Package 'ecocomDP'

July 31, 2021

Type Package

Title Work with Datasets in the Ecological Community Design Pattern

Description Tools to create, use, and convert 'ecocomDP' datasets. 'ecocomDP' is a dataset design pattern for harmonizing ecological community surveys in a research question agnostic format, from source datasets published across multiple repositories, and with methods that keep the derived datasets up-to-date as the underlying sources change. Described in O'Brien et al. (2021), <doi:10.1016/j.ecoinf.2021.101374>.

Version 1.1.0

License MIT + file LICENSE

URL https://github.com/EDIorg/ecocomDP

BugReports https://github.com/EDIorg/ecocomDP/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.4.0)

Imports RColorBrewer (>= 1.1-2), data.table (>= 1.13.0), dplyr (>= 0.7.1), EML (>= 2.0.5), emld (>= 0.5.1), geosphere (>= 1.5), ggplot2 (>= 3.3.3), httr (>= 1.3.1), lubridate (>= 1.7.4), magrittr, neonUtilities (>= 2.1.1), rlang, rmarkdown, stats, stringr (>= 1.2.0), tidyr (>= 0.6.1), tools, utils, uuid (>= 0.1-4), xml2 (>= 1.3.0)

Suggests knitr, mime (>= 0.9), reader (>= 1.0.6), ritis (>= 1.0.0), taxize (<= 0.9.99), testthat, worrms (>= 0.4.2)

VignetteBuilder knitr

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Colin Smith [aut, cre, cph] (<https://orcid.org/0000-0003-2261-9931>), Eric Sokol [aut] (<https://orcid.org/0000-0001-5923-0917>), Margaret O'Brien [aut] (<https://orcid.org/0000-0002-1693-8322>), Matt Bitters [ctb],

Melissa Chen [ctb],
Savannah Gonzales [ctb],
Matt Helmus [ctb],
Brendan Hobart [ctb],
Ruvi Jaimes [ctb],
Lara Janson [ctb],
Marta Jarzyna [ctb],
Michael Just [ctb],
Daijiang Li [ctb],
Wynne Moss [ctb],
Kari Norman [ctb],
Stephanie Parker [ctb],
Natalie Robinson [ctb],
Thilina Surasinghe [ctb]

Maintainer Colin Smith <colin.smith@wisc.edu>

Repository CRAN

Date/Publication 2021-07-31 11:50:08 UTC

R topics documented:

`ants_L0_flat` *Joined and flat version of EDI data package knb-lter-hfr.118.33*

Description

A fully joined and flat version of EDI data package knb-lter-hfr.118.33 (Ant Assemblages in Hemlock Removal Experiment at Harvard Forest since 2003) with all relevant ecocomDP L1 identifiers and content added. Use this dataset as an input to the `L0_flat` argument of the "create" functions.

Usage

```
ants_L0_flat
```

Format

A data frame with 2931 rows and 45 variables:

datetime dates

block block

plot plot number

treatment treatment type

moose.cage location of grid with respect to moose exclosure

trap.type trap type

trap.num applies only to pitfall cups

subfamily ant subfamily

hl head length. We used trait definitions from Del Toro et al. (2015) and filled in missing species' data with information from Ellison et al.

rel eye length relative to body size

rll femur length relative to body size

colony.size size of colony for each species

feeding.preference feeding preference for each species

nest.substrate nest substrate

primary.habitat primary habitat

secondary.habitat secondary habitat associations

seed.disperser whether or not a seed dispersing species

slavemaker.sp whether or not a slavemaking species

behavior classifications based on behavioral interactions with other ants

biogeographic.affinity biogeographic affinity based on available occurrence records

source where trait information was found. Full citations for literature are as follows: Del Toro, I., R.R. Silva, and A.M. Ellison. 2015. Predicated impacts of climatic change on ant functional diversity and distributions in eastern North American forests. Diversity and Distributions 21:781-791; Ellison, A.M., N.J. Gotelli, G. Alpert, and E.J. Farnsworth. 2012. A field guide to the ants of New England. Yale University Press, New Haven, Connecticut, USA.

unit_hl units for "hl" variable

unit_rel units for "rel" variable

unit_rll units for "rll" variable

variable_name variables of the primary observation table

value values of variable_name

unit units of variable_name

observation_id the observation id

location_id the location id

event_id the event id

latitude approximate latitude of study area

longitude approximate longitude of study area

elevation approximate elevation of study area

taxon_name name of organism

taxon_id the taxon id

taxon_rank the taxon rank

authority_system the authority system taxon_name was resolved to

authority_taxon_id the id of taxon_name in authority_system

package_id the identifier of this ecocomDP dataset

original_package_id the identifier of the source dataset

length_of_survey_years number of years the survey has been ongoing

number_of_years_sampled number of years during the survey that samples were taken

std_dev_interval_betw_years the standard deviation between surveys in years

max_num_taxa number of unique taxa in this dataset

geo_extent_bounding_box_m2 the study area in meters squared

Source

https://portal.edirepository.org/nis/mapbrowse?scope=knb-lter-hfr&identifier=118&revision=33

ants_L1 *The ecocomDP version of EDI data package knb-lter-hfr.118.33*

Description

The the ecocomDP (L1) formatted version of EDI data package knb-lter-hfr.118.33 (Ant Assemblages in Hemlock Removal Experiment at Harvard Forest since 2003) read from the EDI API with `read_data(id = "edi.193.5")`. Use this dataset as an input to data "use" functions.

Usage

```
ants_L1
```

Format

A list of:

edi.193.5 The dataset identifier

- **metadata** See source url for metadata
- **tables** A list of data frames, each an ecocomDP table
- **validation_issues** Is NULL because there are no validation issues for this dataset

Source

https://portal.edirepository.org/nis/mapbrowse?scope=edi&identifier=193&revision=5

calc_geo_extent_bounding_box_m2 *Calculate geo_extent_bounding_box_m2 for the dataset_summary table*

Description

Calculate geo_extent_bounding_box_m2 for the dataset_summary table

Usage

```
calc_geo_extent_bounding_box_m2(west, east, north, south)
```

Arguments

west	(numeric) West longitude in decimal degrees and negative if west of the prime meridian.
east	(numeric) East longitude in decimal degrees and negative if west of the prime meridian.
north	(numeric) North latitude in decimal degrees and negative if south of the equator.
south	(numeric) South latitude in decimal degrees and negative if south of the equator.

Value

(numeric) Area of study site in meters squared.

`calc_length_of_survey_years` *Calculate length_of_survey_years for the dataset_summary table*

Description

Calculate length_of_survey_years for the dataset_summary table

Usage

```
calc_length_of_survey_years(dates)
```

Arguments

`dates`	(Date) Dates from the L0 source dataset encompassing the entire study duration.

Value

(numeric) Number of years the study has been ongoing.

`calc_number_of_years_sampled` *Calculate number_of_years_sampled for the dataset_summary table*

Description

Calculate number_of_years_sampled for the dataset_summary table

Usage

```
calc_number_of_years_sampled(dates)
```

Arguments

`dates`	(Date) Dates from the L0 source dataset encompassing the entire study duration.

Value

(numeric) Number of survey years in which a sample was taken.

`calc_std_dev_interval_betw_years` *Calculate std_dev_interval_betw_years for the dataset_summary table*

Description

Calculate std_dev_interval_betw_years for the dataset_summary table

Usage

```
calc_std_dev_interval_betw_years(dates)
```

Arguments

`dates`	(Date) Dates from the L0 source dataset encompassing the entire study duration.

Value

(numeric) The standard deviation between sampling events (in years).

`convert_to_dwca` *Convert an ecocomDP dataset to a Darwin Core Archive dataset*

Description

Convert an ecocomDP dataset to a Darwin Core Archive dataset

Usage

```
convert_to_dwca(
  path,
  core_name,
  source_id,
  derived_id,
  url = NULL,
  user_id,
  user_domain
)
```

Arguments

path	(character) Path to which the DwC-A data objects and EML will be written.
core_name	(character) The central table of the DwC-A dataset being created. Can be: "event" (event core). Occurrence core is not yet supported.
source_id	(character) Identifier of an ecocomDP dataset published in a supported repository. Currently, the EDI Data Repository is supported.
derived_id	(character) Identifier of the DwC-A dataset being created.
url	(character) URL to the publicly accessible directory containing DwC-A data objects. This argument supports direct download of the data entities by a data repository and is used for automated revisioning and publication.
user_id	(character) Identifier of user account associated with the data repository in which this ecocomDP dataset will be archived. Only `user_id` from the EDI is currently supported.
user_domain	(character) Domain (data repository) the `user_id` belongs to. Currently, EDI is supported.

Details

Reads in an ecocomDP dataset from a supported repository and converts it to a DwC-A package.

Value

DwC-A tables, meta.xml, and corresponding EML metadata.

Examples

```
## Not run:
# Create directory for DwC-A outputs
mypath <- paste0(tempdir(), "/data")
dir.create(mypath)

# Convert an EDI published ecocomDP dataset to a DwC-A
convert_to_dwca(
  path = mypath,
  core_name = "event",
  source_id = "edi.193.5",
  derived_id = "edi.834.2",
  user_id = "ecocomdp",
  user_domain = "EDI")

dir(mypath)

# Clean up
unlink(mypath, recursive = TRUE)

## End(Not run)
```

create_dataset_summary *Create the dataset_summary table*

Description

Create the dataset_summary table

Usage

```
create_dataset_summary(
  L0_flat,
  package_id,
  original_package_id = NULL,
  length_of_survey_years,
  number_of_years_sampled,
  std_dev_interval_betw_years,
  max_num_taxa,
  geo_extent_bounding_box_m2 = NULL
)
```

Arguments

L0_flat	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
package_id	(character) Column in L0_flat containing the identifier of the derived L1 dataset.
original_package_id	(character) An optional column in L0_flat containing the identifier of the source L0 dataset.
length_of_survey_years	(character) Column in L0_flat containing the number of years the study has been ongoing. Use calc_length_of_survey_years() to calculate this value.
number_of_years_sampled	(character) Column in L0_flat containing the number of years within the period of study that samples were taken. Use calc_number_of_years_sampled() to calculate this value.
std_dev_interval_betw_years	(character) Column in L0_flat containing the standard deviation of the interval between sampling events. Use calc_std_dev_interval_betw_years() to calculate this value.
max_num_taxa	(character) Column in L0_flat containing the number of unique taxa in the source L0 dataset.
geo_extent_bounding_box_m2	(character) An optional column in L0_flat containing the area (in meters) of the study location, if applicable (some L0 were collected at a single point). Use calc_geo_extent_bounding_box_m2() to calculate this value.

Details

This function collects specified columns from `L0_flat` and returns distinct rows.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The dataset_summary table.

Examples

```
flat <- ants_L0_flat

dataset_summary <- create_dataset_summary(
  L0_flat = flat,
  package_id = "package_id",
  original_package_id = "original_package_id",
  length_of_survey_years = "length_of_survey_years",
  number_of_years_sampled = "number_of_years_sampled",
  std_dev_interval_betw_years = "std_dev_interval_betw_years",
  max_num_taxa = "max_num_taxa",
  geo_extent_bounding_box_m2 = "geo_extent_bounding_box_m2")

dataset_summary
```

---

create_eml *Create EML metadata for an ecocomDP dataset*

Description

Create EML metadata for an ecocomDP dataset

Usage

```
create_eml(
  path,
  source_id,
  derived_id,
  script,
  script_description,
  is_about = NULL,
  contact,
  user_id,
  user_domain,
```

```
  basis_of_record = NULL,
  url = NULL
)
```

Arguments

path	(character) Path to the directory containing ecocomDP tables, conversion script, and where EML metadata will be written.
source_id	(character) Identifier of a data package published in a supported repository. Currently, the EDI Data Repository is supported.
derived_id	(character) Identifier of the dataset being created.
script	(character) Name of file used to convert source_id to derived_id.
script_description	(character) Description of script.
is_about	(named character) An optional argument for specifying dataset level annotations describing what this dataset "is about".
contact	(data.frame) Contact information for the person that created this ecocomDP dataset, containing these columns: • givenName • surName • organizationName • electronicMailAddress
user_id	(character) Identifier of user associated with user_domain.
user_domain	(character) Domain (data repository) the user_id belongs to. Currently, EDI is supported.
basis_of_record	(character) An optional argument to facilitate creation of a Darwin Core record from this dataset using convert_to_dwca(). Use this to define the Darwin Core property basisOfRecord as HumanObservation or MachineObservation.
url	(character) URL to the publicly accessible directory containing ecocomDP tables, conversion script, and EML metadata. This argument supports direct download of the data entities by a data repository and is used for automated revisioning and publication.

Details

This function creates an EML record for an ecocomDP by combining metadata from source_id with boiler-plate metadata describing the ecocomDP model. Changes to the source_id EML include:

- **<access>** Adds user_id to the list of principals granted read and write access to the ecocomDP data package this EML describes.
- **<title>** Adds a note that this is a derived data package in the ecocomDP format.
- **<pubDate>** Adds the date this EML was created.
- **<abstract>** Adds a note that this is a derived data package in the ecocomDP format.

- **<keywordSet** Adds the "ecocomDP" keyword to enable search and discovery of all ecocomDP data packages in the data repository it is published, and 7 terms from the LTER Controlled vocabulary: "communities", "community composition", "community dynamics", "community patterns", "species composition", "species diversity", and "species richness". Darwin Core Terms listed under `basis_of_record` are listed and used by `convert_to_dwca()` to create a Darwin Core Archive of this ecocomDP data package.
- **<intellectualRights>** Keeps intact the original intellectual rights license `source_id` was released under, or uses CCO if missing.
- **<taxonomicCoverage>** Appends to the taxonomic coverage element with data supplied in the ecocomDP taxon table.
- **<contact>** Adds the ecocomDP creator as a point of contact.
- **<methodStep>** Adds a note that this data package was created by the `script`, and adds provenance metadata noting that this is a derived dataset and describes where the `source_id` can be accessed.
- **<dataTables>** Replaces the `source_id` table metadata with descriptions of the the ecocomDP tables.
- **<otherEntity>** Adds `script` and `script_description`. otherEntities of `source_id` are removed.
- **<annotations>** Adds boilerplate annotations describing the ecocomDP at the dataset, entity, and entity attribute levels.

Taxa listed in the taxon table, and resolved to one of the supported authority systems (i.e. ITIS, WORMS, or GBIF), will have their full taxonomic hierarchy expanded, including any common names for each level.

Value

An EML metadata file.

Examples

```
## Not run:
# Create directory with ecocomDP tables for create_eml()
mypath <- paste0(tempdir(), "/data")
dir.create(mypath)
inpts <- c(ants_L1$edi.193.5$tables, path = mypath)
do.call(write_tables, inpts)
file.copy(system.file("extdata", "create_ecocomDP.R", package = "ecocomDP"), mypath)

dir(mypath)

# Describe, with annotations, what the source L0 dataset "is about"
dataset_annotations <- c(
  `species abundance` = "http://purl.dataone.org/odo/ECSO_00001688",
  Population = "http://purl.dataone.org/odo/ECSO_00000311",
  `level of ecological disturbance` = "http://purl.dataone.org/odo/ECSO_00002588",
  `type of ecological disturbance` = "http://purl.dataone.org/odo/ECSO_00002589")

# Add self as contact information incase questions arise
```

```
additional_contact <- data.frame(
  givenName = 'Colin',
  surName = 'Smith',
  organizationName = 'Environmental Data Initiative',
  electronicMailAddress = 'ecocomdp@gmail.com',
  stringsAsFactors = FALSE)

# Create EML
eml <- create_eml(
  path = mypath,
  source_id = "knb-lter-hfr.118.33",
  derived_id = "edi.193.5",
  is_about = dataset_annotations,
  script = "create_ecocomDP.R",
  script_description = "A function for converting knb-lter-hrf.118 to ecocomDP",
  contact = additional_contact,
  user_id = 'ecocomdp',
  user_domain = 'EDI',
  basis_of_record = "HumanObservation")

dir(mypath)
# View(eml)

# Clean up
unlink(mypath, recursive = TRUE)

## End(Not run)
```

`create_location` *Create the location table*

Description

Create the location table

Usage

```
create_location(
  L0_flat,
  location_id,
  location_name,
  latitude = NULL,
  longitude = NULL,
  elevation = NULL
)
```

Arguments

L0_flat	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
location_id	(character) Column in L0_flat containing the identifier assigned to each unique location at the observation level.
location_name	(character) One or more columns in L0_flat of sampling locations ordered from high to low in terms of nesting, where the lowest is the level of observation (e.g. location_name = c("plot","subplot")).
latitude	(character) An optional column in L0_flat containing the latitude in decimal degrees of location_id. Latitudes south of the equator are negative.
longitude	(character) An optional column in L0_flat containing the longitude in decimal degrees of location_id. Longitudes west of the prime meridian are negative.
elevation	(character) An optional column in L0_flat containing the elevation in meters relative to sea level of location_id. Above sea level is positive. Below sea level is negative.

Details

This function collects specified columns from L0_flat, creates data frames for each location_name, assigns latitude, longitude, and elevation to the lowest nesting level (i.e. the observation level) returning NA for higher levels (these will have to be filled manually afterwards), and determines the relationships between location_id and parent_location_id from L0_flat and location_name.

To prevent the listing of duplicate location_name values, and to enable the return of location_name columns by flatten_data(), location_name values are suffixed with the column they came from according to: paste0(<column name>,"__",<column value>). Example: A column named "plot" with values "1", "2", "3", in L0_flat would be listed in the resulting location table under the location_name column as "1", "2", "3" and therefore no way to discern these values correspond with "plot". Applying the above listed solution returns "plot__1", "plot__2", "plot__3" in the location table and returns the column "plot" with values c("1", "2", "3") by flatten_data().

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Additionally, latitude, longitude, and elevation of sites nested above the observation level will have to be manually added after the location table is returned.

Value

(tbl_df, tbl, data.frame) The location table.

Examples

```
flat <- ants_L0_flat

location <- create_location(
```

```
  L0_flat = flat,
  location_id = "location_id",
  location_name = c("block", "plot"),
  latitude = "latitude",
  longitude = "longitude",
  elevation = "elevation")

location
```

`create_location_ancillary` *Create the location_ancillary table*

Description

Create the location_ancillary table

Usage

```
create_location_ancillary(
  L0_flat,
  location_id,
  datetime = NULL,
  variable_name,
  unit = NULL
)
```

Arguments

`L0_flat`	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
`location_id`	(character) Column in `L0_flat` containing the identifier assigned to each unique location at the observation level.
`datetime`	(character) An optional column in `L0_flat` containing the date, and if applicable time, of ancillary location data following the ISO-8601 standard format (e.g. YYYY-MM-DD hh:mm:ss).
`variable_name`	(character) Columns in `L0_flat` containing the ancillary location data.
`unit`	(character) An optional column in `L0_flat` containing the units of each `variable_name` following the column naming convention: unit_<variable_name> (e.g. "unit_depth").

Details

This function collects specified columns from `L0_flat`, converts into long (attribute-value) form by gathering `variable_name`. Regular expression matching joins `unit` to any associated `variable_name` and is listed in the resulting table's "unit" column.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The location_ancillary table.

Examples

```
flat <- ants_L0_flat

location_ancillary <- create_location_ancillary(
  L0_flat = flat,
  location_id = "location_id",
  variable_name = "treatment")

location_ancillary
```

`create_observation` *Create the observation table*

Description

Create the observation table

Usage

```
create_observation(
  L0_flat,
  observation_id,
  event_id = NULL,
  package_id,
  location_id,
  datetime,
  taxon_id,
  variable_name,
  value,
  unit = NULL
)
```

Arguments

L0_flat	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
observation_id	(character) Column in L0_flat containing the identifier assigned to each unique observation.
event_id	(character) An optional column in L0_flat containing the identifier assigned to each unique sampling event.
package_id	(character) Column in L0_flat containing the identifier of the derived L1 dataset.
location_id	(character) Column in L0_flat containing the identifier assigned to each unique location at the observation level.
datetime	(character) Column in L0_flat containing the date, and if applicable time, of the observation following the ISO-8601 standard format (e.g. YYYY-MM-DD hh:mm:ss).
taxon_id	(character) Column in L0_flat containing the identifier assigned to each unique organism at the observation level.
variable_name	(character) Column in L0_flat containing the names of variables measured.
value	(character) Column in L0_flat containing the values of variable_name.
unit	(character) An optional column in L0_flat containing the units of variable_name.

Details

This function collects specified columns from L0_flat and returns distinct rows.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The observation table.

Examples

```
flat <- ants_L0_flat

observation <- create_observation(
  L0_flat = flat,
  observation_id = "observation_id",
  event_id = "event_id",
  package_id = "package_id",
  location_id = "location_id",
  datetime = "datetime",
  taxon_id = "taxon_id",
  variable_name = "variable_name",
  value = "value",
```

```
  unit = "unit")

observation
```

`create_observation_ancillary` *Create the observation_ancillary table*

Description

Create the observation_ancillary table

Usage

```
create_observation_ancillary(
  L0_flat,
  observation_id,
  variable_name,
  unit = NULL
)
```

Arguments

`L0_flat`	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
`observation_id`	(character) Column in `L0_flat` containing the identifier assigned to each unique observation.
`variable_name`	(character) Columns in `L0_flat` containing the ancillary observation data.
`unit`	(character) An optional column in `L0_flat` containing the units of each `variable_name` following the column naming convention: unit_<variable_name> (e.g. "unit_temperature").

Details

This function collects specified columns from `L0_flat`, converts into long (attribute-value) form by gathering `variable_name`. Regular expression matching joins `unit` to any associated `variable_name` and is listed in the resulting table's "unit" column.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The observation_ancillary table.

Examples

```
flat <- ants_L0_flat

observation_ancillary <- create_observation_ancillary(
  L0_flat = flat,
  observation_id = "observation_id",
  variable_name = c("trap.type", "trap.num", "moose.cage"))

observation_ancillary
```

---

create_taxon *Create the taxon table*

Description

Create the taxon table

Usage

```
create_taxon(
  L0_flat,
  taxon_id,
  taxon_rank = NULL,
  taxon_name,
  authority_system = NULL,
  authority_taxon_id = NULL
)
```

Arguments

L0_flat	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
taxon_id	(character) Column in L0_flat containing the identifier assigned to each unique organism at the observation level.
taxon_rank	(character) An optional column in L0_flat containing the taxonomic rank of the organism in taxon_name.
taxon_name	(character) Column in L0_flat containing the taxonomic name of the organism.
authority_system	(character) An optional column in L0_flat containing the name of the authority system authority_taxon_id is from (e.g. "ITIS").
authority_taxon_id	(character) An optional column in L0_flat containing the identifier corresponding to taxon_name in the authority_system.

Details

This function collects specified columns from `L0_flat` and returns distinct rows.

Taxa listed in the taxon table, and resolved to one of the supported authority systems (i.e. ITIS, WORMS, or GBIF), will have their full taxonomic hierarchy expanded, including any common names for each level.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The taxon table.

Examples

```
flat <- ants_L0_flat

taxon <- create_taxon(
  L0_flat = flat,
  taxon_id = "taxon_id",
  taxon_rank = "taxon_rank",
  taxon_name = "taxon_name",
  authority_system = "authority_system",
  authority_taxon_id = "authority_taxon_id")

taxon
```

`create_taxon_ancillary` *Create the taxon_ancillary table*

Description

Create the taxon_ancillary table

Usage

```
create_taxon_ancillary(
  L0_flat,
  taxon_id,
  datetime = NULL,
  variable_name,
  unit = NULL,
  author = NULL
)
```

Arguments

L0_flat	(tbl_df, tbl, data.frame) The fully joined source L0 dataset, in "flat" format (see details).
taxon_id	(character) Column in L0_flat containing the identifier assigned to each unique organism at the observation level.
datetime	(character) An optional in L0_flat containing the date, and if applicable time, of ancillary location data following the ISO-8601 standard format (e.g. YYYY-MM-DD hh:mm:ss).
variable_name	(character) Columns in L0_flat containing the ancillary taxon data.
unit	(character) An optional column in L0_flat containing the units of each variable_name following the column naming convention: unit_<variable_name> (e.g. "unit_average_length").
author	(character) An optional column in L0_flat containing the person associated with identification of taxa in the taxon table.

Details

This function collects specified columns from L0_flat, converts into long (attribute-value) form by gathering variable_name. Regular expression matching joins unit to any associated variable_name and is listed in the resulting table's "unit" column.

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) The taxon_ancillary table.

Examples

```
flat <- ants_L0_flat

taxon_ancillary <- create_taxon_ancillary(
  L0_flat = flat,
  taxon_id = "taxon_id",
  variable_name = c(
    "subfamily", "hl", "rel", "rll", "colony.size",
    "feeding.preference", "nest.substrate", "primary.habitat",
    "secondary.habitat", "seed.disperser", "slavemaker.sp",
    "behavior", "biogeographic.affinity", "source"),
  unit = c("unit_hl", "unit_rel", "unit_rll"))

taxon_ancillary
```

create_variable_mapping *Create the variable_mapping table*

Description

Create the variable_mapping table

Usage

```
create_variable_mapping(
  observation,
  observation_ancillary = NULL,
  location_ancillary = NULL,
  taxon_ancillary = NULL
)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
observation_ancillary	(tbl_df, tbl, data.frame) The optional observation_ancillary table.
location_ancillary	(tbl_df, tbl, data.frame) The optional location_ancillary table.
taxon_ancillary	(tbl_df, tbl, data.frame) The optional taxon_ancillary table.

Details

This function collects specified data tables, extracts unique variable_name values from each, converts into long (attribute-value) form with the table name and variable_name values to the resulting table's "table_name" and "variable_name" columns, respectively. The resulting table's "mapped_system", "mapped_id", and "mapped_label" are filled with NA and are to be manually filled.

Value

(tbl_df, tbl, data.frame) The variable_mapping table.

Examples

```
flat <- ants_L0_flat

# Create inputs to variable_mapping()

observation <- create_observation(
  L0_flat = flat,
```

```
  observation_id = "observation_id",
  event_id = "event_id",
  package_id = "package_id",
  location_id = "location_id",
  datetime = "datetime",
  taxon_id = "taxon_id",
  variable_name = "variable_name",
  value = "value",
  unit = "unit")

observation_ancillary <- create_observation_ancillary(
  L0_flat = flat,
  observation_id = "observation_id",
  variable_name = c("trap.type", "trap.num", "moose.cage"))

location_ancillary <- create_location_ancillary(
  L0_flat = flat,
  location_id = "location_id",
  variable_name = "treatment")

taxon_ancillary <- create_taxon_ancillary(
  L0_flat = flat,
  taxon_id = "taxon_id",
  variable_name = c(
    "subfamily", "hl", "rel", "rll", "colony.size",
    "feeding.preference", "nest.substrate", "primary.habitat",
    "secondary.habitat", "seed.disperser", "slavemaker.sp",
    "behavior", "biogeographic.affinity", "source"),
  unit = c("unit_hl", "unit_rel", "unit_rll"))

# Create variable_mapping table

variable_mapping <- create_variable_mapping(
  observation = observation,
  observation_ancillary = observation_ancillary,
  location_ancillary = location_ancillary,
  taxon_ancillary = taxon_ancillary)

variable_mapping
```

`flatten_data` *Flatten an ecocomDP dataset*

Description

Flatten an ecocomDP dataset

Usage

```
flatten_data(tables)
```

Arguments

tables	(list of tbl_df, tbl, data.frame) A named list of ecocomDP tables.

Details

"flat" format refers to the fully joined source L0 dataset in "wide" form with the exception of the core observation variables, which are in "long" form (i.e. using the variable_name, value, unit columns of the observation table). This "flat" format is the "widest" an L1 ecocomDP dataset can be consistently spread due to the frequent occurrence of L0 source datasets with > 1 core observation variable.

Value

(tbl_df, tbl, data.frame) A single flat table created by joining and spreading all `tables`, except the observation table. See details for more information on this "flat" format.

Note

Warnings/Errors from `flatten_data()` can most often be fixed by addressing any validation issues reported by `read_data()` (e.g. non-unique composite keys).

Ancillary identifiers are dropped from the returned object.

Examples

```
dataset <- ants_L1

flat <- flatten_data(dataset[[1]]$tables)

flat
```

---

plot_taxa_accum_sites *Plot taxa accumulation by site accumulation*

Description

Plot taxa accumulation by site accumulation

Usage

```
plot_taxa_accum_sites(observation, id, alpha = 1)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
id	(character) Identifier of dataset to be used in plot subtitles.
alpha	(numeric) Alpha-transparency scale of data points. Useful when many data points overlap. Allowed values are between 0 and 1, where 1 is 100% opaque. Default is 1.

Value

(gg, ggplot) A gg, ggplot object if assigned to a variable, otherwise a plot to your active graphics device

Examples

```
observation <- ants_L1[[1]]$tables$observation
id <- names(ants_L1)

plot_taxa_accum_sites(observation, id)
```

---

plot_taxa_accum_time *Plot taxa accumulation through time*

Description

Plot taxa accumulation through time

Usage

```
plot_taxa_accum_time(observation, id, alpha = 1)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
id	(character) Identifier of dataset to be used in plot subtitles.
alpha	(numeric) Alpha-transparency scale of data points. Useful when many data points overlap. Allowed values are between 0 and 1, where 1 is 100% opaque. Default is 1.

Value

(gg, ggplot) A gg, ggplot object if assigned to a variable, otherwise a plot to your active graphics device

Examples

```
observation <- ants_L1[[1]]$tables$observation
id <- names(ants_L1)

plot_taxa_accum_time(observation, id)
```

plot_taxa_diversity *Plot diversity (taxa richness) through time*

Description

Plot diversity (taxa richness) through time

Usage

```
plot_taxa_diversity(observation, id, alpha = 1)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
id	(character) Identifier of dataset to be used in plot subtitles.
alpha	(numeric) Alpha-transparency scale of data points. Useful when many data points overlap. Allowed values are between 0 and 1, where 1 is 100% opaque. Default is 1.

Value

(gg, ggplot) A gg, ggplot object if assigned to a variable, otherwise a plot to your active graphics device

Examples

```
observation <- ants_L1[[1]]$tables$observation
id <- names(ants_L1)

plot_taxa_diversity(observation, id)
```

plot_taxa_sample_time *Plot dates and times samples were taken*

Description

Plot dates and times samples were taken

Usage

```
plot_taxa_sample_time(observation, id, alpha = 1)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
id	(character) Identifier of dataset to be used in plot subtitles.
alpha	(numeric) Alpha-transparency scale of data points. Useful when many data points overlap. Allowed values are between 0 and 1, where 1 is 100% opaque. Default is 1.

Value

(gg, ggplot) A gg, ggplot object if assigned to a variable, otherwise a plot to your active graphics device

Examples

```
observation <- ants_L1[[1]]$tables$observation
id <- names(ants_L1)

plot_taxa_sample_time(observation, id)
```

---

plot_taxa_shared_sites

Plot number of unique taxa shared across sites

Description

Plot number of unique taxa shared across sites

Usage

```
plot_taxa_shared_sites(observation, id)
```

Arguments

observation	(tbl_df, tbl, data.frame) The observation table.
id	(character) Identifier of dataset to be used in plot subtitles.

Value

(gg, ggplot) A gg, ggplot object if assigned to a variable, otherwise a plot to your active graphics device

Examples

```
observation <- ants_L1[[1]]$tables$observation
id <- names(ants_L1)

plot_taxa_shared_sites(observation, id)
```

`read_data` *Read an ecocomDP dataset*

Description

Read an ecocomDP dataset

Usage

```
read_data(
  id = NULL,
  parse_datetime = TRUE,
  unique_keys = FALSE,
  site = "all",
  startdate = NA,
  enddate = NA,
  package = "basic",
  check.size = FALSE,
  nCores = 1,
  forceParallel = FALSE,
  token = NA,
  neon.data.save.dir = NULL,
  neon.data.read.path = NULL,
  ...,
  from = NULL
)
```

Arguments

`id`	(character) Identifier of dataset to read. Identifiers are listed in the "id" column of the `search_data()` output. Older versions of datasets can be read, but a warning is issued.
`parse_datetime`	(logical) Parse datetime values if TRUE, otherwise return as character strings.
`unique_keys`	(logical) Whether to create globally unique primary keys (and associated foreign keys). Useful in maintaining referential integrity when working with multiple datasets. If TRUE, `id` is appended to each table's primary key and associated foreign key. Default is FALSE.
`site`	(character) For NEON data, a character vector of site codes to filter data on. Sites are listed in the "sites" column of the `search_data()` output. Defaults to "all", meaning all sites.
`startdate`	(character) For NEON data, the start date to filter on in the form YYYY-MM. Defaults to NA, meaning all available dates.
`enddate`	(character) For NEON data, the end date to filter on in the form YYYY-MM. Defaults to NA, meaning all available dates.
`package`	(character) For NEON data, either 'basic' or 'expanded', indicating which data package to download. Defaults to basic.

`check.size` (logical) For NEON data, should the user approve the total file size before downloading? Defaults to FALSE.

`nCores` (integer) For NEON data, the number of cores to parallelize the stacking procedure. Defaults to 1.

`forceParallel` (logical) For NEON data, if the data volume to be processed does not meet minimum requirements to run in parallel, this overrides. Defaults to FALSE.

`token` (character) For NEON data, a user specific API token (generated within neon.datascience user accounts).

`neon.data.save.dir` (character) For NEON data, an optional and experimental argument (i.e. may not be supported in future releases), indicating the directory where NEON source data should be saved upon download from the NEON API. Data are downloaded using `neonUtilities::loadByProduct()` and saved in this directory as an .rds file. The filename will follow the format <NEON data product ID>_<timestamp>.rds

`neon.data.read.path` (character) For NEON data, an optional and experimental argument (i.e. may not be supported in future releases), defining a path to read in an .rds file of 'stacked NEON data' from `neonUtilities::loadByProduct()`. See details below for more information.

`...` For NEON data, other arguments to `neonUtilities::loadByProduct()`

`from` (character) Full path of file to be read (if .rds), or path to directory containing saved datasets (if .csv).

Details

Validation checks are applied to each dataset ensuring it complies with the ecocomDP model. A warning is issued when any validation checks fail. All datasets are returned, even if they fail validation.

Column classes are coerced to those defined in the ecocomDP specification.

Validation happens each time files are read, from source APIs or local environments.

Details for `read_data()` function regarding NEON data: Using this function to read data with an `id` that begins with "neon.ecocomdp" will result in a query to download NEON data from the NEON Data Portal API using `neonUtilities::loadByProduct()`. If a query includes provisional data (or if you are not sure if the query includes provisional data), we recommend saving a copy of the data in the original format provided by NEON in addition to the derived ecocomDP data package. To do this, provide a directory path using the `neon.data.read.path` argument. For example, the query `my_ecocomdp_data <-read_data(id = "neon.ecocomdp.10022.001.001",neon.data.save.dir = "my_neon_data")` will download the data for NEON Data Product ID DP1.10022.001 (ground beetles in pitfall traps) and convert it to the ecocomDP data model. In doing so, a copy of the original NEON download will be saved in the directory "my_ neon_data with the filename "DP1.10022.001_<timestamp>.RDS" and the derived data package in the ecocomDP format will be stored in your R environment in an object named "my_ecocomdp_data". Further, if you wish to reload a previously downloaded NEON dataset into the ecocomDP format, you can do so using `my_ecocomdp_data <-read_data(id = "neon.ecocomdp.10022.001.001",neon.data.read.path = "my_neon_data/DP1.10022.001_<timestamp>.RDS")`

Provisional NEON data. Despite NEON's controlled data entry, at times, errors are found in published data; for example, an analytical lab may adjust its calibration curve and re-calculate past analyses, or field scientists may discover a past misidentification. In these cases, Level 0 data are edited

and the data are re-processed to Level 1 and re-published. Published data files include a time stamp in the file name; a new time stamp indicates data have been re-published and may contain differences from previously published data. Data are subject to re-processing at any time during an initial provisional period; data releases are never re-processed. All records downloaded from the NEON API will have a "release" field. For any provisional record, the value of this field will be "PROVISIONAL", otherwise, this field will have a value indicating the version of the release to which the record belongs. More details can be found at https://www.neonscience.org/data-samples/data-management/data-revisions-releases.

Value

(list) with the structure:

- id - Dataset identifier
  - metadata - List of info about the dataset. NOTE: This object is underdevelopment and content may change in future releases.
  - tables - List of dataset tables as data.frames.
  - validation_issues - List of validation issues. If the dataset fails any validation checks, then descriptions of each issue are listed here.

Note

This function may not work between 01:00 - 03:00 UTC on Wednesdays due to regular maintenance of the EDI Data Repository.

Examples

```
## Not run:
# Read from EDI
dataset <- read_data("edi.193.5")
# str(dataset)

## End(Not run)

## Not run:
# Read from NEON (full dataset)
dataset <- read_data("neon.ecocomdp.20120.001.001")

## End(Not run)

## Not run:
# Read from NEON with filters (partial dataset)
dataset <- read_data(
  id = "neon.ecocomdp.20120.001.001",
  site = c("COMO", "LECO", "SUGG"),
  startdate = "2017-06",
  enddate = "2019-09",
  check.size = FALSE)

## End(Not run)
```

```
## Not run:
# Read with datetimes as character
dataset <- read_data("edi.193.5", parse_datetime = FALSE)
is.character(dataset$edi.193.5$tables$observation$datetime)

## End(Not run)

# Save a list of datasets for reading
datasets <- c(ants_L1, ants_L1, ants_L1)  # 3 of the same, w/different names
names(datasets) <- c("ds1", "ds2", "ds3")
mypath <- paste0(tempdir(), "/datasets")      # A place for saving
dir.create(mypath)
save_data(datasets, mypath)               # Save as .rds
save_data(datasets, mypath, type = ".csv")# Save as .csv

# Read from local .rds
datasets <- read_data(from = paste0(mypath, "/datasets.rds"))

# Read from local .csv
datasets <- read_data(from = mypath)

# A dataset is returned as a list of metadata, tables, and validation issues
# (if there are any). The dataset ID is assigned to the top level for
# reference.
str(datasets[1])

# Clean up
unlink(mypath, recursive = TRUE)
```

`save_data` *Save an ecocomDP dataset*

Description

Save an ecocomDP dataset

Usage

```
save_data(dataset, path, type = ".rds", name = NULL)
```

Arguments

`dataset`	(list) One or more datasets of the structure returned by `read_data()`. Name of the `dataset` object will become the file name if `name` is not used.
`path`	(character) Path to the directory in which `dataset` will be written.
`type`	(character) Type of file to save the `dataset` as. Default is ".rds" but can also be ".csv". Note: metadata and validation_issues are lost when using ".csv".
`name`	(character) An optional argument for setting the saved file name (for .rds) if you'd like it to be different than `dataset`'s object name.

Value

`.rds`	If `type = ".rds"`, then an .rds representation of `dataset` is returned.
`.csv`	If `type = ".csv"`, then an set of .csv files are written to a sub-directory of `path` named after the data package/product ID.

Note

Subsequent calls won't overwrite files or directories

Examples

```
# Create a list of datasets
datasets <- c(ants_L1, ants_L1, ants_L1)  # 3 of the same, with different names
names(datasets) <- c("ds1", "ds2", "ds3")

# Create directory for the data
mypath <- paste0(tempdir(), "/data")
dir.create(mypath)

# Save as .rds
save_data(datasets, mypath)
dir(mypath)

# Save as .rds with the name "mydata"
save_data(datasets, mypath, name = "mydata")
dir(mypath)

# Save as .csv
save_data(datasets, mypath, type = ".csv")
dir(mypath)

# Clean up
unlink(mypath, recursive = TRUE)
```

---

search_data *Search across all ecocomDP datasets*

Description

Search across all ecocomDP datasets

Usage

```
search_data(text, taxa, num_taxa, num_years, sd_years, area, boolean = "AND")
```

Arguments

text	(character) Text to search for in dataset titles, descriptions, and abstracts. Datasets matching any exact words or phrase will be returned. Can be a regular expression as used by `stringr::str_detect()`. Is not case sensitive. Works with `boolean`.
taxa	(character) Taxonomic rank values to search on. The full taxonomic hierarchy of each taxa in a dataset is searchable for EDI (including common names) but not yet NEON, in which cases the lowest level rank value is searchable.
num_taxa	(numeric) Minimum and maximum number of taxa the dataset should contain. Any datasets within this range will be returned.
num_years	(numeric) Minimum and maximum number of years sampled the dataset should contain. Any datasets within this range will be returned.
sd_years	(numeric) Minimum and maximum standard deviation between survey dates (in years). Any datasets within this range will be returned.
area	(numeric) Bounding coordinates within which the data should originate. Accepted values are in decimal degrees and in the order: North, East, South, West. Any datasets with overlapping areas or contained points will be returned.
boolean	(character) Boolean operator to use when searching `text` and `taxa`. Supported operators are: "AND", "OR". Default is "AND".

Details

Currently, to accommodate multiple L1 versions of NEON data products, search results for a NEON L0 will also list all the L1 versions available for the match. This method is based on the assumption that the summary data among L1 versions is the same, which may need to be addressed in the future. A list of L0 and corresponding L1 identifiers are listed in /inst/L1_versions.txt. Each L1 version is accompanied by qualifying text that's appended to the title, abstract, and descriptions for comprehension of the differences among L1 versions.

Value

(tbl_df, tbl, data.frame) Search results with these feilds:

- source - Source from which the dataset originates. Currently supported are "EDI" and "NEON".
- id - Identifier of the dataset.
- title - Title of the dataset.
- description - Description of dataset. Only returned for NEON datasets.
- abstract - Abstract of dataset.
- years - Number of years sampled.
- sampling_interval - Standard deviation between sampling events in years.
- sites - Sites names or abbreviations. Only returned for NEON datasets.
- url - URL to dataset.
- source_id - Identifier of source L0 dataset.
- source_id_url - URL to source L0 dataset.

Note

This function may not work between 01:00 - 03:00 UTC on Wednesdays due to regular maintenance of the EDI Data Repository.

Examples

```
## Not run:
# Empty search returns all available datasets
search_data()

# "text" searches titles, descriptions, and abstracts
search_data(text = "Lake")

# "taxa" searches taxonomic ranks for a match
search_data(taxa = "Plantae")

# "num_years" searches the number of years sampled
search_data(num_years = c(10, 20))

# Use any combination of search fields to find the data you're looking for
search_data(
  text = c("Lake", "River"),
  taxa = c("Plantae", "Animalia"),
  num_taxa = c(0, 10),
  num_years = c(10, 100),
  sd_years = c(.01, 100),
  area = c(47.1, -86.7, 42.5, -92),
  boolean = "OR")

## End(Not run)
```

`validate_data` *Validate a dataset against the ecocomDP model*

Description

Validate a dataset against the ecocomDP model

Usage

```
validate_data(dataset = NULL, path = NULL)
```

Arguments

`dataset`	(list) A dataset of the structure returned by `read_data()`.
`path`	(character) Path to a directory containing ecocomDP tables as files.

Details

Validation checks:

- File names - File names are the ecocomDP table names.
- Table presence - Required tables are present.
- Column names - Column names of all tables match the model.
- Column presence - Required columns are present.
- Column classes - Column classes match the model specification.
- Datetime format - Date and time formats follow the model specification.
- Primary keys - Primary keys of tables are unique.
- Composite keys - Composite keys (unique constraints) of each table are unique.
- Referential integrity - Foreign keys have a corresponding primary key.
- Coordinate format - Values are in decimal degree format.
- Coordinate range - Values are within -90 to 90 and -180 to 180.
- Elevation - Values are less than Mount Everest (8848 m) and greater than Mariana Trench (-10984 m).
- Variable mapping - variable_name is in table_name.

Value

(list) If any checks fail, then a list of validation issues are returned along with a warning. If no issues are found then NULL is returned.

Note

This function is used by ecocomDP creators (to ensure what has been created is valid), maintainers (to improve the quality of archived ecocomDP datasets), and users (to ensure the data being used is free of error).

Examples

```
# Write a set of ecocomDP tables to file for validation
mydir <- paste0(tempdir(), "/dataset")
dir.create(mydir)
write_tables(
  path = mydir,
  observation = ants_L1$edi.193.5$tables$observation,
  observation_ancillary = ants_L1$edi.193.5$tables$observation_ancillary,
  location = ants_L1$edi.193.5$tables$location,
  location_ancillary = ants_L1$edi.193.5$tables$location_ancillary,
  taxon = ants_L1$edi.193.5$tables$taxon,
  taxon_ancillary = ants_L1$edi.193.5$tables$taxon_ancillary,
  dataset_summary = ants_L1$edi.193.5$tables$dataset_summary,
  variable_mapping = ants_L1$edi.193.5$tables$variable_mapping)

# Validate
validate_data(path = mydir)
```

```
# Clean up
unlink(mydir, recursive = TRUE)
```

`view_descriptions` *View descriptions and requirements of ecocomDP tables*

Description

View descriptions and requirements of ecocomDP tables

Usage

```
view_descriptions()
```

Value

(NULL) Opens a webpage, in your default browser, with a list of descriptions and requirements of the ecocomDP tables

Examples

```
## Not run:
view_descriptions()

## End(Not run)
```

`view_diagram` *View diagram of ecocomDP tables and relationships*

Description

View diagram of ecocomDP tables and relationships

Usage

```
view_diagram()
```

Value

(NULL) Opens a webpage, in your default browser, with a diagram keys and linkages among ecocomDP tables

Examples

```
## Not run:
view_diagram()

## End(Not run)
```

`write_tables` *Write ecocomDP tables to file*

Description

Write ecocomDP tables to file

Usage

```
write_tables(
  path,
  sep = ",",
  observation = NULL,
  location = NULL,
  taxon = NULL,
  dataset_summary = NULL,
  observation_ancillary = NULL,
  location_ancillary = NULL,
  taxon_ancillary = NULL,
  variable_mapping = NULL
)
```

Arguments

`path`	(character) A path to the directory in which the files will be written.
`sep`	(character) Field delimiter to use when writing files. Default is comma.
`observation`	(tbl_df, tbl, data.frame) The observation table.
`location`	(tbl_df, tbl, data.frame) The location table.
`taxon`	(tbl_df, tbl, data.frame) The taxon table.
`dataset_summary`	(tbl_df, tbl, data.frame) The dataset_summary table.
`observation_ancillary`	(tbl_df, tbl, data.frame) The observation_ancillary table.
`location_ancillary`	(tbl_df, tbl, data.frame) The location_ancillary table.
`taxon_ancillary`	(tbl_df, tbl, data.frame) The taxon_ancillary table.
`variable_mapping`	(tbl_df, tbl, data.frame) The variable_mapping table.

Value

ecocomDP tables as sep delimited files

Examples

```
# Create directory for the tables
mypath <- paste0(tempdir(), "/data")
dir.create(mypath)

# Create a couple inputs to write_tables()

flat <- ants_L0_flat

observation <- create_observation(
  L0_flat = flat,
  observation_id = "observation_id",
  event_id = "event_id",
  package_id = "package_id",
  location_id = "location_id",
  datetime = "datetime",
  taxon_id = "taxon_id",
  variable_name = "variable_name",
  value = "value",
  unit = "unit")

observation_ancillary <- create_observation_ancillary(
  L0_flat = flat,
  observation_id = "observation_id",
  variable_name = c("trap.type", "trap.num", "moose.cage"))

# Write tables to file

write_tables(
  path = mypath,
  observation = observation,
  observation_ancillary = observation_ancillary)

dir(mypath)

# Clean up
unlink(mypath, recursive = TRUE)
```

Index