `data.table` with 15+15 (duplicated) rows and 4 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

## Source

Simulated data.

---

`survey_data_all` *Survey Data Variation All*

---

## Description

Survey Data Variation All

## Usage

```
survey_data_all
```

## Format

A `data.table` with 12 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

## Source

Simulated data.

survey_data_m1 *Survey Data m:1*

## Description

Survey Data m:1

## Usage

```
survey_data_m1
```

## Format

A `data.table` with 15 rows and 6 variables:

**country** Factor with levels: "A", "B".

**year** Numeric, with values: 2010-2017.

**variable1** Numeric.

**variable2** Numeric.

**variable3** Numeric.

**variable4** Numeric.

## Source

Variation of `survey_data` with non-unique ids and a m:1 relationship between ids and values.

# Index