# Package 'manydata'

May 6, 2024

**Title** A Portal for Global Governance Data

**Version** 0.9.3

**Date** 2024-05-06

**Description** This is the core package for the many packages universe. It includes functions to help researchers work with and contribute to event datasets on global governance.

**License** CC BY 4.0

**URL** https://github.com/globalgov/manydata, https://manydata.ch/

**BugReports** https://github.com/globalgov/manydata/issues

**Depends** R (>= 3.5.0)

**Imports** dplyr, messydates (>= 0.4.1), purrr, stringr, usethis, jsonlite, remotes, httr, ggplot2 (>= 3.4.0), tidyr, plyr, zoo, cli

**Suggests** testthat, readr, knitr, rmarkdown, ggVennDiagram, manynet, rlang

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**NeedsCompilation** no

**Author** James Hollway [cre, aut, ctb] (IHEID, <https://orcid.org/0000-0002-8361-9647>), Henrique Sposito [ctb] (IHEID, <https://orcid.org/0000-0003-3420-6085>), Bernhard Bieri [ctb] (IHEID, <https://orcid.org/0000-0001-5943-9059>), Esther Peev [ctb] (IHEID, <https://orcid.org/0000-0002-9678-2777>), Jael Tan [ctb] (IHEID, <https://orcid.org/0000-0002-6234-9764>)

**Maintainer** James Hollway <james.hollway@graduateinstitute.ch>

**Repository** CRAN

**Date/Publication** 2024-05-06 19:00:02 UTC

# R topics documented:



---

`call_packages` *Call, download, and update many packages*

---

## Description

Call, download, and update many packages

## Usage

```
call_packages(package, develop = FALSE)
```

## Arguments

| | |
|---|---|
| `package` | A character vector of package name. For multiple packages, please declare package names as a vector (e.g. c("package1", "package2")). |
| `develop` | Would you like to download the develop version of the package? FALSE by default. |

## Details

`call_packages()` finds and download other packages that belong to the many universe of packages. It allows users to rapidly access the names and other descriptive information of these packages. If users intend to download and install a package listed, they can type the package name within the function.

## Value

`call_packages()` returns a tibble with the 'many packages' currently available. If one or more package names are provided, these will be installed from Github.

## See Also

Other call_: `call_releases()`, `call_sources()`, `call_treaties()`

## Examples

```
#call_packages()
#call_packages("manyenviron")
```

---

`call_releases` *Call releases historical milestones/releases*

---

## Description

The function will take a data frame that details this information, or more usefully, a Github repository listing.

## Usage

```
call_releases(repo, begin = NULL, end = NULL)
```

## Arguments

| | |
|---|---|
| `repo` | the github repository to track, e.g. "globalgov/manydata" |
| `begin` | When to begin tracking repository milestones. By default NULL, two months before the first release. |
| `end` | When to end tracking repository milestones. By default NULL, two months after the latest release. |

## Details

The function creates a project timeline graphic using ggplot2 with historical milestones and milestone statuses gathered from a specified GitHub repository.

## Value

A ggplot graph object

## Source

https://benalexkeen.com/creating-a-timeline-graphic-using-r-and-ggplot2/

## See Also

Other call_: `call_packages()`, `call_sources()`, `call_treaties()`

## Examples

```
#call_releases("globalgov/manydata")
#call_releases("manypkgs")
```

---

`call_sources` *Call sources for datacubes and datasets in 'many' packages*

---

## Description

Call sources for datacubes and datasets in 'many' packages

## Usage

```
call_sources(
  package,
  datacube,
  dataset = NULL,
  open_script = FALSE,
  open_codebook = FALSE
)
```

## Arguments

| | |
|---|---|
| `package` | A character vector of package name. For multiple packages, please declare package names as a vector (e.g. c("package1", "package2")). |
| `datacube` | A datacube from one of the many packages. |
| `dataset` | A dataset in a datacube from one of the many packages. NULL by default. That is, all datasets in the datacube are used. For multiple datasets, please declare datasets as a vector (e.g. c("dataset1", "dataset2")). |
| `open_script` | Would you like to open the preparation script for the dataset? By default false. |
| `open_codebook` | Would you like to open the codebook for the dataset? By default false. |

## Details

`call_sources()` displays sources of the datacubes and datasets in 'many' packages. Please declare package, datacube, and dataset

## Value

`call_sources` returns a tibble with information on the dataset, their sources, URL, and mapping to facilitate understanding variable name changes from original data.

## See Also

Other call_: call_packages(), call_releases(), call_treaties()

## Examples

```
call_sources("manydata", "emperors")
```

---

call_treaties *Call treaties from 'many' datasets*

---

## Description

Call treaties from 'many' datasets

## Usage

```
call_treaties(
  dataset,
  treaty_type = NULL,
  variable = NULL,
  actor = NULL,
  key = "manyID"
)
```

## Arguments

| | |
|---|---|
| dataset | A dataset in a datacube from one of the many packages. NULL by default. That is, all datasets in the datacube are used. For multiple datasets, please declare datasets as a vector (e.g. c("dataset1", "dataset2")). |
| treaty_type | The type of treaties to be returned. NULL, by default. Other options are "bilateral" or "multilateral". |
| variable | Would you like to get one, or more, specific variables present in one or more datasets in the 'many' datacube? NULL by default. For multiple variables, please declare variable names as a vector. |
| actor | An actor variable in dataset. NULL by default. If declared, a tibble of the treaties and their member actors is returned. |
| key | A variable key to join datasets. 'manyID' by default. |

## Details

Certain datasets, or consolidated datacubes, in 'many' packages contains information on treaties which can be retrieved with call_treaties().

## Value

`call_treaties()` returns a tibble with a list of the agreements.

## See Also

Other call_: `call_packages()`, `call_releases()`, `call_sources()`

## Examples

```
membs <- dplyr::tibble(manyID = c("ROU-RUS[RFP]_1901A",
"ROU-RUS[RFP]_1901A", "GD16FI_1901A"),
stateID = c("ROU", "RUS", "DNK"),
Title = c("Convention Between Roumania And Russia Concerning Fishing
In The Danube And The Pruth",
"Convention Between Roumania And Russia Concerning Fishing
In The Danube And The Pruth",
"Convention Between The Governments Of Denmark And
The United Kingdom Of Great Britain
And Northern Ireland For Regulating The Fisheries
Of Their Respective Subjects Outside
Territorial Waters In The Ocean Surrounding The Faroe Islands"),
Begin = c("1901-02-22", "1901-02-22", "1901-06-24"))
call_treaties(membs)
call_treaties(membs, treaty_type = "bilateral",
variable = c("Title", "Begin"))
call_treaties(membs, variable = c("Title", "Begin"), actor = "stateID")
```

---

# coalesce_rows *Get first non-missing*

---

## Description

For use with dplyr::summarise, for example

## Usage

```
coalesce_rows(x)
```

## Arguments

| | |
|---|---|
| x | A vector |

## Details

This function operates similarly to coalesce for columns, that is picking the first non-missing observation, but on observations rather than variables.

## Value

A single value

## Source

https://stackoverflow.com/questions/40515180/dplyr-how-to-find-the-first-non-missing-string-by-groups

## Examples

```
dplyr::reframe(emperors$wikipedia, coalesce_rows(emperors$wikipedia))
coalesce_rows(emperors$wikipedia$Begin)
```

---

# compare_categories *Compare categories in 'many' datacubes*

---

## Description

Compare categories in 'many' datacubes

## Usage

```
compare_categories(
  datacube,
  dataset = "all",
  key = "manyID",
  variable = "all",
  category = "all"
)
```

## Arguments

| | |
|---|---|
| datacube | A datacube from one of the many packages. |
| dataset | A dataset in a datacube from one of the many packages. By default "all". That is, all datasets in the datacube are used. To select two or more datasets, please declare them as a vector. |
| key | A variable key to join datasets. 'manyID' by default. |
| variable | Would you like to focus on one, or more, specific variables present in one or more datasets in the 'many' datacube? By default "all". For multiple variables, please declare variable names as a vector. |
| category | Would you like to focus on one specific code category? By default "all" are returned. Other options include "confirmed", "unique", "missing", "conflict", or "majority". For multiple variables, please declare categories as a vector. |

## Details

Confirmed values are the same in all datasets in datacube. Unique values appear once in datasets in datacube. Missing values are missing in all datasets in datacube. Conflict values are different in the same number of datasets in datacube. Majority values have the same value in multiple, but not all, datasets in datacube.

## See Also

Other compare_: `compare_dimensions()`, `compare_missing()`, `compare_overlap()`, `compare_ranges()`

## Examples

```
compare_categories(emperors, key = "ID")
compare_categories(datacube = emperors, dataset = c("wikipedia", "UNRV"),
key = "ID", variable = c("Beg", "End"), category = c("conflict", "unique"))
plot(compare_categories(emperors, key = "ID"))
plot(compare_categories(datacube = emperors, dataset = c("wikipedia", "UNRV"),
key = "ID", variable = c("Beg", "End"), category = c("conflict", "unique")))
```

---

# compare_dimensions *Compare dimensions for 'many' data*

---

## Description

Compare dimensions for 'many' data

## Usage

```
compare_dimensions(datacube, dataset = "all")
```

## Arguments

| | |
|---|---|
| `datacube` | A datacube from one of the many packages. |
| `dataset` | A dataset in a datacube from one of the many packages. By default, "all". That is, all datasets in the datacube are used. To select two or more datasets, please declare them as a vector. |

## Details

`compare_dimensions()` compares the number of observations, variables, the earliest date, and the latest date in all observations for datasets in a 'many' datacube.

## Value

`compare_dimensions()` returns a tibble with information about each dataset including the number of observations, the number of variables, the earliest date, and the latest date in all observations.

## See Also

Other compare_: compare_categories(), compare_missing(), compare_overlap(), compare_ranges()

## Examples

```
compare_dimensions(emperors)
```

---

`compare_missing` *Compare missing observations for 'many' data*

---

## Description

Compare missing observations for 'many' data

## Usage

```
compare_missing(datacube, dataset = "all", variable = "all")
```

## Arguments

| | |
|---|---|
| `datacube` | A datacube from one of the many packages. |
| `dataset` | A dataset in a datacube from one of the many packages. NULL by default. That is, all datasets in the datacube are used. To select two or more datasets, please declare them as a vector. |
| `variable` | Would you like to focus on one, or more, specific variables present in one or more datasets in the 'many' datacube? By default "all". For multiple variables, please declare variable names as a vector. |

## Details

`compare_missing()` compares the missing observations for variables in each dataset in a 'many' datacube.

## Value

`compare_missing()` returns a tibble with information about each dataset including the number of observations, the number of variables, the earliest date, and the latest date in all observations.

## See Also

Other compare_: compare_categories(), compare_dimensions(), compare_overlap(), compare_ranges()

## Examples

```
compare_missing(emperors)
plot(compare_missing(emperors))
```

---

`compare_overlap` *Compare the overlap between datasets in 'many' datacubes*

---

## Description

Compare the overlap between datasets in 'many' datacubes

## Usage

```
compare_overlap(datacube, dataset = "all", key = "manyID")
```

## Arguments

| | |
|---|---|
| `datacube` | A datacube from one of the many packages. |
| `dataset` | A dataset in a datacube from one of the many packages. By default "all". That is, all datasets in the datacube are used. To select two or more datasets, please declare them as a vector. |
| `key` | A variable key to join datasets. 'manyID' by default. |

## Details

`compare_overlap()` compares the overlap between "key" observations in each dataset in a 'many' datacube.

## Value

`compare_overlap()` returns a tibble with information about each dataset and the number of overlapping observations.

## See Also

Other compare_: `compare_categories()`, `compare_dimensions()`, `compare_missing()`, `compare_ranges()`

## Examples

```
compare_overlap(emperors, key = "ID")
plot(compare_overlap(emperors, key = "ID"))
```

compare_ranges *Compare ranges of variables in 'many' data*

## Description

Compare ranges of variables in 'many' data

## Usage

```
compare_ranges(datacube, dataset = "all", variable)
```

## Arguments

| | |
|---|---|
| datacube | A datacube from one of the many packages. |
| dataset | A dataset in a datacube from one of the many packages. By default, "all". That is, all datasets in the datacube are used. To select two or more datasets, please declare them as a vector. |
| variable | Please declare a variable present in one or more datasets in the 'many' datacube. For multiple variables, please declare variable names as a vector. |

## Details

`compare_ranges()` compares the number of observations, variables, the earliest and latest date in each dataset in a 'many' datacube.

## Value

`compare_ranges()` returns a tibble with information about the minimal, maximal, average, and median values for selected variables in datacubes.

## See Also

Other compare_: `compare_categories()`, `compare_dimensions()`, `compare_missing()`, `compare_overlap()`

## Examples

```
compare_ranges(emperors, variable = c("Begin", "End"))
```

`consolidate` *Consolidate datacube into a single dataset*

## Description

This function consolidates a set of datasets in a 'many* package' datacube into a single dataset with some combination of the rows, columns, and observations of the datasets in the datacube. The function includes separate arguments for the rows and columns, as well as for how to resolve conflicts for observations across datasets. This provides users with considerable flexibility in how they combine data. For example, users may wish to stick to units that appear in every dataset but include variables coded in any dataset, or units that appear in any dataset but only those variables that appear in every dataset. Even then there may be conflicts, as the actual unit-variable observations may differ from dataset to dataset. We offer a number of resolve methods that enable users to choose how conflicts between observations are resolved.

## Usage

```
consolidate(
  datacube,
  rows = "any",
  cols = "any",
  resolve = "coalesce",
  key = "manyID"
)
```

## Arguments

| | |
|---|---|
| `datacube` | A datacube from one of the many packages |
| `rows` | Which rows or units to retain. By default "any" (or all) units are retained, but another option is "every", which retains only those units that appear in all parent datasets. |
| `cols` | Which columns or variables to retain. By default "any" (or all) variables are retained, but another option is "every", which retains only those variables that appear in all parent datasets. |
| `resolve` | How should conflicts between observations be resolved? By default "coalesce", but other options include: "min", "max", "mean", "median", and "random". "coalesce" takes the first non-NA value. "max" takes the largest value. "min" takes the smallest value. "mean" takes the average value. "median" takes the median value. "random" takes a random value. For different variables to be resolved differently, you can specify the variables' names alongside how each is to be resolved in a list (e.g. `resolve = c(var1 = "min", var2 = "max")`). In this case, only the variables named will be resolved and returned. |
| `key` | An ID column to collapse by. By default "manyID". Users can also specify multiple key variables in a list. For multiple key variables, the key variables must be present in all the datasets in the datacube (e.g. `key = c("key1", "key2")`). For |

equivalent key columns with different names across datasets, matching is possible if keys are declared (e.g. `key = c("key1" = "key2")`). Missing observations in the key variable are removed.

## Details

Text variables are dropped for more efficient consolidation.

## Value

A single tibble/data frame.

## Examples

```
consolidate(datacube = emperors, key = "ID")
consolidate(datacube = favour(emperors, "UNRV"), rows = "every",
cols = "every", resolve = "coalesce", key = "ID")
consolidate(datacube = emperors, rows = "any", cols = "every",
resolve = "min", key = "ID")
consolidate(datacube = emperors, rows = "every", cols = "any",
resolve = "max", key = "ID")
consolidate(datacube = emperors, rows = "every", cols = "every",
resolve = "median", key = "ID")
consolidate(datacube = emperors, rows = "every", cols = "every",
resolve = "mean", key = "ID")
consolidate(datacube = emperors, rows = "every", cols = "every",
resolve = "random", key = "ID")
consolidate(datacube = emperors, rows = "every", cols = "every",
resolve = c(Begin = "min", End = "max"), key = "ID")
consolidate(datacube = emperors, rows = "any", cols = "any",
resolve = c(Death = "max", Cause = "coalesce"),
key = c("ID", "Begin"))
```

---

`emperors` *Emperors datacube documentation*

---

## Description

Emperors datacube documentation

## Usage

```
emperors
```

## Format

The emperors datacube is a list that contains the following 3 datasets: wikipedia, UNRV, britannica. For more information and references to each of the datasets used, please use the `data_source()` and `data_contrast()`. functions.

**wikipedia:** A dataset with 68 observations and the following 15 variables: ID, Begin, End, FullName, Birth, Death, CityBirth, ProvinceBirth, Rise, Cause, Killer, Dynasty, Era, Notes, Verif.

**UNRV:** A dataset with 99 observations and the following 7 variables: ID, Begin, End, Birth, Death, FullName, Dynasty.

**britannica:** A dataset with 87 observations and the following 3 variables: ID, Begin, End.

## Details

```
#> $wikipedia
#> ------------------------------------------------------------------------------
#> |  Column Name  |  Data Type  |  Observations  |  Missing  |  Missing (%)  |
#> ------------------------------------------------------------------------------
#> |      ID       |  character  |             68|     0     |       0       |
#> |     Begin     |    mdate    |             68|     0     |       0       |
#> |      End      |    mdate    |             68|     0     |       0       |
#> |   FullName    |  character  |             68|     0     |       0       |
#> |     Birth     |  character  |             68|     5     |     7.35      |
#> |     Death     |  character  |             68|     0     |       0       |
#> |   CityBirth   |  character  |             68|    17     |      25       |
#> | ProvinceBirth |  character  |             68|     0     |       0       |
#> |     Rise      |  character  |             68|     0     |       0       |
#> |     Cause     |  character  |             68|     0     |       0       |
#> |    Killer     |  character  |             68|     0     |       0       |
#> |    Dynasty    |  character  |             68|     0     |       0       |
#> |      Era      |  character  |             68|     0     |       0       |
#> |     Notes     |  character  |             68|    22     |     32.35     |
#> |     Verif     |  character  |             68|    57     |     83.82     |
#> ------------------------------------------------------------------------------
#>
#>
#> $UNRV
#> ------------------------------------------------------------------------------
#> |  Column Name  |  Data Type  |  Observations  |  Missing  |  Missing (%)  |
#> ------------------------------------------------------------------------------
#> |      ID       |  character  |             99|     0     |       0       |
#> |     Begin     |    mdate    |             99|     0     |       0       |
#> |      End      |    mdate    |             99|     0     |       0       |
#> |     Birth     |  character  |             99|     0     |       0       |
#> |     Death     |  character  |             99|     0     |       0       |
#> |   FullName    |  character  |             99|     5     |     5.05      |
#> |    Dynasty    |  character  |             99|    37     |     37.37     |
#> ------------------------------------------------------------------------------
#>
```

```
#>
#> $britannica
#> ------------------------------------------------------------------------
#> |  Column Name  |  Data Type  |  Observations  |  Missing  |  Missing (%)  |
#> ------------------------------------------------------------------------
#> |      ID       |  character  |             87|     0     |       0       |
#> |     Begin     |    mdate    |             87|     0     |       0       |
#> |      End      |    mdate    |             87|     0     |       0       |
#> ------------------------------------------------------------------------
```

## URL

- wikipedia: https://en.wikipedia.org/wiki/List_of_Roman_emperors
- UNRV: https://www.unrv.com/government/emperor.php
- britannica: https://www.britannica.com/topic/list-of-Roman-emperors-2043294

## Mapping

- wikipedia: Variable Mapping

| *from* | *to* |
|---|---|
| name | ID |
| reign.start | Begin |
| reign.end | End |
| name.full | FullName |
| birth | Birth |
| death | Death |
| birth.cty | CityBirth |
| birth.prv | ProvinceBirth |
| rise | Rise |
| cause | Cause |
| killer | Killer |
| dynasty | Dynasty |
| era | Era |
| notes | Notes |
| verif.who | Verif |

- UNRV: Variable Mapping

| *from* | *to* |
|---|---|
| 'Common Name' | ID |
| Beg | Begin |
| 'Full Name/Imperial Name' | FullName |
| 'Dynasty/Class/Notes' | Dynasty |

- britannica: Variable Mapping

| *from* | *to* |
|---|---|
| Name | ID |
| reign_start | Begin |
| reign_end | End |

## Source

- wikipedia: Wikipedia, List_of_Roman_emperors, https://en.wikipedia.org/wiki/List_of_Roman_emperors, Accessed on 2021-07-22
- UNRV: UNRV, Roman Emperor list, https://www.unrv.com/government/emperor.php, Accessed on 2021-07-22
- britannica: Britannica, List of Roman emperors, https://www.britannica.com/topic/list-of-Roman-emperors-2043294, Accessed on 2021-07-22

---

# favour *Favour datasets in a datacube*

---

## Description

Favour datasets in a datacube

## Usage

```
favour(datacube, dataset)

favor(datacube, dataset)
```

## Arguments

| | |
|---|---|
| datacube | A many datacube |
| dataset | The name of one, or more, datasets within the datacube to be favoured over others. |

## Details

The dataset declared becomes the reference for the first non NA value. If more than one dataset is declared, please list datasets in increasing order of importance (.i.e. last dataset should be favoured over previous).

## Value

The datacube with datasets re-ordered accordingly

## Examples

```
favour(emperors, "UNRV")
favour(emperors, c("wikipedia", "UNRV", "britannica"))
```

---

`recollect` *Pastes unique string vectors*

---

## Description

For use with dplyr::summarise, for example

## Usage

```
recollect(x, collapse = "_")
```

## Arguments

| | |
|---|---|
| `x` | A vector |
| `collapse` | String indicating how elements separated |

## Details

This function operates similarly to reunite, but instead of operating on columns/observations, it pastes together unique rows/observations.

## Value

A single value

## Examples

```
data <- data.frame(ID = c(1,2,3,3,2,1))
data1 <- data.frame(ID = c(1,2,3,3,2,1), One = c(1,NA,3,NA,2,NA))
recollect(data$ID)
recollect(data1$One)
```

## repaint *Fills missing data by lookup*

### Description

Fills missing data where known by other observations with the same id/index

### Usage

```
repaint(df, id, var)
```

### Arguments

| | |
|---|---|
| df | a dataframe |
| id | a string identifying a column in the dataframe for indexing |
| var | a string identifying a column or columns in the dataframe to be filled |

### Value

A dataframe

### Examples

```
data <- data.frame(ID = c(1,2,3,3,2,1),
                   One = c(1,NA,3,NA,2,NA),
                   Two = c(NA,"B",NA,"C",NA,"A"))
repaint(data, "ID", c("One","Two"))
```

## reunite *Pastes unique string vectors*

### Description

A vectorised function for use with dplyr's mutate, etc

### Usage

```
reunite(..., sep = "_")
```

### Arguments

| | |
|---|---|
| ... | Variables to pass to the function, currently only two at a time |
| sep | Separator when vectors reunited, by default "_" |

## Value

A single vector with unique non-missing information

## Examples

```
data <- data.frame(fir=c(NA, "two", "three", NA),
                   sec=c("one", NA, "three", NA), stringsAsFactors = FALSE)
transmutate(data, single = reunite(fir, sec))
```

---

`transmutate` *Drop only columns used in formula*

---

## Description

A function between dplyr's transmute and mutate

## Usage

```
transmutate(.data, ...)
```

## Arguments

| | |
|---|---|
| `.data` | Data frame to pass to the function |
| `...` | Variables to pass to the function |

## Value

Data frame with mutated variables and none of the variables used in the mutations, but, unlike `dplyr::transmute()`, all other unnamed variables.

## Source

https://stackoverflow.com/questions/51428156/dplyr-mutate-transmute-drop-only-the-columns-used-in-the-formula

## Examples

```
pluck(emperors, "wikipedia")
transmutate(emperors$wikipedia, Beginning = Begin)
```

# Index