# Package ‘ecocomDP’

May 8, 2026

**Type** Package

**Title** Tools to Create, Use, and Convert ecocomDP Data

**Description** Work with the Ecological Community Data Design Pattern. 'ecocomDP' is a flexible data model for harmonizing ecological community surveys, in a research question agnostic format, from source data published across repositories, and with methods that keep the derived data up-to-date as the underlying sources change. Described in O'Brien et al. (2021), <doi:10.1016/j.ecoinf.2021.101374>.

**Version** 1.3.2

**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, data.table, dplyr, EML (>= 2.0.5), emld (>= 0.5.1), geosphere, ggplot2, httr, lubridate, magrittr, methods, neonUtilities (>= 2.1.1), rlang, rmarkdown, stats, stringr, tidyr, tools, utils, uuid, xml2, neonOS

**Suggests** knitr, mime, reader, ritis, taxize, testthat, worrms, ggrepel, usmap (>= 0.6.1), sf (>= 1.0.9), maps

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**Language** en-US

**NeedsCompilation** no

**Author** Colin Smith [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2261-9931>),
Eric Sokol [aut] (ORCID: <https://orcid.org/0000-0001-5923-0917>),
Margaret O'Brien [aut] (ORCID: <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],
Rafael Rangel [ctb] (ORCID: <https://orcid.org/0009-0000-6265-3064>),
Natalie Robinson [ctb],
Thilina Surasinghe [ctb],
Kyle Zollo-Venecek [ctb] (ORCID:
<https://orcid.org/0000-0002-1615-590X>)

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

**Repository** CRAN

**Date/Publication** 2024-08-16 21:40:02 UTC

# Contents

annotation_dictionary . . . 3
ants_L0_flat . . . 3
ants_L1 . . . 5
calc_geo_extent_bounding_box_m2 . . . 6
calc_length_of_survey_years . . . 6
calc_number_of_years_sampled . . . 7
calc_std_dev_interval_betw_years . . . 7
convert_to_dwca . . . 8
create_dataset_summary . . . 9
create_eml . . . 11
create_location . . . 14
create_location_ancillary . . . 15
create_observation . . . 16
create_observation_ancillary . . . 18
create_taxon . . . 19
create_taxon_ancillary . . . 21
create_variable_mapping . . . 22
flatten_data . . . 24
plot_sample_space_time . . . 25
plot_sites . . . 26
plot_taxa_abund . . . 28
plot_taxa_accum_sites . . . 30
plot_taxa_accum_time . . . 31
plot_taxa_diversity . . . 33

plot_taxa_occur_freq . . . . . 34
plot_taxa_rank . . . . . 36
plot_taxa_shared_sites . . . . . 37
read_data . . . . . 39
save_data . . . . . 42
search_data . . . . . 43
validate_data . . . . . 45
write_tables . . . . . 47

**Index** **49**

---

`annotation_dictionary` *Annotations of published data*

---

## Description

View the collection of dataset- and attribute-level annotations from existing ecocomDP datasets.

## Usage

```
annotation_dictionary()
```

## Details

Use the search field to find the annotation terms and URIs.

## Examples

```
## Not run:
View(annotation_dictionary())

## End(Not run)
```

---

`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 (L1) 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:

**id** 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 a dataset to the Darwin Core Archive format*

## Description

Convert a dataset to the Darwin Core Archive format

## 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*

---

## Description

Create EML metadata

## 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:<br>• givenName<br>• surName<br>• organizationName<br>• 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$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 = 'csmith@mail.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 a dataset*

---

## Description

Flatten a dataset

## Usage

```
flatten_data(data)
```

## Arguments

| | |
|---|---|
| `data` | (list) The dataset object returned by `read_data()`, or a named list of ecocoomDP tables. |

## Details

The "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

```
# Flatten a dataset object
flat <- flatten_data(ants_L1)
flat

# Flatten a list of tables
tables <- ants_L1$tables
flat <- flatten_data(tables)
flat
```

---

`plot_sample_space_time` *Plot dates and times samples were collected or observations were made*

---

## Description

Plot dates and times samples were collected or observations were made

## Usage

```
plot_sample_space_time(
  data,
  id = NA_character_,
  alpha = 1,
  color_var = "package_id",
  shape_var = "package_id",
  observation = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation table, or a flat table containing columns of the observation table. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `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. |
| `color_var` | (character) Name of column to use to assign colors to the points on the plot |
| `shape_var` | (character) Name of column to use to assign shapes to the points on the plot |
| `observation` | (tbl_df, tbl, data.frame) DEPRECATED: Use `data` instead. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 9 columns of the observation table.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_sample_space_time(dataset)

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  dplyr::filter(as.numeric(location_id) > 4) %>%
  plot_sample_space_time()

## End(Not run)

# Plot the example dataset
plot_sample_space_time(ants_L1)
```

---

## `plot_sites` *Plot sites on US map*

---

### Description

Plot sites on US map

### Usage

```
plot_sites(
  data,
  id = NA_character_,
  alpha = 1,
  labels = TRUE,
  color_var = "package_id",
  shape_var = "package_id"
)
```

### Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation and taxon tables, or a flat table containing columns of the observation and location tables. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `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. |
| `labels` | (logical) Argument to show labels of each US state. Default is TRUE. |
| `color_var` | (character) Name of column to use to assign colors to the points on the plot |
| `shape_var` | (character) Name of column to use to assign shapes to the points on the plot |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 14 columns of the combined observation and location tables.

### Value

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

### Examples

```
## Not run:
library(dplyr)
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_sites(dataset)

# Flatten dataset then plot
dataset %>%
 flatten_data() %>%
 plot_sites()

# Download a NEON dataset
dataset2 <- read_data(
 id = "neon.ecocomdp.20120.001.001",
 site= c('COMO','LECO'),
 startdate = "2017-06",
 enddate = "2021-03",
 token = Sys.getenv("NEON_TOKEN"), # option to use a NEON token
 check.size = FALSE)

# Combine the two datasets and plot. This requires the datasets be first
```

```
# flattened and then stacked.
flattened_data1 <- dataset %>% flatten_data()
flattened_data2 <- dataset2 %>% flatten_data()
stacked_data <- bind_rows(flattened_data1,flattened_data2)
plot_sites(stacked_data)

## End(Not run)

# Plot the example dataset
plot_sites(ants_L1)
```

---

`plot_taxa_abund` *Plot mean taxa abundances per 'observation_id'*

---

## Description

Plot taxon abundances averaged across observation records for each taxon. Abundances are reported using the units provided in the dataset. In some cases, these counts are not standardized to sampling effort.

## Usage

```
plot_taxa_abund(
  data,
  id = NA_character_,
  min_relative_abundance = 0,
  trans = "identity",
  facet_var = NA_character_,
  color_var = NA_character_,
  facet_scales = "free",
  alpha = 1
)
```

## Arguments

`data` (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation and taxon tables, or a flat table containing columns of the observation and taxon tables.

`id` (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column.

`min_relative_abundance` (numeric) Minimum relative abundance allowed for taxa included in the plot; a value between 0 and 1, inclusive.

| | |
|---|---|
| trans | (character) Define the transform applied to the response variable; "identity" is default, "log1p" is x+1 transform. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time". |
| facet_var | (character) Name of column to use for faceting. Must be a column of the observation or taxon table. |
| color_var | (character) Name of column to use for plot colors. |
| facet_scales | (character) Should scales be free ("free", default value), fixed ("fixed"), or free in one dimension ("free_x", "free_y")? |
| 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. |

## Details

The data parameter accepts a range of input types but ultimately requires the 13 columns of the combined observation and taxon tables.

## Value

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

## Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# plot ecocomDP formatted dataset
plot_taxa_abund(dataset)

# plot flattened ecocomDP dataset, log(x+1) transform abundances
plot_taxa_abund(
  data = flatten_data(dataset),
  trans = "log1p")

# facet by location color by taxon_rank, log 10 transform
plot_taxa_abund(
  data = dataset,
  facet_var = "location_id",
  color_var = "taxon_rank",
  trans = "log10")

# facet by location, minimum rel. abund = 0.05, log 10 transform
plot_taxa_abund(
  data = dataset,
  facet_var = "location_id",
  min_relative_abundance = 0.05,
```

```
  trans = "log1p")

# color by location, log 10 transform
plot_taxa_abund(
  data = dataset,
  color_var = "location_id",
  trans = "log10")

# tidy syntax, flatten then filter data by date
dataset %>%
  flatten_data() %>%
  dplyr::filter(
    lubridate::as_date(datetime) > "2003-07-01") %>%
  plot_taxa_abund(
    trans = "log1p",
    min_relative_abundance = 0.01)

## End(Not run)

# Plot the example dataset
plot_taxa_abund(ants_L1)
```

---

`plot_taxa_accum_sites` *Plot taxa accumulation by site accumulation*

---

## Description

Plot taxa accumulation by site accumulation

## Usage

```
plot_taxa_accum_sites(data, id = NA_character_, alpha = 1, observation = NULL)
```

## Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation table, or a flat table containing columns of the observation table. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `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. |
| `observation` | (tbl_df, tbl, data.frame) DEPRECATED: Use `data` instead. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 9 columns of the observation table.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_taxa_accum_sites(dataset)

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  plot_taxa_accum_sites()

# Plot from the observation table directly
plot_taxa_accum_sites(dataset$tables$observation)

## End(Not run)

# Plot the example dataset
plot_taxa_accum_sites(ants_L1)
```

---

## `plot_taxa_accum_time` *Plot taxa accumulation through time*

---

### Description

Plot taxa accumulation through time

### Usage

```
plot_taxa_accum_time(data, id = NA_character_, alpha = 1, observation = NULL)
```

### Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation table, or a flat table containing columns of the observation table. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `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. |
| `observation` | (tbl_df, tbl, data.frame) DEPRECATED: Use `data` instead. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 9 columns of the observation table.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_taxa_accum_time(dataset)

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  plot_taxa_accum_time()

# Plot from the observation table directly
plot_taxa_accum_time(dataset$tables$observation)

## End(Not run)

# Plot the example dataset
plot_taxa_accum_time(ants_L1)
```

## plot_taxa_diversity *Plot diversity (taxa richness) through time*

### Description

Plot diversity (taxa richness) through time

### Usage

```
plot_taxa_diversity(
  data,
  id = NA_character_,
  time_window_size = "day",
  observation = NULL,
  alpha = 1
)
```

### Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation table, or a flat table containing columns of the observation table. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `time_window_size` | (character) Define the time window over which to aggregate observations for calculating richness. Can be: "day" or "year" |
| `observation` | (tbl_df, tbl, data.frame) DEPRECATED: Use `data` instead. |
| `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. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 9 columns of the observation table.

### Value

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

## Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_taxa_diversity(dataset)

# Plot the dataset with observations aggregated by year
plot_taxa_diversity(dataset, time_window_size = "year")

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(
    lubridate::as_date(datetime) > "2007-01-01") %>%
  plot_taxa_diversity()

# Plot from the observation table directly
plot_taxa_diversity(dataset$tables$observation)

## End(Not run)

# Plot the example dataset
plot_taxa_diversity(ants_L1)
```

---

# plot_taxa_occur_freq *Plot taxon occurrence frequencies*

---

## Description

Plot taxon occurrence frequences as the number of 'event_id' by 'location_id' combinations in which a taxon is observed.

## Usage

```
plot_taxa_occur_freq(
  data,
  id = NA_character_,
  min_occurrence = 0,
  facet_var = NA_character_,
  color_var = NA_character_,
  facet_scales = "free",
  alpha = 1
)
```

### Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation and taxon tables, or a flat table containing columns of the observation and taxon tables. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `min_occurrence` | (numeric) Minimum number of occurrences allowed for taxa included in the plot. |
| `facet_var` | (character) Name of column to use for faceting. Must be a column of the observation or taxon table. |
| `color_var` | (character) Name of column to use for plot colors. |
| `facet_scales` | (character) Should scales be free ("free", default value), fixed ("fixed"), or free in one dimension ("free_x", "free_y")? |
| `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. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 13 columns of the combined observation and taxon tables.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_taxa_occur_freq(dataset)

# Facet by location and color by taxon_rank
plot_taxa_occur_freq(
  data = dataset,
  facet_var = "location_id",
  color_var = "taxon_rank")

# Color by location and only include taxa with >= 5 occurrences
plot_taxa_occur_freq(
  data = dataset,
  color_var = "location_id",
  min_occurrence = 5)
```

```
# Flatten, filter using a time cutoff, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  plot_taxa_occur_freq()

## End(Not run)
# Plot the example dataset
plot_taxa_occur_freq(ants_L1)
```

---

`plot_taxa_rank` *Plot taxa ranks*

---

## Description

Plot the number of observations that use each taxonomic rank in the dataset.

## Usage

```
plot_taxa_rank(
  data,
  id = NA_character_,
  facet_var = NA_character_,
  facet_scales = "free_x",
  alpha = 1
)
```

## Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation and taxon tables, or a flat table containing columns of the observation and taxon tables. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `facet_var` | (character) Name of column to use for faceting. Must be a column of the observation or taxon table. |
| `facet_scales` | (character) Should scales be free ("free", default value), fixed ("fixed"), or free in one dimension ("free_x", "free_y")? |
| `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. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 13 columns of the combined observation and taxon tables.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data(
  id = "neon.ecocomdp.20120.001.001",
  site= c('COMO','LECO'),
  startdate = "2017-06",
  enddate = "2019-09",
  check.size = FALSE)

# Plot the dataset
plot_taxa_rank(dataset)

# Plot with facet by location
plot_taxa_rank(dataset, facet_var = "location_id")

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  dplyr::filter(grepl("COMO",location_id)) %>%
  plot_taxa_rank()

## End(Not run)

# Plot the example dataset
plot_taxa_rank(ants_L1)
```

---

## 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(data, id = NA_character_, observation = NULL)
```

### Arguments

| | |
|---|---|
| `data` | (list or tbl_df, tbl, data.frame) The dataset object returned by `read_data()`, a named list of tables containing the observation table, or a flat table containing columns of the observation table. |
| `id` | (character) Identifier of dataset to be used in plot subtitles. Is automatically assigned when `data` is a dataset object containing the `id` field, or is a table containing the package_id column. |
| `observation` | (tbl_df, tbl, data.frame) DEPRECATED: Use `data` instead. |

### Details

The `data` parameter accepts a range of input types but ultimately requires the 9 columns of the observation table.

### Value

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

### Examples

```
## Not run:
# Read a dataset of interest
dataset <- read_data("edi.193.5")

# Plot the dataset
plot_taxa_shared_sites(dataset)

# Flatten the dataset, manipulate, then plot
dataset %>%
  flatten_data() %>%
  dplyr::filter(lubridate::as_date(datetime) > "2003-07-01") %>%
  dplyr::filter(as.numeric(location_id) > 4) %>%
  plot_taxa_shared_sites()

# Plot from the observation table directly
plot_taxa_shared_sites(dataset$tables$observation)

## End(Not run)

# Plot the example dataset
plot_taxa_shared_sites(ants_L1)
```

read_data *Read published data*

## Description

Read published data

## 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,
  format = "new"
)
```

## 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).

`format` (character) Format of returned object, which can be: "new" (the new implementation) or "old" (the original implementation; deprecated). In the new format, the top most level of nesting containing the "id" field has been moved to the same level as the "tables", "metadata", and "validation_issues" fields.

## 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) A dataset 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, max.level = 2)

# Read from NEON (full dataset)
dataset <- read_data("neon.ecocomdp.20120.001.001")

# 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)

# Read with datetimes as character
dataset <- read_data("edi.193.5", parse_datetime = FALSE)
is.character(dataset$tables$observation$datetime)

# Read from saved .rds
save_data(dataset, tempdir())
dataset <- read_data(from = paste0(tempdir(), "/dataset.rds"))

# Read from saved .csv
save_data(dataset, tempdir(), type = ".csv")# Save as .csv
dataset <- read_data(from = tempdir())

## End(Not run)
```

---

`save_data` *Save a dataset*

---

## Description

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

# Save as .rds
save_data(ants_L1, mypath)
dir(mypath)

# Save as .rds with the name "mydata"
save_data(ants_L1, mypath, name = "mydata")
dir(mypath)

# Save as .csv
save_data(ants_L1, mypath, type = ".csv")
dir(mypath)

## Not run:
# Save multiple datasets
ids <- c("edi.193.5", "edi.303.2", "edi.290.2")
datasets <- lapply(ids, read_data)
save_data(datasets, mypath)
dir(mypath)

## End(Not run)

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

---

`search_data` *Search published data*

---

## Description

Search published data

## 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 names to search on. To effectively search the taxonomic tree, it is advisable to start with specific taxonomic names and then gradually broaden the search to higher rank levels when needed. For instance, if searching for "Astragalus gracilis" (species) doesn't produce any results, try expanding the search to "Astragalus" (Genus), "Fabaceae" (Family), and so on. This approach accounts for variations in organism identification, ensuring a more comprehensive exploration of the taxonomic hierarchy.

`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". Note, other parameters used in a search are combined with an implicit "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 tables against the model*

---

## Description

Validate tables against the 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.
- Mapped_id - values in mapped_id are valid URIs

## 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

```
## Not run:
# Write a set of ecocomDP tables to file for validation
mydir <- paste0(tempdir(), "/dataset")
dir.create(mydir)
write_tables(
  path = mydir,
  observation = ants_L1$tables$observation,
  observation_ancillary = ants_L1$tables$observation_ancillary,
  location = ants_L1$tables$location,
  location_ancillary = ants_L1$tables$location_ancillary,
  taxon = ants_L1$tables$taxon,
  taxon_ancillary = ants_L1$tables$taxon_ancillary,
  dataset_summary = ants_L1$tables$dataset_summary,
  variable_mapping = ants_L1$tables$variable_mapping)
```

```
# Validate
validate_data(path = mydir)

# Clean up
unlink(mydir, recursive = TRUE)

## End(Not run)
```

---

`write_tables` *Write tables to file*

---

## Description

Write 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


∗ **datasets**
    ants_L0_flat, 3
    ants_L1, 5

annotation_dictionary, 3
ants_L0_flat, 3
ants_L1, 5

calc_geo_extent_bounding_box_m2, 6
calc_length_of_survey_years, 6
calc_number_of_years_sampled, 7
calc_std_dev_interval_betw_years, 7
convert_to_dwca, 8
create_dataset_summary, 9
create_eml, 11
create_location, 14
create_location_ancillary, 15
create_observation, 16
create_observation_ancillary, 18
create_taxon, 19
create_taxon_ancillary, 21
create_variable_mapping, 22

flatten_data, 24

plot_sample_space_time, 25
plot_sites, 26
plot_taxa_abund, 28
plot_taxa_accum_sites, 30
plot_taxa_accum_time, 31
plot_taxa_diversity, 33
plot_taxa_occur_freq, 34
plot_taxa_rank, 36
plot_taxa_shared_sites, 37

read_data, 39

save_data, 42
search_data, 43

validate_data, 45

write_tables, 47