Package: data4health (via r-universe)

July 7, 2026

Title A Practical Workflow for Health Data Wrangling

Version 0.1.1

Description Provides a streamlined workflow for cleaning, transforming, filtering, aggregating, and exporting epidemiological line list data. The package is designed for public health surveillance and clinical datasets where each row represents an individual case. It supports common data-wrangling tasks and multi-format data import/export (e.g., 'csv', 'rds', 'xlsx', 'json', 'dbf'). The functions are designed to be combined into a clear and reproducible pipeline while remaining flexible enough for use in standalone data-processing steps. 'data4health' is part of the '4health' toolkit, which integrates health, climate, land-use, and socioeconomic data workflows. More information on the '4health' tools can be found on the HARMONIZE website <https://harmonize-tools.org/toolkits>.

Imports foreign, readxl, writexl, shiny, jsonlite, GHRexplore, lubridate, plotly, shinyAce

BugReports https://gitlab.earth.bsc.es/ghr/data4health/-/issues

License AGPL (>= 3)

Encoding UTF-8

URL https://bsc-es.github.io/GHRtools/docs/data4health/data4health.html

Suggests knitr, rmarkdown, testthat (>= 3.0.0), shinytest2, withr

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 4.2.0)

LazyData true

Config/Needs/website rmarkdown

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Daniela Lührsen [aut, cre] (ORCID: <https://orcid.org/0009-0002-6340-5964>), Carles Milà [aut] (ORCID: <https://orcid.org/0000-0003-0470-0760>), Raquel Lana [aut] (ORCID: <https://orcid.org/0000-0002-7573-1364>), Rachel Lowe [aut] (ORCID: <https://orcid.org/0000-0003-3939-7343>), Mark Adler [cph] (Author of src/blast.c), Daniela Petruzalek [cph] (Author of src/dbc2dbf.c, see github.com/danicat/read.dbc)

Maintainer Daniela Lührsen <daniela.luhrsen@bsc.es>

Repository https://cran.r-universe.dev

Date/Publication 2026-07-07 09:40:08 UTC

RemoteUrl https://github.com/cran/data4health

RemoteRef HEAD

RemoteSha 1934dd890e52a5010a896378e047ab3b13042a7b

Contents

d4h_aggregate 2
d4h_clean 4
d4h_example 7
d4h_filter 8
d4h_load 9
d4h_save 10
d4h_ui 12
example_dataset 12

Index **14**

`d4h_aggregate` *Aggregate the health data*

Description

The 'd4h_aggregate()' function aggregates health data by specified spatial and temporal columns. It can also include additional columns for aggregation. The function supports filling in missing combinations of spatial and temporal values, ensuring a complete dataset for analysis.

Usage

```
d4h_aggregate(
  data,
  space_col = NULL,
  time_col = NULL,
  all_spaces = NULL,
  all_times = NULL,
```

```
  add_col = NULL,
  ...
)
```

Arguments

data	The dataframe to be aggregated. It can be a dataframe from the R environment, or a filepath (accepted formats are csv, xls, xlsx, dbf, and dbc).
space_col	The name of the column, or a vector of column names, to use for spatial aggregation. If two or more columns are passed (e.g. the region name and the region code), they must be in 1-to-1 correspondence — each code maps to exactly one region name. If they are inconsistent, only the first column is used. When `all_spaces` is also provided, its values must match the format of the first column in `space_col`.
time_col	The name of the column, or a vector of column names, to use for temporal aggregation. If two or more columns are passed (e.g. an epiweek column and a date of the first day whithin said epiweek column), they must be in 1-to-1 correspondence — each date maps to exactly one epiweek. If they are inconsistent, only the first column is used. When `all_times` is also provided, its values must match the format of the first column in `time_col`.
all_spaces	A vector with all area codes/names. If all_spaces is not passed, and there is a missing area in the raw data, it will also be missing in the resultant dataset. Optional.
all_times	A vector with all continuous dates. If all_times is not passed, and there is a missing timestep in the raw data, it will also be missing in the resultant dataset. Optional.
add_col	Any other columns that should be aggregated.
...	Reserved for internal use. If you misspell an argument name, the function will return a helpful error message suggesting the correct one.

Value

An aggregated dataframe.

Examples

```
# aggregate number of cases by geographic area
agg_space <- d4h_aggregate(
  data = example_dataset,
  space_col = "NAME_GEO"
)

# aggregate number of cases by date of symptom onset
agg_time <- d4h_aggregate(
  data = example_dataset,
  time_col = "DATE_INI"
)
```

```
# aggregate by geographic area and date
agg_space_time <- d4h_aggregate(
  data = example_dataset,
  space_col = "NAME_GEO",
  time_col = "DATE_INI"
)

# aggregate by space, time, and an additional column (e.g. sex)
agg_with_sex <- d4h_aggregate(
  data = example_dataset,
  space_col = "NAME_GEO",
  time_col = "DATE_INI",
  add_col = "SEXO"
)

# ensure all areas are present in output (even if zero cases)
agg_complete_space <- d4h_aggregate(
  data = example_dataset,
  space_col = "NAME_GEO",
  all_spaces = c("Arendelle", "Monstropolis", "Radiator Springs",
                 "Atlantica", "Zootopia", "Agrabah", "Motunui")
)

# ensure all dates in a sequence are present
all_dates <- seq(min(example_dataset$DATE_INI, na.rm = TRUE),
                 max(example_dataset$DATE_INI, na.rm = TRUE),
                 by = "day")

agg_complete_time <- d4h_aggregate(
  data = example_dataset,
  time_col = "DATE_INI",
  all_times = all_dates
)
```

`d4h_clean` *Clean the health data*

Description

'd4h_clean()' performs basic data cleaning operations on health-related datasets. Among others, it handles missing values, column renaming, date transformations, and outlier detection.

Usage

```
d4h_clean(
  data,
  cols_to_remove = NULL,
  cols_to_include = NULL,
  threshold_remove = NULL,
  remove_rows_missing = NULL,
```

```
  rename_columns = NULL,
  rename_categories = NULL,
  week_to_date = NULL,
  month_to_date = NULL,
  date_to_weekdate = NULL,
  date_to_weeknumber = NULL,
  week_start = "Sunday",
  date_to_monthdate = NULL,
  date_to_monthnumber = NULL,
  date_to_yearnumber = NULL,
  ...
)
```

Arguments

`data`	A data.frame containing your health data.
`cols_to_remove`	Columns to be removed.
`cols_to_include`	Columns to be included.
`threshold_remove`	Numerical from 0-100. Columns with a higher percentage of missing values than this will be excluded.
`remove_rows_missing`	Character vector of variables where empty or NA entries should be deleted from the table.
`rename_columns`	Either "lower" to transform all column to lower case, "upper" to transform all column names in upper case, "no_spaces" to replace all spaces by underscores or a named character vector of current column names and how to rename them.
`rename_categories`	A named list where each name is a column and the value is a named vector containing the current and new name.
`week_to_date`	`week_to_date` must be NULL or a character vector specifying where epiweek information is stored. It can the name of a column containing year and week combined, in 'YYYY-MM' format, or the names of two columns, where the first contains the year and the second contains the week. A new column with the original column name and the appendix "_weekdate" will be created. The start of the week can be set by using parameter "week_start". Default is "Sunday".
`month_to_date`	`month_to_date` must be NULL or a character vector specifying where the month and year information is stored. It can the name of a column containing year and month combined, in 'YYYY-MM' format, or the names of two columns, where the first contains the year and the second contains the month. A new column with the original column name and the appendix "_monthdate" will be created.
`date_to_weekdate`	column name that contains the date to be transformed. Returns the first day of the week in dateformat. A new column with the original column name and the appendix "_weekdate" will be created. The start of the week can be set by using parameter "week_start". Default is "Sunday".

`date_to_weeknumber`
: column name that contains the date to be transformed. Returns a column with the week number. A new column with the original column name and the appendix "_epiyw" will be created. The start of the week can be set by using parameter "week_start". Default is "Sunday".

`week_start`
: The day which indicates the start of the week. Default is "Sunday".

`date_to_monthdate`
: Which column should be transformed to the date of the first day of the month? A new column named with the original column name and the appendix "_monthdate" will be created.

`date_to_monthnumber`
: Which column should be transformed to the number of the month (i.e. 1-12)? A new column named with the original column name and the appendix "_monthnumber" will be created.

`date_to_yearnumber`
: Which column should be transformed to year? A new column named with the original column name and the appendix "_year" will be created".

`...`
: Reserved for internal use. If you misspell an argument name, the function will return a helpful error message suggesting the correct one.

Value

A data.frame with the cleaned data.

Examples

```
# basic cleaning
head(example_dataset)

# rename all columns to lower case
clean_lower <- d4h_clean(
  data = example_dataset,
  rename_columns = "lower"
)
colnames(example_dataset)
colnames(clean_lower)

# remove selected columns
clean_removed <- d4h_clean(
  data = example_dataset,
  cols_to_remove = c("NOTES", "LOCALCOD")
)
colnames(example_dataset)
colnames(clean_removed)

# remove rows with missing sex
clean_no_missing_sex <- d4h_clean(
  data = example_dataset,
  remove_rows_missing = "SEXO"
)
```

```
nrow(example_dataset)
nrow(clean_no_missing_sex)

# remove variables with more than 50% missing values
clean_threshold <- d4h_clean(
  data = example_dataset,
  threshold_remove = 50
)
colnames(example_dataset)
colnames(clean_threshold)

# transform DATE_INI to epidemiological week variables
clean_epiweek <- d4h_clean(
  data = example_dataset,
  date_to_weekdate = "DATE_INI",
  date_to_weeknumber = "DATE_INI"
)
colnames(clean_epiweek)
head(clean_epiweek[, c("DATE_INI_weekdate", "DATE_INI_epiyw")])

# transform DATE_INI to month and year variables
clean_month_year <- d4h_clean(
  data = example_dataset,
  date_to_monthdate = "DATE_INI",
  date_to_monthnumber = "DATE_INI",
  date_to_yearnumber = "DATE_INI"
)
colnames(clean_month_year)
head(clean_month_year[, c("DATE_INI_monthdate",
                          "DATE_INI_monthnumber",
                          "DATE_INI_year")])
```

`d4h_example` *Get path to data4health example datasets*

Description

The 'data4health' package includes some example files in its `inst/extdata` directory. This function to access them. Use it without parameter to retrieve a list of all examples files, and then again to retrieve the path to your specific file.

Usage

```
d4h_example(path = NULL)
```

Arguments

`path`	Name of file. If `NULL`, the example files will be listed.

Value

If `path` is `NULL`, a character vector of available example file names. If `path` is provided, the full file path to that example file.

Examples

```
d4h_example()
d4h_example("example_dataset.xlsx")
```

`d4h_filter` *Filter the health data*

Description

The 'data4health' package provides a function for filtering dataframes. It allows users to filter data based on various conditions and column types.

Usage

```
d4h_filter(data, ...)
```

Arguments

`data`	The dataframe to be filtered.
`...`	Any columns that you want to be filtered. Each columns should be passed with a list including the filter condition and the filter values. Filter conditions depend on the column type: • numeric = "over", "under", "between", "equal" • Date = "after", "before", "during", • character = "include", "exclude", "match", "starts", "ends"

Value

A filtered dataframe.

Examples

```
## View the first rows
head(example_dataset)

# filter numeric variable (CODE_GEO greater than 100050)
filtered_numeric <- d4h_filter(
  example_dataset,
  CODE_GEO = list(over = 100050)
)

# filter Date variable (cases after Jan 1, 2024)
```

```
filtered_date_after <- d4h_filter(
  example_dataset,
  DATE_INI = list(after = as.Date("2024-01-01"))
)

# filter Date variable between two dates
filtered_date_between <- d4h_filter(
  example_dataset,
  DATE_INI = list(
    between = as.Date(c("2023-06-01", "2023-12-31"))
  )
)

# filter character variable (include only Positive results)
filtered_positive <- d4h_filter(
  example_dataset,
  RESULT_LABORATORY = list(include = "Positive")
)

# exclude specific geographic areas
filtered_exclude_geo <- d4h_filter(
  example_dataset,
  NAME_GEO = list(exclude = c("Agrabah", "Atlantica"))
)

# combine multiple filters
filtered_combined <- d4h_filter(
  example_dataset,
  RESULT_LABORATORY = list(include = "Positive"),
  HOSPITALISED = list(include = "S"),
  DATE_INI = list(after = as.Date("2024-01-01"))
)
```

`d4h_load` *Load the health data*

Description

The 'd4h_load()' function loads files into the R environment. It supports various file formats, including 'csv', 'rds', 'xls', 'xlsx', 'json', 'dbc' and 'dbf'. Multiple files can be passed as long as there is column name consistency across multiple files.

Usage

```
d4h_load(input_file, sheet = NULL, header = NULL)
```

Arguments

`input_file`	Complete path to the input file.
`sheet`	The name(s) of the 'xlsx'/ 'xls' sheet(s) to be loaded. Optional.

header — The user can indicate whether the file has a header or not. If not, a header will be introduced with generic column names (column_1, column_2, etc). If this parameter is not passed the function will try to detect automatically. Optional.

Value

A data.frame with the loaded data.

Examples

```
# load single files of different formats
# json
json_path <- d4h_example("example_dataset.json")
head(d4h_load(json_path))

# csv
csv_path <- d4h_example("example_dataset.csv")
head(d4h_load(csv_path))

# load multiple files with the same columns
rds2023_path <- d4h_example("example_dataset_2023.rds")
rds2024_path <- d4h_example("example_dataset_2024.rds")
head(d4h_load(rds2023_path))
head(d4h_load(rds2024_path))

# you can see that the two datasets have the same columns
# thus they can be loaded together
rds <- d4h_load(c(rds2023_path, rds2024_path))
head(rds)
nrow(rds) # the number of rows is the sum of the two datasets

# load an 'xlsx' file and specify the sheet to be loaded
xlsx_path <- d4h_example("example_dataset.xlsx")
d4h_load(xlsx_path)
# the function will automatically detect if there are multiple sheets and
# load the first one, if you want a different one you can specify it with
# the sheet argument
head(d4h_load(xlsx_path, sheet = "year2024"))
```

---

d4h_save *Save files in desired data type*

Description

The 'd4h_save()' function saves dataframes to your local disk. It supports various file formats, including 'csv', 'rds', 'xlsx', 'json', 'dbc' and 'dbf'.

Usage

```
d4h_save(data, name = NULL, format = "csv")
```

Arguments

data	data.frame that will be saved.
name	The name of the output file. If no argument is passed, it will received the name "data4health_" followed by the date and time.
format	Possible options are: "csv", "dbf", "rds", "xlsx", "csv2". Default is csv.

Value

None.

Examples

```
## Not run:
# save as 'csv' (default)
d4h_save(
  data = example_dataset,
  name = "dengue_export",
  format = "csv"
)

# save as 'rds'
d4h_save(
  data = example_dataset,
  name = "dengue_export",
  format = "rds"
)

# save as 'xlsx' file
d4h_save(
  data = example_dataset,
  name = "dengue_export",
  format = "xlsx"
)

# save as 'dbf'
d4h_save(
  data = example_dataset,
  name = "dengue_export",
  format = "dbf"
)

# save as 'csv2' (semicolon separator)
d4h_save(
  data = example_dataset,
  name = "dengue_export",
  format = "csv2"
)

## End(Not run)
```

d4h_ui *Opens the data4health user interface*

Description

The 'd4h_ui()' function opens the data4health user interface in a web browser. It provides an interactive environment for users to explore and manipulate health-related datasets. The interface allows for data cleaning, filtering, aggregation, and visualization, making it easier to work with epidemiological line list data.

Usage

```
d4h_ui()
```

Value

Called for its side effects. Opens the Shiny app in the browser.

Examples

```
if(interactive()){
   # open the data4health user interface
   d4h_ui()
}
```

example_dataset *Simulated dengue surveillance dataset*

Description

A simulated dataset of 500 dengue notification records including demographic, geographic, clinical, and laboratory information.

Format

A data frame with 500 rows and 11 variables:

NAME_GEO Character. Name of the geographic area. May be missing.

CODE_GEO Integer. Numeric geographic code.

LOCALCOD Character. Local health unit code (e.g., "LOC-1").

DATE_BRTHD Date. Patient's date of birth.

DATE_INI Date. Date of symptom onset.

SEXO Character. Sex ("M", "F"). May be missing.

DENGUE_TYPE Character. Dengue classification code ("1", "2", "3"). May be missing.

HOSPITALISED Character. Hospitalisation status ("S" = Yes, "N" = No). May be missing.

RESULT_LABORATORY Character. Laboratory result ("Positive", "Negative", "Pending"). May be missing.

NOTES Character. Epidemiological notes (e.g., "Imported case"). May be missing.

SEROTYPE Character. Dengue virus serotype ("1"–"4"). Frequently missing.

Details

The dataset was simulated for demonstration and testing purposes. Missing values were introduced to resemble real-world surveillance data. Generated using random sampling with `set.seed(42)`. See `data-raw/example_dataset.R` for full reproducible code.

Source

Simulated data generated within the package.

Index

∗ **datasets**
example_dataset, 12

d4h_aggregate, 2
d4h_clean, 4
d4h_example, 7
d4h_filter, 8
d4h_load, 9
d4h_save, 10
d4h_ui, 12

example_dataset, 12