Package 'epidict'

May 8, 2026

Title Epidemiology Data Dictionaries and Random Data Generators

Version 0.3.0

Description The 'R4EPIs' project <https://r4epi.github.io/sitrep/> seeks to provide a set of standardized tools for analysis of outbreak and survey data in humanitarian aid settings. This package currently provides standardized data dictionaries from Medecins Sans Frontieres Operational Centre Amsterdam for outbreak scenarios (Acute Jaundice Syndrome, Cholera, Diphtheria, Measles, Meningitis) and surveys (Retrospective mortality and access to care, Malnutrition, Vaccination coverage and Event Based Surveillance) - as described in the following <https://scienceportal.msf.org/assets/standardised-mortality-surveys?utm_source=chatgpt.com>.
In addition, a data generator from these dictionaries is provided. It is also possible to read in any Open Data Kit format data dictionary.

License GPL-3

URL https://github.com/R4EPI/epidict/,
https://r4epi.github.io/epidict/

Imports clipr, dplyr, readxl, rlang, stats, tibble, tidyr, utils

Suggests covr, DT, knitr, matchmaker, rmarkdown, sitrep, testthat (>= 2.1.0), withr

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Alexander Spina [aut, cre] (ORCID: <https://orcid.org/0000-0001-8425-1867>),
Zhian N. Kamvar [aut] (ORCID: <https://orcid.org/0000-0003-1458-7108>),
Lukas Richter [aut],
Patrick Keating [aut],
Annick Lenglet [ctb],
Applied Epi Incorporated [cph],
Medecins Sans Frontieres Operational Centre Amsterdam [fnd]

Maintainer Alexander Spina <aspina@appliedepi.org>

Repository CRAN

Date/Publication 2026-01-20 10:10:10 UTC

Contents

`dict_rename_helper` *Dictionary-based helper for aligning your data to variables used in a script*

Description

Dictionary-based helper for aligning your data to variables used in a script

Usage

```
dict_rename_helper(
  dictionary,
  varnames,
  varnames_type,
  rmd,
  copy_to_clipboard = TRUE
)
```

Arguments

`dictionary`	A dataframe of the dictionary which you would like to use.
`varnames`	The name of `dictionary` column that contains variable names.
`varnames_type`	The name of `dictionary` column that contains the variable type. This variable needs to be the same number of rows as as `varnames`.
`rmd`	Path to the Rmarkdown file which you would like to compare to.
`copy_to_clipboard`	if `TRUE` (default), the rename template will be copied to the user's clipboard with `clipr::write_clip()`. If `FALSE`, the rename template will be printed to the user's console.

Value

A dplyr command used to rename columns in your data frame according to the dictionary

See Also

read_dict()

---

gen_data *Generate random linelist or survey data*

Description

Based on a dictionary generator like msf_dict(), this function will generate a randomized dataset based on values defined in the dictionaries. The randomized dataset produced should mimic an excel export from DHIS2 or ODK.

Usage

```
gen_data(dictionary, varnames = "name", numcases = 300, org = "MSF")
```

Arguments

dictionary	Specify which dictionary you would like to use.
varnames	Specify name of column that contains variable names. If dictionary is in ODK format, varnames needs to be "name" (Default), if in DHIS2 format then change to "data_element_shortname".
numcases	Specify the number of cases you want (default is 300)
org	Specify the organisation which the dictionary belongs to. Currently, only MSF exists. In the future, dictionaries from WHO and other organizations may become available.

Value

a data frame with cases in rows and variables in columns. The number of columns will vary from dictionary to dictionary, so please use the dictionary functions to generate a corresponding dictionary.

Examples

```
if (require("dplyr") & require("matchmaker")) {
  withAutoprint({

    # You will often want to use MSF dictionaries to translate codes to human-
    # readable variables. Here, we generate a data set of 20 cases:
    dat <- gen_data(
      dictionary = "Cholera",
      varnames = "data_element_shortname",
      numcases = 20,
      org = "MSF"
    )
    print(dat)
```

```
    # We want the expanded dictionary, so we will select `compact = FALSE`
    dict <- msf_dict(dictionary = "Cholera", long = TRUE, compact = FALSE, tibble = TRUE)
    print(dict)

    # Now we can use matchmaker to filter the data:
    dat_clean <- matchmaker::match_df(dat, dict,
      from = "option_code",
      to = "option_name",
      by = "data_element_shortname",
      order = "option_order_in_set"
    )
    print(dat_clean)

  })
}
```

`msf_dict` *MSF data dictionaries and dummy datasets*

Description

These function produce MSF dictionaries based on DHIS2 (for OCA outbreaks) and ODK (for intersectional outbreaks and surveys) data sets defining the data element name, code, short names, types, and key/value pairs for translating the codes into human-readable format.

Usage

```
msf_dict(dictionary, tibble = TRUE, long = TRUE, compact = TRUE, clean = TRUE)
```

Arguments

`dictionary`	Specify which dictionary you would like to use. • MSF OCA outbreaks include: "AJS", "Cholera", "Measles", "Meningitis" • MSF intersectional outbreaks include: "AJS_intersectional", "Cholera_intersectional", "Diphtheria_intersectional", "Measles_intersectional", "Meningitis_intersectional" • MSF OCA surveys include "Mortality", "Nutrition", "Vaccination_long", "Vaccination_short" and "ebs"
`tibble`	If `TRUE` (default), return data dictionary as a tidyverse tibble otherwise will return a list.
`long`	If `TRUE` (default), the returned data dictionary is in long format with each option getting one row. If `FALSE`, then two data frames are returned, one with variables and the other with content options.
`compact`	If `TRUE` (default), then a nested data frame is returned where each row represents a single variable and a nested data frame column called "options", which can be expanded with `tidyr::unnest()`. This only works if `long = TRUE`.
`clean`	If `TRUE` (default), then will clean variable names and variable options. This will set text to lower snake case and remove any accents.

Value

A data frame (tibble) containing the specified MSF data dictionary. If `long = TRUE`, each variable-option pair is represented as a row. If `compact = TRUE`, the options are nested as a data frame column named "options". If `long = FALSE`, a list is returned with two data frames: `dictionary` and `options`.

See Also

`read_dict()` `gen_data()` `matchmaker::match_df()`

Examples

```
if (require("dplyr") & require("matchmaker")) {
  withAutoprint({
    # You will often want to use MSF dictionaries to translate codes to human-
    # readable variables. Here, we generate a data set of 20 cases:
    dat <- gen_data(
      dictionary = "Cholera",
      varnames = "data_element_shortname",
      numcases = 20,
      org = "MSF"
    )
    print(dat)

    # We want the expanded dictionary, so we will select `compact = FALSE`
    dict <- msf_dict(dictionary = "Cholera", long = TRUE, compact = FALSE, tibble = TRUE)
    print(dict)

    # Now we can use matchmaker to filter the data:
    dat_clean <- matchmaker::match_df(dat, dict,
      from = "option_code",
      to = "option_name",
      by = "data_element_shortname",
      order = "option_order_in_set"
    )
    print(dat_clean)
  })
}
```

---

msf_dict_rename_helper

Helper for aligning your data to MSF standardised dictionaries and analysis templates.

Description

Helper for aligning your data to MSF standardised dictionaries and analysis templates.

Usage

```
msf_dict_rename_helper(dictionary, copy_to_clipboard = TRUE)
```

Arguments

dictionary	Specify which MSF dictionary you would like to use. See msf_dict() for options.
copy_to_clipboard	if TRUE (default), the rename template will be copied to the user's clipboard with clipr::write_clip(). If FALSE, the rename template will be printed to the user's console.

Value

A dplyr command used to rename columns in your data frame according to the dictionary

---

read_dict *Data dictionaries*

Description

These function read dictionaries in ODK and DHIS2 formats, and reformats them for dataset recoding into human-readable format.

Usage

```
read_dict(
  path,
  sheet,
  format,
  tibble = TRUE,
  long = TRUE,
  compact = TRUE,
  clean = TRUE
)
```

Arguments

path	Define the path to .xlsx file where the dictionary is stored
sheet	Optional, if your sheets have non-standard names (e.g. using a disease pre-fix) - this can be specified here.
format	The format which the dictionary is in. Currently supports "DHIS2" and "ODK".
tibble	If TRUE (default), return data dictionary as a tidyverse tibble otherwise will return a list.

`long`	If `TRUE` (default), the returned data dictionary is in long format with each option getting one row. If `FALSE`, then two data frames are returned, one with variables and the other with content options.
`compact`	If `TRUE` (default), then a nested data frame is returned where each row represents a single variable and a nested data frame column called "options", which can be expanded with `tidyr::unnest()`. This only works if `long = TRUE`.
`clean`	If `TRUE` (default), then will clean variable names and variable options. This will set text to lower snake case and remove any accents.

Value

If `long = TRUE`, returns a tibble of the merged dictionary and value options. If `long = FALSE`, returns a list with elements `dictionary` and `options`. If `compact = TRUE`, options are nested as a column of data frames under "options".

Index