Package: rdhs (via r-universe)

November 1, 2024

Type Package

Title API Client and Dataset Management for the Demographic and Health Survey (DHS) Data

Version 0.8.2

Maintainer OJ Watson <oj.watson@hotmail.co.uk>

URL https://docs.ropensci.org/rdhs/

BugReports https://github.com/ropensci/rdhs/issues

Description Provides a client for (1) querying the DHS API for survey indicators and metadata (<https://api.dhsprogram.com/#/index.html>), (2) identifying surveys and datasets for analysis, (3) downloading survey datasets from the DHS website, (4) loading datasets and associate metadata into R, and (5) extracting variables and combining datasets for pooled analysis.

LazyData TRUE

Depends R (>= 3.3.0)

Imports brio, R6, httr, jsonlite, foreign, magrittr, rappdirs, digest, storr, xml2, qdapRegex, getPass, haven, iotools, sf, cli, rlang

Suggests testthat, knitr, rmarkdown, ggplot2, survey, data.table, microbenchmark

License MIT + file LICENSE

RoxygenNote 7.2.3

VignetteBuilder knitr

Language en-GB

Encoding UTF-8

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

RemoteUrl https://github.com/ropensci/rdhs

RemoteRef main

RemoteSha 4f6074ae01a4f47fc132819d4ecb75f523a03a3c

Contents

`as_factor.labelled` *Archived dataset capable as_factor*

Description

Changes in ‘haven‘ have meant that ‘labelled‘ class are now referred to as ‘haven_labelled‘ classes. If ‘haven::as_factor‘ is used on old datasets they will fail to find the suitable method. rdhs::as_factor.labelled will work on old archived datasets that have a ‘labelled‘ class.

Usage

```
as_factor.labelled(
  x,
  levels = c("default", "labels", "values", "both"),
  ordered = FALSE,
  ...
)
```

Arguments

`x`	Object to coerce to a factor.
`levels`	How to create the levels of the generated factor: • "default": uses labels where available, otherwise the values. Labels are sorted by value. • "both": like "default", but pastes together the level and value • "label": use only the labels; unlabelled values become `NA` • "values: use only the values
`ordered`	If `TRUE` create an ordered (ordinal) factor, if `FALSE` (the default) create a regular (nominal) factor.
`...`	Other arguments passed down to method.

Details

For more details see `haven::as_factor`

Examples

```
## Not run:
# create a data.frame using the new haven_labelled class
df1 <- data.frame(
area = haven::labelled(c(1L, 2L, 3L), c("reg 1"=1,"reg 2"=2,"reg 3"=3)),
climate = haven::labelled(c(0L, 1L, 1L), c("cold"=0,"hot"=1))
)

# manually change it to the old style
class(df1$area) <- "labelled"
class(df1$climate) <- "labelled"

# with rdhs attached, i.e. library(rdhs), we can now do the following
haven::as_factor(df1$area)

# we can also use this on the data.frame by using the only_labelled argument
haven::as_factor(df1, only_labelled = TRUE)

## End(Not run)
```

`authenticate_dhs` *DHS Website Authentication*

Description

Authenticate Users for DHS website

Usage

```
authenticate_dhs(config)
```

Arguments

`config`	Object of class 'rdhs_config' as produced by 'read_rdhs_config' that must contain a valid 'email', 'project' and 'password'.

Details

If the user has more than one project that contains the first 30 characters of the provided project they will be prompted to choose which project they want. This choice will be saved so they do not have to enter it again in this R session.

Value

Returns list of length 3:

- user_name: your email usually
- user_pass: your password you provided
- proj_id: your project number

Note

Credit for some of the function to https://github.com/ajdamico/lodown/blob/master/R/dhs.R

`available_datasets` *Create a data frame of datasets that your log in can download*

Description

DHS datasets that can be downloaded

Usage

```
available_datasets(
  config,
  datasets_api_results = NULL,
  surveys_api_results = NULL
)
```

Arguments

`config`	Object of class 'rdhs_config' as produced by 'read_rdhs_config' that must contain a valid 'email', 'project' and 'password'.
`datasets_api_results`	Data.table for the api results for the datasets endpoint. Default = NULL and generated by default if not declared.
`surveys_api_results`	Data.table for the api results for the surveys endpoint. Default = NULL and generated by default if not declared.

Value

Returns `"data.frame"` of length 14:

- "FileFormat"
- "FileSize"
- "DatasetType"
- "SurveyNum"
- "SurveyId"
- "FileType"
- "FileDateLastModified"
- "SurveyYearLabel"
- "SurveyType"
- "SurveyYear"

- "DHS_CountryCode"
- "FileName"
- "CountryName"
- "URLS"

Note

Inspiration for function to https://github.com/ajdamico/lodown/blob/master/R/dhs.R

`client_cache_date` *Pull last cache date*

Description

Pull last cache date

Usage

```
client_cache_date(root)
```

Arguments

`root`	Character for root path to where client, caches, surveys etc. will be stored.

`client_dhs` *Make a dhs client*

Description

Make a DHS API client

Usage

```
client_dhs(config = NULL, root = rappdirs_rdhs(), api_key = NULL)
```

Arguments

`config`	config object, as created using `read_rdhs_config`
`root`	Character for root directory to where client, caches, surveys etc. will be stored. Default = `rappdirs_rdhs()`
`api_key`	Character for DHS API KEY. Default = NULL

Methods

`dhs_api_request` Makes a call to the DHS websites API. You can make requests to any of their declared api endpoints (see `vignette(rdhs)` for more on these). API queries can be filtered by providing query terms, and you can control how many search results you want returned. The default parameters will return all of the results, and will format it nicely into a data.frame for you. N.B. This is easier to now do by using the bespoke functions that are included within the package. These take the form dhs_<endpoint>, e.g. `dhs_data`. These functions can also take your client as an argument that will cache the response for you

Usage: `dhs_api_request(api_endpoint, query = list(), api_key = private$api_key, num_results = 100, just_results = TRUE)`

Arguments:

- `api_endpoint`: API endpoint. Must be one of the 12 possible endpoints.
- `query`: List of query filters. To see possible query filter terms for each endpoint then head to the DHS api website.
- `api_key`: DHS API key. Default will grab the key provided when the client was created.
- `num_results`: The Number of results to return. Default = "ALL" which will loop through all the api search results pages for you if there are more results than their API will allow you to fetch in one page. If you specify a number this many results will be returned (but probably best to just leave default).
- `just_results`: Boolean whether to return just the results or all the http API response. Default = TRUE (probably best again to leave as this.)

Value: Data.frame with search results if just_results=TRUE, otherwise a nested list with all the API responses for each page required.

`available_datasets` Searches the DHS website for all the datasets that you can download. The results of this function are cached in the client. If you have recently requested new datasets from the DHS website then you can specify to clear the cache first so that you get the new set of datasets available to you.

Usage: `available_datasets(clear_cache_first = FALSE)`

Arguments:

- `clear_cache_first`: Boolean detailing if you would like to clear the cached available datasets first. The default is set to FALSE. This option is available so that you can make sure your client fetches any new datasets that you have recently been given access to.

Value: Data.frame object with 14 variables that detail the surveys you can download, their url download links and the country, survey, year etc info for that link.

`get_datasets` Gets datasets from your cache or downloads from the DHS website. By providing the filenames, as specified in one of the returned fields from `dhs_datasets`, the client will log in for you and download all the files you have requested. If any of the requested files are unavailable for your log in, these will be flagged up first as a message so you can make a note and request them through the DHS website. You also have the option to control whether the downloaded zip file is then extracted and converted into a more convenient R `data.frame`. This converted object will then be subsequently saved as a ".rds" object within the client root directory datasets folder, which can then be more quickly loaded when needed with `readRDS`. You also have the option to reformat the dataset, which will ensure that a suitable parser is used to preserve the meta information in your dataset, such as what different survey response codes mean.

Usage: `get_datasets(dataset_filenames, download_option = "rds", reformat = FALSE, all_lower = TRUE, output_dir_root = file.path(private$root, "datasets"), clear_cache = FALSE, ...)`

Arguments:

- `dataset_filenames`: The desired filenames to be downloaded. These can be found as one of the returned fields from `dhs_datasets`. Alternatively you can also pass the desired rows from `dhs_datasets`.
- `download_option`: Character specifying whether the dataset should be just downloaded ("zip"), imported and saved as an .rds object ("rds"), or both extract and rds ("both"). Conveniently you can just specify any letter from these options.
- `reformat`: Boolean concerning whether to reformat read in datasets by removing all factors and labels. Default = FALSE.
- `all_lower`: Logical indicating whether all value labels should be lower case. Default to 'TRUE'.
- `output_dir_root`: Root directory where the datasets will be stored within. The default will download datasets to a subfolder of the client root called "datasets"
- `clear_cache`: Should your available datasets cache be cleared first. This will allow newly accessed datasets to be available. Default = 'TRUE'
- `...`: Any other arguments to be passed to `read_dhs_dataset`

Value: Depends on the download_option requested, but ultimately it is a file path to where the dataset was downloaded to, so that you can interact with it accordingly.

`survey_questions` Use this function after download_survey to query downloaded surveys for what questions they asked. This function will look for the downloaded and imported survey datasets from the cache, and will download them if not previously downloaded.

Usage: `survey_questions(dataset_filenames, search_terms = NULL, essential_terms = NULL, regex = NULL, rm_na = TRUE, ...)`

Arguments:

- `dataset_filenames`: The desired filenames to be downloaded. These can be found as one of the returned fields from `dhs_datasets`.
- `search_terms`: Character vector of search terms. If any of these terms are found within the surveys question descriptions, the corresponding code and description will be returned.
- `essential_terms`: Character pattern that has to be in the description of survey questions. I.e. the function will first find all survey_questions that contain your search terms (or regex) OR essential_terms. It will then remove any questions that did not contain your essential_terms. Default = NULL.
- `regex`: Regex character pattern for matching. If you want to specify your regex search pattern, then specify this argument. N.B. If both search_terms and regex are supplied as arguments then regex will be ignored.
- `rm_na`: Should NAs be removed. Default is 'TRUE'
- `...`: Any other arguments to be passed to `download_datasets`

Value: Data frame of the surveys where matches were found and then all the resultant codes and descriptions.

`survey_variables` Use this function after download_survey to look up all the surveys that have the provided codes.

Usage: `survey_variables(dataset_filenames, variables, essential_variables = NULL, rm_na = TRUE, ...)`

Arguments:

- `dataset_filenames`: The desired filenames to be downloaded. These can be found as one of the returned fields from `dhs_datasets`.
- `variables`: Character vector of survey variables to be looked up
- `essential_variables`: Character vector of variables that need to present. If any of the codes are not present in that survey, the survey will not be returned by this function. Default = NULL.
- `rm_na`: Should NAs be removed. Default is 'TRUE'
- `...`: Any other arguments to be passed to `download_datasets`

Value: Data frame of the surveys where matches were found and then all the resultant codes and descriptions.

`extract` Function to extract datasets using a set of survey questions as taken from the output from `survey_questions`

Usage: `extract(questions, add_geo = FALSE)`

Arguments:

- `questions`: Questions to be queried, in the format from `survey_questions`
- `add_geo`: Add geographic information to the extract. Default = TRUE

`get_variable_labels` Returns information about a dataset's survey variables and definitions.

Usage: `get_variable_labels(dataset_filenames = NULL, dataset_paths = NULL, rm_na = FALSE)`

Arguments:

- `dataset_filenames`: Vector of dataset filenames to look up
- `dataset_paths`: Vector of dataset file paths to where datasets have been saved to
- `rm_na`: Should variables and labels with NAs be removed. Default = FALSE

Value: Data frame of survey variable names and definitions

`get_cache_date` Returns the private member variable cache-date, which is the date the client was last created/validated against the DHS API.

Usage: `get_cache_date()`

Value: POSIXct and POSIXt time

`get_root` Returns the file path to the client's root directory

Usage: `get_root()`

Value: Character string file path

`get_config` Returns the client's configuration

Usage: `get_config()`

Value: Config data.frame

`get_downloaded_datasets` Returns a named list of all downloaded datasets and their file paths

Usage: `get_downloaded_datasets()`

Value: List of dataset names and file paths.

`set_cache_date` Sets the private member variable cache-date, which is the date the client was last created/validated against the DHS API. This should never really be needed but is included to demonstrate the cache clearing properties of the client in the vignette.

Usage: `set_cache_date(date)`

Arguments:

- `date`: POSIXct and POSIXt time to update cache time to.

`save_client` Internally save the client object as an .rds file within the root directory for the client.

Usage: `save_client()`

`clear_namespace` Clear the keys and values associated within a cache context. The dhs client caches a number of different tasks, and places these within specific contexts using the package `storr::storr_rds`.

Usage: `clear_namespace(namespace)`

Arguments:

- `namespace`: Character string for the namespace to be cleared.

Examples

```
## Not run:
# create an rdhs config file at "rdhs.json"
conf <- set_rdhs_config(
config_path = "rdhs.json",global = FALSE, prompt = FALSE
)
td <- tempdir()
cli <- rdhs::client_dhs(api_key = "DEMO_1234", config = conf, root = td)

## End(Not run)
```

`collapse_api_responses` *collapse API response list*

Description

collapse API response list

Usage

```
collapse_api_responses(x)
```

Arguments

`x`	List of lists from API to be collapsed

data_and_labels *Create list of dataset and its variable names*

Description

Function to give the former output of get_datasets as it can be nice to have both the definitions and the dataset attached together

Usage

```
data_and_labels(dataset)
```

Arguments

dataset	Any read in dataset created by get_datasets, either as the file path or after having been read using readRDS

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download one of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1])
dl <- data_and_labels(g$zzbr62dt)

# now we easily have our survey question labels easily accessible
grep("bed net", dl$variable_names$description, value = TRUE)

## End(Not run)
```

delabel_df *convert labelled data frame to data frame of just characters*

Description

convert labelled data frame to data frame of just characters

Usage

```
delabel_df(df)
```

Arguments

df	data frame to convert labelled elements of. Likely this will be the output of extract_dhs.

Value

A data frame of de-labelled elements

Examples

```
df1 <- data.frame(
area = haven::labelled(c(1L, 2L, 3L), c("reg 1"=1,"reg 2"=2,"reg 3"=3)),
climate = haven::labelled(c(0L, 1L, 1L), c("cold"=0,"hot"=1))
)

df_char <- delabel_df(df = df1)
```

---

dhs_countries *API request of DHS Countries*

Description

API request of DHS Countries

Usage

```
dhs_countries(
  countryIds = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  surveyCharacteristicIds = NULL,
  tagIds = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
indicatorIds	Specify a comma separated list of indicators ids to filter by. For a list of indicators use `dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))`

surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Countries on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Countries on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
surveyCharacteristicIds	Specify a survey characteristic id to filter countries in surveys with the specified survey characteristic. For a list of survey characteristics use dhs_surveys(returnFields=c("SurveyId'
tagIds	Specify a tag id to filter countries with surveys containing indicators with the specified tag. For a list of tags use dhs_tags()
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 12 (or less if returnFields is provided) countries with their corresponding details. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/countries/fields

Examples

```
## Not run:
# A common use for the countries API endpoint is to query which countries
# ask questions about a given topic. For example to find all countries that
# record data on malaria prevalence by RDT:

dat <- dhs_countries(indicatorIds = "ML_PMAL_C_RDT")

# Additionally you may want to know all the countries that have conducted
# MIS (malaria indicator surveys):
```

```
dat <- dhs_countries(surveyType="MIS")

# A complete list of examples for how each argument to the countries API
# endpoint can be provided is given below, which is a copy of each of
# the examples listed in the API at:

# https://api.dhsprogram.com/#/api-countries.cfm


dat <- dhs_countries(countryIds="EG",all_results=FALSE)
dat <- dhs_countries(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_countries(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_countries(surveyYear="2010",all_results=FALSE)
dat <- dhs_countries(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_countries(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_countries(surveyType="DHS",all_results=FALSE)
dat <- dhs_countries(surveyCharacteristicIds="32",all_results=FALSE)
dat <- dhs_countries(tagIds="1",all_results=FALSE)
dat <- dhs_countries(f="html",all_results=FALSE)

## End(Not run)
```

---

dhs_data *API request of DHS Indicator Data*

Description

API request of DHS Indicator Data

Usage

```
dhs_data(
  countryIds = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  selectSurveys = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  surveyCharacteristicIds = NULL,
  characteristicCategory = NULL,
  characteristicLabel = NULL,
  tagIds = NULL,
  breakdown = NULL,
  returnGeometry = NULL,
  f = NULL,
  returnFields = NULL,
```

```
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))
indicatorIds	Specify a comma separated list of indicator ids to filter by. For a list of indicators use dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel", "SurveyType","CountryName"))
selectSurveys	Specify to filter Data from the latest survey by adding 'selectSurveys="latest"' in conjunction with a Country Code and/or Survey Type. Please Note: Not all indicators are present in the latest surveys. To filter your API Indicator Data call to return the latest survey data in which a specific set of indicators is present, add 'selectSurveys="byIndicator"' in conjunction with Indicator IDs, Country Code, and/or Survey Type instead of using 'selectSurveys="latest"'
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Data on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Data on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
surveyCharacteristicIds	Specify a survey characteristic id to filter data on surveys with the specified survey characteristic. For a list of survey characteristics use dhs_surveys(returnFields=c("SurveyId","SurveyType","CountryName"))
characteristicCategory	Specify a survey characteristic category to filter data on surveys with the specified survey characteristic category. This query is case insensitive, but it only recognizes exact phrase matches. For example, 'characteristicCategory="wealth"' will return results that have a characteristic category of 'Wealth' while 'characteristicCategory="wealth quintile"' will return results that have a characteristic category of 'Wealth Quintile'.
characteristicLabel	Specify a survey characteristic category to filter data on surveys with the specified survey characteristic category. This query is case insensitive, but it only recognizes exact phrase matches. You can use characteristicLabel on its own or in conjunction with characteristicCategory.

tagIds	Specify a tag id to filter data on indicators with the specified tag. For a list of tags use dhs_tags()
breakdown	Data can be requested at different levels via the breakdown parameter. By default national data is returned and provides totals on a national level. 'breakdown="subnational"' data provides values on a subnational level. 'breakdown="background"' provides totals on categorized basis. Examples are urban/rural, education and wealth level. 'breakdown="all"' provides all the data including disaggregated data.
returnGeometry	Coordinates can be requested from the API by including 'returnGeometry=TRUE' in your request. The default for this value is false.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE

Value

Returns a data.table of 27 (or less if returnFields is provided) data for your particular query. Details of properties returned with each row of data are provided at https://api.dhsprogram.com/rest/dhs/data/fields

Examples

```
## Not run:
# A common use for the indicator data API will be to search for a specific
# health indicator for a given country. For example to return the total
# malaria prevalence according to RDT, given by the indicator ML_PMAL_C_RDT,
# in Senegal since 2010:

dat <- dhs_data(
indicatorIds="ML_PMAL_C_RDT",
countryIds="SN",
surveyYearStart="2006"
)

# A complete list of examples for how each argument to the data api
# endpoint can be provided is given below, which is a copy of each of
# the examples listed in the API at:
```

```
# https://api.dhsprogram.com/#/api-data.cfm


dat <- dhs_data(countryIds="EG",all_results=FALSE)
dat <- dhs_data(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_data(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_data(selectSurveys="latest",all_results=FALSE)
dat <- dhs_data(selectSurveys="byIndicator", indicatorIds="FE_CEBA_W_CH0",
all_results=FALSE)
dat <- dhs_data(surveyYear="2010",all_results=FALSE)
dat <- dhs_data(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_data(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_data(surveyType="DHS",all_results=FALSE)
dat <- dhs_data(surveyCharacteristicIds="32",all_results=FALSE)
dat <- dhs_data(characteristicCategory="wealth quintile",all_results=FALSE)
dat <- dhs_data(breakdown="all", countryIds="AZ", characteristicLabel="6+",
all_results=FALSE)
dat <- dhs_data(tagIds="1",all_results=FALSE)
dat <- dhs_data(breakdown="subnational",all_results=FALSE)
dat <- dhs_data(breakdown="background",all_results=FALSE)
dat <- dhs_data(breakdown="all",all_results=FALSE)
dat <- dhs_data(f="html",all_results=FALSE)
dat <- dhs_data(f="geojson", returnGeometry="true",all_results=FALSE)

## End(Not run)
```

`dhs_datasets` *API request of DHS Datasets*

Description

API request of DHS Datasets

Usage

```
dhs_datasets(
  countryIds = NULL,
  selectSurveys = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  fileFormat = NULL,
  fileType = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
```

```
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
selectSurveys	Specify to filter data from the latest survey by including 'selectSurveys=TRUE' in your request. Note: Please use this parameter in conjunction with countryCode, surveyType, or indicatorIds for best results.
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use `dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))`
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Datasets on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Datasets on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
fileFormat	Specify the file format for the survey. Can use file format type name (SAS, Stata, SPSS, Flat, Hierarchical) or file format code. View list of file format codes - https://dhsprogram.com/data/File-Types-and-Names.cfm
fileType	Specify the type of dataset generated for the survey (e.g. household, women, men, children, couples, etc.). View list of file type codes - https://dhsprogram.com/data/File-Types-and-Names.cfm
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a `data.table` of 13 (or less if `returnFields` is provided) datasets with their corresponding details. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/datasets/fields

Examples

```
## Not run:
# The API endpoint for the datasets available within the DHS website
# is a very useful endpoint, which is used a lot within `rdhs`. For example,
# it is used to find the file names and size of the dataset files, as well
# as when they were last modified. This enables us to see which datasets
# have been updated and may thus be out of date. For example to find all
# datasets that have been modified in 2018:

dat <- dhs_datasets()
dates <- rdhs:::mdy_hms(dat$FileDateLastModified)
years <- as.POSIXlt(dates, tz = tz(dates))$year + 1900
modified_in_2018 <- which(years == 2018)

# A complete list of examples for how each argument to the datasets
# API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-datasets.cfm


dat <- dhs_datasets(countryIds="EG",all_results=FALSE)
dat <- dhs_datasets(selectSurveys="latest",all_results=FALSE)
dat <- dhs_datasets(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_datasets(surveyYear="2010",all_results=FALSE)
dat <- dhs_datasets(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_datasets(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_datasets(surveyType="DHS",all_results=FALSE)
dat <- dhs_datasets(fileFormat="stata",all_results=FALSE)
dat <- dhs_datasets(fileFormat="DT",all_results=FALSE)
dat <- dhs_datasets(fileType="KR",all_results=FALSE)
dat <- dhs_datasets(f="geojson",all_results=FALSE)

## End(Not run)
```

---

dhs_data_updates *API request of DHS Data Updates*

Description

API request of DHS Data Updates

Usage

```
dhs_data_updates(
  lastUpdate = NULL,
  f = NULL,
  returnFields = NULL,
```

```
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

lastUpdate	Specify a date or Unix time to filter the updates by. Only results for data that have been updated on or after the specified date will be returned.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 9 (or less if returnFields is provided) indicators or surveys that have been added/updated or removed. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/dataupdates/fields

Examples

```
## Not run:
# The API endpoint for the data updates available within the DHS
# is a very useful endpoint, which is used a lot within `rdhs`. For example,
# we use it to keep the end user's cache up to date. For example to find all
# updates that have occurred in 2018:

dat <- dhs_data_updates(lastUpdate="20180101")

# A complete list of examples for how each argument to the data updates
# API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-dataupdates.cfm
```

```
dat <- dhs_data_updates(lastUpdate="20150901",all_results=FALSE)
dat <- dhs_data_updates(f="html",all_results=FALSE)

## End(Not run)
```

---

dhs_geometry *API request of DHS Geometry*

Description

API request of DHS Geometry

Usage

```
dhs_geometry(
  countryIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use `dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))`
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Geometry on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Geometry on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.

returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 7 (or less if returnFields is provided) geometry with their corresponding details. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/geometry/fields

Examples

```
## Not run:
# The geometry API endpoint returns the spatial geometry for countries, and
# can then be used to recreate the spatial polygon for a given country. For
# example to return the coordinates for the Senegal 2010 DHS survey:

dat <- dhs_geometry(surveyIds="SN2010DHS")

# At the moment there is no function to convert the coordinates returned by
# the API but this will be in the next version of rdhs. For those interested
# look at the geojson vignette for an alternative way to produce plots.

# A complete list of examples for how each argument to the geometry
# API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-geometry.cfm


dat <- dhs_geometry(countryIds="EG",all_results=FALSE)
dat <- dhs_geometry(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_geometry(surveyYear="2010",all_results=FALSE)
dat <- dhs_geometry(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_geometry(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_geometry(surveyType="DHS",all_results=FALSE)
dat <- dhs_geometry(f="geojson",all_results=FALSE)

## End(Not run)
```

dhs_gps_data_format *DHS GPS Data Format*

Description

Data frame to describe the data encoded in DHS GPS files

Usage

```
data(dhs_gps_data_format)
```

Format

A dataframe of 20 observations of 3 variables:

`dhs_gps_data_format`: A dataframe of GPS data descriptions.

- "Name"
- "Type"
- "Description"

dhs_indicators *API request of DHS Indicators*

Description

API request of DHS Indicators

Usage

```
dhs_indicators(
  countryIds = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  surveyCharacteristicIds = NULL,
  tagIds = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))
indicatorIds	Specify a comma separated list of indicators ids to filter by. For a list of indicators use dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel", "SurveyType","CountryName"))
surveyYear	Specify a survey year to filter by.
surveyYearStart	Specify a range of Survey Years to filter Indicators on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Indicators on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a comma separated list of survey years to filter by.
surveyCharacteristicIds	Specify a survey characteristic id to filter indicators in surveys with the specified survey characteristic. For a list of survey characteristics use dhs_surveys(returnFields=c("SurveyId'
tagIds	Specify a tag id to filter indicators with the specified tag. For a list of tags use dhs_tags()
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 18 (or less if returnFields is provided) indicators with attributes for each indicator. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/indicators/fields

Examples

```
## Not run:
# A common use for the indicators data API will be to search for a list of
```

```
# health indicators within a given characteristic category, such as anemia
# testing, HIV prevalence, micronutrients etc. For example to return all the
# indicators relating to malaria testing by RDTs:

dat <- dhs_indicators(surveyCharacteristicIds="90")

# A list of the different `surveyCharacteristicIds` can be found
# [here](https://api.dhsprogram.com/rest/dhs/surveycharacteristics?f=html)

# A complete list of examples for how each argument to the indicator API
# endpoint can be provided is given below, which is a copy of each of
# the examples listed in the API at:

# https://api.dhsprogram.com/#/api-indicators.cfm


dat <- dhs_indicators(countryIds="EG",all_results=FALSE)
dat <- dhs_indicators(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_indicators(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_indicators(surveyYear="2010",all_results=FALSE)
dat <- dhs_indicators(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_indicators(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_indicators(surveyType="DHS",all_results=FALSE)
dat <- dhs_indicators(surveyCharacteristicIds="32",all_results=FALSE)
dat <- dhs_indicators(tagIds="1",all_results=FALSE)
dat <- dhs_indicators(f="html",all_results=FALSE)

## End(Not run)
```

---

dhs_info *API request of DHS Info*

Description

API request of DHS Info

Usage

```
dhs_info(
  infoType = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

infoType	Specify a type of info to obtain the information requested. Default is version. ‘infoType="version"‘ (default) Provides the version of the API. Example: https://api.dhsprogram.com/rest/dhs/info?infoType=version ‘infoType="citation"‘ Provides the citation for the API to include with your application or data. Example: https://api.dhsprogram.com/rest/dhs/info?infoType=citation
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 2 (or less if returnFields is provided) fields describing the type of information that was requested and a value corresponding to the information requested. https://api.dhsprogram.com/rest/dhs/info/fields

Examples

```
## Not run:
# The main use for the info API  will be to confirm the version of the API
# being used to providing the most current citation for the data.

dat <- dhs_info(infoType="version")

# A complete list of examples for how each argument to the info API
# endpoint can be provided is given below, which is a copy of each of
# the examples listed in the API at:

# https://api.dhsprogram.com/#/api-info.cfm


dat <- dhs_info(infoType="version",all_results=FALSE)
dat <- dhs_info(infoType="citation",all_results=FALSE)
dat <- dhs_info(f="html",all_results=FALSE)

## End(Not run)
```

dhs_publications *API request of DHS Publications*

Description

API request of DHS Publications

Usage

```
dhs_publications(
  countryIds = NULL,
  selectSurveys = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  surveyCharacteristicIds = NULL,
  tagIds = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
selectSurveys	Specify to filter data from the latest survey by including 'selectSurveys=TRUE' in your request. Note: Please use this parameter in conjunction with countryCode, surveyType, or indicatorIds for best results.
indicatorIds	Specify a comma separated list of indicators ids to filter by. For a list of indicators use `dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))`
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use `dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))`
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Publications on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Publications on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.

surveyType	Specify a survey type to filter by.
surveyCharacteristicIds	Specify a survey characteristic id to filter publications with countries with the specified survey characteristics. For a list of survey characteristics use dhs_surveys(returnFields=c("
tagIds	Specify a tag id to filter publications with surveys containing indicators with the specified tag. For a list of tags use dhs_tags()
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 10 (or less if returnFields is provided) publications with detailed information for each publication. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/publications/fields

Examples

```
## Not run:
# A main use for the publications API endpoint is to find which surveys have
# a published report resulting from the conducted survey:

dat <- dhs_publications()

# A complete list of examples for how each argument to the publications
# API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-publications.cfm


dat <- dhs_publications(countryIds="EG",all_results=FALSE)
dat <- dhs_publications(selectSurveys="latest",all_results=FALSE)
dat <- dhs_publications(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_publications(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_publications(surveyYear="2010",all_results=FALSE)
dat <- dhs_publications(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_publications(surveyYearStart="1991", surveyYearEnd="2006",
```

```
all_results=FALSE)
dat <- dhs_publications(surveyType="DHS",all_results=FALSE)
dat <- dhs_publications(surveyCharacteristicIds="32",all_results=FALSE)
dat <- dhs_publications(tagIds=1,all_results=FALSE)
dat <- dhs_publications(f="html",all_results=FALSE)

## End(Not run)
```

`dhs_surveys` *API request of DHS Surveys*

Description

API request of DHS Surveys

Usage

```
dhs_surveys(
  countryIds = NULL,
  indicatorIds = NULL,
  selectSurveys = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  surveyStatus = NULL,
  surveyCharacteristicIds = NULL,
  tagIds = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

`countryIds`	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
`indicatorIds`	Specify a comma separated list of indicators ids to filter by. For a list of indicators use `dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))`
`selectSurveys`	Specify to filter data from the latest survey by including 'selectSurveys=TRUE' in your request. Note: Please use this parameter in conjunction with countryCode, surveyType, or indicatorIds for best results.

surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Surveys on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Surveys on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
surveyStatus	Every survey is assigned a surveys status and can be queried based on the surveyStatus parameter. ‘surveyStatus="available"‘ (default) provides a list of all surveys for which the DHS API contains Indicator Data. ‘surveyStatus="Completed"‘ provides a list of all completed surveys. NOTE: Data may not be available for every completed survey. ‘surveyStatus="Ongoing"‘ provides a list of all ongoing surveys. ‘surveyStatus="all"‘ provides a list of all surveys.
surveyCharacteristicIds	Specify a survey characteristic id to filter surveys with the specified survey characteristic. For a list of survey characteristics use dhs_surveys(returnFields=c("SurveyId","SurveyY
tagIds	Specify a tag id to filter surveys containing indicators with the specified tag. For a list of tags use dhs_tags()
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 28 (or less if returnFields is provided) surveys with detailed information for each survey. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/surveys/fields

Examples

```
## Not run:
# A common use for the surveys API endpoint is to query which countries
# have conducted surveys since a given year, e.g. since 2010
```

```
dat <- dhs_surveys(surveyYearStart="2010")

# Additionally, some countries conduct non DHS surveys, but the data for
# thse is also available within the DHS website/API. To query these:

dat <- dhs_surveys(surveyType="MIS")

# Lastly, you may be interested to know about anything peculiar about a
# particular survey's implementation. This can be found by looking within
# the footnotes variable within the data frame returned. For example, the
# Madagascar 2013 MIS:

dat$Footnotes[dat$SurveyId == "MD2013MIS"]

# A complete list of examples for how each argument to the surveys API
# endpoint can be provided is given below, which is a copy of each of
# the examples listed in the API at:

# https://api.dhsprogram.com/#/api-surveys.cfm


dat <- dhs_surveys(countryIds="EG",all_results=FALSE)
dat <- dhs_surveys(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_surveys(selectSurveys="latest",all_results=FALSE)
dat <- dhs_surveys(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_surveys(surveyYear="2010",all_results=FALSE)
dat <- dhs_surveys(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_surveys(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_surveys(surveyType="DHS",all_results=FALSE)
dat <- dhs_surveys(surveyStatus="Surveys",all_results=FALSE)
dat <- dhs_surveys(surveyStatus="Completed",all_results=FALSE)
dat <- dhs_surveys(surveyStatus="Ongoing",all_results=FALSE)
dat <- dhs_surveys(surveyStatus="All",all_results=FALSE)
dat <- dhs_surveys(surveyCharacteristicIds="32",all_results=FALSE)
dat <- dhs_surveys(tagIds="1",all_results=FALSE)
dat <- dhs_surveys(f="html",all_results=FALSE)

## End(Not run)
```

`dhs_survey_characteristics`

API request of DHS Survey Characteristics

Description

API request of DHS Survey Characteristics

Usage

```
dhs_survey_characteristics(
  countryIds = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
indicatorIds	Specify a comma separated list of indicators ids to filter by. For a list of indicators use `dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))`
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use `dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))`
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Survey Characteristics on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Survey Characteristics on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by `client_dhs`
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a `data.table` of 2 (or less if `returnFields` is provided) survey characteristics. A survey can be labelled with one or more of these survey characteristics. A description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/surveycharacteristics/fields

Examples

```
## Not run:
# A good use for the survey characteristics API endpoint is to query what the
# IDs are for each survey characteristic. These are useful for passing as
# arguments to other API endpoints.For example to show all the ids:

dat <- dhs_survey_characteristics()

# Or if your analysis is foucssed on a particular country, and you want to
# see all the characteristics surveyed for e.g. Senegal

dat <- dhs_countries(countryIds="SN")

# A complete list of examples for how each argument to the survey
# characteristics API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-surveycharacteristics.cfm


dat <- dhs_survey_characteristics(countryIds="EG",all_results=FALSE)
dat <- dhs_survey_characteristics(indicatorIds="FE_FRTR_W_TFR",
all_results=FALSE)
dat <- dhs_survey_characteristics(surveyIds="SN2010DHS,all_results=FALSE")
dat <- dhs_survey_characteristics(surveyYear="2010,all_results=FALSE")
dat <- dhs_survey_characteristics(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_survey_characteristics(surveyYearStart="1991",
surveyYearEnd="2006",all_results=FALSE)
dat <- dhs_survey_characteristics(surveyType="DHS",all_results=FALSE)
dat <- dhs_survey_characteristics(f="html",all_results=FALSE)

## End(Not run)
```

`dhs_tags` *API request of DHS Tags*

Description

API request of DHS Tags

Usage

```
dhs_tags(
  countryIds = NULL,
  indicatorIds = NULL,
  surveyIds = NULL,
  surveyYear = NULL,
  surveyYearStart = NULL,
  surveyYearEnd = NULL,
  surveyType = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

countryIds	Specify a comma separated list of country ids to filter by. For a list of countries use `dhs_countries(returnFields=c("CountryName","DHS_CountryCode"))`
indicatorIds	Specify a comma separated list of indicators ids to filter by. For a list of indicators use `dhs_indicators(returnFields=c("IndicatorId","Label","Definition"))`
surveyIds	Specify a comma separated list of survey ids to filter by. For a list of surveys use `dhs_surveys(returnFields=c("SurveyId","SurveyYearLabel","SurveyType","CountryName"))`
surveyYear	Specify a comma separated list of survey years to filter by.
surveyYearStart	Specify a range of Survey Years to filter Tags on. surveyYearStart is an inclusive value. Can be used alone or in conjunction with surveyYearEnd.
surveyYearEnd	Specify a range of Survey Years to filter Tags on. surveyYearEnd is an inclusive value. Can be used alone or in conjunction with surveyYearStart.
surveyType	Specify a survey type to filter by.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by `client_dhs`
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a `data.table` of 4 (or less if `returnFields` is provided) tags with detailed information. An indicators can be tagged with one or more tags to help identify certain topics an indicator can be identified by. A description of the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/tags/fields

Examples

```
## Not run:
# A good use for the tags API endpoint is to query what the
# IDs are for each tag. These are useful for passing as
# arguments to other API endpoints.For example to show all the ids:

dat <- dhs_tags()

# Or if your analysis is foucssed on a particular country, and you want to
# see all the characteristics surveyed for e.g. Senegal

dat <- dhs_tags(countryIds="SN")

# A complete list of examples for how each argument to the survey
# tags API endpoint can be provided is given below, which is a
# copy of each of the examples listed in the API at:

# https://api.dhsprogram.com/#/api-tags.cfm


dat <- dhs_tags(countryIds="EG",all_results=FALSE)
dat <- dhs_tags(indicatorIds="FE_FRTR_W_TFR",all_results=FALSE)
dat <- dhs_tags(surveyIds="SN2010DHS",all_results=FALSE)
dat <- dhs_tags(surveyYear="2010",all_results=FALSE)
dat <- dhs_tags(surveyYearStart="2006",all_results=FALSE)
dat <- dhs_tags(surveyYearStart="1991", surveyYearEnd="2006",
all_results=FALSE)
dat <- dhs_tags(surveyType="DHS",all_results=FALSE)
dat <- dhs_tags(f="html",all_results=FALSE)

## End(Not run)
```

`dhs_ui_updates` *API request of DHS UI Updates*

Description

API request of DHS UI Updates

Usage

```
dhs_ui_updates(
  lastUpdate = NULL,
  f = NULL,
  returnFields = NULL,
  perPage = NULL,
  page = NULL,
  client = NULL,
  force = FALSE,
  all_results = TRUE
)
```

Arguments

lastUpdate	Specify a date or Unix time to filter the updates by. Only results for interfaces that has been updated on or after the specified date will be returned.
f	You can specify the format of the data returned from the query as HTML, JSON, PJSON, geoJSON, JSONP, XML or CSV. The default data format is JSON.
returnFields	Specify a list of attributes to be returned.
perPage	Specify the number of results to be returned per page. By default the API will return 100 results.
page	Allows specifying a page number to obtain for the API request. By default the API will return page 1.
client	If the API request should be cached, then provide a client object created by client_dhs
force	Should we force fetching the API results, and ignore any cached results we have. Default = FALSE
all_results	Boolean for if all results should be returned. If FALSE then the specified page only will be returned. Default = TRUE.

Value

Returns a data.table of 3 (or less if returnFields is provided) interfaces that have been added/updated or removed. A detailed description of all the attributes returned is provided at https://api.dhsprogram.com/rest/dhs/uiupdates/fields

Examples

```
## Not run:
# The main use for the ui updates API will be to search for the last time
# there was a change to the UI. For example to return all the
# changes since 2018:

dat <- dhs_ui_updates(lastUpdate="20180101")

# A complete list of examples for how each argument to the ui updates API
# endpoint can be provided is given below, which is a copy of each of
```

```
# the examples listed in the API at:

# https://api.dhsprogram.com/#/api-uiupdates.cfm


dat <- dhs_ui_updates(lastUpdate="20150901",all_results=FALSE)
dat <- dhs_ui_updates(f="html",all_results=FALSE)

## End(Not run)
```

`download_boundaries` *DHS Spatial Boundaries*

Description

Download Spatial Boundaries

Usage

```
download_boundaries(
  surveyNum = NULL,
  surveyId = NULL,
  countryId = NULL,
  method = "sf",
  quiet_download = FALSE,
  quiet_parse = TRUE,
  server_sleep = 5,
  client = NULL
)
```

Arguments

`surveyNum`	Numeric for the survey number to be downloaded. Values for surveyNum can be found in the datasets or surveys endpoints in the DHS API that can be accessed using `dhs_datasets` and `dhs_surveys`. Default is NULL, which will cause the SurveyId to be used to find the survey.
`surveyId`	Numeric for the survey ID to be downloaded. Values for surveyId can be found in the datasets or surveys endpoints in the DHS API that can be accessed using `dhs_datasets` and `dhs_surveys`. Default is NULL, which will cause the SurveyNum to be used to find the survey.
`countryId`	2-letter DHS country code for the country of the survey being downloaded. Default = NULL, which will cause the countrycode to be looked up from the API.
`method`	Character for how the downloaded shape file is read in. Default = "sf", which uses `sf::st_read`. Currently, this is the only option available(‘rgdal‘ used to be available) but for development reasons this will be left as a parameter option for possible future alternatives to read in spatial files. To just return the file paths for the files use method = "zip".

quiet_download	Whether to download file quietly. Passed to [‘download_file()‘]. Default is ‘FALSE‘.
quiet_parse	Whether to read boundaries dataset quietly. Applies to ‘method = "sf"‘. Default is ‘TRUE‘.
server_sleep	Numeric for length of sleep prior to downloading file from their survey. Default 5 seconds.
client	If the request should be cached, then provide a client object created by client_dhs. Default = ‘NULL‘, which will search for a client to use

Details

Downloads the spatial boundaries from the DHS spatial repository, which can be found at https://spatialdata.dhsprogram.com/home/.

Value

Returns either the spatial file as a ‘sf‘ (see [sf::sf]) object, or a vector of the file paths of where the boundary was downloaded to.

Examples

```
## Not run:
 # using the surveyNum
 res <- download_boundaries(surveyNum = 471, countryId = "AF")

 # using the surveyId and no countryID
 res <- download_boundaries(surveyId = "AF2010OTH")


## End(Not run)
```

---

download_datasets *Create a data frame of datasets that your log in can download*

Description

Download datasets specified using output of available_datasets.

Usage

```
download_datasets(
  config,
  desired_dataset,
  download_option = "both",
  reformat = TRUE,
  all_lower = TRUE,
  output_dir_root = NULL,
  ...
)
```

Arguments

config	Object of class ‘rdhs_config‘ as produced by ‘read_rdhs_config‘ that must contain a valid ‘email‘, ‘project‘ and ‘password‘.
desired_dataset	Row from available_datasets
download_option	Character dictating how the survey is stored when downloaded. Must be one of: • "zip" - Just the zip. "z", "i", "p" or "zip" will match • "rds" - Just the read in and saved rds. "r", "d", "s" or "rdhs" will match • "both" - Both the rds and extract. "b", "o", "t", "h" or "both" will match
reformat	Boolean detailing whether dataset rds should be reformatted for ease of use later. Default = TRUE
all_lower	Logical indicating whether all value labels should be lower case. Default to ‘TRUE‘.
output_dir_root	Directory where files are to be downloaded to
...	Any other arguments to be passed to read_dhs_dataset

---

extraction *DHS survey questions extracted from datasets*

Description

Create a list of survey responses extracted using output of R6_client_dhs$public_methods$survey_questions

Usage

```
extraction(questions, available_datasets, geo_surveys, add_geo = FALSE)
```

Arguments

questions	Output of R6_client_dhs$public_methods$survey_questions
available_datasets	Datasets that could be available. Output of R6_client_dhs$public_methods$available_datasets
geo_surveys	Geographic Data Survey file paths.
add_geo	Boolean detailing if geographic datasets should be added.

Value

Returns ‘data.frame‘ with variables corresponding to the requested variables in the questions object. Will also have geographic data related columns if ‘add_geo=TRUE‘ is set. Lastly a SurveyId variable will also be appended corresponding to dhs_datasets$SurveyId

extract_dhs *Extract Data*

Description

Extracts data from your downloaded datasets according to a data.frame of requested survey variables or survey definitions

Usage

```
extract_dhs(questions, add_geo = FALSE)
```

Arguments

questions	Questions to be queried, in the format from search_variables or search_variable_labels
add_geo	Add geographic information to the extract. Defaut = 'TRUE'

Details

Function to extract datasets using a set of survey questions as taken from the output from search_variables or search_variable_labels

Value

A list of 'data.frames' for each survey data extracted.

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download one of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1])

# create some terms of data me may want to extrac
st <- search_variable_labels(names(g), "bed net")

# and now extract it
ex <- extract_dhs(st)

## End(Not run)
```

factor_format *reformat haven and labelled read ins to have no factors or labels*

Description

reformat haven and labelled read ins to have no factors or labels

Usage

```
factor_format(res, reformat = FALSE, all_lower = TRUE)
```

Arguments

res	dataset to be formatted
reformat	Boolean whether to remove all factors and labels and just return the unfactored data. Default = FALSE
all_lower	Logical indicating whether all value labels should be lower case. Default to 'TRUE'.

Value

list with the formatted dataset and the code descriptions

file_dataset_format *Returns what the dataset file ending should be for a given filename*

Description

Returns what the dataset file ending should be for a given filename

Usage

```
file_dataset_format(file_format)
```

Arguments

file_format	FileFormat for a file as taken from the API, e.g. dhs_datasets(returnFields = "FileFormat")

Value

One of "dat","dat","sas7bdat","sav" or "dta"

Examples

```
file_format <- "Stata dataset (.dta)"
identical(rdhs:::file_dataset_format(file_format),"dta")
```

`get_available_datasets` *Get Available Datasets*

Description

Details the datasets that your login credentials have access to

Usage

```
get_available_datasets(clear_cache = FALSE)
```

Arguments

`clear_cache`	Boolean detailing if you would like to clear the cached available datasets first. The default is set to FALSE. This option is available so that you can make sure your client fetches any new datasets that you have recently been given access to.

Details

Searches the DHS website for all the datasets that you can download. The results of this function are cached in the client. If you have recently requested new datasets from the DHS website then you can specify to clear the cache first so that you get the new set of datasets available to you. This function is used by `get_datasets` and should thus be used with 'clear_cache_first = TRUE' before using 'get_datasets' if you have recently requested new datasets.

Value

A `data.frame` with 14 variables that detail the surveys you can download, their url download links and the country, survey, year etc info for that link.

Examples

```
## Not run:
# grab the datasets
datasets <- get_available_datasets()

# and if we look at the last one it will be the model datasets from DHS
tail(datasets, 1)

## End(Not run)
```

`get_datasets` *Get Datasets*

Description

Downloads datasets you have access to from the DHS website

Usage

```
get_datasets(
  dataset_filenames,
  download_option = "rds",
  reformat = FALSE,
  all_lower = TRUE,
  output_dir_root = NULL,
  clear_cache = FALSE,
  ...
)
```

Arguments

`dataset_filenames`	The desired filenames to be downloaded. These can be found as one of the returned fields from `dhs_datasets`. Alternatively you can also pass the desired rows from `dhs_datasets`.
`download_option`	Character specifying whether the dataset should be just downloaded ("zip"), imported and saved as an .rds object ("rds"), or both extract and rds ("both"). Conveniently you can just specify any letter from these options.
`reformat`	Boolean concerning whether to reformat read in datasets by removing all factors and labels. Default = FALSE.
`all_lower`	Logical indicating whether all value labels should be lower case. Default to 'TRUE'.
`output_dir_root`	Root directory where the datasets will be stored within. The default will download datasets to a subfolder of the client root called "datasets"
`clear_cache`	Should your available datasets cache be cleared first. This will allow newly accessed datasets to be available. Default = 'FALSE'
`...`	Any other arguments to be passed to `read_dhs_dataset`

Details

Gets datasets from your cache or downloads from the DHS website. By providing the filenames, as specified in one of the returned fields from `dhs_datasets`, the client will log in for you and download all the files you have requested. If any of the requested files are unavailable for your log in, these will be flagged up first as a message so you can make a note and request them through

the DHS website. You also have the option to control whether the downloaded zip file is then extracted and converted into a more convenient R `data.frame`. This converted object will then be subsequently saved as a ".rds" object within the client root directory datasets folder, which can then be more quickly loaded when needed with `readRDS`. You also have the option to reformat the dataset, which will ensure that the datasets returned are encoded simply as character strings, i.e. there are no factors or labels.

Value

Depends on the download_option requested, but ultimately it is a file path to where the dataset was downloaded to, so that you can interact with it accordingly.

Examples

```
 ## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download one of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1])

## End(Not run)
```

`get_downloaded_datasets` *Get Downloaded Datasets*

Description

Detail the datasets that you have already downloaded

Usage

```
get_downloaded_datasets()
```

Details

Returns a `data.frame` of the datasets that have been downloaded within this client. This could be useful if you are without an internet connection and wish to know which saved dataset files in your root directory correspond to which dataset

Value

A `data.frame` of downloaded datasets

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download one of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1])

# these will then be stored so that we know what datasets we have downloaded
d <- get_downloaded_datasets()

# which returns a names list of file paths to the datasets
d[1]

## End(Not run)
```

`get_labels_from_dataset` *Return variable labels from a dataset*

Description

Returns variable labels stored as `"label"` attribute.

Usage

```
get_labels_from_dataset(data, return_all = TRUE)
```

Arguments

`data`	A `data.frame` from which to extract variable labels.
`return_all`	Logical whether to return all variables (`TRUE`) or only those with labels.

Value

A `data.frame` consisting of the variable name and labels.

get_rdhs_config *Get rdhs config*

Description

Gets the rdhs config being used

Usage

```
get_rdhs_config()
```

Details

Returns the config being used by rdhs at the moment. This will either be a ‘data.frame‘ with class ‘rdhs_config‘ or will be NULL if this has not been set up yet

Value

A `data.frame` containing your rdhs config

get_variable_labels *Get Survey Variable Labels*

Description

Return variable labels from a dataset

Usage

```
get_variable_labels(dataset, return_all = TRUE)
```

Arguments

`dataset`	Can be either the file path to a dataset, the dataset as a ‘data.frame‘ or the filenames of datasets. See details for more information
`return_all`	Logical whether to return all variables (`TRUE`) or only those with labels.

Details

Returns variable names and their labels from a dataset. You can pass for the ‘data‘ argument any of the following:

- The file path to a saved dataset. This would be the direct output of `get_datasets`
- A read in dataset, i.e. produced by using `readRDS` to load a dataset from a file path produced by `get_datasets`
- Dataset filenames. These can be found as one of the returned fields from `dhs_datasets`. If these datasets have not been downloaded before this will download them for you.

Value

A data.frame consisting of the variable name and labels.

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download one of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1])

# we can pass the list of filepaths to the function
head(get_variable_labels(g))

# or we can pass the full dataset
r <- readRDS(g[[1]])
head(get_variable_labels(r))

## End(Not run)
```

---

last_api_update *Pull last DHS API database update time*

Description

Pull last DHS API database update time

Usage

```
last_api_update(timeout = 30)
```

Arguments

timeout	Numeric for API timeout. Default = 30

---

model_datasets *DHS model datasets*

Description

The model datasets from the DHS website in a ‘data.frame‘ that is analogous to those returned by ‘get_available_datasets()‘

Usage

```
data(model_datasets)
```

Format

A dataframe of 36 observations of 14 variables:

`model_datasets`: A dataframe of model datasets

- "FileFormat"
- "FileSize"
- "DatasetType"
- "SurveyNum"
- "SurveyId"
- "FileType"
- "FileDateLastModified"
- "SurveyYearLabel"
- "SurveyType"
- "SurveyYear"
- "DHS_CountryCode"
- "FileName"
- "CountryName"
- "URLS"

`parse_map` *Create dictionary from DHS .MAP codebook*

Description

Create dictionary from DHS .MAP codebook

Usage

```
parse_map(map, all_lower = TRUE)
```

Arguments

`map`	A character vector containing .MAP file, e.g. from ‘readLines()‘.
`all_lower`	Logical indicating whether all value labels should be converted to lower case

Details

Currently hardcoded for 111 char width .MAP files, which covers the vast majority of DHS Phase V, VI, and VIII. To be extended in the future and perhaps add other useful options.

Value

A data frame containing metadata, principally variable labels and a vector of value labels.

Examples

```
mrdt_zip <- tempfile()
download.file("https://dhsprogram.com/data/model_data/dhs/zzmr61fl.zip",
              mrdt_zip, mode="wb")

map <- rdhs::read_zipdata(mrdt_zip, "\\.MAP", readLines)
dct <- rdhs:::parse_map(map)
```

`parse_meta` *Parse fixed-width file metadata*

Description

Parse dataset metadata

Usage

```
parse_dcf(dcf, all_lower = TRUE)

parse_sps(sps, all_lower = TRUE)

parse_do(do, dct, all_lower = TRUE)
```

Arguments

`dcf`	.DCF file path to parse
`all_lower`	logical indicating whether to convert variable labels to lower case. Defaults to 'TRUE'.
`sps`	.SPS file as character vector (e.g. from readLines / brio::read_lines)
`do`	.DO file as character vector (e.g. from readLines / brio::read_lines)
`dct`	.DCT file as character vector (e.g. from readLines / brio::read_lines)

Value

data.frame with metadata for parsing fixed-width flat file

Examples

```
mrfl_zip <- tempfile()
download.file("https://dhsprogram.com/data/model_data/dhs/zzmr61fl.zip",
              mrfl_zip, mode = "wb")

dcf <- rdhs::read_zipdata(mrfl_zip, "\\.DCF", readLines)
dct <- rdhs:::parse_dcf(dcf)

sps <- rdhs::read_zipdata(mrfl_zip, "\\.SPS", readLines)
```

```
dct <- rdhs:::parse_sps(sps)

do <- rdhs::read_zipdata(mrfl_zip, "\\.DO", readLines)
dctin <- rdhs::read_zipdata(mrfl_zip, "\\.DCT", readLines)
dct <- rdhs:::parse_do(do, dctin)
```

`rbind_labelled` *Combine data frames with columns of class 'labelled'*

Description

Combine data frames with columns of class 'labelled'

Usage

```
rbind_labelled(..., labels = NULL, warn = TRUE)
```

Arguments

`...`	data frames to bind together, potentially with columns of class "labelled". The first argument can be a list of data frames, similar to 'plyr::rbind.fill'.
`labels`	A named list providing vectors of value labels or describing how to handle columns of class 'labelled'. See details for usage.
`warn`	Logical indicating to warn if combining variables with different value labels. Defaults to TRUE.

Details

The argument 'labels' provides options for how to handle binding of columns of class 'labelled'. Typical use is to provide a named list with elements for each labelled column. Elements of the list are either a vector of labels that should be applied to the column or the character string "concatenated", which indicates that labels should be concatenated such that all unique labels are distinct values in the combined vector. This is accomplished by converting to character strings, binding, and then casting back to labelled. For labelled columns for which labels are not provided in the 'label' argument, the default behaviour is that the labels from the first data frame with labels for that column are inherited by the combined data.

See examples.

Value

A data frame.

Examples

```
df1 <- data.frame(
area = haven::labelled(c(1L, 2L, 3L), c("reg 1"=1,"reg 2"=2,"reg 3"=3)),
climate = haven::labelled(c(0L, 1L, 1L), c("cold"=0,"hot"=1))
)
df2 <- data.frame(
area    = haven::labelled(c(1L, 2L), c("reg A"=1, "reg B"=2)),
climate = haven::labelled(c(1L, 0L), c("cold"=0, "warm"=1))
)

# Default: all data frames inherit labels from first df. Incorrect if
# "reg 1" and "reg A" are from different countries, for example.
dfA <- rbind_labelled(df1, df2)
haven::as_factor(dfA)

# Concatenate value labels for "area". Regions are coded separately,
# and original integer values are lost (by necessity of more levels now).
# For "climate", codes "1 = hot" and "1 = warm", are coded as the same
# outcome, inheriting "1 = hot" from df1 by default.
dfB <- rbind_labelled(df1, df2, labels=list(area = "concatenate"))
dfB
haven::as_factor(dfB)

# We can specify to code as "1=warm/hot" rather than inheriting "hot".
dfC <- rbind_labelled(df1, df2,
labels=list(area = "concatenate", climate = c("cold"=0, "warm/hot"=1)))

dfC$climate
haven::as_factor(dfC)

# Or use `climate="concatenate"` to code "warm" and "hot" as different.
dfD <- rbind_labelled(df1, df2,
labels=list(area = "concatenate", climate="concatenate"))

dfD
haven::as_factor(dfD)
```

---

rbind_list_base *implementation of data.tables rbindlist*

Description

implementation of data.tables rbindlist

Usage

```
rbind_list_base(x)
```

Arguments

x	List of lists to be converted to a data.frame

`rdhs` **rdhs** *DHS database through R*

Description

Provides a client for (1) querying the DHS API for survey indicators and metadata, (2) identifying surveys and datasets for analysis, (3) downloading survey datasets from the DHS website, (4) loading datasets and associate metadata into R, and (5) extracting variables and combining datasets for pooled analysis.

Author(s)

Maintainer: OJ Watson <oj.watson@hotmail.co.uk> (ORCID)

Authors:

- Jeff Eaton (ORCID)

Other contributors:

- Lucy D'Agostino McGowan (ORCID) [reviewer]
- Duncan Gillespie [reviewer]

See Also

Useful links:

- https://docs.ropensci.org/rdhs/
- Report bugs at https://github.com/ropensci/rdhs/issues

`read_dhs_dataset` *read in dhs standard file types*

Description

read in dhs standard file types

Usage

```
read_dhs_dataset(file, dataset, reformat = FALSE, all_lower = TRUE, ...)
```

Arguments

file	path to zip file to be read
dataset	row from dhs_datasets that corresponds to the file
reformat	boolean detailing if datasets should be nicely reformatted. Default = 'FALSE'
all_lower	Logical indicating whether all value labels should be lower case. Default to 'TRUE'.
...	Extra arguments to be passed to either read_dhs_dta or read_dhs_flat

---

read_dhs_dta *Read DHS Stata data set*

Description

This function reads a DHS recode dataset from the zipped Stata dataset. By default ('mode = "haven"'), it reads in the stata data set using read_dta

Usage

```
read_dhs_dta(zfile, mode = "haven", all_lower = TRUE, ...)
```

Arguments

zfile	Path to '.zip' file containing Stata dataset, usually ending in filename 'XXXXXXDT.zip'
mode	Read mode for Stata '.dta' file. Defaults to "haven", see 'Details' for other options.
all_lower	Logical indicating whether all value labels should be lower case. Default to 'TRUE'.
...	Other arguments to be passed to read_zipdata. Here this will be arguments to pass to either read_dta or read.dta depending on the mode provided

Details

The default 'mode="haven"' uses read_dta to read in the dataset. We have chosen this option as it is more consistent with respect to variable labels and descriptions than others. The other options either use use read.dta or they use the '.MAP' dictionary file provided with the DHS Stata datasets to reconstruct the variable labels and value labels. In this case, value labels are stored are stored using the the 'labelled' class from 'haven'. See '?haven::labelled' for more information. Variable labels are stored in the "label" attribute of each variable, the same as 'haven::read_dta()'.

Currently, 'mode="map"' is only implemented for 111 character fixed-width .MAP files, which comprises the vast majority of recode data files from DHS Phases V, VI, and VII and some from Phase IV. Parsers for other .MAP formats will be added in future.

Other available modes read labels from the Stata dataset with various options available in R:

* 'mode="map"' uses the '.MAP' dictionary file provided with the DHS Stata datasets to reconstruct the variable labels and value labels. In this case, value labels are stored are stored using the the

'labelled' class from 'haven'. See '?haven::labelled' for more information. Variable labels are stored in the "label" attribute of each variable, the same as 'haven::read_dta()'.

* 'mode="haven"': use 'haven::read_dta()' to read dataset. This option retains the native value codings with value labels affixed with the 'labelled' class.

* 'mode="foreign"': use 'foreign::read.dta()', with default options convert.factors=TRUE to add variable labels. Note that variable labels will not be added if labels are not present for all values, but variable labels are available via the "val.labels" attribute.

* 'mode="foreignNA"': use 'foreign::read.dta(..., convert.factors=NA)', which converts any values without labels to 'NA'. This risks data loss if labelling is incomplete in Stata datasets.

* 'mode="raw"': use 'foreign::read.dta(..., convert.factors=FALSE)', which simply loads underlying value coding. Variable labels and value labels are still available through dataset attributes (see examples).

Value

A data frame. If mode = 'map', value labels for each variable are stored as the 'labelled' class from 'haven'.

See Also

read.dta, labelled, read_dta.

For more information on the DHS filetypes and contents of distributed dataset .ZIP files, see https://dhsprogram.com/data/File-Types-and-Names.cfm#CP_JUMP_10334.

Examples

```
mrdt_zip <- tempfile()
download.file("https://dhsprogram.com/data/model_data/dhs/zzmr61dt.zip",
              mrdt_zip, mode="wb")

mr <- rdhs::read_dhs_dta(mrdt_zip,mode="map")
attr(mr$mv213, "label")
class(mr$mv213)
head(mr$mv213)
table(mr$mv213)
table(haven::as_factor(mr$mv213))

## If Stata file codebook is complete, `mode="map"` and `"haven"`
## should be the same.
mr_hav <- rdhs::read_dhs_dta(mrdt_zip, mode="haven")
attr(mr_hav$mv213, "label")
class(mr_hav$mv213)
head(mr_hav$mv213)  # "9=missing" omitted from .dta codebook
table(mr_hav$mv213)
table(haven::as_factor(mr_hav$mv213))

## Parsing codebook when using foreign::read.dta()
# foreign issues with duplicated factors
# Specifying foreignNA can help but often will not as below.
# Thus we would recommend either using mode = "haven" or mode = "raw"
```

```
## Not run:
mr_for <- rdhs::read_dhs_dta(mrdt_zip, mode="foreign")
mr_for <- rdhs::read_dhs_dta(mrdt_zip, mode = "foreignNA")

## End(Not run)
## Don't convert factors
mr_raw <- rdhs::read_dhs_dta(mrdt_zip, mode="raw")
table(mr_raw$mv213)
```

`read_dhs_flat` *Read DHS flat file data set*

Description

This function reads a DHS recode dataset from the zipped flat file dataset.

Usage

```
read_dhs_flat(zfile, all_lower = TRUE, meta_source = NULL)
```

Arguments

`zfile`	Path to '.zip' file containing flat file dataset, usually ending in filename 'XXXXXXFL.zip'
`all_lower`	Logical indicating whether all value labels should be lower case. Default to 'TRUE'.
`meta_source`	character string indicating metadata source file for data dictionary. Default `NULL` first tried to use `.DCF` and then `.SPS` if not found.

Value

A data frame. Value labels for each variable are stored as the 'labelled' class from 'haven'.

See Also

`labelled`, `read_dhs_dta`.

For more information on the DHS filetypes and contents of distributed dataset .ZIP files, see https://dhsprogram.com/data/File-Types-and-Names.cfm#CP_JUMP_10334.

Examples

```
mrfl_zip <- tempfile()
download.file("https://dhsprogram.com/data/model_data/dhs/zzmr61fl.zip",
              mrfl_zip,mode="wb")

mr <- rdhs:::read_dhs_flat(mrfl_zip)
attr(mr$mv213, "label")
class(mr$mv213)
```

```
head(mr$mv213)
table(mr$mv213)
table(haven::as_factor(mr$mv213))
```

`read_zipdata` *Read filetype from a zipped folder based on the file ending*

Description

Read filetype from a zipped folder based on the file ending

Usage

```
read_zipdata(zfile, pattern = ".dta$", readfn = haven::read_dta, ...)
```

Arguments

`zfile`	Path to '.zip' file containing flat file dataset, usually ending in filename 'XXXXXXFL.zip'
`pattern`	String detailing which filetype is to be read from within the zip by means of a grep. Default = ".dta$"
`readfn`	Function object to be used for reading in the identified file within the zip. Default = 'haven::read_dta'
`...`	additional arguments to readfn

Examples

```
## Not run:
# get the model datasets included in the package
model_datasets <- model_datasets

# download just the zip
g <- get_datasets(
dataset_filenames = model_datasets$FileName[1],
download_option = "zip"
)

# and then read from the zip. This function is used internally by rdhs
# when using `get_datasets` with `download_option = .rds` (default)
r <- read_zipdata(
g[[1]], pattern = ".dta"
)

# and we can pass a function to read the file and any other args with ...
r <- read_zipdata(
g[[1]], pattern = ".dta", readfn = haven::read_dta, encoding = "UTF-8"
)

## End(Not run)
```

response_is_json *checks if the response is json or not by looking at the responses headers*

Description

checks if the response is json or not by looking at the responses headers

Usage

```
response_is_json(x)
```

Arguments

x	A response

response_to_json *converts response to json by first converting the response to text*

Description

converts response to json by first converting the response to text

Usage

```
response_to_json(x)
```

Arguments

x	A response

search_variables *Search Survey Variables*

Description

Searches across datasets specified for requested survey variables. This function (or search_variable_labels) should be used to provide the 'questions' argument for extract_dhs.

Usage

```
search_variables(dataset_filenames, variables, essential_variables = NULL, ...)
```

Arguments

dataset_filenames	The desired filenames to be downloaded. These can be found as one of the returned fields from dhs_datasets.
variables	Character vector of survey variables to be looked up
essential_variables	Character vector of variables that need to present. If any of the codes are not present in that survey, the survey will not be returned by this function. Default = 'NULL'.
...	Any other arguments to be passed to download_datasets

Details

Use this function after get_datasets to look up all the survey variables that have the required variable.

Value

A data.frame of the surveys where matches were found and then all the resultant codes and descriptions.

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download two of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1:2])

# and now seearch within these for survey variables
search_variables(
dataset_filenames = names(g), variables = c("v002","v102","ml13"),
)

# if we specify an essential variable then that dataset has to have that
# variable or else no variables will be returned for that datasets
search_variables(
dataset_filenames = names(g),
variables = c("v002","v102","ml13"),
essential_variables = "ml13"
)

## End(Not run)
```

search_variable_labels *Search Survey Variable Definitions*

Description

Searches across datasets specified for requested survey variable definitions. This function (or search_variable_labels) should be used to provide the 'questions' argument for extract_dhs.

Usage

```
search_variable_labels(
  dataset_filenames,
  search_terms = NULL,
  essential_terms = NULL,
  regex = NULL,
  ...
)
```

Arguments

dataset_filenames	The desired filenames to be downloaded. These can be found as one of the returned fields from dhs_datasets.
search_terms	Character vector of search terms. If any of these terms are found within the survey question definitions, the corresponding survey variable and definitions will be returned.
essential_terms	Character pattern that has to be in the definitions of survey question definitions. I.e. the function will first find all survey variable definitions that contain your 'search_terms' (or regex) OR 'essential_terms'. It will then remove any questions that did not contain your 'essential_terms'. Default = 'NULL'.
regex	Regex character pattern for matching. If you want to specify your regex search pattern, then specify this argument. N.B. If both 'search_terms' and 'regex" are supplied as arguments then regex will be ignored.
...	Any other arguments to be passed to download_datasets

Details

Use this function after get_datasets to query downloaded datasets for what survey questions they asked. This function will look for your downloaded and imported survey datasets from your cached files, and will download them if not downloaded.

Value

A data.frame of the surveys where matches were found and then all the resultant codes and descriptions.

Examples

```
## Not run:
# get the model datasets included with the package
model_datasets <- model_datasets

# download two of them
g <- get_datasets(dataset_filenames = model_datasets$FileName[1:2])

# and now seearch within these for survey variable labels of interest
vars <- search_variable_labels(
dataset_filenames = names(g), search_terms = "fever"
)

head(vars)

# if we specify an essential term then no results will be returned from
# a dataset if it does not have any results from the search with this term
search_variable_labels(
dataset_filenames = names(g),
search_terms = "fever",
essential_terms = "primaquine",
)

# we can also use regex queries if we prefer, by passing `regex = TRUE`
vars <- search_variable_labels(
dataset_filenames = names(g), search_terms = "fever|net", regex = TRUE
)

## End(Not run)
```

---

set_rdhs_config *Set rdhs config*

Description

Sets the configuration settings for using rdhs.

Usage

```
set_rdhs_config(
  email = NULL,
  project = NULL,
  cache_path = NULL,
  config_path = NULL,
  global = TRUE,
  verbose_download = FALSE,
  verbose_setup = TRUE,
  data_frame = NULL,
  timeout = 30,
```

```
  password_prompt = FALSE,
  prompt = TRUE
)
```

Arguments

email	Character for email used to login to the DHS website.
project	Character for the name of the DHS project from which datasets should be downloaded.
cache_path	Character for directory path where datasets and API calls will be cached. If left bank, a suitable directory will be created within your user cache directory for your operating system (permission granting).
config_path	Character for where the config file should be saved. For a global configuration, 'config_path' must be '~/.rdhs.json'. For a local configuration, 'config_path' must be 'rdhs.json'. If left bank, the config file will be stored within your user cache directory for your operating system (permission granting).
global	Logical for the config_path to be interpreted as a global config path or a local one. Default = TRUE.
verbose_download	Logical for dataset download progress bars to be shown. Default = FALSE.
verbose_setup	Logical for rdhs setup and messages to be printed. Default = TRUE.
data_frame	Function with which to convert API calls into. If left blank `data_frame` objects are returned. Must be passed as a character. Examples could be: `data.table::as.data.table` `tibble::as.tibble`
timeout	Numeric for how long in seconds to wait for the DHS API to respond. Default = 30.
password_prompt	Logical whether user is asked to type their password, even if they have previously set it. Default = FALSE. Set to TRUE if you have mistyped your password when using `set_rdhs_config`.
prompt	Logical for whether the user should be prompted for permission to write to files. This should not need be changed by the user. Default = TRUE.

Details

Setting up a configuration will enable API results to be cached, as well as enabling datasets from the DHS website to be downloaded and also cached. To enable results to be cached you have to either provide a valid 'cache_path' argument, or allow rdhs to write to the user cache directory for your operating system. To do the later, leave the 'cache_path' argument blank and you will be explicitly prompted to give permission to 'rdhs' to save your results in this directory. If you do not then your API calls and any downloaded datasets will be saved in the temp directory and deleted after your R session closes. To allow 'rdhs' to download datasets from the DHS website, you have to provide both an 'email' and 'project' argument. You will then be prompted to type in your login password securely. Your provided config (email, project, password, cache_path etc) will be saved at the location provided by 'config_path'. If no argument is provided 'config_path' will be either set to within your user cache directory if you have given permission to do so, otherwise it will be placed within your temp directory.

When creating your config you also have the option to specify whether the ‘config_path‘ provided should be used as a local configuration or a global one. This is controlled using the ‘global‘ argument, which by default is set equal to ‘TRUE‘. A global config is saved within your R root directory (the directory that a new R session will start in). If you set ‘global‘ to ‘FALSE‘ the config file will be saved within the current directory. This can be useful if you create a new DHS project for each new piece of work, and want to keep the datasets you download for this project separate to another. If you want to have your config file saved in a different directory, then you must create a file "rdhs.json" first in that directory before specifying the full path to it, as well as setting ‘global‘ equal to ‘FALSE‘.

As an aside, it is useful for the DHS program to see how the surveys they conducted are being used, and thus it is helpful for them if you do create a new project for each new piece of work (e.g. a different publication). However, we would still recommend setting up a global config and using the same ‘cache_path‘ for different projects as this will save you time downloading the same datasets as you have downloaded before.

Lastly, you can decide how API calls from the DHS API are formatted by providing an argument for ‘data_frame‘. If left blank API calls will be returned as ‘data.frame‘ objects, however, you could return API calls as ‘data.table‘ objects using ‘data.table::as.data.table‘.

Value

Invisibly returns the rdhs config object

Examples

```
## Not run:
# normal set up we would prvide the email and project, and be prompted for
# the password. (not run as it requires a prompt)
set_rdhs_config(email = "blah@gmail.com", project = "Blahs",
config_path = "rdhs.json", global = FALSE)


# otherwise we can do this by specifying prompt to FALSE
set_rdhs_config(
config_path = "rdhs.json", global = FALSE, prompt = FALSE
)

# you can look at what you have set these to using \code{get_rdhs_config}
config <- get_rdhs_config()

## End(Not run)
```

`unzip_special` *unzip special that catches for 4GB+*

Description

unzip special that catches for 4GB+

Usage

```
unzip_special(
  zipfile,
  files = NULL,
  overwrite = TRUE,
  junkpaths = FALSE,
  exdir = ".",
  unzip = "internal",
  setTimes = FALSE
)
```

Arguments

zipfile	The pathname of the zip file: tilde expansion (see path.expand will be performed.)
files	A character vector of recorded filepaths to be extracted: the default is to extract all files.
overwrite	If TRUE, overwrite existing files (the equivalent of unzip -o), otherwise ignore such files (the equivalent of unzip -n).
junkpaths	If TRUE, use only the basename of the stored filepath when extracting. The equivalent of unzip -j.
exdir	The directory to extract files to (the equivalent of unzip -d). It will be created if necessary.
unzip	The method to be used. An alternative is to use getOption("unzip"), which on a Unix-alike may be set to the path to a unzip program.
setTimes	logical. For the internal method only, should the file times be set based on the times in the zip file? (NB: this applies to included files, not to directories.)

---

update_rdhs_config *Update your current rdhs config*

Description

update_rdhs_config allows you to update elements of your rdhs config, without having to set it completely via set_rdhs_config. For each config element, provide the new changes required. To update your password, set password = TRUE and you will be asked securely for your new password.

Usage

```
update_rdhs_config(
  password = FALSE,
  email = NULL,
  project = NULL,
  cache_path = NULL,
  config_path = NULL,
```

```
  global = NULL,
  verbose_download = NULL,
  verbose_setup = NULL,
  timeout = NULL,
  data_frame = NULL,
  project_choice = NULL
)
```

Arguments

password	Logical for updating your password securely. Default = FALSE
email	Character for email used to login to the DHS website.
project	Character for the name of the DHS project from which datasets should be downloaded.
cache_path	Character for directory path where datasets and API calls will be cached. If left bank, a suitable directory will be created within your user cache directory for your operating system (permission granting).
config_path	Character for where the config file should be saved. For a global configuration, 'config_path' must be '~/.rdhs.json'. For a local configuration, 'config_path' must be 'rdhs.json'. If left bank, the config file will be stored within your user cache directory for your operating system (permission granting).
global	Logical for the config_path to be interpreted as a global config path or a local one. Default = TRUE.
verbose_download	Logical for dataset download progress bars to be shown. Default = FALSE.
verbose_setup	Logical for rdhs setup and messages to be printed. Default = TRUE.
timeout	Numeric for how long in seconds to wait for the DHS API to respond. Default = 30.
data_frame	Function with which to convert API calls into. If left blank `data_frame` objects are returned. Must be passed as a character. Examples could be: `data.table::as.data.table` `tibble::as.tibble`
project_choice	Numeric for project choice. See `authenticate_dhs` for more info.

Index