Package: ReLTER (via r-universe)

August 20, 2026

Type Package

Title An Interface for the eLTER Community

Version 3.1.1

Description ReLTER provides access to DEIMS-SDR (https://deims.org/), and allows interaction with data and software implemented by eLTER Research Infrastructure (RI) thus improving data sharing among European LTER projects. ReLTER uses the R language to access and interact with the DEIMS-SDR archive of information shared by the Long Term Ecological Research (LTER) network. This package grew within eLTER H2020 as a major project that will help advance the development of European Long-Term Ecosystem Research Infrastructures (eLTER RI - https://elter-ri.eu). The ReLTER package functions in particular allow to: - retrieve the information about entities (e.g. sites, datasets, and activities) shared by DEIMS-SDR (see e.g. get_site_info function); - interact with the ODSEurope (maps.opendatascience.eu) starting with the dataset shared by DEIMS-SDR (https://deims.org/) (see e.g. the get_site_ODS() function); - use the eLTER site informations to download and crop geospatial data from other platforms (see e.g. get_site_ODS function()); - improve the quality of the dataset (see e.g. get_id_worms()). Functions currently implemented are derived from discussions of the needs among the eLTER users community. The ReLTER package will continue to follow the progress of eLTER-RI and evolve, adding new tools and improvements as required.

License GPL (>= 3)

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

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

Depends R (>= 4.1.0)

Imports dplyr, dtplyr, ggforce, ggspatial, geojsonsf, ggplot2, httr2, jqr, jsonlite, leaflet (>= 2.1.1), lifecycle, lubridate, magrittr, purrr, qrcode, sf (>= 0.9-5), utils, units, tibble, stringr, terra

Suggests rbibutils, markdown, cowplot, geodata, httptest2, ISOcodes, knitr, leaflet.extras, prettymapr, RColorBrewer, shiny, spocc, tidyr, taxize (>= 0.9.97), testthat (>= 3.0.0), withr, worrms, zen4R

Remotes hrbrmstr/waffle, XML

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libfontconfig1-dev libfreetype6-dev libgdal-dev gdal-bin libgeos-dev make libicu-dev libjpeg-dev libjq-dev libpng-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-07-21 09:41:48 UTC

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

RemoteRef main

RemoteSha 023bc15b846d24c647f79e3b71eae264871aa7f9

Contents

`EDC_construct_full_url`

Construct full url for download

Description

[Stable] Construct full URL, including replacing year and month where required.

Usage

```
EDC_construct_full_url(edc_row, dataset_year, dataset_month)
```

Arguments

`edc_row`	`vector` one row for chosen dataset from edc_df.
`dataset_year`	`string` Chosen year
`dataset_month`	`string` Chosen month

Author(s)

Micha Silver, phD (2020) <silverm@post.bgu.ac.il>

eLTER_data_reporting_format *eLTER Data Reporting Format (Rhrefhttps://doi.org/10.5281/zenodo.6373410DRF)*

Description

[Experimental] `eLTER_data_reporting_format` data is a nested `list` describing the field specifications used in the eLTER data reporting process. It includes two versions of the format: version 1.3 (used in eLTER-Plus) and version 2.0 (proposed update). Each version contains multiple components such as DATA, REFERENCE, METHOD, STATION, EVENT, and SAMPLE, which define the expected fields for each section.

Usage

```
data(eLTER_data_reporting_format)
```

Format

A named list with two main elements:

version1.3 Field names grouped by section as used in eLTER-Plus. Each section (e.g. DATA, METHOD, STATION) contains field names for core and extended data structures.

version2.0 Updated field names grouped by section, reflecting refinements in naming conventions and structure.

Source

eLTER-Plus field specification document: https://doi.org/10.5281/zenodo.6373410 Proposed updated format: https://docs.google.com/spreadsheets/d/1wTiwb_5uM_XGsrSWUx9h2t4maSSFg7mpggq2nIydWco

get_activity_info *Obtain the information about of an eLTER activity.*

Description

[Stable] This function obtains the information about of an eLTER activity (e.g. https://deims.org/activity/8786fc6d-5d70-495c-b901-42f480182845) provided in DEIMS-SDR catalogue.

Usage

```
get_activity_info(activityid, show_map = FALSE)
```

Arguments

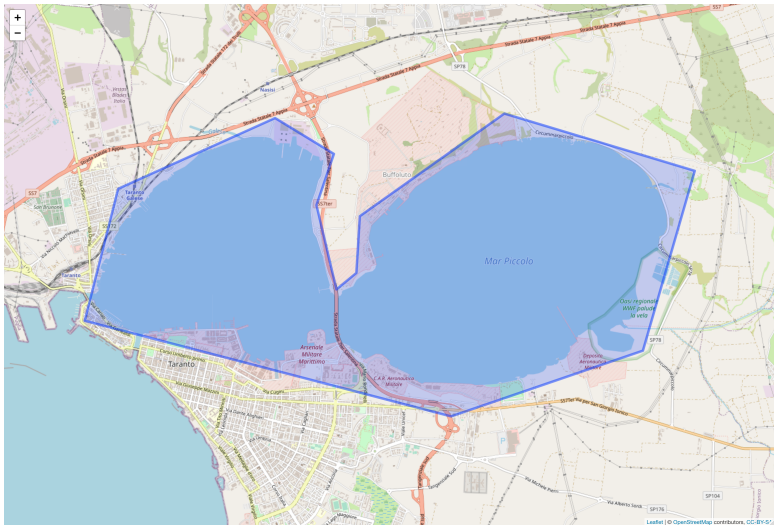
`activityid`	A `character`. It is the DEIMS ID of activity make from DEIMS-SDR website. DEIMS ID information here. The DEIMS.iD of activity is the URL for the activity page.
`show_map`	A `boolean`. If TRUE a Leaflet map with occurrences is shown. Default FALSE.

Value

The output of the function is a `list` with two elements:

- `map` A Leaflet map with the activity location, if requested with `show_map`.
- `data` A `data.frame` with the information about the activity.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
activities <- get_activity_info(
  activityid =
    "https://deims.org/activity/8786fc6d-5d70-495c-b901-42f480182845",
  show_map = TRUE
)
activities
```

`get_dataset_info` *Obtain the information about of an eLTER dataset.*

Description

[Deprecated] This function obtains the information about of an eLTER dataset (e.g. https://deims.org/dataset/38d604ef-decb-4d67-8ac3-cc843d10d3ef) provided in DEIMS-SDR catalogue.

Usage

```
get_dataset_info(datasetid, show_map = FALSE)
```

Arguments

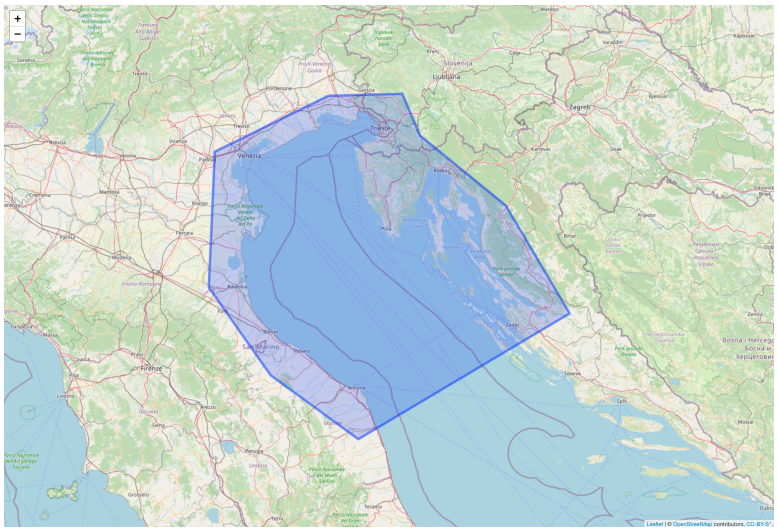
`datasetid`	A `character`. It is the DEIMS ID of dataset make from DEIMS-SDR website. DEIMS ID information here. The DEIMS ID of dataset is the URL for the dataset page.
`show_map`	A `boolean`. If TRUE a Leaflet map with occurrences is shown. Default FALSE.

Value

The output of the function is a `list` with two elements:

- `map` A Leaflet map with the dataset location, if requested with `show_map`.
- `data` A `data.frame` with the information about the dataset.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
## Not run:
tDataset <- get_dataset_info(
  datasetid =
  "https://deims.org/dataset/38d604ef-decb-4d67-8ac3-cc843d10d3ef",
  show_map = TRUE
)
tDataset

## End(Not run)
```

---

get_deims_API_version *Get version of DEIMS-SDR API*

Description

[Stable] This function obtains the version of the DEIMS-SDR API.

Usage

```
get_deims_API_version(deims_url = get_deims_base_url())
```

Arguments

deims_url	A character. DEIMS-SDR base URL. Defaults to package settings.

Value

version number.

---

get_deims_base_url *Get DEIMS-SDR base URL*

Description

[Stable]

Usage

```
get_deims_base_url()
```

Value

DEIMS-SDR base URL

See Also

Other package_customizable_settings: package_settings

`get_location_info` *Obtain the information about of an eLTER location.*

Description

[Stable] This function obtains the information about of an eLTER location (e.g. https://deims.org/location/12b38f3f-7e72-425a-80c7-7cad35ce4c7b) provided in DEIMS-SDR catalogue.

Usage

```
get_location_info(locationid, show_map = FALSE)
```

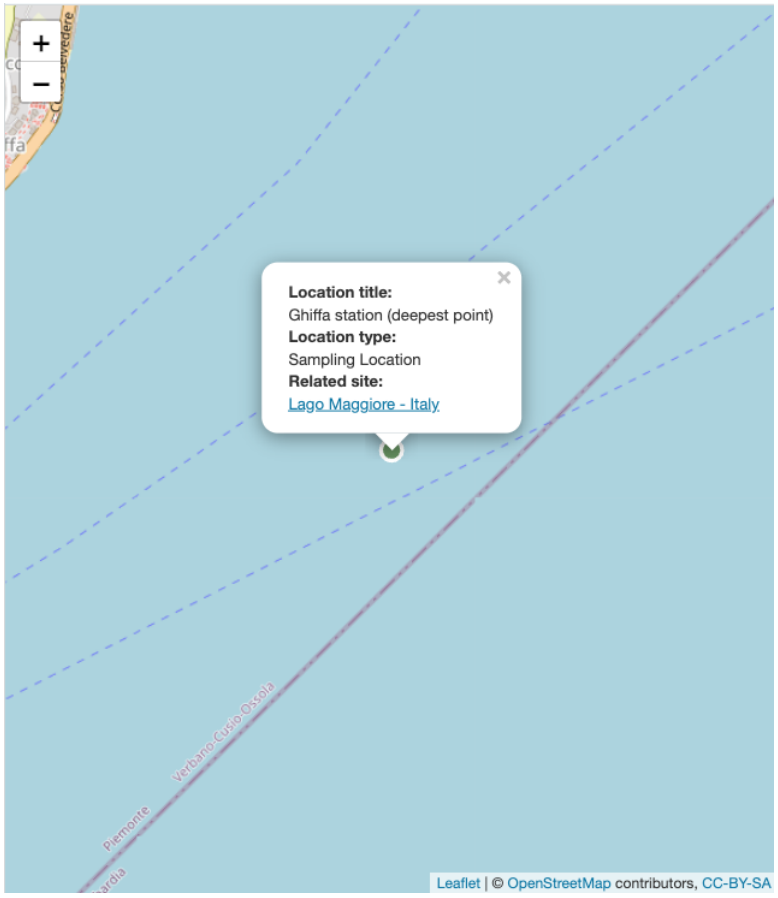
Arguments

locationid	A character. It is the DEIMS ID of location make from DEIMS-SDR website. DEIMS ID information here. The DEIMS.iD of activity is the URL for the location page.
show_map	A boolean. If TRUE a Leaflet map with occurrences is shown. Default FALSE.

Value

The output of the function is a tibble with the information about the location.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Examples

```
## Not run:
# Sampling location multipolygon
location <- get_location_info(
  locationid =
    "https://deims.org/location/85dc6019-9654-4ba0-8338-08c4ffe8fe47",
  show_map = TRUE
)
location

# Sampling location polygon
location <- get_location_info(
  locationid =
```

```
    "https://deims.org/location/12b38f3f-7e72-425a-80c7-7cad35ce4c7b",
  show_map = TRUE
)
location

# Equipment location polygon
location <- get_location_info(
  locationid =
    "https://deims.org/locations/04de8301-b481-4ed2-89ff-2f48562e2514",
  show_map = TRUE
)
location

# Sampling location point
location <- get_location_info(
  locationid =
    "https://deims.org/location/ec1a58f7-1aee-4e3f-bec3-4eb1516ee905",
  show_map = TRUE
)
location

# Sampling location point with location type null
location <- get_location_info(
  locationid =
    "https://deims.org/location/c3db70c3-5d2c-4905-801c-7b7a5c4d00d9",
  show_map = TRUE
)
location

## End(Not run)
```

`get_network_sites` *Retrieve a list of sites in an eLTER Network.*

Description

[Stable] This function return a spatial point vector object including title, date late updated, URI, and coordinates, stored in DEIMS-SDR catalogue, of all the eLTER sites belonging to an eLTER Network (e.g. LTER- Italy network).

Usage

```
get_network_sites(networkDEIMSID)
```

Arguments

`networkDEIMSID`	A `character`. The DEIMS ID of the network from DEIMS-SDR website. DEIMS ID information here and Complete list of networks here. The DEIMS ID of network is the URL for the network page.

Value

The output of the function is a point vector of `sf` class (package sf) of the network's sites.

Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
## Not run:
# The sites of LTER-Italy network
listSites <- get_network_sites(
  networkDEIMSID =
    "https://deims.org/network/7fef6b73-e5cb-4cd2-b438-ed32eb1504b3"
)
listSites

# The sites of LTER Europe network
euSites <- get_network_sites(
  networkDEIMSID =
    "https://deims.org/networks/4742ffca-65ac-4aae-815f-83738500a1fc"
)
euSites

## End(Not run)
```

`get_sensor_info` *Obtain the information about of an eLTER sensor.*

Description

[Deprecated] This function obtains the information about of an eLTER sensor (e.g. https://deims.org/sensors/3845475c-4aec-4dd7-83b4-0ab6ba95db35) provided in DEIMS-SDR catalogue.

Usage

```
get_sensor_info(sensorid, show_map = FALSE)
```

Arguments

`sensorid`	A `character`. It is the DEIMS ID of sensor make from DEIMS-SDR website. DEIMS ID information here. The DEIMS.iD of sensor is the URL for the sensor page.
`show_map`	A `boolean`. If TRUE a Leaflet map with occurrences is shown. Default FALSE.

Value

The output of the function is a `list` with two elements:

- `map` A Leaflet map with the sensor location, if requested with `show_map`.
- `data` A `data.frame` with the information about the sensor.

The function output

Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
## Not run:
# print the map of the sensor
sensor_B3 <- get_sensor_info(
  sensorid =
    "https://deims.org/sensors/3845475c-4aec-4dd7-83b4-0ab6ba95db35",
```

```
  show_map = TRUE
)
sensor_B3

# only table of sensor information
Licor <- get_sensor_info(
  sensorid =
    "https://deims.org/sensors/4a7ad644-f2e7-4224-965b-ec5ef5365655",
  show_map = FALSE
)
Licor

# Moldaenke FluoroProbe sensor
sensor_FP <- get_sensor_info(
  sensorid = "https://deims.org/sensors/82635223-a4f4-498c-b283-9c95999d9d2f",
  show_map = FALSE
)
sensor_FP

## End(Not run)
```

`get_site_EcoDataCube` *Acquire various raster layers from* R*hrefhttps://ecodatacube.eu/EcoDataCube EU and crop to an eLTER site boundary.*

Description

[Stable] Download and return a SpatRaster object containing the requested dataset from EcoDataCube EU, cropped to an eLTER site boundary, which is obtained from the DEIMS-SDR API.

Usage

```
get_site_EcoDataCube(
  deimsid,
  dataset = "",
  dataset_year = NA,
  dataset_month = NA,
  show_map = TRUE,
  output_dir = tempdir()
)
```

Arguments

`deimsid`	`string`. The DEIMS ID of the site from DEIMS-SDR website. DEIMS ID information here.
`dataset`	`string` The requested dataset. One of: "CHELSA_precip", "clc_2017", "clc_2020", "crop_map", "DTM_30m", "MODIS_LST_day", "MODIS_LST_night", "NDVI_bimonthly",

	"NDVI_yearly", "NDWI_monthly", "soil_type" If dataset is NA or an empty string, then a list of the available EcoDataCube layers is printed. Default is "". (See Details for explanation of each dataset)
`dataset_year`	`string` indicating which year to choose (for multi-year datasets) Entered as four digits (i.e. '2020')
`dataset_month`	`string` indicating which month (for multi-month datasets). Entered as two digits. i.e. '02', '03', or '11' etc.
`show_map`	`Bool` whether to show `leaflet` map of downloaded dataset. Default TRUE
`output_dir`	`string` where to save the EcoDataCube dataset and style file. The *.tif, *.sld and *.qml files will be saved to the specified directory, with filename as <chosen_dataset>_.tif Default tempdir()

Details

Supported datasets from the EcoDataCube repository include:

dataset	full name	date required	
CHELSA_precip	CHELSA Monthly accumulated precipitation	yes	2000-01-01 00:00:00 UTC–2019
clc_2017	Corine Landcover (CLC+) 2017-2019	no	2017-01-01 00:00:00 UTC–2022
clc_2020	Corine Landcover (CLC+) 2020-2022	no	2017-01-01 00:00:00 UTC–2022
crop_map	EUCROPMAP Pan-EU year 2022	no	2022-01-01 00:00:00 UTC–2022
DTM_30m	OpenLandMap Ensemble Digital Terrain Model	no	2006-01-01 00:00:00 UTC–2015
MODIS_LST_day	MOD11A2 monthly land surface temp. (day)	yes	2000-01-01 00:00:00 UTC–2021
MODIS_LST_night	MOD11A2 monthly land surface temp. (night)	yes	2000-01-01 00:00:00 UTC–2021
NDVI_bimonthly	Cloud-free reconstructed Landsat bimonthly NDVI	yes	2000-01-01 00:00:00 UTC–2022
NDVI_yearly	Cloud free reconstructed yearly Landsat NDVI	yes	2000-01-01 00:00:00 UTC–2022
NDWI_bimonthly	Cloud free reconstructed bi-monthly NDWI (Gao)	yes	2000-01-01 00:00:00 UTC–2022
soil_type	AI4SoilHealth: Soil type dominant class	no	2000-01-01 00:00:00 UTC–2022

All datasets are georeferenced to the EPSG:3035 coordinate reference system.

Value

The function returns a SpatRaster object (from the `terra` package) of the requested dataset, cropped to the site boundaries. The SpatRaster is also saved as GeoTiff, along with the style files, in the chosen `output_dir`.

Author(s)

Micha Silver, phD (2020) <silverm@post.bgu.ac.il>

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
# Example of TERENO Harsleben
deimsid = "https://deims.org/c945abe4-3d40-46d1-b5d0-33127c35c6ab"
# NDVI
harsleben_ndvi <- get_site_EcoDataCube(
  deimsid = deimsid,
  dataset = "NDVI_bimonthly",
  dataset_year = "2021",
  dataset_month = "06"
)

# EUCrop map
harsleben_crop <- get_site_EcoDataCube(
  deimsid = deimsid,
  dataset = "crop_map",
)
# MODIS Land Surface Temperature,
# Kalkalpen National Park - Austria
## Not run:
deimsid = "https://deims.org/49515dda-1198-4013-8f43-c33e107af081"
kalkalpen_LST_day <- get_site_EcoDataCube(
  deimsid = deimsid,
  dataset = "MODIS_LST_day",
  dataset_year = "2010",
  dataset_month = "01"
)

## End(Not run)
```

---

get_site_info *Obtain details about an eLTER site.*

Description

[Stable] This function obtains information of a single eLTER site, as a stored in DEIMS-SDR catalogue, through the DEIMS-SDR API.

Usage

```
get_site_info(
  deimsid,
  categories = NA,
  show_map = FALSE,
  with_locations = FALSE
)
```

Arguments

deimsid	A character. The DEIMS ID of the site from DEIMS-SDR website. DEIMS ID information here.
categories	A categories. This parameter selects which categories or categories are retrieved and returned in the result. Possible value are: 'Affiliations', 'Contacts', 'EnvCharacts', 'General', 'Infrastructure', 'observedProperties', 'RelateRes'. Multiple values can be indicated. A site's boundary is always returned.
show_map	A boolean. When TRUE a leaflet map is plotted as side effect. Default FALSE.
with_locations	A boolean. When TRUE, and only show_map is TRUE, all site related locations are showed in the plotted map. Default FALSE.

Value

The output of the function is a sf with the information about the site. If the boundary is missing from DEIMS-SDR a tibble is returned.

Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Examples

```
site <- get_site_info(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe",
  categories = c("EnvCharacts", "Affiliations"),
  show_map = TRUE,
  with_locations = FALSE
)
site
```

---

get_site_speciesOccurrences

Retrieve species occurrences within an eLTER site boundary

Description

[Stable] This function downloads species occurrence records from GBIF https://www.gbif.org, iNaturalist https://www.inaturalist.org/ and OBIS https://obis.org/ and intersects them with the boundary of an eLTER site retrieved from the DEIMS-SDR API https://deims.org/. Only occurrences falling within the site polygon are returned, enriched with eLTER site metadata.

Usage

```
get_site_speciesOccurrences(
  deimsid,
  list_DS,
  show_map = TRUE,
  limit = 500,
  exclude_inat_from_gbif = TRUE
)
```

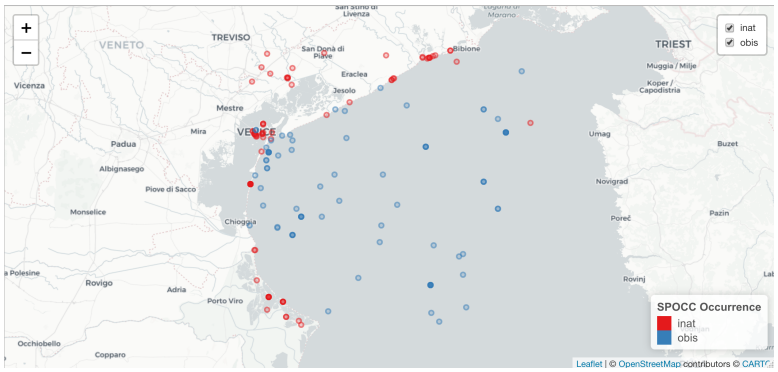
Arguments

deimsid	A character. The DEIMS ID of the site from DEIMS-SDR website. DEIMS ID information here.
list_DS	A character vector. Data sources to query; any combination of "gbif", "inat", and/or "obis".
show_map	A boolean. If TRUE the leaflet map is both printed and returned in the output list as $map. If FALSE the map is not printed but is still built and returned in $map for later use. Default FALSE.
limit	A numeric. Maximum number of records to download per source. Default 500. Note that when querying for many species the total number of records can be large and slow to download; start with a small value (e.g. 10) to verify results before increasing.
exclude_inat_from_gbif	A boolean. If TRUE, and both "gbif" and "inat" are in list_DS, records originating from iNaturalist are removed from the GBIF results to avoid duplicates. Default TRUE.

Value

A list with one sf element per data source (named gbif, inat, obis) containing only occurrences that fall within the site boundary, and a map element with a leaflet object. Each sf element contains the occurrence geometry and the following eLTER site metadata fields (prefixed with eLTER_): title, uri, created, changed, geoCoord, country, geoElev.avg, geoElev.min, geoElev.max, biogeographicalRegion, biome, ecosystemType, eunisHabitat, landforms, geoBonBiome, geology, hydrology, soils, vegetation, size.value. If no occurrences are found within the boundary for a given source, that source is omitted from the list and an informative message is printed. Returns invisible(NULL) if the site has no boundary or if no occurrences are found for any source.

The function output



Author(s)

Alessandro Oggioni, PhD (2020) <oggioni.a@irea.cnr.it>

Paolo Tagliolato, PhD (2020) <tagliolato.p@irea.cnr.it>

Martina Zilioli <zilioli.m@irea.cnr.it>

See Also

spocc::occ()

spocc::obis_search()

RColorBrewer::brewer.pal()

get_site_info()

Examples

```
## Not run:
# Terrestrial site: Saldur River Catchment (GBIF and iNaturalist, excluding records sourced from iNaturalist)
occ_SRC <- get_site_speciesOccurrences(
  deimsid = "https://deims.org/97ff6180-e5d1-45f2-a559-8a7872eb26b1",
  list_DS = c("gbif", "inat"),
  show_map = TRUE,
  limit = 50,
  exclude_inat_from_gbif = TRUE
)
occ_SRC

# Terrestrial site: Gran Paradiso National Park (only GBIF considering, excluding records sourced from iNaturalist)
occ_GPNP <- get_site_speciesOccurrences(
  deimsid = "https://deims.org/15c3e841-8494-42d2-a44e-c49a0ff25946",
  list_DS = "gbif",
  show_map = TRUE,
  limit = 50,
  exclude_inat_from_gbif = TRUE
)
occ_GPNP
```

```
# Marine site: Gulf of Venice (OBIS only)
occ_GoV <- get_site_speciesOccurrences(
  deimsid = "https://deims.org/758087d7-231f-4f07-bd7e-6922e0c283fd",
  list_DS = "obis",
  show_map = FALSE,
  limit = 10
)
occ_GoV

# Marine site: Gulf of Venice (all sources excluding records of GBIF sourced from iNaturalist)
occ_GoV_all <- get_site_speciesOccurrences(
  deimsid = "https://deims.org/758087d7-231f-4f07-bd7e-6922e0c283fd",
  list_DS = c("gbif", "inat", "obis"),
  show_map = TRUE,
  limit = 10,
  exclude_inat_from_gbif = TRUE
)
occ_GoV_all$obis
occ_GoV_all$map

## End(Not run)
```

`get_sites_interactive` *Select sites by choosing a 3d bounding box through an interactive GUI.*

Description

[Experimental] Open a GUI (a shiny widget) to interact with the user for the selection of a 3D bounding box. The user can visualise the sites within the selected bounding box and ask for returning the contained sites or simply the bounding box.

Usage

```
get_sites_interactive()
```

Value

`tibble` with selected sites or a `list` with selected bounding box (with slots `bbx` and `elevation_range`).

Author(s)

Paolo Tagliolato

See Also

shiny::runGadget

`leaflet.extras::addDrawToolbar()`

Examples

```
## Not run:
sites_tbl_sf <- get_sites_interactive()

## End(Not run)
```

`get_sites_within_3d_bounding_box` *select sites within a 3d bounding box*

Description

[Experimental] select sites within a 3d bounding box

Usage

```
get_sites_within_3d_bounding_box(bbox, elevation_range = NULL, show_map = TRUE)
```

Arguments

`bbox`	bounding box
`elevation_range`	elevation range
`show_map`	show the map

See Also

`geojsonsf::geojson_sf()`

Examples

```
## Not run:
bbx3d<-list(bbx=data.frame(x=c(6.718290, 9.805761),
                y=c(44.79938, 47.11089)),
            elevation_range=c(1200, 9000))
get_sites_within_3d_bounding_box(bbx3d)

## End(Not run)
```

`get_sites_within_radius` *Get all sites within a given distance of a point location*

Description

[Experimental] retrieve all sites within a given distance from the center point (in coordinates lat, lon), and in the (optionally) specified elevation range.

Usage

```
get_sites_within_radius(
  lat = 39.1386,
  lon = -8.33305,
  distance = 1,
  elevation_range = c(NULL, NULL),
  show_map = TRUE
)
```

Arguments

`lat`	latitude of radius center
`lon`	longitude of radius center
`distance`	distance from point (actually it is in degrees for a limitation of the endpoint)
`elevation_range`	min and max elevation in meters
`show_map`	`logical` print a map with retrieved sites (default is TRUE)

Value

list of deims_id, distance from given coordinates

See Also

`geojsonsf::geojson_sf()`

Examples

```
# example code
get_sites_within_radius(lat=39.1386, lon=-8.33305, distance=900)
```

get_zenodo_data *Obtain the data from a dataset deposited in Zenodo record.*

Description

[Deprecated] The function download the file(s) deposited in Zenodo record and returns a tibble with metadata.

Usage

```
get_zenodo_data(doi, rdata_exist = TRUE)
```

Arguments

doi	A character. It is the DOI of the Zenodo record.
rdata_exist	A logical. Is the .RData or .rds file in the record we are questioning? Default TRUE.

Value

a file(s) containing in the Zenodo record and tibble contains some metadata of Zenodo record.

Author(s)

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

See Also

zen4R::ZenodoManager()

tidyr::unnest()

tidyr::nest()

Examples

```
## Not run:
## Not run:

record <- get_zenodo_data(
  doi = "10.5281/zenodo.7041152", # test dataset
  rdata_exist = TRUE
)
record

## End (Not run)

## End(Not run)
```

`map_occ_gbif2elter` *Harmonize outputs of get_site_speciesOccurrence and map them into eLTER reporting format*

Description

[Experimental]

Usage

```
map_occ_gbif2elter(x, deimsid, version = "1.3")
```

Arguments

`x`	A `tibble` like one that can be obtained by as_tibble(get_site_speciesOccurrences(deimsid, "gbif")$gbif)
`deimsid`	A `character`. The DEIMS.iD of the site from DEIMS-SDR website. DEIMS ID information
`version`	A `character` for select the version of eLTER Data Reporting Format. Default 1.3

Value

list with the following named elements:

- deimsid: the same deimsid passed in input
- source: one of "gbif", "inat", "obis"
- data_mapping: tibble structured according to data_mapping of eLTER reporting format
- reference_TAXA: tibble structured according to reference_TAXA of eLTER reporting format
- reference_VARIABLES: tibble structured according to reference_VARIABLES of eLTER reporting format

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Martina Zilioli <zilioli.m@irea.cnr.it>

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

`map_occ_inat2elter` *Harmonize outputs of get_site_speciesOccurrence and map them into eLTER reporting format*

Description

[Experimental]

Usage

```
map_occ_inat2elter(x, deimsid, version = "1.3")
```

Arguments

`x`	A `tibble` like one that can be obtained by as_tibble(get_site_speciesOccurrences(deimsid, "gbif")$gbif)
`deimsid`	A `character`. The DEIMS.iD of the site from DEIMS-SDR website. DEIMS ID information
`version`	A `character` for select the version of eLTER Data Reporting Format. Default 1.3

Value

list with the following named elements:

- deimsid: the same deimsid passed in input
- source: one of "gbif", "inat", "obis"
- data_mapping: tibble structured according to data_mapping of eLTER reporting format
- reference_TAXA: tibble structured according to reference_TAXA of eLTER reporting format
- reference_VARIABLES: tibble structured according to reference_VARIABLES of eLTER reporting format

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Martina Zilioli <zilioli.m@irea.cnr.it>

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

`map_occ_obis2elter` *Harmonize outputs of get_site_speciesOccurrence and map them into eLTER reporting format*

Description

[Experimental]

Usage

```
map_occ_obis2elter(x, deimsid, version = "1.3")
```

Arguments

`x`	A `tibble` like one that can be obtained by as_tibble(get_site_speciesOccurrences(deimsid, "gbif")$gbif)
`deimsid`	A `character`. The DEIMS.iD of the site from DEIMS-SDR website. DEIMS ID information
`version`	A `character` for select the version of eLTER Data Reporting Format. Default 1.3

Value

list with the following named elements:

- deimsid: the same deimsid passed in input
- source: one of "gbif", "inat", "obis"
- data_mapping: tibble structured according to data_mapping of eLTER reporting format
- reference_TAXA: tibble structured according to reference_TAXA of eLTER reporting format
- reference_VARIABLES: tibble structured according to reference_VARIABLES of eLTER reporting format

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Martina Zilioli <zilioli.m@irea.cnr.it>

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

package_settings *Package settings that can be changed by the user*

Description

[Stable]

Usage

```
package_settings
```

See Also

Other package_customizable_settings: `get_deims_base_url()`

produce_network_points_map *Provide a map (image) of sites in an eLTER Network.*

Description

[Stable] Return a image map object of all of the eLTER sites belonging to an eLTER Network (e.g. LTER Italy network), as a stored into DEIMS-SDR.

Usage

```
produce_network_points_map(networkDEIMSID, countryCode)
```

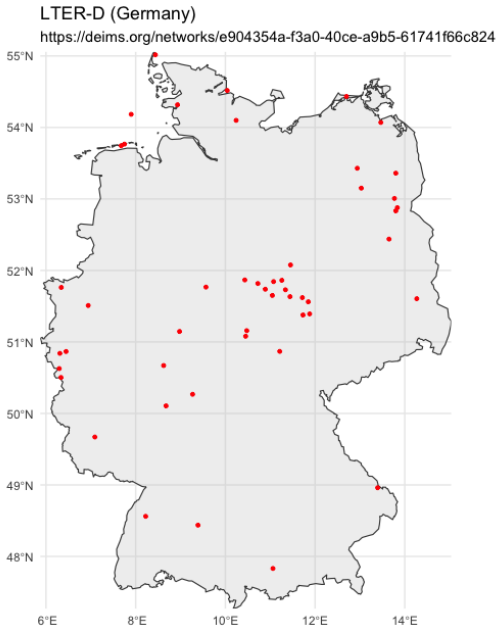
Arguments

networkDEIMSID	A `character`. The DEIMS ID of the network from DEIMS-SDR website. DEIMS ID information here and Complete list of ILTER networks here.
countryCode	A `character` following the ISO 3166-1 alpha-3 codes. This ISO convention consists of three-letter country codes as defined in ISO 3166-1. The ISO 3166 standard published by the International Organization for Standardization (ISO), to represent countries, dependent territories, and special areas of geographical interest. The map produced by this function will be limited only to the country indicated in this parameter, if the network has a extraterritorial sites those will not represented.

Value

The output of the function is a `ggplot2` plot containing an image of geographic distribution of the network of sites present in the chosen country. If the network contains extraterritorial sites, a map of these can be generated by specifying the network's DEIMS-ID and providing, in the country parameter, the country in which the extraterritorial sites are located.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

See Also

geodata::gadm()

Examples

```
## Not run:
# Italian sites
map <- produce_network_points_map(
  networkDEIMSID =
    "https://deims.org/networks/7fef6b73-e5cb-4cd2-b438-ed32eb1504b3",
  countryCode = "ITA"
)

# Italian extraterritorial sites (Nepal)
map <- produce_network_points_map(
  networkDEIMSID =
    "https://deims.org/networks/7fef6b73-e5cb-4cd2-b438-ed32eb1504b3",
  countryCode = "NPL"
)

# German sites
map_LTERGermanSites <- produce_network_points_map(
  networkDEIMSID =
```

```
    "https://deims.org/networks/e904354a-f3a0-40ce-a9b5-61741f66c824",
  countryCode = "DEU"
)

# Remove scale bar and a north arrow
map_LTERGermanSites <- map_LTERGermanSites +
  ggplot2::theme(
    panel.grid = ggplot2::element_blank(),
    axis.text = ggplot2::element_blank(),
    axis.ticks = ggplot2::element_blank(),
    axis.title = ggplot2::element_blank()
  )
 map_LTERGermanSites

 # Add annotation scale and North arrow
 map_LTERGermanSites <- map_LTERGermanSites +
   ggspatial::annotation_scale(
     location = "br", # bottom right
     width_hint = 0.2) +
   ggspatial::annotation_north_arrow(
     location = "bl", # bottom right
     which_north = "true",
     style = ggspatial::north_arrow_fancy_orienteering(),
     height = ggplot2::unit(1, "cm"),
     width = ggplot2::unit(1, "cm")
   )
 map_LTERGermanSites

## End(Not run)
```

`produce_site_map` *Provide a map object of a sites LTER.*

Description

[Stable] This function produces a map of the site boundaries as provided by the DEIMS-SDR catalogue, within a given country and network.

Usage

```
produce_site_map(
  deimsid,
  scale_location = "bl",
  arrow_location = "tl",
  inset_position = "br"
)
```

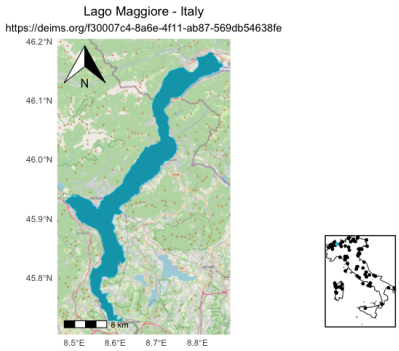
Arguments

`deimsid`	A `character`. The DEIMS ID of network from DEIMS-SDR website. DEIMS ID information here.
`scale_location`	A `character`. Position of the map scale (e.g. "bl", "br"). Options: `"tl"` = top-left, `"tr"` = top-right, `"bl"` = bottom-left, `"br"` = bottom-right. Default is `"bl"`.
`arrow_location`	A `character`. Position of the north arrow (e.g. "tl", "tr"). Options: `"tl"` = top-left, `"tr"` = top-right, `"bl"` = bottom-left, `"br"` = bottom-right. Default is `"tl"`.
`inset_position`	A `character`. Position of the country overview inset map. Options: `"tl"` = top-left, `"tr"` = top-right, `"bl"` = bottom-left, `"br"` = bottom-right. Default is `"br"`.

Value

The output of the function is an `image` of the boundary of the site, OSM as base map and all country sites map.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

See Also

`ggspatial::annotation_map_tile()`

`ggspatial::annotation_scale()`

`ggspatial::annotation_north_arrow()`

`cowplot::ggdraw()`

`cowplot::draw_plot()`

`geodata::gadm()`

Examples

```
## Not run:
# Example of Lange Bramke site
siteMap <- produce_site_map(
  deimsid = "https://deims.org/8e24d4f8-d6f6-4463-83e9-73cac2fd3f38"
)

# Example of Eisenwurzen site
siteMap <- produce_site_map(
  deimsid = "https://deims.org/d0a8da18-0881-4ebe-bccf-bc4cb4e25701",
  inset_position = "bl"
)

# Example of Lake Maggiore site
siteMap <- produce_site_map(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe",
  scale_location = "bl",
  arrow_location = "tl",
  inset_position = "br"
)

## End(Not run)
```

`produce_site_observedProperties_pie` *Produce a pie chart of the observed properties collected in a site LTER.*

Description

[Stable] Return a pie chart of Environmental observed properties, as a stored in DEIMS-SDR catalogue, of a single eLTER site.

Usage

```
produce_site_observedProperties_pie(deimsid)
```

Arguments

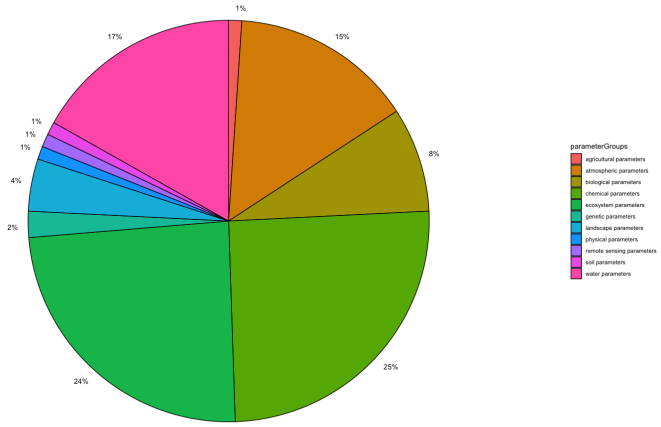
`deimsid`	A `character`. It is the DEIMS ID of site/network from DEIMS-SDR website. DEIMS ID information here.

Value

The output of the function is a pie chart and a `tibble`. The percentages, as a label in the pie charts and in the output table (column 'perc'), refer to the number of the observed properties, belonging to a type (e.g. biological, atmospheric, etc.), measured compared to all of observed properties measured into selected eLTER site. This function allows to show what type of observed

properties are most measured into a site. In the example below the atmospheric observed properties corresponds to the 15 percent of all observed properties measured into the site.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

See Also

`ggforce::geom_arc_bar()`

`RColorBrewer::brewer.pal()`

Examples

```
## Not run:
pie <- produce_site_observedProperties_pie(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe"
)

## End(Not run)
```

`produce_site_observedProperties_waffle`

Produce a waffle chart of the observed properties collected in an eLTER site

---

Description

[Stable] Returns a waffle chart of environmental observed properties, as stored in DEIMS-SDR catalogue, for a single eLTER site. The chart is built with **ggplot2** and requires no additional packages. Each square represents one observed property; squares of the same colour belong to the same parameter group (e.g. biological, atmospheric, etc.).

Usage

```
produce_site_observedProperties_waffle(deimsid)
```

Arguments

`deimsid`	A `character`. The DEIMS ID of the site from DEIMS-SDR website. DEIMS ID information here.

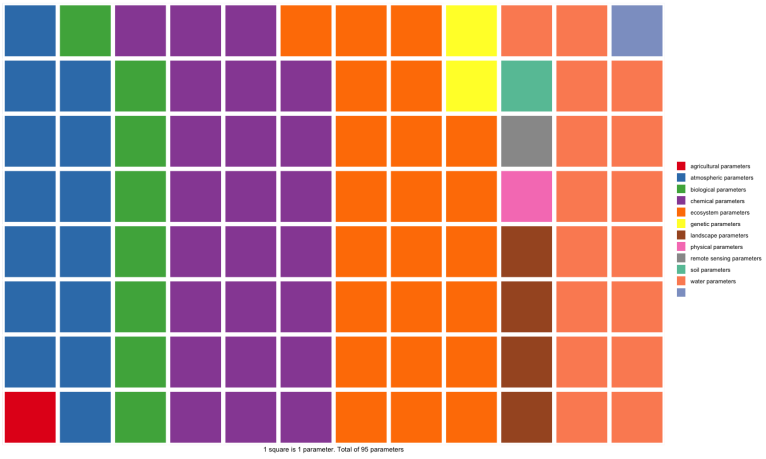
Value

The function prints a waffle chart as a side effect and returns a `tibble` with four columns:

- `parameterGroups` `character`. Name of the parameter group.
- `n` `integer`. Number of observed properties in the group.
- `freq` `double`. Relative frequency of the group.
- `label` `character`. Percentage label for the group.

Returns `invisible(NULL)` if the DEIMS ID is invalid or the site has no observed properties.

The function output



Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

See Also

RColorBrewer::brewer.pal()

get_site_info()

Examples

```
## Not run:
waffle <- produce_site_observedProperties_waffle(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe"
)
waffle

## End(Not run)
```

---

produce_site_qrcode *Obtain the QRCode of any DEIMS-SDR entities.*

Description

[Stable] Return a QR code image of any provided DEIMS ID (e.g. dataset, site, activity).

Usage

```
produce_site_qrcode(deimsid, do_plot = FALSE)
```

Arguments

deimsid	A character. The DEIMS ID of entities from DEIMS-SDR website. DEIMS ID information here.
do_plot	A boolean. Plot the computed QRCode. Default FALSE.

Value

The QR code as a logical matrix with "qr_code" class.

The function output

Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

Examples

```
qrcode <- produce_site_qrcode(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe"
)

a <- produce_site_qrcode(
  deimsid = "https://deims.org/f30007c4-8a6e-4f11-ab87-569db54638fe",
  do_plot = TRUE
)
```

---

reporting_produce_data_object_v1.3

Compose an eLTER Data Reporting Format object

Description

[Experimental] Given several tables, creates an eLTER data reporting format object

Usage

```
reporting_produce_data_object_v1.3(
  data = NULL,
  station = NULL,
```

```
  method = NULL,
  reference = NULL,
  event = NULL,
  sample = NULL,
  license = "",
  deimsid = "",
  data_type = "measurement",
  filename = NULL
)
```

Arguments

data	A tibble. See eLTER data specification format for details
station	A tibble
method	A tibble
reference	A tibble
event	A tibble
sample	A tibble
license	A character
deimsid	A character. The DEIMS ID of the site from DEIMS-SDR website. DEIMS ID information here.
data_type	A character. Data must be provided by one of measurement or mapping. Default 'measurement'
filename	optional filename associated with the object, of the form provided as output by the function reporting_compose_file_name

Value

list with eLTER reporting format slots

Note

This method must be intended as a signpost for future implementation

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

See Also

Peterseil, Geiger et al. (2020) Field Specification for data reporting. Technical Document. TechDoc.01. EU Horizon 2020 eLTER PLUS Project, Grant agreement No. 871128 https://zenodo.org/record/6373410

Examples

```
## Not run:
## Not run:

deimsid <- "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6"
time_span <- 2015 # e.g. whole year
# time_span <- "20150302-20180415" # e.g. span between two dates
data_topic <- "VEG" # data provider defined abbreviation of "vegetation"
variable_group <- "SPECCOVER" # data provider defined abbreviation
version <- "V20220907"

filename <- reporting_compose_file_name(
  deimsid = deimsid,
  data_topic = data_topic,
  variable_group = variable_group,
  time_span = time_span,
  version = version
)

data <- tibble::tribble(
  ~SITE_CODE,
  ~VARIABLE,
  ~TIME,
  ~VALUE,
  ~UNIT,
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "TEMP", "2016-03-15", "5.5",  "\u00b0C",
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "PREC", "2016-03-03", "10.2", "mm",
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "TEMP", "2016-02-15", "2.5",  "\u00b0C",
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "NH4N", "2016-03",   "5.5",  "mg/l",
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "SO4S", "2016-03",   "10.2", "mg/l",
 "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "CA",  "2016-03",   "2.5",  "mg/l"
)

station <- dplyr::tribble(
  ~SITE_CODE, ~STATION_CODE, ~STYPE, ~LAT,      ~LON,       ~ALTITUDE,
  deimsid,    "IP2",         "AREA",  45.340805, 7.88887495, 265
)
method <- dplyr::tribble(
  ~VARIABLE, ~METH_DESCR,
  "COVE_F",  "Analysis of ammonium..."
)

research_object <- reporting_produce_data_object_v1.3(
 filename = filename,
 deimsid = deimsid,
 data = data,
 station = station,
 method = method
)


## End(Not run)
```

```
## End (Not run)
```

`reporting_produce_data_object_v2.0`
Compose an eLTER Data Reporting Format object

Description

[Experimental] Given several tables, creates an eLTER data reporting format object

Usage

```
reporting_produce_data_object_v2.0(
  station = NULL,
  method = NULL,
  data = NULL,
  reference = NULL,
  event = NULL,
  sample_event = NULL,
  license = NULL,
  deimsid = NULL,
  data_topic,
  variable_group = "",
  time_span,
  version = Sys.Date() %>% format("V%Y%m%d")
)
```

Arguments

`station`	A `tibble` containing the station information. Station is an observation entity within a site or platform. Station in this respect is synonym to plot, observation location, sensor location, etc. and is defined by a location, elevation and installation height (if relevant) which is located within a given LTER site or platform.
`method`	A `tibble` containing the method information. Method describes the procedure to generate and manipulate the data.
`data`	A `tibble` containing the data. Data are defined as the sum of observation values being observed at a station/plot either by sensor, measurement device or human observation.
`reference`	A `tibble` containing the reference information. Reference is the listing and description of additional codes used in the data provision.
`event`	A `tibble` containing the event information. Event is defined as activity to observe or collect information on the ecosystem characteristic of interest.

sample_event	A tibble containing the sample event information. Sample event is the key observational units in environmental sciences, i.e. ecology, geosciences, biogeochemistry, and hydrobiology, and are essential to document and further analyse biological communities in laboratories (e.g., phytoplankton communities in water samples, benthic communities in sediments, etc.).
license	A character. It is a textual description of the conditions to use for the data.
deimsid	A character The DEIMS ID of the site from DEIMS-SDR website. More information about DEIMS ID in this pages: page.
data_topic	A character. Max 5-digit code for data topic or observation programme, e.g. METEO (Meteorology), BIODIV (Biodiversity), DEPO (deposition), GHG (Green House gas), SW (Soil water), VEG (Vegetation). The abbreviation is defined by the data provider depending on the data.
variable_group	A character. Optional, list of variables or variable groups contained in the data. The abbreviation is defined by the data provider depending on the data.
time_span	A numeric or a character. Time span covered in the data. E.g. 2015, 20150302-20180415. The time span is defined by the data provider.
version	version in format "VYYYYMMDD". Data version in the format "V"YYYYMMDD. Defaults to current system date.

Value

list with eLTER reporting format slots

Author(s)

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

See Also

Peterseil, Geiger et al. (2020) Field Specification for data reporting. Technical Document. TechDoc.01. EU Horizon 2020 eLTER PLUS Project, Grant agreement No. 871128 https://zenodo.org/record/6373410

Examples

```
## Not run:
## Not run:
deimsid <- "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6"
time_span <- 2015 # e.g. whole year
# time_span <- "20150302-20180415" # e.g. span between two dates
data_topic <- "VEG" # data provider defined abbreviation of "vegetation"
variable_group <- "SPECCOVER" # data provider defined abbreviation
version <- "V20220907"

data <- tibble::tribble(
  ~SITE_CODE,
  ~VARIABLE,
  ~TIME,
  ~VALUE,
```

```
  ~UNIT,
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "TEMP", "2016-03-15", "5.5",  "\u00b0C",
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "PREC", "2016-03-03", "10.2", "mm",
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "TEMP", "2016-02-15", "2.5",  "\u00b0C",
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "NH4N", "2016-03",   "5.5",  "mg/l",
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "SO4S", "2016-03",   "10.2", "mg/l",
  "https://deims.org/8eda49e9-1f4e-4f3e-b58e-e0bb25dc32a6", "CA",  "2016-03",   "2.5",  "mg/l"
)

station <- dplyr::tribble(
  ~SITE_CODE, ~STATION_CODE, ~STYPE, ~LAT,      ~LON,       ~ALTITUDE,
  deimsid,    "IP2",         "AREA",  45.340805, 7.88887495, 265
)
method <- dplyr::tribble(
  ~VARIABLE, ~METH_DESCR,
  "COVE_F",  "Analysis of ammonium..."
)

research_object <- reporting_produce_data_object_v2.0(
  station = station,
  method = method,
  data = data,
  deimsid = deimsid,
  data_topic = data_topic,
  variable_group = variable_group,
  time_span = time_span,
  version = version
)


## End(Not run)
## End (Not run)
```

`reporting_save_archive` *Creates an archive with files following the eLTER reportingFormat*

Description

[Experimental] Creates a zip archive "filename".zip

Usage

```
reporting_save_archive(x, filepath = tempdir(), saveRDS = FALSE)
```

Arguments

`x`	A `list` like the one created by function `reporting_produce_data_object_v2.0`
`filepath`	A `character` file path. Defaults to temporary directory
`saveRDS`	A `logical`. Save also object in RDS format. Defaults to FALSE

Value

named A `list` containing paths to saved files filepaths. Slots are named "zip" and possibly "RDS". A path to the zip file, or rds file, can be defined by the parameter `filepath`.

Note

This method must be intended as a signpost for future implementation

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

Alessandro Oggioni, phD <oggioni.a@irea.cnr.it>

See Also

Peterseil, Geiger et al. (2020) Field Specification for data reporting. Technical Document. TechDoc.01. EU Horizon 2020 eLTER PLUS Project, Grant agreement No. 871128 https://zenodo.org/record/6373410

Examples

```
## Not run:
## Not run:

archive <- reporting_save_archive(
  x = research_object,
  # obtained from the function `reporting_produce_data_object_v2.0()`
  filepath = ".",
  saveRDS = TRUE
)

## End (Not run)

## End(Not run)
```

---

save_occ_eLTER_reporting_Archive

Creates an archive with files following the eLTER reportingFormat

---

Description

[Experimental] Creates a zip archive named biodiv_occurrence_site_"deimsid_code"_"source".zip where "deimsid_code" is the uuid in the last part of the deimsid, and "source" is one of "gbif", "inat", "obis"

Usage

```
save_occ_eLTER_reporting_Archive(lterReportOut, path = tempdir())
```

Arguments

lterReportOut	A list like the one created by map_occ_gbif2elter
path	path of the zip file. Defaults to temporary folder

Value

the path to the created file

Author(s)

Paolo Tagliolato, phD <tagliolato.p@irea.cnr.it>

---

set_deims_base_url *Set DEIMS-SDR API base URL*

Description

[Stable]

Usage

```
set_deims_base_url(url = "https://deims.org/", force = FALSE)
```

Arguments

url	A character. Set the base URL to DEIMS-SDR.
force	A boolean. Default FALSE.

---

taxon_id_pesi *Provide a taxon ID* R*hrefhttps://en.wikipedia.org/wiki/LSIDLSID to a taxon list.*

Description

[Stable] This function provide a taxon ID, usually a LSID, from a taxonomic list. The input of the function is a csv file with a list of taxa. The Taxon ID provided by this function is currently taken from Pan-European Species directories Infrastructure - PESI. This function takes advantage of taxize's eubon_search function https://docs.ropensci.org/taxize/ and the PESI RestAPI.

Usage

```
taxon_id_pesi(table, taxaColumn)
```

Arguments

`table`	A `data.frame` containing column with a taxa (e.g. Sphaerosoma seidlitzi, Malthinus, etc.).
`taxaColumn`	A `numeric` that identify the column containing taxa value.

Value

The output of the function is a `tibble` containing all the columns provided as input and new columns as: 'canonicalName', 'authorship', 'synonyms', 'LSID', 'url', 'accordingTo', 'checkStatus' gathered from PESI.

An example to export dataset obtained by this function is: datasetMerged <- dplyr::bind_rows(table) write.csv(datasetMerged, "table.csv", row.names = FALSE, fileEncoding = "UTF-8")

Someone could have problems of characters encoding when CSV file is written. To resolve we suggest two different solutions:

Solution 1 -

1. Open the CSV in Notepad.
2. Click "File" and "Save As".
3. In the new popup that displays, select "ANSI" from the "Encoding" field.
4. Click "Save".
5. Now, you should be able to open the file in Excel and display the characters correctly.

Solution 2 -

1. Open Excel
2. Click "File" and "New"
3. Click on the "Data" tab
4. Click "From Text" and select the CSV file
5. Select "Delimited"
6. For "File origin", select "65001 : Unicode (UTF-8)"
7. Click "Next"
8. Select "Comma"
9. Click "Finish"
10. Excel should now show you the CSV file and display the characters correctly.

Author(s)

Alessandro Oggioni, phD (2020) <oggioni.a@irea.cnr.it>

See Also

`taxize::eubon_search()`

Examples

```
## Not run:
insects <- data.frame(
   taxonID = c(1, 2, 3, 4, 5, 6),
   family = c(
     "Alexiidae", "Anthicidae",
     "Anthribidae", "Anthribidae",
     "Biphyllidae", "Brentidae"
   ),
   scientificName = c(
     "Sphaerosoma seidlitzi", "Endomia tenuicollis tenuicollis",
     "Anthribus fasciatus", "Phaenotherion fasciculatum fasciculatum",
     "Diplocoelus fagi", "Holotrichapion (Apiops) pisi"
   )
)

output <- taxon_id_pesi(
  table = insects,
  taxaColumn = 3
)

# The annotated URIs of columns label are achieved by:
attributes(output)$uri


## End(Not run)
```

`taxon_id_worms` *Enrich and certify a list of species names by comparing with* R*hrefhttps://www.marinespecies.orgWoRMS.*

Description

[Stable] This function provide tibble object with all the columns of input table of taxa plus new columns such as valid_name, valid_authority, valid_AphiaID, status, synonyms, LSID, url, matchType, nOfWormsRecords, wormsRecords obtained from Word Register of Marine Species WoRMS rest API.

Usage

```
taxon_id_worms(input, taxaColumn = 1, verbose = TRUE, refine = FALSE)
```

Arguments

`input`	A `tibble`. The table that contain the species names list to be checked.
`taxaColumn`	A `numeric`. The cardinal number of the column where species list is. Default is `1`.

verbose — A logical. Whit this selection, the function returns a message with number of record(s) that don't match with any Worms names and the number of record(s) that match with more that one Worms name. Default is TRUE.

refine — A logical. With this selection, the function allows to refine the result(s) that match with more Worms records. By a interactive use of the terminal, the user can chose the result. Default is FALSE.

Value

The output of the function is a tibble with the columns provided and new columns such as: valid_name, valid_authority, valid_AphiaID, status, synonyms, LSID, url, matchType, nOfWormsRecords, wormsRecords obtained by Worms rest API. The function also return, if verbose is TRUE, the list of records that don't match with Worms name species.

Most of the labels of the columns are the terms of Darwin Core terms. The columns labels are annotate with the link (URI) of the Darwin Core terms as attributes of the tibble.

Author(s)

Alessandro Oggioni, phD (2021) <oggioni.a@irea.cnr.it>

Paolo Tagliolato, phD (2021) <tagliolato.p@irea.cnr.it>

See Also

worrms::wm_records_names()

Examples

```
phytoplankton <- tibble::tibble(
   ID = c(1, 2, 3, 4, 5, 6, 7),
   species = c(
   "Asterionella formosa", "Chrysococcus sp.",
   "Cryptomonas rostrata", "Dinobryon divergens",
   "Mallomonas akrokomos", "Melosira varians",
   "Cryptomonas rostrata"
 )
)
table <- taxon_id_worms(
  input = phytoplankton,
  taxaColumn = 2,
  verbose = TRUE,
  refine = TRUE
)
table

# The annotated URIs of columns label are achieved by:
attributes(table)$uri
```

Index