Package 'safedata'

May 5, 2023

Type Package

Title Interface to Data from the SAFE Project

Version 1.1.3

Date 2023-05-05

Description The SAFE Project (<https://www.safeproject.net/>) is a large scale ecological experiment in Malaysian Borneo that explores the impact of habitat fragmentation and conversion on ecosystem function and services. Data collected at the SAFE Project is made available under a common format through the Zenodo data repository and this package makes it easy to discover and load that data into R.

License GPL-3

Imports readxl, jsonlite, chron, curl, sf, httr, igraph

Suggests knitr, rmarkdown, ape, testthat (>= 2.1.0)

Encoding UTF-8

URL https://imperialcollegelondon.github.io/safedata/index.html

BugReports https://github.com/ImperialCollegeLondon/safedata/issues

RoxygenNote 7.2.0

VignetteBuilder knitr

NeedsCompilation no

Author Andy Aldersley [aut],
David Orme [aut, cre] (<https://orcid.org/0000-0002-7005-1394>)

Maintainer David Orme <d.orme@imperial.ac.uk>

Repository CRAN

Date/Publication 2023-05-05 15:00:02 UTC

R topics documented:

`add_locations` *Adds location data to a SAFE data worksheet.*

Description

This function adds location data to the rows of a SAFE data worksheet, converting it into a `sf` spatial features object. Rows are matched to values in a location_field: if there is only one location field, it will be used automatically; otherwise, the specific field must be provided.

Usage

```
add_locations(
  obj,
  location_field = NULL,
  location_table = NULL,
  gazetteer_info = FALSE
)
```

Arguments

`obj`	An existing object of class `safedata`
`location_field`	The name of a location field in a `safedata` object.
`location_table`	An existing location table for a dataset, as generated by `get_locations`.
`gazetteer_info`	Should all the gazetteer fields be included in the returned `sf` object. See `load_gazetteer` for details.

Value

A modified safedata object including geometry data.

See Also

get_locations, load_gazetteer, load_location_aliases

Examples

```
set_example_safe_dir()
beetle_abund <- load_safe_data(1400562, "Ant-Psel")
beetle_abund <- add_locations(beetle_abund)
unset_example_safe_dir()
```

---

add_taxa *Add a taxonomic hierarchy to a SAFE data worksheet.*

Description

All datasets containing taxon observations provide taxonomic data (see get_taxa) that can be linked to rows in data worksheets that contain "taxa" fields. This function matches the taxonomic hierarchy for a dataset for a given taxon field in a data worksheet and inserts fields to show that hierarchy.

Usage

```
add_taxa(
  obj,
  taxon_field = NULL,
  taxon_table = NULL,
  prefix = NULL,
  which = NULL
)
```

Arguments

obj	An existing object of class safedata
taxon_field	The name of a taxon field in a safedata for which to add taxonomic data.
taxon_table	An existing taxon table for a dataset, as generated by get_taxa.
prefix	A string to be appended to taxon field names, primarily to discriminate between fields of multiple taxonomies are to be added.
which	A vector specifying a subset of taxonomy table field names to add.

Details

An existing taxon table can be provided to the function, but the table will be automatically loaded if no table is provided. If a data worksheet only contains a single taxon field, then that field will be used automatically, otherwise users have to specify which taxon field to use. A prefix can be added to taxon fields in order to discrimate between fields from multuple taxon fields. By default, the function adds all of the fields included in the output of `get_taxa`, but `which` allows a subset of field names to be used.

Value

A modified `safedata` object with added taxonomic columns.

See Also

`get_taxa`

Examples

```
set_example_safe_dir()
beetle_morph <- load_safe_data(1400562, "MorphFunctTraits")
beetle_morph <- add_taxa(beetle_morph)
unset_example_safe_dir()
```

---

download_safe_files *Download SAFE dataset files*

Description

This downloads files associated with SAFE datasets, either all of the files included in a set of records (`xlsx_only = FALSE`) or just the core .xlsx files (`xlsx_only = FALSE`), and stores them in the SAFE data directory. See `insert_dataset` for details on using embargoed or restricted datasets.

Usage

```
download_safe_files(
  record_ids,
  confirm = TRUE,
  xlsx_only = TRUE,
  download_metadata = TRUE,
  refresh = FALSE,
  token = NULL
)
```

Arguments

record_ids	A vector of SAFE dataset record ids or a safe_record_set object.
confirm	Requires the user to confirm before download (logical)
xlsx_only	Should all files be downloaded or just the core .xslx file (logical)
download_metadata	Should the metadata record for the file be downloaded (logical)
refresh	Should the function check if local copies have been modified and download fresh copies. This is useful if the local copies have unintentionally been modified but note the warning above.
token	An access token for restricted datasets. These tokens are requested through the Zenodo page for a restricted dataset and are long alphanumeric strings. If you are providing a token, you should only provide the record id for that dataset.

Details

By default, the function will also download the dataset metadata. This information is required by many of the functions in the package but users can turn off automatic metadata download.

Value

Invisibly, a vector of paths within the 'safe_dir' for successfully downloaded files. If the download is not successful then the function returns FALSE.

Warning

Using refresh = TRUE will **overwrite locally modified files** and replace them with the versions of record from Zenodo.

Examples

```
set_example_safe_dir()
recs <- validate_record_ids(c(3247631, 3266827, 3266821))
download_safe_files(recs, confirm = FALSE)
unset_example_safe_dir()

## Not run:
    # This example requires a private token
    download_safe_files(1237730, confirm = FALSE,
                        token="longStringFromZenodo")

## End(Not run)
```

get_file_details *Get file details for a record*

Description

This function returns a list of the files associated with a record. The access status of the record is reported, along with the local absolute file path and whether a local copy is present. If there is no local copy and the dataset is open access, you can use download_safe_files to get local copies. If the dataset is embargoed or restricted, see insert_dataset.

Usage

```
get_file_details(record_id)
```

Arguments

record_id	A record id

Value

A data frame with fields: filename, dataset_access, path and local.

Examples

```
set_example_safe_dir()
files <- get_file_details(1400562)
unset_example_safe_dir()
```

get_locations *Obtaining locations for a SAFE dataset.*

Description

This function returns a spatial 'simple features' (sf) object for a dataset record, including basic information about locations for the SAFE gazetteer if requested.

Usage

```
get_locations(obj, gazetteer_info = FALSE)
```

Arguments

obj	A single record id, or an existing safedata dataframe.
gazetteer_info	Should all the gazetteer fields be included in the returned sf object.

Details

All SAFE datasets recording observations from the field must include a Locations worksheet, which must contain all of the location names used in Locations type fields in data worksheets. These entries are matched against existing location names in the SAFE gazetteer. Users can also include "new" location names in their dataset, which do not necessarily have coordinates. Processing locations works as follows:

1. Known locations (`new_location = FALSE`) are matched against both the official location names in the gazetteer and the accepted aliases.
2. New locations are first checked against the location aliases list to see if any new locations in this dataset have been matched to a gazetteer location. The gazetteer location is then used in preference to any new location data within the dataset.
3. Finally, remaining new locations are assigned any feature geometry within the dataset. New locations do not necessarily have geometry data, so may have null or empty geometries.

Value

An object of classes "safe_locations", "sf" and "data.frame".

Note

Not all SAFE project datasets contain locations. In this case `get_locations` will return NULL.

See Also

`add_locations`, `load_gazetteer`, `load_location_aliases`

Examples

```
set_example_safe_dir()
locations <- get_locations(1400562)
unset_example_safe_dir()
```

---

get_taxa *Obtaining taxonomy for a SAFE dataset.*

Description

This function generates a taxonomy table for a specified SAFE dataset record. Each row represent a taxon used in the data worksheets in the dataset and the table fields show the taxonomic hierarchy, taken from GBIF, for those taxa.

Usage

```
get_taxa(obj)
```

Arguments

obj	A single record id, or an existing safedata dataframe.

Details

All SAFE datasets containing taxa must include a Taxa worksheet, which must contain all of the taxa referred to in taxon fields in data worksheets. All entries in this worksheet are validated against the GBIF database before publication and are used to populate the taxonomic index for the SAFE datasets. The validated taxonomic hierarchy for the data is also available in the metadata for a record and this function converts the metadata into a taxonomy table. If the Taxa worksheet for a SAFE project dataset is empty, because the data does not contain observations on taxa, `get_taxa` will return NULL.

The taxonomy table includes the eight "backbone" taxonomic ranks used in the GBIF database: Kingdom, Phylum, Class, Order, Family, Genus, Species and Subspecies. It also includes three further fields: `taxon_name`, `taxon_rank` and `gbif_status`. These record the original taxa provided in the Taxa worksheet and show where the taxonomic name used in a dataset differs from the canonical name used in GBIF. The row names of the taxon table are the labels used in data worksheets and are used to match a taxonomy table to a loaded data worksheet (see `add_taxa`).

For more details on the structure of the Taxa worksheet see: https://safedata-validator.readthedocs.io/en/latest/data_format/taxa.html

Value

A taxonomy table of classes "safe_taxa" and "data.frame".

See Also

`add_taxa`

Examples

```
set_example_safe_dir()
taxa <- get_taxa(1400562)
unset_example_safe_dir()
```

---

get_taxon_coverage *Retrieve the current taxon index*

Description

This function downloads the current taxon index across published datasets and returns a formatted taxon table showing the taxonomic hierarchy of each taxon. It requires an internet connection.

Usage

```
get_taxon_coverage()
```

Details

Note that the use of "user" defined taxa can lead to some unusual entries. There is nothing to stop a user defining "Gallus gallus" as a "user" taxon with Animalia as a parent taxon.

Value

A data frame containing higher taxon level information for each taxon in the database taxon index. If the API is unavailable, the function returns NULL.

See Also

get_taxa

Examples

```
all_taxa <- get_taxon_coverage()
```

---

get_taxon_graph *Get a graph of the taxa in a dataset*

Description

This function loads the taxon index for a dataset (see get_taxa) and creates a taxonomic graph for the dataset. The graph vertex attributes contain further details of each taxon, including GBIF id and taxonomic status.

Usage

```
get_taxon_graph(record)
```

Arguments

record	A single dataset record id

Details

This function may modify the taxon index data to represent it as a graph, primarily to avoid duplicating taxon names, which are used as the primary vertex id. Possible issues are:

1. A user maps two worksheet names onto the _same_ taxon name: for example, "moth" and "butterfly" both using Lepidoptera as a taxon name. This is resolved by recreating the two worksheet names as children ofthe shared taxon name.
2. The taxon index may contain two GBIF taxonomic concepts with the same name (e.g. an accepted and doubtful usage). In this case, the function appends the GBIF ID to make taxon names unique.

3. Simple duplication of identical taxa - this should not happen but has been observed and the function removes all but one entry.
4. Missing parent taxa - again this should not happen but sometimes the GBIF backbone chain stops above the Kingdom level. The function adds a unique name for unknown parents.

Value

An graph object.

Examples

```
set_example_safe_dir()
beetle_graph <- get_taxon_graph(1400562)
plot(beetle_graph, vertex.label.cex = 0.6, vertex.size = 15,
     vertex.size2 = 3, vertex.shape = "rectangle")
# show worksheet names for tips
wsn <- igraph::vertex_attr(beetle_graph, "worksheet_name")
txn <- igraph::vertex_attr(beetle_graph, "name")
labels <- ifelse(is.na(wsn), txn, wsn)
is_leaf <- igraph::vertex_attr(beetle_graph, "leaf")
vert_col <- ifelse(is_leaf, "cornflowerblue", "grey")
plot(beetle_graph, vertex.label.cex = 0.6, vertex.size = 15,
     vertex.size2 = 3, vertex.shape = "rectangle",
     vertex.label = labels, vertex.color= vert_col)
unset_example_safe_dir()
```

---

igraph_to_phylo *Getting a phylogeny for a dataset*

Description

The function igraph_to_phylo takes a taxon graph (see get_taxon_graph) and attempts to convert that to a a phylo object from **ape**. This will fail if the graph is not simple (no loops or multiple edges) or is not connected (has isolated taxa). Neither of these conditions should happen in datasets but they do.

Usage

```
igraph_to_phylo(g)

get_phylogeny(record)
```

Arguments

g	A taxon graph returned by get_taxon_graph
record	A single dataset record id

Details

The phylogeny is assigned with equal branch lengths and so displays showing the taxonomic hierarchy of taxa. Note that the phylogeny will contain singleton nodes if an internal taxon has a single descendant - see the example below showing internal node labels. The ape functions has.singles and collapse.singles can be used to detect and remove these if required.

The function get_phylogeny is simply a wrapper that calls get_taxon_graph and then igraph_to_phylo.

Value

An phylo object.

Functions

- get_phylogeny: Get a phylogeny for a dataset

See Also

get_taxon_graph

Examples

```
set_example_safe_dir()
beetle_graph <- get_taxon_graph(1400562)
beetle_phylo <- igraph_to_phylo(beetle_graph)
ape::plot.phylo(beetle_phylo, show.node.label = TRUE)
# Or wrapped into a single function
beetle_phylo <- get_phylogeny(1400562)
ape::plot.phylo(beetle_phylo, show.node.label = TRUE)
unset_example_safe_dir()
```

---

insert_dataset *Inserts local copies of files from a dataset into a SAFE data directory*

Description

If files are embargoed or restricted, then users may request the datafiles from the authors. This function allows provided files to be incorporated into a SAFE data directory, so that they will then work seamlessly alongside openly available data.

Usage

```
insert_dataset(record_id, files)
```

Arguments

record_id	A SAFE dataset record id
files	A vector of files to insert into the data directory

Examples

```
set_example_safe_dir()
files <- system.file("safedata_example_dir",
                     "template_ClareWfunctiondata.xlsx",
                     package = "safedata")
insert_dataset(1237719, files)
dat <- load_safe_data(1237719, "Data")
str(dat)
unset_example_safe_dir()
```

---

load_gazetteer *Load and cache the SAFE gazetteer*

Description

This function loads the SAFE gazetteer, stored as a geojson file in the root of the SAFE data directory, as an `sf` GIS object. The GIS data uses the WGS84 (EPSG:4326) geographic coordinate system.

Usage

```
load_gazetteer()
```

Details

The gazetteer contains the following fields:

location The official gazetteer name for a sampling site.

type A short description of location type - typically the project that created the location

plot_size Where applicable, the size of the plot at a location. Note that point locations may define a sampling area with a plot size.

display_order DELETE

parent DELETE

region One of the four major large scale sampling areas: SAFE, Maliau, Danum and VJR

fractal_order Only defined for the SAFE core sampling points, which follow a fractal layout.

transect_order Again, for SAFE core sampling points, the location of a point along the sampling design transect.

centroid_x, centroid_y The centroid of the feature

source The original source GIS file that the feature was described in.

bbox_xmin, bbox_ymin, bbox_xmax, bbox_ymax The bounding box of the feature.

geometry The GIS geometry for the data - a column of class `sfc`.

When this function is first called in a session, the loaded `sf` object is cached for re-use (see `load_index`).

Value

An sf object containing the SAFE gazetteer locations.

See Also

load_location_aliases, load_index

---

load_safe_data *Loads data from a SAFE dataset.*

Description

This function returns a data frame containing the data from a data worksheet in a SAFE dataset. Note that SAFE dataset .xlsx files include the other (non-data) worksheets Summary, Taxa, Locations that contain metadata: see get_taxa, get_locations, add_taxa and add_locations for accessing and using this metadata.

Usage

```
load_safe_data(record_id, worksheet)

## S3 method for class 'safedata'
str(object, ...)

## S3 method for class 'safedata'
print(x, n = 10, ...)
```

Arguments

record_id	A SAFE dataset record id
worksheet	The name of the worksheet to load
...	Further arguments to str and print methods.
x, object	A safedata object.
n	The number of rows to show in the print method.

Details

In particular, the large amount of data worksheet summary metadata is not attached as attributes to the data frame returned by this function. This is largely to avoid excessive output to the console during normal use of the data frame: an extended description of a worksheet can be displayed using show_worksheet.

Currently, this function only loads data from SAFE formatted .xlsx files - data stored in external files is not yet handled.

Value

A data frame with the additional `safedata` class and additional attribute data containing metadata for the data.

Methods (by generic)

- `str`: Display structure of a safedata data frame
- `print`: Print safedata data frame

Examples

```
set_example_safe_dir()
beetle_abund <- load_safe_data(1400562, "Ant-Psel")
str(beetle_abund)
# See also the show_worksheet function for further worksheet metadata
show_worksheet(beetle_abund)
unset_example_safe_dir()
```

`safedata` *Finding and using data from the SAFE Project.*

Description

The SAFE Project is one of the largest ecological experiments in the world. We are studying how biodiversity and ecosystem function change as forests are modified by human activities. As well as studying the change, we also explore whether preserving sections of forest within modified landscapes and around waterways can protect biodiversity and ecosystem function, and how much protection is needed to be effective.

The project is a multinational collaboration that has generated a wide range of research data. We archive and document this data using the Zenodo file repository and maintain spatial, taxonomic and other indices to support data discovery and reuse. This package provides a range of tools to make it easier to find and analyse our research data outputs.

Links

https://www.safeproject.net/datasets/view_datasets https://www.zenodo.org/communities/safe

safedata_network *Network resources and the safedata package.*

Description

The safedata package requires access to the internet in order to:

1. maintain an up-to-date index of datasets and the locations gazeteer, 2. download datasets and their metadata, 3. download metadata about the taxonomic coverage of datasets, and 4. search dataset metadata for relevant datasets.

These actions use two resources. The first is the safedata web server API which provides everything except the actual datasets. The second is the Zenodo API, which provides the datasets.

If you do not have an internet connection - or if either of the two APIs is unavailable - the safedata package can be used as normal to load and use datasets that have already been downloaded to the local safedata directory. However, it will not be possible to update the dataset index, download new datasets or search for datasets until the APIs are available. The package should handle internet failures gracefully and provide meaningful messages.

Note on SSL certificates

Downloading data uses the curl package and the underlying libcurl library. Some older versions of Mac OS X (10.14 and earlier) provide a built-in libcurl with an outdated set of certificates that prevents curl from connecting to resources using LetsEncrypt for HTTPS, which includes https://safeproject.net. To use safedata on these systems, you have to install a newer version of curl (e.g. using brew) and then compile curl from source, linking it to that newer libcurl. The simplest way to do this is to use `export PKG_CONFIG_PATH="/usr/local/opt/curl/lib/pkgconfig"` before installing the package: this points the installation to the package configuration for the brew installed version of curl.

search_safe *SAFE dataset search functions.*

Description

In addition to the datasets stored on Zenodo, the SAFE Project website provides an API to search dataset metadata in more depth. The search functions access this API and return `safe_record_set` objects identifying datasets that match a particular query.

Usage

```
search_dates(dates, match_type = "intersect", most_recent = FALSE, ids = NULL)

search_fields(
  field_text = NULL,
  field_type = NULL,
```

```
  ids = NULL,
  most_recent = FALSE
)

search_authors(author, ids = NULL, most_recent = FALSE)

search_taxa(
  taxon_name = NULL,
  taxon_rank = NULL,
  gbif_id = NULL,
  ids = NULL,
  most_recent = FALSE
)

search_text(text, ids = NULL, most_recent = FALSE)

search_spatial(
  wkt = NULL,
  location = NULL,
  distance = NULL,
  ids = NULL,
  most_recent = FALSE
)
```

Arguments

dates	A vector of length 1 or 2, containing either ISO format date character strings ("yyyy-mm-dd") or POSIXt dates.
match_type	A character string (see Details).
most_recent	Logical indicating whether to restrict the API to returning only the most recent versions of the datasets found. By default all versions of matching dataset concepts are returned.
ids	A set of SAFE dataset record IDs to restrict a search. This will typically be a safe_record_set object returned by another search but can also be a vector of record ids in any of the formats accepted by validate_record_ids.
field_text	Text to search for within the data worksheet field name and description.
field_type	A data worksheet field type (see Links).
author	A character string used to search for datasets by author full (or partial) names.
taxon_name	The scientific name of a taxon to search for.
taxon_rank	A taxonomic rank to search for.
gbif_id	A GBIF taxonomic ID number.
text	Character string to look for within a SAFE dataset, worksheet, title, field description, and dataset keywords.
wkt	A well-known text geometry string, assumed to use latitude and longitude in WGS84 (EPSG:4326).

location	The name of a location in the SAFE gazetteer.
distance	A buffer distance for spatial searches, giving the distance in metres within which to match either location or wkt searches.

Details

The API provides endpoints to search datasets by date extents, data worksheet fields, authors, taxa, free text and by spatial query. All of the functions accept the argument most_recent, which restricts the returned datasets to the most recent versions of each matching dataset concept. The functions can also be passed an existing safe_record_set object to search within the results of a previous search.

The match_type parameter specifies how to match date ranges and must be one of "intersect" (default), "contain", or "within". The "contain" option returns datasets that span a date range, "within" returns datasets that fall within the given range and "intersect" selects datasets that overlap any part of the date range. Note that match_type is ignored when only a single date is provided.

Value

An object of class safe_record_set of datasets that match the query.

Functions

- search_dates: Search datasets by date extent
- search_fields: Search data worksheet field metadata.
- search_authors: Search by dataset author
- search_taxa: Search by taxon name, rank or GBIF ID.
- search_text: Search dataset, worksheet and field titles and descriptions
- search_spatial: Search by spatial sampling area/named location.

Spatial searches

For spatial searches, users can select a location name from a SAFE data gazetteer (see e.g. https://www.safeproject.net/info/gazetteer or load_gazetteer) or provide a WKT geometry. The sampling locations provided in each SAFE dataset are tested to see if they intersect the search geometry.

A buffer distance can aso be provided to extend the search around the query geometry. Note that although WKT geometries should be provided using WGS84 lat/long coordinates, since this is typical field GPS data, distances must be provided as metres and all proximity calculations take place in the UTM50N projected coordinate system.

The search_spatial function will not retrieve datasets that have not provided sampling locations or use newly defined locations that are missing coordinate information.

Links

SAFE data API e.g. https://www.safeproject.net/api

Worksheet field types https://safedata-validator.readthedocs.io/en/latest/data_format/data.html#field-types

SAFE gazetteer See load_gazetteer and e.g. https://www.safeproject.net/info/gazetteer

WKT https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

Examples

```
search_dates("2014-06-12")
search_dates(as.POSIXct(c("2014-06-12", "2015-06-11")))
search_dates(c("2014-06-12", "2015-06-11"), match_type = "contain")
search_fields(field_text = "temperature")
search_fields(field_type = "numeric")
search_fields(field_text = "temperature", field_type = "numeric")
search_authors("Ewers")
search_taxa(taxon_name = "Formicidae")
search_taxa(gbif_id = 4342)
search_taxa(taxon_rank = "family")
search_text("forest")
search_text("ant")
search_spatial(wkt = "Point(116.5 4.75)")
search_spatial(wkt = "Point(116.5 4.75)", distance = 100000)
search_spatial(wkt = "Polygon((110 0, 110 10,120 10,120 0,110 0))")
search_spatial(location = "A_1")
search_spatial(location = "A_1", distance = 2500)

# combining searches using logical operators
fish <- search_taxa('Actinopterygii')
odonates <- search_taxa("Odonata")
ewers <- search_authors("Ewers")
aquatic <- fish | odonates
aquatic_ewers <- aquatic & ewers
all_in_one <- (fish | odonates) & ewers
```

---

set_example_safe_dir *Functions to use an example data directory for package examples*

Description

The documentation of safedata includes code examples using a data directory. A zipped example directory is included in the package files (data/safedata_example_dir.zip) but package code must not write within the package structure. The set_example_safe_dir function is used in code examples to unpack this example directory into a temporary folder and set it for use in the example code. The function unset_example_safe_dir is then used to restore any existing data directory set by the user. The example directory should only be created once per session.

Usage

```
set_example_safe_dir()

unset_example_safe_dir()
```

Value

The set_example_safe_dir function returns the path of the example directory invisibly.

Functions

- `unset_example_safe_dir`: Restores a user data directory after running an code example.

See Also

`set_safe_dir`

`set_safe_dir` *Set the local SAFE data directory*

Description

This function sets the local directory used to store SAFE dataset files along with record and index metadata. The function can also initialise a new data directory, downloading the required index files. By default, it will update indices if needed and will validate the directory contents. Once set, the location of the directory is stored in options("safedata.dir"). The `url` argument specifies a URL to a website that exposes the SAFE data API.

Usage

```
set_safe_dir(
  safedir,
  update = TRUE,
  create = FALSE,
  url = "https://www.safeproject.net"
)
```

Arguments

`safedir`	A path to the directory to set as the SAFE data directory (str).
`update`	Should the local dataset index be updated (logical)?
`create`	Should a new data directory be created at the provided path (logical)?
`url`	A URL providing the SAFE Data API, defaulting to the SAFE Project's own URL.

Details

The safedata package uses a data directory to store local copies of dataset files along with index files. Files for a dataset record are stored in subdirectories using the zenodo concept id for the record and then record id: `3342494/3342495`, for example. In addition to data files from these records, these folders can also contain a JSON file containing record metadata (e.g. `3342495.json`): this is a structured version of the summary information shown on the Zenodo page.

The root of the data directory also holds three index files:

`index.json` , containing a full list of the files and dataset records available in the SAFE data repository;

`gazetteer.geojson` , containing the official list of known sampling locations and GIS data; and

`location_aliases.csv` , a list of alternative names permitted for some locations.

If `create = TRUE`, the function will try to create the named directory and populate it with the three index files. This requires an internet connection.

By default, the function also needs an internet connection to check for updates to the three index files. Updating can be turned off for offline use.

The default behaviour is also to validate the directory structure. The function will warn when files other than those found in datasets are present within the data structure and when any dataset files that are present have been modified. Although this can be turned off, it is not recommended to modify or add files within a SAFE data directory.

Value

Invisibly, a boolean showing whether a SAFE data directory was set successfully.

`show_concepts` *Show SAFE dataset metadata*

Description

These functions provide access to the metadata associated with SAFE datasets. The functions provide three levels of information:

`show_concepts` displays the record versions grouped under dataset concepts,

`show_record` displays summary information about a specific record, and

`show_worksheet` displays metadata about data worksheet fields within a record.

All three functions accept a first argument `obj`, which can be one of three things:

1. A character or numeric vector of SAFE dataset records or concepts, which will be validated using `validate_record_ids`, or
2. An already validated `safe_record_set` object, or
3. A `safedata` data frame loaded using `load_safe_data`.

If `show_concepts` is passed a record id, then the function looks up the relevant concept. The version table indicates which versions are available ("*" for the most recent available version and "o" for older available versions), and which are unavailable due to embargo or retriction ("x"). A "!" is used to show that a private local copy of an embargoed or restricted dataset has been inserted using `insert_dataset`.

Usage

```
show_concepts(obj)

show_record(obj)

show_worksheet(obj, worksheet = NULL, extended_fields = FALSE)
```

Arguments

obj	A reference to SAFE records or a loaded worksheet (see above)
worksheet	The name of a worksheet to show. Obviously, if obj is a loaded worksheet, that will be the worksheet described and this can be left as NULL.
extended_fields	Logical - show a compact description of worksheet fields or a longer output including full metadata descriptors.

Value

Invisibly, a SAFE metadata object or a list of such objects. These are not really intended for end user consumption.

Functions

- show_concepts: Show the records associated with a dataset concept.
- show_record: Show details of a specific dataset
- show_worksheet: Show details of a data worksheet

Examples

```
set_example_safe_dir()
recs <- validate_record_ids(c(1400562, 3266827, 3266821))
show_concepts(recs)
show_record(recs[1,])
# Show worksheet metadata from a record or from a loaded worksheet
show_worksheet(1400562, "EnvironVariables")
beetle_abund <- load_safe_data(1400562, "Ant-Psel")
show_worksheet(beetle_abund, extended_fields = TRUE)
unset_example_safe_dir()
```

---

validate_record_ids *Validates dataset record ids from user input*

Description

This takes a vector of user supplied record identifiers and validates them against the index. Typically the identifiers are provided as integers, but the function will also handle Zenodo URLs and DOIs.

Usage

```
validate_record_ids(record_set)

## S3 method for class 'safe_record_set'
print(x, ...)

## S3 method for class 'safe_record_set'
x & y

## S3 method for class 'safe_record_set'
x | y
```

Arguments

record_set	A vector of values containing Zenodo concept or record ids.
x, y	Objects of class safe_record_set
...	Further arguments to print methods, unused.

Details

The function returns a data frame with class safe_record_set, containing the columns concept, record, available and, finally, mra containing the most recent available record (if any). The function can be run on an existing safe_record_set to update this information.

Note that record will be NA when a value represents a concept id. Inputs that do not match a record or concept ids are returned in the attribute mismatches of the record set.

This function is largely used internally to validate user inputs and to provide a common output for the search functions but is exported to allow users to check record ids and display summary information using the print method.

Value

An object of class safe_record_set (see Details)

Methods (by generic)

- print: Print a brief summary of "safe_record_set" objects.
- &: Combine two record sets, retaining only records that are present in both.
- |: Combine two record sets, including the records that are present in either.

Examples

```
set_example_safe_dir()
validate_record_ids(c(3247631, 3266827, 3266821, -1000))
validate_record_ids(c("https://doi.org/10.5281/zenodo.3247631",
                      "10.5281/zenodo.3266827",
                      "https://zenodo.org/record/3266821",
                      "not_this_one/3266821"))
unset_example_safe_dir()
```

Index