Package ‘manydata’

October 13, 2022

Title A Portal for Global Governance Data
Version 0.7.5
Date 2022-06-07

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

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

Depends R (>= 3.5.0)
Imports tibble, dplyr, messydates, lubridate, purrr, rlang, stringr, usethis, jsonlite, remotes, httr, progress, ggplot2

Suggests testthat, anytime, mice, withr, haven, readxl, readr, knitr, rmarkdown

Encoding UTF-8
LazyData true
RoxygenNote 7.2.0
VignetteBuilder knitr

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 2022-06-07 10:30:02 UTC
R topics documented:

coalesce_compatible ............................................. 2
coalesce_rows ....................................................... 3
consolidate .......................................................... 3
emperors ............................................................... 5
favour ................................................................. 5
get_packages .......................................................... 6
get_treaty ............................................................... 7
plot_releases ........................................................... 8
recollect ............................................................... 9
repa.int ............................................................... 10
report ................................................................. 10
reunite ............................................................... 11
transmutate ............................................................ 12

Index 13

coalesce_compatible Coalesce all compatible rows of a data frame

Description

This function identifies and coalesces all compatible rows in a data frame. Compatible rows are defined as those rows where all present elements are equal, allowing for equality where one row has an element present and the other is missing the observation.

Usage

coalesce_compatible(.data)

Arguments

.data data frame to consolidate

Value

A tibble with the missing observations coalesced for compatible rows

Examples

eg1 <- tibble::tribble(
  ~x, ~y, ~z,
  "a", "b", NA,
  "a", "b", "c",
  "j", "k", NA,
  NA, "k", "l"
)                   
coalesce_compatible(eg1)
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::summarise(emperors$wikipedia, coalesce_rows(emperors$wikipedia))
coalesce_rows(emperors$wikipedia$Beg)

consolidate  Consolidate database into a single dataset

Description
This function consolidates a set of datasets in a 'many' package database into a single dataset with some combination of the rows, columns, and observations of the datasets in the database. 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(
  database,
  rows = "any",
  cols = "any",
  resolve = "coalesce",
  key = "manyID"
)

Arguments

database \hspace{1cm} A database object from one of the many packages
rows \hspace{1cm} 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 \hspace{1cm} 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 \hspace{1cm} 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 \hspace{1cm} An ID column to collapse by. By default "many_ID". 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 database (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").

Value

A single tibble/data frame.

Examples

consolidate(database = emperors, key = "ID")
consolidate(database = favour(emperors, "UNRV"), rows = "every",
cols = "every", resolve = "coalesce", key = "ID")
consolidate(database = emperors, rows = "any", cols = "every",
resolve = "min", key = "ID")
consolidate(database = emperors, rows = "every", cols = "any",
resolve = "max", key = "ID")
consolidate(database = emperors, rows = "every", cols = "every",
resolve = "median", key = "ID")
emperors

consolidate(database = emperors, rows = "every", cols = "every", resolve = "mean", key = "ID")
consolidate(database = emperors, rows = "every", cols = "every", resolve = "random", key = "ID")
consolidate(database = emperors, rows = "every", cols = "every", resolve = c(Beg = "min", End = "max"), key = "ID")
consolidate(database = emperors, rows = "any", cols = "any", resolve = c(Death = "max", Cause = "coalesce"), key = c("ID", "Beg"))

---

emperors  Emperors database documentation

Description
Emperors database documentation

Usage
emperors

Format
The emperors database 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, Beg, 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, Beg, End, Birth, Death, FullName, Dynasty.
britannica: A dataset with 87 observations and the following 3 variables: ID, Beg, End.

---

favour  Favour datasets in a database

Description
Favour datasets in a database

Usage
favour(database, dataset)
favor(database, dataset)
get_packages

Description
Find and download packages in the many packages universe

Usage
get_packages(pkg)

Arguments
pkg A character vector of package names or number of a package

Details
The function 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 by simply calling the function. If users intend to download and install a package from the universe, they can type the package name within the function.
Value

If no package name is provided, this function prints a table (tibble) to the console with details on packages that are currently available within the many universe. This includes the name and description of the package, the latest installed and release version number, and the latest release date. It also include a list of numbers which orders the package and can be used to load the respective package instead of the name. If one or more package names are provided, these will be installed from Github.

get_treaty

Get international treaties

Description

Some datasets in the membership databases across the 'many* packages' (e.g. manyenviron) contain a myriad of information on international treaties governing an international domain. The get_treaty functions help researchers retrieve all bilateral agreements, or multilateral agreements, from these datasets. Researchers can, for example, use get_bilaterals() to retrieve which countries have signed a specific international agreement, or several international agreements signed in a respective year. As well, researchers can use get_multilaterals() to retrieve the titles of all multilateral agreements signed in the past 10 years, for instance. Alternatively, to get information from several datasets in a memberships database, researchers can consolidate() the database into one dataset with some combination of the rows, columns, and observations of the datasets before getting the desired bilateral or multilateral treaties.

Usage

get_bilaterals(membs)

get_multilaterals(membs)

Arguments

membs A memberships dataset from one of the many packages

Value

A tibble of bilateral agreements
A tibble of multilateral agreements

Examples

membs <- tibble::tibble(CountryID = c("ROU", "RUS", "DNK"),
manyID = c("ROU-RUS[RFP]_1901A", "ROU-RUS[RFP]_1901A", "GD16FI_1901A"),
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"),
Beg = c("1901-02-22", "1901-02-22", "1901-06-24"),
End = c(NA, NA, NA))
get_bilaterals(membs)
membs <- tibble::tibble(CountryID = c("ROU", "RUS", "DNK"),
manyID = c("ROU-RUS[RFP]_1901A", "ROU-RUS[RFP]_1901A", "GD16FI_1901A"),
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"),
Beg = c("1901-02-22", "1901-02-22", "1901-06-24"),
End = c(NA, NA, NA))
get_multilaterals(membs)

---

**plot_releases**

*A plotting function that visualises historical milestones/releases*

**Description**

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

**Usage**

```r
plot_releases(repo)
```

**Arguments**

- `repo` the github repository to track, e.g. "globalgov/manydata"

**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
recollect

Source

Examples

```r
if(!httr::http_error("https://api.github.com/repos/globalgov/manypkgs/releases")) {
  plot_releases("globalgov/manypkgs")
}
```

---

recollect  

*Pastes unique string vectors*

---

Description

For use with dplyr::summarise, for example

Usage

```r
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

```r
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  
\textit{Fills missing data by lookup}

\textbf{Description}

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

\textbf{Usage}

\begin{verbatim}
repaint(df, id, var)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{df} \hspace{1cm} a dataframe
\item \texttt{id} \hspace{1cm} a string identifying a column in the dataframe for indexing
\item \texttt{var} \hspace{1cm} a string identifying a column or columns in the dataframe to be filled
\end{itemize}

\textbf{Value}

A dataframe

\textbf{Examples}

\begin{verbatim}
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"))
\end{verbatim}

\textbf{report}  
\textit{Set of data structure exploration functions for users}

\textbf{Description}

The report family of functions allows users to quickly get information about and compare several aspects of a package in the many packages universe, and its’ databases and datasets.

\textbf{Usage}

\begin{verbatim}
data_source(pkg, database = NULL, dataset = NULL)
data_contrast(pkg, database = NULL, dataset = NULL)
open_codebook(pkg, database, dataset)
\end{verbatim}
Arguments

pkg character string of the many package to report data on. Required input.
database vector of character strings of the many package to report data on a specific
database in a many package If NULL, the function returns a summary of all
databases in the many package NULL by default for data_source() and data_contrast().
dataset character string of the many package to report data on a specific dataset in a
specific database of a many package If NULL and database is specified, returns
database level metadata. NULL by default for data_source() and data_contrast().

Details

data_source() displays names of the database/datasets and source material of data in a many
package.
data_contrast() displays information about databases and datasets contained in them. Namely
the number of unique ID’s, the percentage of missing data, the number of observations, the number
of variables, the minimum beginning date and the maximum ending date as well as the most direct
URL to the original dataset.
open_codebook() opens the original codebook of the specified dataset to allow users to look up the
original coding rules. Note that no original codebook might exist for certain datasets. In the latter
case, please refer to the source URL provided with each dataset by running manydata::data_contrast()
as further information on coding rules available online.

Value

A dataframe with the data sources
A list with the desired metadata to compare various datasets in a many package.
Opens a pdf version of the original codebook of the specified dataset, if available.

Examples

data_source(pkg = "manydata")
data_contrast(pkg = "manydata")
transmutate

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


Examples

pluck(emperors, "wikipedia")
transmutate(emperors$wikipedia, Beginning = Beg)
Index

* datasets
  - emperors, 5
coalesce_compatible, 2
coalesce_rows, 3
consolidate, 3
data_contrast (report), 10
data_source (report), 10
emperors, 5
favor (favour), 5
favour, 5
get_bilaterals (get_treaty), 7
get_multilaterals (get_treaty), 7
get_packages, 6
get_treaty, 7
open_codebook (report), 10
plot_releases, 8
recollect, 9
repaint, 10
report, 10
reunite, 11
transmutate, 12