Data Retriever

Release v2.2.0

Nov 05, 2018

Contents

Contents:

CHAPTER 1

User Guide

1.1 We handle the data so you can focus on the science

Finding data is one thing. Getting it ready for analysis is another. Acquiring, cleaning, standardizing and importing publicly available data is time consuming because many datasets lack machine readable metadata and do not conform to established data structures and formats.

The Data Retriever automates the first steps in the data analysis pipeline by downloading, cleaning, and standardizing datasets, and importing them into relational databases, flat files, or programming languages. The automation of this process reduces the time for a user to get most large datasets up and running by hours, and in some cases days.

1.2 What data tasks does the Retriever handle

The Data Retriever handles a number of common tasks including:

1. Creating the underlying database structures, including automatically determining the data types
2. Downloading the data
3. Transforming data into appropriately normalized forms for database management systems (e.g., "wide" data into "long" data and splitting tables into proper sub-tables to reduce duplication)
4. Converting heterogeneous null values (e.g., 999.0, -999, NaN) into standard null values
5. Combining multiple data files into single tables; and 6) placing all related tables in a single database or schema.

A couple of examples on the more complicated end include the Breeding Bird Survey of North America (breed-bird-survey) and the Alwyn Gentry Tree Transect data(gentry-forest-transects):

- Breeding bird survey data consists of multiple tables. The main table is divided into one file per region in 70 individual compressed files. Supplemental tables required to work with the data are posted in a variety of locations and formats. The Data Retriever automates: downloading all data files, extracting data from region-specific raw data files into single tables, correcting typographic errors, replacing non-standard null values, and adding a Species table that links numeric identifiers to actual species names.

- The Gentry forest transects data is stored in over 200 Excel spreadsheets, each representing an individual study site, and compressed in a zip archive. Each spreadsheet contains counts of individuals found at a given site and all stems measured from that individual; each stem measurement is placed in a separate column, resulting in variable numbers of columns across rows, a format that is difficult to work with in both database and analysis software. There is no information on the site in the data files themselves, it is only present in the names of the files. The Retriever downloads the archive, extracts the files, and splits the data they contain into four tables: Sites, Species, Stems, and Counts, keeping track of which file each row of count data originated from in the Counts table and placing a single stem on each row in the Stems table.

Adapted from Morris & White 2013.

1.3 Installing (binaries)

Precompiled binaries of the most recent release are available for Windows, OS X, and Ubuntu/Debian at the project website.

1.4 Installing From Source

Required packages

To install the Data Retriever from source, you'll need Python 2.6+ or Python 3.3+ with the following packages installed:

- xlrd

The following packages are optional

- PyMySQL (for MySQL)
- sqlite3 (for SQLite, v3.8 or higher required)
- psycopg2 (for PostgreSQL)
- pypyodbc (for MS Access)

Steps to install from source

1. Clone the repository
2. From the directory containing setup.py, run the following command: `python setup.py install` or use pip `pip install . --upgrade` to install and `pip uninstall retriever` to uninstall the retriever
3. After installing, type `retriever` from a command prompt to see the available options of the Data Retriever. Use `retriever --version` to confirm the version installed on your system.

1.5 Using the Data Retriever Commands

After installing, run `retriever update` to download all of the available dataset scripts. Run `retriever ls` to see the available datasets

To see the full list of command line options and datasets run `retriever --help`. The output will look like this:

```
usage: retriever [-h] [-v] [-q]
                 {download,install,defaults,update,new,new_json,edit_json,delete_json,
↪ls,citation,reset,help}
                 ...

positional arguments:
  {download,install,defaults,update,new,new_json,edit_json,delete_json,ls,citation,
↪reset,help}
                        sub-command help
    download            download raw data files for a dataset
    install             download and install dataset
    defaults            displays default options
    update              download updated versions of scripts
    new                 create a new sample retriever script
    new_json            CLI to create retriever datapackage.json script
    edit_json           CLI to edit retriever datapackage.json script
    delete_json         CLI to remove retriever datapackage.json script
    ls                  display a list all available dataset scripts
    citation            view citation
    reset               reset retriever: removes configation settings,
                        scripts, and cached data
    help

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -q, --quiet           suppress command-line output
```

To install datasets, use the `install` command.

1.6 Examples

Using install

The install command downloads the datasets and installs them in the desired engine.

$ `retriever install -h` (gives install options)

```
usage: retriever install [-h] [--compile] [--debug]
                         {mysql,postgres,sqlite,msaccess,csv,json,xml} ...
positional arguments:
  {mysql,postgres,sqlite,msaccess,csv,json,xml}
                        engine-specific help
    mysql               MySQL
    postgres            PostgreSQL
    sqlite              SQLite
    msaccess            Microsoft Access
    csv                 CSV
    json                JSON
    xml                 XML
optional arguments:
  -h, --help            show this help message and exit
  --compile             force re-compile of script before downloading
  --debug               run in debug mode
```

Examples using install

These examples use Breeding Bird Survey data (breed-bird-survey). The retriever has support for various databases and flat file formats (mysql, postgres, sqlite, msaccess, csv, json, xml). All the engines have a variety of options or flags. Run `retriever defaults` to see the defaults. For example, the default options for mysql and postgres engines are given below.

```
retriever defaults

Default options for engine  MySQL
user   root
password
host   localhost
port   3306
database_name   {db}
table_name   {db}.{table}

Default options for engine  PostgreSQL
user   postgres
password
host   localhost
port   5432
database   postgres
database_name   {db}
table_name   {db}.{table}
```

Help information for a particular engine can be obtained by running retriever install [engine name] [-h] [–help], for example, `retriever install mysql -h`. Both mysql and postgres require the database user name `--user [USER], -u [USER]` and password `--password [PASSWORD], -p [PASSWORD]`. MySQL and PostgreSQL database management systems support the use of configuration files. The configuration files provide a mechanism to support using the engines without providing authentication directly. To set up the configuration files please refer to the respective database management systems documentation.

Install data into Mysql:

```
retriever install mysql --user myusername --password ***** --host localhost --port␣
↪8888 --database_name testdbase breed-bird-survey
retriever install mysql --user myusername breed-bird-survey (using attributes in the␣
↪client authentication configuration file)
```

Install data into postgres:

```
retriever install postgres --user myusername --password ***** --host localhost --port␣
↪5432 --database_name testdbase breed-bird-survey
retriever install postgres breed-bird-survey (using attributes in the client␣
↪authentication configuration file)
```

Install data into sqlite:

```
retriever install sqlite breed-bird-survey -f mydatabase.db (will use mydatabase.db)
retriever install sqlite breed-bird-survey (will use or create default sqlite.db in␣
↪working directory)
```

Install data into csv:

```
retriever install csv breed-bird-survey --table_name  "BBS_{table}.csv"
retriever install csv breed-bird-survey
```

Using download

The `download` command downloads the raw data files exactly as they occur at the source without any clean up or modification. By default the files will be stored in the working directory.

`--path` can be used to specify a location other than the working directory to download the files to. E.g., `--path ./data`

`--subdir` can be used to maintain any subdirectory structure that is present in the files being downloaded.

```
retriever download -h (gives you help options)
retriever download breed-bird-survey (download raw data files to the working␣
↪directory)
retriever download breed-bird-survey -path  C:\Users\Documents (download raw data␣
↪files to path)
```

Using citation

The `citation` command show the citation for the retriever and for the scripts.

```
retriever citation (citation of the Data retriever)
retriever citation breed-bird-survey (citation of Breed bird survey data)
```

To create new, edit, delete scripts please read the documentation on scripts

1.7 Storing database connection details

The retriever reads from the standard configuration files for the database management systems. If you want to store connection details they should be stored in those files. Make sure to secure these files appropriately.

For postgreSQL, create or modify *~/.pgpass*. This is a file named *.pgpass* located in the users home directory. It should take the general form:

`hostname:port:database:username:password`

where each word is replaced with the correct information for your database connection or replaced with an $\star$ to apply to all values for that section.

For MySQL, create or modify *~/.my.cnf*. This is a file named *.my.cnf* located in the users home directory. The relevant portion of this file for the retriever is the *client* section which should take the general form:

```
[client]
host=hostname
port=port
user=username
password=password
```

where each word to the right of the = is replaced with the correct information for your database connection. Remove or comment out the lines for any values you don't want to set.

1.8 Acknowledgments

Development of this software was funded by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4563 to Ethan White and the National Science Foundation as part of a CAREER award to Ethan White.

CHAPTER 2

Using the Data Retriever from R

2.1 rdataretriever

The Data Retriever provides an R interface to the Data Retriever so that the `retriever`'s data handling can easily be integrated into R workflows.

2.2 Installation

To use the R package `rdataretriever`, you first need to install the retriever.

The `rdataretriever` can then be installed using `install.packages("rdataretriever")`

To install the development version, use `devtools`

```
# install.packages("devtools")
library(devtools)
install_github("ropensci/rdataretriever")
```

Note: The R package takes advantage of the Data Retriever's command line interface, which must be available in the path. This path is given to the `rdataretriever` using the function `use_RetrieverPath()`. The location of `retriever` is dependent on the Python installation (Python.exe, Anaconda, Miniconda), the operating system and the presence of virtual environments in the system. The following instances exemplify this reliance and how to find retriever's path.

2.2.1 Ubuntu OS with default Python:

If `retriever` is installed in default Python, it can be found out in the system with the help of `which` command in the terminal. For example:

```
$ which retriever
/home/<system_name>/.local/bin/retriever
```

The path to be given as input to `use_RetrieverPath()` function is */home/<system_name>/.local/bin/* as shown below:

```
library(rdataretriever)
use_RetrieverPath("/home/<system_name>/.local/bin/")
```

The `which` command in the terminal finds the location of `retriever` including the name of the program, but the path required by the function is the directory that contains `retriever`. Therefore, the *retriever* needs to be removed from the path before using it.

2.2.2 Ubuntu OS with Anaconda environment:

When `retriever` is installed in an virtual environment, the user can track its location only when that particular environment is activated. To illustrate, assume the virtual environment is *py27*:

```
$ conda activate py27
(py27) $ which retriever
/home/<system_name>/anaconda2/envs/py27/bin/retriever
```

This path can be used for `rdataretriever` after removing *retriever* as follows:

```
library(rdataretriever)
use_RetrieverPath("/home/<system_name>/anaconda2/envs/py27/bin/")
```

Note: `rdataretriever` will be able to locate `retriever` even if the virtual environment is deactivated.

2.3 rdataretriever functions:

2.3.1 datasets()

Description : The function returns a list of available datasets.

Arguments : No arguments needed.

Example :

```
rdataretriever::datasets()
```

2.3.2 fetch()

Description : Each datafile in a given dataset is downloaded to a temporary directory and then imported as a data.frame as a member of a named list.

Arguments :

- `dataset` (String): Name of dataset to be downloaded
- `quiet` (Bool): The argument decides if warnings need to be displayed (TRUE/FALSE)
- `data_name` (String): Name assigned to dataset once it is downloaded

Example :

```
rdataretriever :: fetch(dataset = 'portal')
```

2.3.3 download()

Description : Used to download datasets directly without cleaning them and when user does not have a specific preference for the format of the data and the kind of database.

Arguments :

- `dataset` (String): Name of the dataset to be downloaded.
- `path` (String): Specify dataset download path.
- `quiet` (Bool): Setting TRUE minimizes the console output.
- `sub_dir` (Bool): Setting TRUE keeps the subdirectories for archived files.
- `debug` (Bool): Setting TRUE helps in debugging in case of errors.

Example :

```
rdataretriever :: download("iris","/Users/username/Desktop")
```

2.3.4 Installation functions

Format specific installation

Description : `rdataretriever` supports installation of datasets in three file formats through different functions:

- csv (`install_csv`)
- json (`install_json`)
- xml (`install_xml`)

Arguments : These functions require same arguments.

- `dataset` (String): Name of the dataset to install.
- `table_name` (String): Specify the table name to install.
- `debug` (Bool): Setting TRUE helps in debugging in case of errors.
- `use_cache` (Bool): Setting FALSE reinstalls scripts even if they are already installed.

Example :

```
rdataretriever :: install_csv("bird-size",table_name = "Bird_Size",debug = TRUE)
```

Database specific installation

Description : `rdataretriever` supports installation of datasets in four different databses through different functions:

- MySQL (`install_mysql`)
- PostgreSQL (`install_postgres`)
- SQLite (`install_sqlite`)
- MSAccess (`install_msaccess`)

Arguments for PostgreSQL and MySQL :

- `database_name` (String): Specify database name.

- `debug` (Bool): Setting True helps in debugging in case of errors.
- `host` (String): Specify host name for database.
- `password` (String): Specify password for database.
- `port` (Int): Specify the port number for installation.
- `quiet` (Bool): Setting True minimizes the console output.
- `table_name` (String): Specify the table name to install.
- `use_cache` (Bool): Setting False reinstalls scripts even if they are already installed.
- `user` (String): Specify the username.

Example :

```
rdataretriever :: install_postgres(dataset = 'portal', user='postgres', password=
↪'abcdef')
```

Arguments for MSAccess and SQLite :

- `file` (String): Enter file_name for database.
- `table_name` (String): Specify the table name to install.
- `debug` (Bool): Setting True helps in debugging in case of errors.
- `use_cache` (Bool): Setting False reinstalls scripts even if they are already installed.

Example :

```
rdataretriever :: install_sqlite(dataset = 'iris', file = 'sqlite.db',debug=FALSE,␣
↪use_cache=TRUE)
```

2.3.5 get_updates()

Description : This function will check if the version of the retriever's scripts in your local directory ' ~/.retriever/scripts/' is up-to-date with the most recent official retriever release.

Example :

```
rdataretriever :: get_updates()
```

2.3.6 reset()

Description : The function will Reset the components of rdataretriever using scope [all, scripts, data, connection]

Arguments :

- `scope` : Specifies what components to reset. Options include: 'scripts', 'data', 'connection' and 'all', where 'all' is the default setting that resets all components.

Example :

```
rdataretriever :: reset(scope = 'data')
```

2.4 Examples

```
library(rdataretriever)

# List the datasets available via the retriever
rdataretriever::datasets()

# Install the Gentry forest transects dataset into csv files in your working directory
rdataretriever::install('gentry-forest-transects', 'csv')

# Download the raw Gentry dataset files without any processing to the
# subdirectory named data
rdataretriever::download('gentry-forest-transects', './data/')

# Install and load a dataset as a list
Gentry = rdataretriever::fetch('gentry-forest-transects')
names(gentry-forest-transects)
head(gentry-forest-transects$counts)
```

To get citation information for the `rdataretriever` in R use `citation(package = 'rdataretriever')`:

CHAPTER 3

Data Retriever using Python

Data Retriever is written purely in python. The Python interface provides the core functionality supported by the CLI (Command Line Interface).

3.1 Installation

The installation instructions for the CLI and module are the same. Links have been provided below for convenience.

- Instructions for installing from binaries project website.
- Instructions for installing from source install from Source.

Note: The python interface requires version 2.1 and above.

3.2 Tutorial

Importing retriever

```
>>> import retriever
```

In this tutorial, the module will be referred to as `rt`.

```
>>> import retriever as rt
```

3.3 List Datasets

Listing available datasets using `dataset_names` function. The function returns a list of all the currently available scripts.

```
>>> rt.dataset_names()

['abalone-age',
 'antarctic-breed-bird',
 .
 .
 'wine-composition',
 'wine-quality']
```

For a more detailed description of the scripts installed in retriever, the `datasets` function can be used. This function returns a list of `Scripts` objects. From these objects, we can access the available Script's attributes as follows.

```
>>> for dataset in rt.datasets():
      print(dataset.name)

abalone-age
airports
amniote-life-hist
antarctic-breed-bird
aquatic-animal-excretion
.
.
```

There are a lot of different attributes provided in the Scripts class. Some notably useful ones are:

```
- name
- citation
- description
- keywords
- title
- urls
- version
```

You can add more datasets locally by yourself. Adding dataset documentation.

3.4 Update Datasets

If there are no scripts available, or you want to update scripts to the latest version, `check_for_updates` will download the most recent version of all scripts.

```
>>> rt.check_for_updates()

Downloading scripts...
Download Progress: [####################] 100.00%
The retriever is up-to-date
```

Downloading recipes for all datasets can take a while depending on the internet connection.

3.5 Download Datasets

To directly download datasets without cleaning them use the `download` function

```
def download(dataset, path='./', quiet=False, subdir=False, debug=False):
```

A simple download for the `iris` dataset can be done using the following.

```
>>> rt.download("iris")
```

Output:

```
=> Downloading iris

Downloading bezdekIris.data...
100%  0 seconds Copying bezdekIris.data
```

The files will be downloaded into your current working directory by default. You can change the default download location by using the `path` parameter. Here, we are downloading the `NPN` dataset to our `Desktop` directory

```
>>> rt.download("NPN","/Users/username/Desktop")
```

Output:

```
=> Downloading NPN

Downloading 2009-01-01.xml...
11  MBB
Downloading 2009-04-02.xml...
42  MBB
.
.
```

```
path (String): Specify dataset download path.

quiet  (Bool): Setting True minimizes the console output.

subdir (Bool): Setting True keeps the subdirectories for archived files.

debug  (Bool): Setting True helps in debugging in case of errors.
```

3.6 Install Datasets

Retriever supports installation of datasets into 7 major databases and file formats.

```
- csv
- json
- msaccess
- mysql
- postgres
- sqlite
- xml
```

There are separate functions for installing into each of the 7 backends:

```
def install_csv(dataset, table_name=None, compile=False, debug=False,
            quiet=False, use_cache=True):
```

(continues on next page)

(continued from previous page)

```
def install_json(dataset, table_name=None, compile=False,
             debug=False, quiet=False, use_cache=True):

def install_msaccess(dataset, file=None, table_name=None,
                 compile=False, debug=False, quiet=False, use_cache=True):

def install_mysql(dataset, user='root', password='', host='localhost',
              port=3306, database_name=None, table_name=None,
              compile=False, debug=False, quiet=False, use_cache=True):

def install_postgres(dataset, user='postgres', password='',
                 host='localhost', port=5432, database='postgres',
                 database_name=None, table_name=None,
                 compile=False, debug=False, quiet=False, use_cache=True):

def install_sqlite(dataset, file=None, table_name=None,
                compile=False, debug=False, quiet=False, use_cache=True):

def install_xml(dataset, table_name=None, compile=False, debug=False,
            quiet=False, use_cache=True):
```

A description of default parameters mentioned above:

```
compile          (Bool): Setting True recompiles scripts upon installation.

database_name (String): Specify database name. For postgres, mysql users.

debug            (Bool): Setting True helps in debugging in case of errors.

file           (String): Enter file_name for database. For msaccess, sqlite users.

host           (String): Specify host name for database. For postgres, mysql users.

password       (String): Specify password for database. For postgres, mysql users.

port              (Int): Specify the port number for installation. For postgres, mysql␣
↪users.

quiet            (Bool): Setting True minimizes the console output.

table_name     (String): Specify the table name to install.

use_cache        (Bool): Setting False reinstalls scripts even if they are already␣
↪installed.

user           (String): Specify the username. For postgres, mysql users.
```

Examples to Installing Datasets:

Here, we are installing the dataset wine-composition as a CSV file in our current working directory.

```
rt.install_csv("wine-composition")

=> Installing wine-composition

Downloading wine.data...
100%  0 seconds Progress: 178/178 rows inserted into ./wine_composition_
↪WineComposition.csv totaling 178
```

(continues on next page)

(continued from previous page)

The installed file is called `wine_composition_WineComposition.csv`

Similarly, we can download any available dataset as a JSON file:

```
rt.install_json("wine-composition")

=> Installing wine-composition

Progress: 178/178 rows inserted into ./wine_composition_WineComposition.json totaling␣
↪17
```

The wine-composition dataset is now installed as a JSON file called wine_composition_WineComposition.json in our current working directory.

CHAPTER 4

Datasets Available

4.1 1. Wine Composition

name wine-composition

reference *Exploration, Classification and Correlation. Institute of Pharmaceutical*

citation Forina, M. et al, PARVUS - An Extendible Package for Data

description A chemical analysis of wines grown in the same region in Italy but derived from three different cultivators.

4.2 2. Mapped plant quadrat time-series from Montana (Anderson et al. 2011)

name mapped-plant-quads-mt

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3551799*

citation Jed Anderson, Lance Vermeire, and Peter B. Adler. 2011. Fourteen years of mapped, permanent quadrats in a northern mixed prairie, USA. Ecology 92:1703.

description Long term plant quadrats of northern mixed prairie in Montana.

4.3 3. Dataset containing information on all airports on ouraiports.com

name airports

reference *http://ourairports.com/data/*

citation OurAirports.com, Megginson Technologies Ltd.

description Dataset containing information on all airports on ourairports.com

4.4 4. Miscellaneous Abundance Database (figshare 2012)

name community-abundance-misc

reference *Not available*

citation Baldridge, Elita, A Data-intensive Assessment of the Species Abundance Distribution(2013). All Graduate Theses and Dissertations. Paper 4276.

description Community abundance data for fish, reptiles, amphibians, beetles, spiders, and birds, compiled from the literature by Elita Baldridge.

4.5 5. Fray Jorge community ecology database (Kelt et al. 2013)

name fray-jorge-ecology

reference fray-jorge-ecology's home link.

citation

D. A. Kelt, P. L. Meserve, J. R. Gutierrez, W. Bryan Milstead, and M. A. Previtali. 2013. Long-term monitoring of mammals in the face of biotic and abiotic influences at a semiarid site in north-central Chile. Ecology 94:977. http://dx.doi.org/10.1890/12-1811.1.

description Long-term monitoring of small mammal and plant communities in the face of biotic and abiotic influences at a semiarid site in north-central Chile.

4.6 6. Database of Vertebrate Home Range Sizes - Tamburello et al., 2015

name home-ranges

reference *http://datadryad.org/resource/doi:10.5061/dryad.q5j65/1*

citation Tamburello N, Cote IM, Dulvy NK (2015) Energy and the scaling of animal space use. The American Naturalist 186(2):196-211. http://dx.doi.org/10.1086/682070.

description Database of mean species masses and corresponding empirically measured home range sizes for 569 vertebrate species from across the globe, including birds, mammals, reptiles, and fishes.

4.7 7. Oosting Natural Area (North Carolina) plant occurrence (Palmer et al. 2007)

name plant-occur-oosting

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3528371*

citation Michael W. Palmer, Robert K. Peet, Rebecca A. Reed, Weimin Xi, and Peter S. White. 2007. A multiscale study of vascular plants in a North Carolina Piedmont forest. Ecology 88:2674.

description A data set collected in 1989 of vascular plant occurrences in overlapping grids of nested plots in the Oosting Natural Area of the Duke Forest, Orange County, North Carolina, USA.

4.8 8. Biovolumes for freshwater phytoplankton - Colin et al. 2014

name phytoplankton-size

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3560628*

citation Colin T. Kremer, Jacob P. Gillette, Lars G. Rudstam, Pal Brettum, and Robert Ptacnik. 2014. A compendium of cell and natural unit biovolumes for >1200 freshwater phytoplankton species. Ecology 95:2984.

description Sampling phytoplankton communities basing on cell size.

4.9 9. Car Evaluation

name car-eval

reference *http://archive.ics.uci.edu/ml/datasets/Car+Evaluation*

citation Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

description A database useful for testing constructive induction and structure discovery methods.

4.10 10. Poker Hand dataset

name poker-hands

reference *http://archive.ics.uci.edu/ml/datasets/Poker+Hand*

citation Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

description A dataset used to predict poker hands

4.11 11. National_Lakes_Assessment_Data

name nla

reference nla's home link.

citation NA

description The National Aquatic Resource Surveys (NARS) are statistical surveys designed to assess the status of and changes in quality of the coastal waters of the nation, lakes and reservoirs, rivers and streams, and wetlands. Using sample sites selected at random, these surveys provide a snapshot of the overall condition of water belonging to the nation. Because the surveys use standardized field and lab methods, we can compare results from different parts of the country and between years. EPA works with state, tribal and federal partners to design and implement the National Aquatic Resource Surveys.

4.12 12. Fish parasite host ecological characteristics (Strona, et al., 2013)

name fish-parasite-hosts

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3555378*

citation Giovanni Strona, Maria Lourdes D. Palomares, Nicolas Bailly, Paolo Galli, and Kevin D. Lafferty. 2013. Host range, host ecology, and distribution of more than 11800 fish parasite species. Ecology 94:544.

description The data set includes 38008 fish parasite records (for Acanthocephala, Cestoda, Monogenea, Nematoda, Trematoda) compiled from scientific literature.

4.13 13. Sagebrush steppe mapped plant quadrats (Zachmann et al. 2010)

name mapped-plant-quads-id

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3550215*

citation Luke Zachmann, Corey Moffet, and Peter Adler. 2010. Mapped quadrats in sagebrush steppe:long-term data for analyzing demographic rates and plant-plant interactions. Ecology 91:3427.

description This data set consists of 26 permanent 1-m2 quadrats located on sagebrush steppe in eastern Idaho, USA.

4.14 14. ND-Gain

name nd-gain

reference *http://index.gain.org/*

citation Chen, C., Noble, I., Hellmann, J., Coffee, J., Murillo, M. and Chawla, N., 2015. University of Notre Dame Global Adaptation Index Country Index Technical Report. ND-GAIN: South Bend, IN, USA.

description The ND-GAIN Country Index summarizes a country's vulnerability to climate change and other global challenges in combination with its readiness to improve resilience. It aims to help governments, businesses and communities better prioritize investments for a more efficient response to the immediate global challenges ahead.

4.15 15. Forest fire data for Montesinho natural park in Portugal

name forest-fires-portugal

reference *http://archive.ics.uci.edu/ml/datasets/Forest+Fires*

citation

P. Cortez and A. Morais. A Data Mining Approach to Predict Forest Fires using Meteorological Data. In J. Neves, M. F. Santos and J. Machado Eds., New Trends in Artificial Intelligence, Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence, December, Guimaraes, Portugal, pp. 512-523, 2007. APPIA, ISBN-13 978-989-95618-0-9.

description A database for regression analysis with the aim of predicting burned areas of forestry using meteorological and other data.

4.16 16. Portal Project Data (Ernest et al. 2016)

name portal-dev

reference *https://github.com/weecology/PortalData*

citation

S. K.M. Ernest, G. M. Yenni, G. Allington, E. M. Christensen, K. Geluso, J. R. Goheen, M. R. Schutzenhofer, S. R. Supp, K. M. Thibault, James H. Brown, and T. J. Valone. 2016. Long-term monitoring and experimental manipulation of a Chihuahuan desert ecosystem near Portal, Arizona (1977-2013). Ecology 97:1082.

description The data set represents a Desert ecosystems using the composition and abundances of ants, plants, and rodents has occurred continuously on 24 plots.

4.17 17. Croche understory vegetation data set

name croche-vegetation-data

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3528707*

citation Alain Paquette, Etienne Laliberté, André Bouchard, Sylvie de Blois, Pierre Legendre, and Jacques Brisson. 2007. Lac Croche understory vegetation data set (1998-2006). Ecology 88:3209.

description The Lac Croche data set covers a nine-year period (1998-2006) of detailed understory vegetation sampling of a temperate North American forest located in the Station de Biologie des Laurentides (SBL), Québec, Canada.

4.18 18. Foraging attributes for birds and mammals (Wilman, et al., 2014)

name elton-traits

reference elton-traits's home link.

citation Hamish Wilman, Jonathan Belmaker, Jennifer Simpson, Carolina de la Rosa, Marcelo M. Rivadeneira, and Walter Jetz. 2014. EltonTraits 1.0: Species-level foraging attributes of the world's birds and mammals. Ecology 95:2027.

description Characterization of species by physiological, behavioral, and ecological attributes that are subjected to varying evolutionary and ecological constraints and jointly determine their role and function in ecosystems.

4.19 19. First-flowering dates of plants in the Northern Great Plains

name ngreatplains-flowering-dates

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3531716*

citation Steven E. Travers and Kelsey L. Dunnell. 2009. First-flowering dates of plants in the Northern Great Plains. Ecology 90:2332.

description Observations data of first-flowering time of native and nonnative plant species in North Dakota and Minnesota over the course of 51 years in the last century

4.20 20. BUPA liver disorders

name bupa-liver-disorders

reference *https://archive.ics.uci.edu/ml/datasets/Liver+Disorders*

citation Richard S. Forsyth, 8 Grosvenor Avenue, Mapperley Park , Nottingham NG3 5DX, 0602-621676

description The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the dataset constitutes the record of a single male individual. The 7th field (selector) has been widely misinterpreted in the past as a dependent variable representing presence or absence of a liver disorder. This is incorrect. The 7th field was created by BUPA researchers as a train/test selector. It is not suitable as a dependent variable for classification. The dataset does not contain any variable representing presence or absence of a liver disorder.

4.21 21. Barnacle, fucoid, and mussel recruitment in the Gulf of Maine, USA, from 1997 to 2007

name marine-recruitment-data

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3530633*

citation Peter S. Petraitis, Harrison Liu, and Erika C. Rhile. 2009. Barnacle, fucoid, and mussel recruitment in the Gulf of Maine, USA, from 1997 to 2007. Ecology 90:571.

description This data set provides access to recruitment data collected in the experimental plots from 1997 to 2007

4.22 22. Nesting ecology and offspring recruitment in a long-lived turtle

name turtle-offspring-nesting

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3531323*

citation Lisa E. Schwanz, Rachel M. Bowden, Ricky-John Spencer, and Fredric J. Janzen. 2009. Nesting ecology and offspring recruitment in a long-lived turtle. Ecology 90:1709. [https://doi.org/10.6084/m9.figshare.3531323.v1]

description Valuable empirical resource for exploring important facets of nesting ecology and hatchling recruitment in a wild population of a long-lived species.

4.23 23. 3-D maps of tree canopy geometries at leaf scale

name tree-canopy-geometries

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3530507*

citation Hervé Sinoquet, Sylvain Pincebourde, Boris Adam, Nicolas Donès, Jessada Phattaralerphong, Didier Combes, Stéphane Ploquin, Krissada Sangsing, Poonpipope Kasemsap, Sornprach Thanisawanyangkura, Géraldine Groussier-Bout, and Jérôme Casas. 2009. 3-D maps of tree canopy geometries at leaf scale. Ecology 90:283

description This data set reports the three-dimensional geometry of a set of fruit and rubber trees at the leaf scale

4.24 24. New York City TreesCount

name nyc-tree-count

reference *https://www.nycgovparks.org/trees/treescount*

citation TreeCount 2015 is citizen science project of NYC Parks'[https://www.nycgovparks.org/trees/treescount].

description Dataset consist of every street tree of New York City on the block

4.25 25. Fire-related traits for plant species of the Mediterranean Basin. Ecology 90:1420

name mediter-basin-plant-traits

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3531092*

citation

S. Paula, M. Arianoutsou, D. Kazanis, Ç. Tavsanoglu, F. Lloret, C. Buhk, F. Ojeda, B. Luna, J. M. Moreno, A. Rodrigo, J. M. Espelta, S. Palacio, B. Fernández-Santos,, P. M. Fernandes, and J. G. Pausas. 2009. Fire-related traits for plant species of the Mediterranean Basin. Ecology 90:1420.

description This data set compiles the most updated and comprehensive information on fire-related traits for vascular plant species of the Mediterranean Basin

4.26 26. The ph_ownership_history dataset

name harvard-forest

reference *http://harvardforest.fas.harvard.edu/*

citation Hall B. 2017. Historical GIS Data for Harvard Forest Properties from 1908 to Present. Harvard Forest Data Archive: HF110.

description ph_ownership_history

4.27 27. Biomass and Its Allocation in Chinese Forest Ecosystems (Luo, et al., 2014)

name forest-biomass-china

reference forest-biomass-china's home link.

citation Yunjian Luo, Xiaoquan Zhang, Xiaoke Wang, and Fei Lu. 2014. Biomass and its allocation in Chinese forest ecosystems. Ecology 95:2026.

description Forest biomass data set of China which includes tree overstory components (stems, branches, leaves, and roots, among all other plant material), the understory vegetation (saplings, shrubs, herbs, and mosses), woody liana vegetation, and the necromass components of dead organic matter (litterfall, suspended branches, and dead trees).

4.28 28. Breed-Bird-Survey-nlcd Data

name breed-bird-survey-nlcd

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3554424*

citation Michael F. Small, Joseph A. Veech, and Jennifer L. R. Jensen. 2012. Local landscape composition and configuration around North American Breeding Bird Survey routes. Ecology 93:2298.

description Landcover data for all North American Breeding Bird Survey routes from the 2006 National Land Cover Database at buffers from 200 m to 10 km..

4.29 29. Mapped plant quadrat time-series from Kansas (Adler et al. 2007)

name mapped-plant-quads-ks

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3528368*

citation Peter B. Adler, William R. Tyburczy, and William K. Lauenroth. 2007. Long-term mapped quadrats from Kansas prairie:demographic information for herbaceaous plants. Ecology 88:2673.

description Demographic data for testing current theories in plant ecology and forecasting the effects of global change.

4.30 30. Shortgrass steppe mapped plants quads - Chu et al. 2013

name mapped-plant-quads-co

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3556779*

citation Cover, density, and demographics of shortgrass steppe plants mapped 1997-2010 in permanent grazed and ungrazed quadrats. Chengjin Chu, John Norman, Robert Flynn, Nicole Kaplan, William K. Lauenroth, and Peter B. Adler. Ecology 2013 94:6, 1435-1435.

description This data set maps and analyzes demographic rates of many common plant species in the shortgrass steppe of North America under grazed and ungrazed conditions.

4.31 31. Wine Quality

name wine-quality

reference wine-quality's home link.

citation

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

description Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests

4.32 32. Bioclim 2.5 Minute Climate Data

name bioclim

reference *http://worldclim.org/bioclim*

citation Hijmans, R.J., S.E. Cameron, J.L. Parra, P.G. Jones and A. Jarvis, 2005. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25: 1965-1978.

description Bioclimatic variables that are derived from the monthly temperature and rainfall values in order to generate more biologically meaningful variables.

4.33 33. Bird Body Size and Life History (Lislevand et al. 2007)

name bird-size

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3527864*

citation Terje Lislevand, Jordi Figuerola, and Tamas Szekely. 2007. Avian body sizes in relation to fecundity, mating system, display behavior, and resource sharing. Ecology 88:1605.

description A comprehensive compilation of data set on avian body sizes that would be useful for future comparative studies of avian biology.

4.34 34. species data on densities and percent cover in the 60 experimental plots from 1996 to 2002

name macroalgal_communities

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3526004*

citation Peter S. Petraitis and Nicholas Vidargas. 2006. Marine intertidal organisms found in experimental clearings on sheltered shores in the Gulf of Maine, USA. Ecology 87:796.

description Experimental clearings in macroalgal stands were established in 1996 to determine if mussel beds and macroalgal stands on protected intertidal shores of New England represent alternative community states

4.35 35. The LakeCat Dataset

name lakecats-final-tables

reference *https://www.epa.gov/national-aquatic-resource-surveys/lakecat*

citation Hill, Ryan A., Marc H. Weber, Rick Debbout, Scott G. Leibowitz, Anthony R. Olsen. 2018. The Lake-Catchment (LakeCat) Dataset: characterizing landscape features for lake basins within the conterminous USA. Freshwater Science doi:10.1086/697966.

description This current lakecat dataset has 136 local catchment (Cat) and 136 watershed (Ws) metrics making a total of 272 metrics.

4.36 36. Mammal Community DataBase (Thibault et al. 2011)

name mammal-community-db

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3552243*

citation Katherine M. Thibault, Sarah R. Supp, Mikaelle Giffin, Ethan P. White, and S. K. Morgan Ernest. 2011. Species composition and abundance of mammalian communities. Ecology 92:2316.

description This data set includes species lists for 1000 mammal communities, excluding bats, with species-level abundances available for 940 of these communities.

4.37 37. Sonoran Desert Lab perennials vegetation plots

name veg-plots-sdl

reference *Not available*

citation Susana Rodriguez-Buritica, Helen Raichle, Robert H. Webb, Raymond M. Turner, and D. Lawrence Venable. 2013. One hundred and six years of population and community dynamics of Sonoran Desert Laboratory perennials. Ecology 94:976.

description The data set constitutes all information associated with the Spalding-Shreve permanent vegetation plots from 1906 through 2012, which is the longest-running plant monitoring program in the world.

4.38 38. Antarctic Site Inventory breeding bird survey data, 1994-2013

name antarctic-breed-bird

reference antarctic-breed-bird's home link.

citation Heather J. Lynch, Ron Naveen, and Paula Casanovas. 2013. Antarctic Site Inventory breeding bird survey data, 1994-2013. Ecology 94:2653.

description The data set represents the accumulation of 19 years of seabird population abundance data which was collected by the Antarctic Site Inventory, an opportunistic vessel-based monitoring program surveying the Antarctic Peninsula and associated sub-Antarctic Islands.

4.39 39. Percentage leaf herbivory across vascular plant species

name leaf-herbivory

reference leaf-herbivory's home link.

citation Martin M. Turcotte, Christina J. M. Thomsen, Geoffrey T. Broadhead, Paul V. A. Fine, Ryan M. Godfrey, Greg P. A. Lamarre, Sebastian T. Meyer, Lora A. Richards, and Marc T. J. Johnson. 2014. Percentage leaf herbivory across vascular plant species. Ecology 95:788. http://dx.doi.org/10.1890/13-1741.1.

description Spatially explicit measurements of population level leaf herbivory on 1145 species of vascular plants from 189 studies from across the globe.

4.40 40. Portal Project Data (Ernest et al. 2009)

name portal

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3531317*

citation

S. K. Morgan Ernest, Thomas J. Valone, and James H. Brown. 2009. Long-term monitoring and experimental manipulation of a Chihuahuan Desert ecosystem near Portal, Arizona, USA. Ecology 90:1708.

description The data set represents a Desert ecosystems using the composition and abundances of ants, plants, and rodents has occurred continuously on 24 plots. Currently includes only mammal data.

4.41 41. MammalDIET

name mammal-diet

reference *Not available*

citation Kissling WD, Dalby L, Flojgaard C, Lenoir J, Sandel B, Sandom C, Trojelsgaard K, Svenning J-C (2014) Establishing macroecological trait datasets:digitalization, extrapolation, and validation of diet preferences in terrestrial mammals worldwide. Ecology and Evolution, online in advance of print. doi:10.1002/ece3.1136

description MammalDIET provides a comprehensive, unique and freely available dataset on diet preferences for all terrestrial mammals worldwide.

4.42 42. Wisconsin Breast Cancer Database

name breast-cancer-wi

reference *http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29*

citation Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

description Database containing information on Wisconsin Breast Cancer Diagnostics

4.43 43. Effects of biodiversity on the functioning of ecosystems:A summary of 164 experimental manipulations of species richness

name species-exctinction-rates

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3530825*

citation Bradley J. Cardinale, Diane S. Srivastava, J. Emmett Duffy, Justin P. Wright, Amy L. Downing, Mahesh Sankaran, Claire Jouseau, Marc W. Cadotte, Ian T. Carroll, Jerome J. Weis, Andy Hector, and Michel Loreau. 2009. Effects of biodiversity on the functioning of ecosystems:A summary of 164 experimental manipulations of species richness. Ecology 90:854.

description A summary of the results on the accelerating rates of species extinction

4.44 44. Iris Plants Database

name iris

reference *http://mlr.cs.umass.edu/ml/datasets/Iris*

citation

R. A. Fisher. 1936. The Use of Multiple Measurements in Taxonomic Problems. and Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

description Famous dataset from R. A. Fisher. This dataset has been corrected. Information Source: Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

4.45 45. The distribution and host range of the pandemic disease chytridiomycosis in Australia, spanning surveys from 1956-2007.

name chytr-disease-distr

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3547077*

citation Kris Murray, Richard Retallick, Keith R. McDonald, Diana Mendez, Ken Aplin, Peter Kirkpatrick, Lee Berger, David Hunter, Harry B. Hines, R. Campbell, Matthew Pauza, Michael Driessen, Richard Speare, Stephen J. Richards, Michael Mahony, Alastair Freeman, Andrea D. Phillott, Jean-Marc Hero, Kerry Kriger, Don Driscoll, Adam Felton, Robert Puschendorf, and Lee F. Skerratt. 2010. The distribution and host range of the pandemic disease chytridiomycosis in Australia, spanning surveys from 1956-2007. Ecology 91:1557.

description The data is of a distribution and host range of this invasive disease in Australia

4.46 46. Masses of Mammals (Smith et al. 2003)

name mammal-masses

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3523112*

citation Felisa A. Smith, S. Kathleen Lyons, S. K. Morgan Ernest, Kate E. Jones, Dawn M. Kaufman, Tamar Dayan, Pablo A. Marquet, James H. Brown, and John P. Haskell. 2003. Body mass of late Quaternary mammals. Ecology 84:3403.

description A data set of compiled body mass information for all mammals on Earth.

4.47 47. GDP Data

name gdp

reference *https://github.com/datasets/gdp/blob/master*

citation NA

description Country, regional and world GDP in current US Dollars ($). Regional means collections of countries e.g. Europe & Central Asia. Data is sourced from the World Bank and turned into a standard normalized CSV.

4.48 48. Body sizes of consumers and their resources

name predator-prey-body-ratio

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3525119*

citation Ulrich Brose, Lara Cushing, Eric L. Berlow, Tomas Jonsson, Carolin Banasek-Richter, Louis-Felix Bersier, Julia L. Blanchard, Thomas Brey, Stephen R. Carpenter, Marie-France Cattin Blandenier, Joel E. Cohen, Hassan Ali Dawah, Tony Dell, Francois Edwards, Sarah Harper-Smith, Ute Jacob, Roland A. Knapp, Mark E. Ledger, Jane Memmott, Katja Mintenbeck, John K. Pinnegar, Bjorn C. Rall, Tom Rayner, Liliane Ruess, Werner Ulrich, Philip Warren, Rich J. Williams, Guy Woodward, Peter Yodzis, and Neo D. Martinez10. 2005. Body sizes of consumers and their resources. Ecology 86:2545.

description Body size ratios between predators and their prey,

4.49 49. USDA plant list - taxonomy for US plant species

name plant-taxonomy-us

reference *http://plants.usda.gov*

citation USDA, NRCS. 2017. The PLANTS Database (http://plants.usda.gov, DATEOFDOWNLOAD). National Plant Data Team, Greensboro, NC 27401-4901 USA.

description Plant taxonomy data for the United States from the USDA plants website

4.50 50. Partners_In_Flight_Species_Assessment_Data

name partners-in-flight

reference *http://rmbo.org/pifassessment/Database.aspx*

citation Partners in Flight. 2017. Avian Conservation Assessment Database, version 2017. Available at http://pif.birdconservancy.org/ACAD. Accessed on 19.2.2018

description The Partners in Flight (PIF) Species Assessment Database is now the Avian Conservation Assessment Database, Whereas the Species Assessment Database contained information only on landbirds in Canada, USA and Mexico, the Avian Conservation Assessment Database contains assessment data for all North American birds from Canada to Panama.

4.51 51. Demography of the endemic mint Dicerandra frutescens in Florida scrub

name dicerandra-frutescens

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3529460*

citation Eric S. Menges. 2008. Demography of the endemic mint Dicerandra frutescens in Florida scrub. Ecology 89:1474.

description Study of the demography of Dicerandra frutescens, an endemic and endangered mint restricted to Florida scrub

4.52 52. A database on visible diurnal spring migration of birds

name bird-migration-data

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3551952*

citation Georg F. J. Armbruster, Manuel Schweizer, and Deborah R. Vogt. 2011. A database on visible diurnal spring migration of birds (Central Europe:Lake Constance). Ecology 92:1865.

description Birds migration data

4.53 53. Phylogeny and metabolic rates in mammals (Ecological Archives 2010)

name mammal-metabolic-rate

reference *https://figshare.com/collections/Phylogeny_and_metabolic_scaling_in_mammals/3303477*

citation Isabella Capellini, Chris Venditti, and Robert A. Barton. 2010. Phylogeny and metabolic rates in mammals. Ecology 20:2783-2793.

description Data on basal metabolic rate (BMR) with experimental animal body mass, field metabolic rate (FMR) with wild animal body mass, and sources of the data. Ecological Archives E091-198-S1.

4.54 54. Mammal abundance indices in the northern portion of the Great Basin

name great-basin-mammal-abundance

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3525485*

citation Rebecca A. Bartel, Frederick F. Knowlton, and Charles Stoddart. 2005. Mammal abundance indices in the northern portion of the Great Basin, 1962-1993. Ecology 86:3130.

description Indices of abundance of selected mammals obtained for two study areas within the Great Basin.

4.55 55. A stream gage database for evaluating natural and altered flow conditions in the conterminous United States.

name streamflow-conditions

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3544358*

citation James A. Falcone, Daren M. Carlisle, David M. Wolock, and Michael R. Meador. 2010. GAGES:A stream gage database for evaluating natural and altered flow conditions in the conterminous United States. Ecology 91:621.

description streamflow in ecosystems

4.56 56. Nematode traits and environmental constraints in 200 soil systems

name nematode-traits

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3552057*

citation Christian Mulder and J. Arie Vonk. 2011. Nematode traits and environmental constraints in 200 soil systems:scaling within the 60–6000 µm body size range. Ecology 92:2004.

description This data set includes information on taxonomy, life stage, sex, feeding habit, trophic level, geographic location, sampling period, ecosystem type, soil type, and soil chemistry

4.57 57. Mammal Life History Database - Ernest, et al., 2003

name mammal-life-hist

reference mammal-life-hist's home link.

citation

S. K. Morgan Ernest. 2003. Life history characteristics of placental non-volant mammals. Ecology 84:3402.

description The purpose of this data set was to compile general life history characteristics for a variety of mammalian species to perform comparative life history analyses among different taxa and different body size groups.

4.58 58. BioTIME species identities and abundances

name biotime

reference *https://zenodo.org/record/1095628#.WskN7dPwYyn*

citation Dornelas M, Antão LH, Moyes F, et al. BioTIME: A database of biodiversity time series for the Anthropocene. Global Ecology & Biogeography. 2018; 00:1 - 26. https://doi.org/10.1111/geb.12729.

description The BioTIME database has species identities and abundances in ecological assemblages through time.

4.59 59. Vascular plant composition - McGlinn, et al., 2010

name plant-comp-ok

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3547209*

citation Daniel J. McGlinn, Peter G. Earls, and Michael W. Palmer. 2010. A 12-year study on the scaling of vascular plant composition in an Oklahoma tallgrass prairie. Ecology 91:1872.

description The data is part of a monitoring project on vascular plant composition at the Tallgrass Prairie Preserve in Osage County, Oklahoma, USA.

4.60 60. Michigan forest canopy dynamics plots - Woods et al. 2009

name forest-plots-michigan

reference forest-plots-michigan's home link.

citation Kerry D. Woods. 2009. Multi-decade, spatially explicit population studies of canopy dynamics in Michigan old-growth forests. Ecology 90:3587.

description The data set provides stem infomation from a regular grid of 256 permanent plots includes about 20% of a 100-ha old-growth forest at the Dukes Research Natural Area in northern Michigan, USA.

4.61 61. The data was used to investigate patterns and causes of variation in NPP by the giant kelp, Macrocystis pyrifera, which is believed to be one of the fastest growing autotrophs on earth.

name macrocystis-variation

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3529700*

citation Andrew Rassweiler, Katie K. Arkema, Daniel C. Reed, Richard C. Zimmerman, and Mark A. Brzezinski. 2008. Net primary production, growth, and standing crop of Macrocystis pyrifera in southern California. Ecology 89:2068.

description

4.62 62. Global Biotic Interactions (GloBI) data

name globi-interaction

reference *https://github.com/jhpoelen/eol-globi-data/wiki*

citation Poelen, J.H., Simons, J.D. and Mungall, C.J., 2014. Global biotic interactions: an open infrastructure to share and analyze species-interaction datasets. Ecological Informatics, 24, pp.148-159.

description GloBI contains code to normalize and integrate existing species-interaction datasets and export the resulting integrated interaction dataset.

4.63 63. Mount St. Helens vegetation recovery plots (del Moral 2010)

name mt-st-helens-veg

reference mt-st-helens-veg's home link.

citation Roger del Moral. 2010. Thirty years of permanent vegetation plots, Mount St. Helens, Washington. Ecology 91:2185.

description Documenting vegetation recovery from volcanic disturbances using the most common species found in non-forested habitats on Mount St. Helens.

4.64 64. Spatial Population Data Alpine Butterfly - Matter et al 2014

name butterfly-population-network

reference butterfly-population-network's home link.

citation Matter, Stephen F., Nusha Keyghobadhi, and Jens Roland. 2014. Ten years of abundance data within a spatial population network of the alpine butterfly, Parnassius smintheus. Ecology 95:2985. Ecological Archives E095-258.

description Stephen F. Matter, Nusha Keyghobadhi, and Jens Roland. 2014. Ten years of abundance data within a spatial population network of the alpine butterfly, Parnassius smintheus. Ecology 95:2985.

4.65 65. Abalone Age and Size Data

name abalone-age

reference *http://archive.ics.uci.edu/ml/datasets/Abalone*

citation Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

description Database to aid in the prediction of the age of an Abalone given physical measurements

4.66 66. The effects of biodiversity on ecosystem community, and population variables reported 1974-2004

name biodiversity-response

reference *https://figshare.com/articles/Data_Paper_Data_Paper/3530822*

citation Bernhard Schmid, Andrea B. Pfisterer, and Patricia Balvanera. 2009. Effects of biodiversity on ecosystem community, and population variables reported 1974-2004. Ecology 90:853

description

4.67 67. Food web including metazoan parasites for a brackish shallow water ecosystem in Germany and Denmark

name flensburg-food-web

reference *https://figshare.com/articles/Full_Archive/3552066*

citation

C. Dieter Zander, Neri Josten, Kim C. Detloff, Robert Poulin, John P. McLaughlin, and David W. Thieltges. 2011. Food web including metazoan parasites for a brackish shallow water ecosystem in Germany and Denmark. Ecology 92:2007.

description This data is of a food web for the Flensburg Fjord, a brackish shallow water inlet on the Baltic Sea, between Germany and Denmark.

4.68 68. Alwyn H. Gentry Forest Transect Dataset

name gentry-forest-transects

reference *http://www.mobot.org/mobot/research/gentry/welcome.shtml*

citation Phillips, O. and Miller, J.S., 2002. Global patterns of plant diversity: Alwyn H. Gentry's forest transect data set. Missouri Botanical Press.

description

4.69 69. USGS North American Breeding Bird Survey 50 stop

name breed-bird-survey-50stop

reference *http://www.pwrc.usgs.gov/BBS/*

citation Pardieck, K.L., D.J. Ziolkowski Jr., M.-A.R. Hudson. 2015. North American Breeding Bird Survey Dataset 1966 - 2014, version 2014.0. U.S. Geological Survey, Patuxent Wildlife Research Center.

description A Cooperative effort between the U.S. Geological Survey's Patuxent Wildlife Research Center and Environment Canada's Canadian Wildlife Service to monitor the status and trends of North American bird populations.

4.70 70. Aquatic Animal Excretion

name aquatic-animal-excretion

reference *http://onlinelibrary.wiley.com/doi/10.1002/ecy.1792/abstract*

citation Vanni, M. J., McIntyre, P. B., Allen, D., Arnott, D. L., Benstead, J. P., Berg, D. J., Brabrand, Å., Brosse, S., Bukaveckas, P. A., Caliman, A., Capps, K. A., Carneiro, L. S., Chadwick, N. E., Christian, A. D., Clarke, A., Conroy, J. D., Cross, W. F., Culver, D. A., Dalton, C. M., Devine, J. A., Domine, L. M., Evans-White, M. A., Faafeng, B. A., Flecker, A. S., Gido, K. B., Godinot, C., Guariento, R. D., Haertel-Borer, S., Hall, R. O., Henry, R., Herwig, B. R., Hicks, B. J., Higgins, K. A., Hood, J. M., Hopton, M. E., Ikeda, T., James, W. F., Jansen, H. M., Johnson, C. R., Koch, B. J., Lamberti, G. A., Lessard-Pilon, S., Maerz, J. C., Mather, M. E., McManamay, R. A., Milanovich, J. R., Morgan, D. K. J., Moslemi, J. M., Naddafi, R., Nilssen, J. P., Pagano, M., Pilati, A., Post, D. M., Roopin, M., Rugenski, A. T., Schaus, M. H., Shostell, J., Small, G. E., Solomon, C. T., Sterrett, S. C., Strand, O., Tarvainen, M., Taylor, J. M., Torres-Gerald, L. E., Turner, C. B., Urabe, J., Uye, S.-I., Ventelä, A.-M., Villeger, S., Whiles, M. R., Wilhelm, F. M., Wilson, H. F., Xenopoulos, M. A. and Zimmer, K. D. (2017), A global database of nitrogen and phosphorus excretion rates of aquatic animals. Ecology. Accepted Author Manuscript. doi:10.1002/ecy.1792

description Dataset containing the nutrient cycling rates of individual animals.

4.71 71. BAAD: a Biomass And Allometry Database for woody plants

name biomass-allometry-db

reference *https://doi.org/10.6084/m9.figshare.c.3307692.v1*

citation Falster, D.S., Duursma, R.A., Ishihara, M.I., Barneche, D.R., FitzJohn, R.G., Varhammar, A., Aiba, M., Ando, M., Anten, N., Aspinwall, M.J. and Baltzer, J.L., 2015. BAAD: a Biomass And Allometry Database for woody plants.

description The data set is a Biomass and allometry database (BAAD) for woody plants containing 259634 measurements collected in 176 different studies from 21084 individuals across 678 species.

4.72 72. Vertnet Mammals

name vertnet-mammals

reference *http://vertnet.org/resources/datatoolscode.html*

citation Bloom, D., Wieczorek J., Russell, L. (2016). VertNet_Mammals_Sept. 2016. CyVerse Data Commons. http://datacommons.cyverse.org/browse/iplant/home/shared/commons_repo/curated/VertNet_Mammals_Sep2016

description Compilation of digitized museum records of mammals including locations, dates of collection, and some trait data.

4.73 73. Commercial Fisheries Monthly Trade Data by Product, Country/Association

name noaa-fisheries-trade

reference noaa-fisheries-trade's home link.

citation No known Citation

description Commercial Fisheries statistics provides a summary of commercial fisheries product data by individual country.

4.74 74. USGS North American Breeding Bird Survey

name breed-bird-survey

reference *http://www.pwrc.usgs.gov/BBS/*

citation Pardieck, K.L., D.J. Ziolkowski Jr., M.-A.R. Hudson. 2015. North American Breeding Bird Survey Dataset 1966 - 2014, version 2014.0. U.S. Geological Survey, Patuxent Wildlife Research Center

description A Cooperative effort between the U.S. Geological Survey's Patuxent Wildlife Research Center and Environment Canada's Canadian Wildlife Service to monitor the status and trends of North American bird populations.

4.75 75. Indian Forest Stand Structure and Composition (Ramesh et al. 2010)

name forest-plots-wghats

reference forest-plots-wghats's home link.

citation

B. R. Ramesh, M. H. Swaminath, Santoshgouda V. Patil, Dasappa, Raphael Pelissier, P. Dilip Venugopal, S. Aravajy, Claire Elouard, and S. Ramalingam. 2010. Forest stand structure and composition in 96 sites along environmental gradients in the central Western Ghats of India. Ecology 91:3118.

description This data set reports woody plant species abundances in a network of 96 sampling sites spread across 22000 km2 in central Western Ghats region, Karnataka, India.

4.76 76. Vertnet Birds

name vertnet-birds

reference *http://vertnet.org/resources/datatoolscode.html*

citation Bloom, D., Wieczorek J., Russell, L. (2016). VertNet_Aves_Sept. 2016. CyVerse Data Commons. http://datacommons.cyverse.org/browse/iplant/home/shared/commons_repo/curated/VertNet_Aves_Sep2016

description Compilation of digitized museum records of birds including locations, dates of collection, and some trait data.

4.77 77. Vertnet Amphibians

name vertnet-amphibians

reference *http://vertnet.org/resources/datatoolscode.html*

citation Bloom, D., Wieczorek J., Russell, L. (2016). VertNet_Amphibia_Sept. 2016. CyVerse Data Commons. http://datacommons.cyverse.org/browse/iplant/home/shared/commons_repo/curated/VertNet_Amphibia_Sep2016

description Compilation of digitized museum records of amphibians including locations, dates of collection, and some trait data.

4.78 78. Vertnet Fishes

name vertnet-fishes

reference *http://vertnet.org/resources/datatoolscode.html*

citation Bloom, D., Wieczorek J., Russell, L. (2016). VertNet_Fishes_Sept. 2016. CyVerse Data Commons. http://datacommons.cyverse.org/browse/iplant/home/shared/commons_repo/curated/VertNet_Fishes_Sep2016

description Compilation of digitized museum records of fishes including locations, dates of collection, and some trait data.

4.79 79. A database on the life history traits of the Northwest European flora

name plant-life-hist-eu

reference *http://www.uni-oldenburg.de/en/biology/landeco/research/projects/leda/*

citation KLEYER, M., BEKKER, R.M., KNEVEL, I.C., BAKKER, J.P, THOMPSON, K., SONNENSCHEIN, M., POSCHLOD, P., VAN GROENENDAEL, J.M., KLIMES, L., KLIMESOVA, J., KLOTZ, S., RUSCH, G.M., HERMY, M., ADRIAENS, D., BOEDELTJE, G., BOSSUYT, B., DANNEMANN, A., ENDELS, P., GoeTZENBERGER, L., HODGSON, J.G., JACKEL, A-K., KueHN, I., KUNZMANN, D., OZINGA, W.A., RoeMERMANN, C., STADLER, M., SCHLEGELMILCH, J., STEENDAM, H.J., TACKENBERG, O., WILMANN, B., CORNELISSEN, J.H.C., ERIKSSON, O., GARNIER, E., PECO, B. (2008): The LEDA Traitbase: A database of life-history traits of Northwest European flora. Journal of Ecology 96: 1266-1274

description The LEDA Traitbase provides information on plant traits that describe three key features of plant dynamics: persistence, regeneration and dispersal.

4.80 80. Forest Inventory and Analysis

name forest-inventory-analysis

reference *http://fia.fs.fed.us/*

citation DATEOFDOWNLOAD. Forest Inventory and Analysis Database, St. Paul, MN: U.S. Department of Agriculture, Forest Service, Northern Research Station. [Available only on internet: http://apps.fs.fed.us/fiadb-downloads/datamart.html]

description WARNING: This dataset requires downloading many large files and will probably take several hours to finish installing.

4.81 81. PREDICTS Database

name predicts

reference *http://data.nhm.ac.uk/dataset/902f084d-ce3f-429f-a6a5-23162c73fdf7*

citation Lawrence N Hudson; Tim Newbold; Sara Contu; Samantha L L Hill et al. (2016). Dataset: The 2016 release of the PREDICTS database. http://dx.doi.org/10.5519/0066354

description A dataset of 3,250,404 measurements, collated from 26,114 sampling locations in 94 countries and representing 47,044 species.

4.82 82. Mammal Super Tree

name mammal-super-tree

reference *http://doi.org/10.1111/j.1461-0248.2009.01307.x*

citation Fritz, S. A., Bininda-Emonds, O. R. P. and Purvis, A. (2009), Geographical variation in predictors of mammalian extinction risk: big is bad, but only in the tropics. Ecology Letters, 12: 538-549. doi:10.1111/j.1461-0248.2009.01307.x

description Mammal Super Tree from Fritz, S.A., O.R.P Bininda-Emonds, and A. Purvis. 2009. Geographical variation in predictors of mammalian extinction risk: big is bad, but only in the tropics. Ecology Letters 12:538-549

4.83 83. Marine Predator and Prey Body Sizes - Barnes et al. 2008

name predator-prey-size-marine

reference predator-prey-size-marine's home link.

citation

C. Barnes, D. M. Bethea, R. D. Brodeur, J. Spitz, V. Ridoux, C. Pusineri, B. C. Chase, M. E. Hunsicker, F. Juanes, A. Kellermann, J. Lancaster, F. Menard, F.-X. Bard, P. Munk, J. K. Pinnegar, F. S. Scharf, R. A. Rountree, K. I. Stergiou, C. Sassa, A. Sabates, and S. Jennings. 2008. Predator and prey body sizes in marine food webs. Ecology 89:881.

description The data set contains relationships between predator and prey size which are needed to describe interactions of species and size classes in food webs.

4.84 84. vertnet:

name vertnet

reference *http://vertnet.org/resources/datatoolscode.html*

citation Not currently available

description

4.85 85. USA National Phenology Network

name NPN

reference *http://www.usanpn.org/results/data*

citation Schwartz, M. D., Ault, T. R., & J. L. Betancourt, 2012: Spring Onset Variations and Trends in the Continental USA: Past and Regional Assessment Using Temperature-Based Indices. International Journal of Climatology (published online, DOI: 10.1002/joc.3625).

description The data set was collected via Nature's Notebook phenology observation program (2009-present), and (2) Lilac and honeysuckle data (1955-present)

4.86 86. Commercial Fisheries Monthly Trade Data by Product, Country/Association

name fao-global-capture-product

reference *http://www.fao.org/fishery/statistics/global-capture-production/*

citation FAO. 2018. FAO yearbook. Fishery and Aquaculture Statistics 2016/FAO annuaire. Statistiques des pêches et de l'aquaculture 2016/FAO anuario. Estadísticas de pesca y acuicultura 2016. Rome/Roma. 104pp.

description Commercial Fisheries statistics provides a summary of commercial fisheries product data by individual country.

4.87 87. Global wood density database - Zanne et al. 2009

name wood-density

reference *http://datadryad.org/resource/doi:10.5061/dryad.234*

citation Chave J, Coomes DA, Jansen S, Lewis SL, Swenson NG, Zanne AE (2009) Towards a worldwide wood economics spectrum. Ecology Letters 12(4): 351-366. http://dx.doi.org/10.1111/j.1461-0248.2009.01285.x and Zanne AE, Lopez-Gonzalez G, Coomes DA, Ilic J, Jansen S, Lewis SL, Miller RB, Swenson NG, Wiemann MC, Chave J (2009) Data from: Towards a worldwide wood economics spectrum. Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.234

description A collection and collation of data on the major wood functional traits, including the largest wood density database to date (8412 taxa), mechanical strength measures and anatomical features, as well as clade-specific features such as secondary chemistry.

4.88 88. Amniote life History database

name amniote-life-hist

reference amniote-life-hist's home link.

citation Myhrvold, N.P., Baldridge, E., Chan, B., Sivam, D., Freeman, D.L. and Ernest, S.M., 2015. An amniote life-history database to perform comparative analyses with birds, mammals, and reptiles:Ecological Archives E096-269. Ecology, 96(11), pp.3109-000.

description Compilation of life history traits for birds, mammals, and reptiles.

4.89 89. Gulf of Maine intertidal density/cover (Petraitis et al. 2008)

name intertidal-abund-me

reference intertidal-abund-me's home link.

citation Peter S. Petraitis, Harrison Liu, and Erika C. Rhile. 2008. Densities and cover data for intertidal organisms in the Gulf of Maine, USA, from 2003 to 2007. Ecology 89:588.

description The data on densities and percent cover in the 60 experimental plots from 2003 to 2007 and to update data from 1996 to 2002 that are already published in Ecological Archives.Includes densities of mussels, herbivorous limpet, herbivorous snails, predatory snail, barnacle , fucoid algae and percent cover by mussels, barnacles, fucoids, and other sessile organisms.

4.90 90. PRISM Climate Data

name prism-climate

reference *http://prism.oregonstate.edu/*

citation Not currently available

description The PRISM data set represents climate observations from a wide range of monitoring networks, applies sophisticated quality control measures, and develops spatial climate datasets to reveal short- and long-term climate patterns.

4.91 91. Vertnet Reptiles

name vertnet-reptiles

reference *http://vertnet.org/resources/datatoolscode.html*

citation Bloom, D., Wieczorek J., Russell, L. (2016). VertNet_Reptilia_Sept. 2016. CyVerse Data Commons. http://datacommons.cyverse.org/browse/iplant/home/shared/commons_repo/curated/VertNet_Reptilia_Sep2016

description Compilation of digitized museum records of reptiles including locations, dates of collection, and some trait data.

4.92 92. 3D Elevation Program (3DEP) high-quality U.S. Geological Survey topographic data

name usgs-elevation

reference *https://pubs.er.usgs.gov/publication/fs20163022*

citation Lukas, Vicki, Stoker, J.M., 2016, 3D Elevation Program—Virtual USA in 3D: U.S. Geological Survey Fact Sheet 2016–3022, 1 p., http://dx.doi.org/10.3133/fs20163022.

description The U.S. Geological Survey (USGS) 3D Elevation Program (3DEP) uses lidar to create a virtual reality maps.

4.93 93. A Southern Ocean dietary database

name socean-diet-data

reference *https://figshare.com/articles/Full_Archive/3551304*

citation Ben Raymond, Michelle Marshall, Gabrielle Nevitt, Chris L. Gillies, John van den Hoff, Jonathan S. Stark, Marcel Losekoot, Eric J. Woehler, and Andrew J. Constable. 2011. A Southern Ocean dietary database. Ecology 92:1188.

description Diet-related data from published and unpublished data sets and studies

4.94 94. Tree growth, mortality, physical condition - Clark, 2006

name la-selva-trees

reference *https://doi.org/10.6084/m9.figshare.c.3299324.v1*

citation David B. Clark and Deborah A. Clark. 2006. Tree growth, mortality, physical condition, and microsite in an old-growth lowland tropical rain forest. Ecology 87:2132.

description The data set helps to examine the post-establishment ecology of 10 species of tropical wet forest trees selected to span a range of predicted life history patterns at the La Selva Biological Station in Costa Rica.

4.95 95. Tree demography in Western Ghats, India - Pelissier et al. 2011

name tree-demog-wghats

reference tree-demog-wghats's home link.

citation Raphael Pelissier, Jean-Pierre Pascal, N. Ayyappan, B. R. Ramesh, S. Aravajy, and S. R. Ramalingam. 2011. Twenty years tree demography in an undisturbed Dipterocarp permanent sample plot at Uppangala, Western Ghats of India. Ecology 92:1376.

description A data set on demography of trees monitored over 20 years in Uppangala permanent sample plot (UPSP).

4.96 96. Pantheria (Jones et al. 2009)

name pantheria

reference pantheria's home link.

citation Kate E. Jones, Jon Bielby, Marcel Cardillo, Susanne A. Fritz, Justin O'Dell, C. David L. Orme, Kamran Safi, Wes Sechrest, Elizabeth H. Boakes, Chris Carbone, Christina Connolly, Michael J. Cutts, Janine K. Foster, Richard Grenyer, Michael Habib, Christopher A. Plaster, Samantha A. Price, Elizabeth A. Rigby, Janna Rist, Amber Teacher, Olaf R. P. Bininda-Emonds, John L. Gittleman, Georgina M. Mace, and Andy Purvis. 2009. PanTHERIA:a species-level database of life history, ecology, and geography of extant and recently extinct mammals. Ecology 90:2648.

description PanTHERIA is a data set of multispecies trait data from diverse literature sources and also includes spatial databases of mammalian geographic ranges and global climatic and anthropogenic variables.

CHAPTER 5

Adding datasets to the Data Retriever

5.1 Script Creation

The Data Retriever uses a simple CLI for developing new dataset scripts. This allows users with no programming experience to quickly add most standard datasets to the Retriever by specifying the names and locations of the tables along with additional information about the configuration of the data. The script is saved as a JSON file, that follows the DataPackage standards.

To create a new script, try `retriever new_json`, which starts the CLI tool for new script creation.

`Required`

1. **name:** A one word name for the dataset

`Strongly recommended`

1. **title:** Give the name of the dataset
2. **description:** A brief description of the dataset of ~25 words.
3. **citation:** Give a citation if available
4. **homepage:** A reference to the data or the home page
5. **keywords:** Helps in classifying the type of data (i.e using Taxon, Data Type, Spatial Scale, etc.)

`Mandatory for any table added; Add Table? (y/N)`

1. **table-name:** Name of the table, URL to the table
2. **table-url:** Name of the table, URL to the table

5.2 Basic Scripts

The most basic scripts structure requires only some general metadata about the dataset,i.e., the shortname of the database and table, and the location of the table.

Example of a basic script, example.script

Creating script from the CLI

```
name (a short unique identifier; only lowercase letters and - allowed): example-mammal
title: Mammal Life History Database - Ernest, et al., 2003
description:
citation: S. K. Morgan Ernest. 2003. Life history characteristics of placental non-
↪volant mammals. Ecology 84:3402.
homepage (for the entire dataset):
keywords (separated by ';'): mammals ; compilation

Add Table? (y/N): y
table-name: species
table-url: http://esapubs.org/archive/ecol/E084/093/Mammal_lifehistories_v2.txt
missing values (separated by ';'):
replace_columns (separated by ';'):
delimiter:
do_not_bulk_insert (bool = True/False):
contains_pk (bool = True/False):
escape_single_quotes (bool = True/False):
escape_double_quotes (bool = True/False):
fixed_width (bool = True/False):
header_rows (int):
Enter columns [format = name, type, (optional) size]:


Add crosstab columns? (y,N): n

Add Table? (y/N): n
```

Created script

```
{
    "citation": "S. K. Morgan Ernest. 2003. Life history characteristics of placental␣
↪non-volant mammals. Ecology 84:3402.",
    "description": "",
    "homepage": "",
    "keywords": [
        "Mammals",
        "Compilation"
    ],
    "name": "example-mammal",
    "resources": [
        {
            "dialect": {},
            "name": "species",
            "schema": {
                "fields": []
            },
            "url": "http://esapubs.org/archive/ecol/E084/093/Mammal_lifehistories_v2.
↪txt"
        }
    ],
    "retriever": "True",
    "retriever_minimum_version": "2.0.dev",
    "title": "Mammal Life History Database - Ernest, et al., 2003"
    "urls": {
```

(continues on next page)

(continued from previous page)

```
        "species": "www.exampleurl.com"
    },
    "version": "1.0.0"
}
```

Explanation for the keys:

- `citation`: Citation for the dataset
- `description`: Description for the dataset
- `homepage`: Homepage or website where the data is hosted
- `keywords`: Keywords/tags for the dataset (for searching and classification)
- `name`: Shortname for the dataset. Unique, URL-identifiable
- `resources`: List of tables within the dataset
  - `dialect`: Metadata for retriever to process the table
    - `missingValues`: (Optional) List of strings which represents missing values in tables
    - `delimiter`: (Optional) a character which represent boundary between two separate value(ex. ',' in csv files)
    - `header_rows`: (Optional) number of header rows in table.
  - `name`: Name of the table
  - `schema`: List of the columns in the table
    - `fields`: (Optional-Recommended) List of columns and their types and (optional) size values
    - `ct_column`: (Optional) Cross-tab column with column names from dataset
  - `url`: URL of the table
- `retriever`: Auto generated tag for script identification
- `retriever_minimum_version`: Minimum version that supports this script
- `title`: Title/Name of the dataset
- `urls`: dictionary of table names and the respective urls
- `version`: "1.0.0"

5.3 Multiple Tables

A good example of data with multiple tables is Ecological Archives E091-124-D1, McGlinn et al. 2010. `plant-comp-ok` Vascular plant composition data. Since there are several csv files, we create a table for each of the files.

Assuming we want to call our dataset McGlinn2010, below is an example of the script that will handle this data

```
...
  "name": "McGlinn2010",
  "resources": [
      {
          "dialect": {},
          "name": "pres",
```

(continues on next page)

(continued from previous page)

```
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_pres.csv"
        },
        {
            "dialect": {},
            "name": "cover",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_cover.csv"
        },
        {
            "dialect": {},
            "name": "richness",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_rich.csv"
        },
        {
            "dialect": {},
            "name": "species",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_specodes.csv"
        },
        {
            "dialect": {},
            "name": "environment",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_env.csv"
        },
        {
            "dialect": {},
            "name": "climate",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E091/124/TGPP_clim.csv"
        }
    ],
    "retriever": "True",
    "retriever_minimum_version": "2.0.dev",
    "title": "Vascular plant composition - McGlinn, et al., 2010",
    "urls": {
        "climate": "http://esapubs.org/archive/ecol/E091/124/TGPP_clim.csv",
        "cover": "http://esapubs.org/archive/ecol/E091/124/TGPP_cover.csv",
        "environment": "http://esapubs.org/archive/ecol/E091/124/TGPP_env.csv",
        "pres": "http://esapubs.org/archive/ecol/E091/124/TGPP_pres.csv",
        "richness": "http://esapubs.org/archive/ecol/E091/124/TGPP_rich.csv",
        "species": "http://esapubs.org/archive/ecol/E091/124/TGPP_specodes.csv"
    }
    ...
```

5.4 Null Values

The Retriever can replace non-standard null values by providing a semi-colon separated list of those null values after the table in which the null values occur.

```
...
Table name: species
```

(continues on next page)

(continued from previous page)

```
Table URL: http://esapubs.org/archive/ecol/E084/093/Mammal_lifehistories_v2.txt
nulls (separated by ';'): -999 ; 'NA'
...
```

For example, the Adler et al. 2010. `mapped-plant-quads-ks` script uses -9999 to indicate null values.

```
...
      {
          "dialect": {},
          "name": "quadrat_info",
          "schema": {},
          "url": "http://esapubs.org/archive/ecol/E088/161/quadrat_info.csv"
      },
      {
          "dialect": {
              "missingValues": [
                  "NA"
              ]
          },
...
```

5.5 Headers

If the first row of a table is the headers then naming the columns will, be default, be handled automatically. If you want to rename an existing header row for some reason, e.g., it includes reserved keywords for a database management system, you can do so by adding a list of semi-colon separated column names, with the new columns provided after a comma for each such column.

```
...
Add Table? (y/N): y
Table name: species
Table URL: http://esapubs.org/archive/ecol/E091/124/TGPP_specodes.csv
replace_columns (separated by ';', with comma-separated values): jan, january ; feb,␣
↪february ; mar, march
...
```

The `mapped-plant-quads-ks` script for the Adler et al. 2007. dataset from Ecological Archives includes this functionality:

```
...
 "name": "mapped-plant-quads-ks",
  "resources": [
      {
          "dialect": {},
          "name": "main",
          "schema": {},
          "url": "http://esapubs.org/archive/ecol/E088/161/allrecords.csv"
      },
      {
          "dialect": {},
          "name": "quadrat_info",
          "schema": {},
          "url": "http://esapubs.org/archive/ecol/E088/161/quadrat_info.csv"
      },
```

(continues on next page)

(continued from previous page)

```
    {
        "dialect": {
            "missingValues": [
                "NA"
            ]
        },
        "name": "quadrat_inventory",
        "schema": {},
        "url": "http://esapubs.org/archive/ecol/E088/161/quadrat_inventory.csv"
    },
    {
        "dialect": {},
        "name": "species",
        "schema": {},
        "url": "http://esapubs.org/archive/ecol/E088/161/species_list.csv"
    },
    {
        "dialect": {
            "missingValues": [
                "NA"
            ],
            "replace_columns": [
                [
                    "jan",
                    "january"
                ],
                [
                    "feb",
                    "february"
                ],
                [
                    "mar",
                    "march"
                ],
                [
                    "apr",
                    "april"
                ],
                [
                    "jun",
                    "june"
                ],
                [
                    "jul",
                    "july"
                ],
                [
                    "aug",
                    "august"
                ],
                [
                    "sep",
                    "september"
                ],
                [
                    "oct",
                    "october"
```

(continues on next page)

(continued from previous page)

```
                    ],
                    [
                        "nov",
                        "november"
                    ],
                    [
                        "dec",
                        "december"
                    ]
                ]
            },
            "name": "monthly_temp",
            "schema": {},
            "url": "http://esapubs.org/archive/ecol/E088/161/monthly_temp.csv"
        },
    ...
```

5.6 Full control over column names and data types

By default the Retriever automatically detects both column names and data types, but you can also exercise complete control over the structure of the resulting database by adding column names and types.

It is recommended to describe the schema of the table while creating the JSON file. This enables processing of the data faster since column detection increases the processing time.

These values are stored in the `fields` array of the `schema` dict of the JSON script.

The `fields` value enables full control of the columns, which includes, renaming columns, skipping unwanted columns, mentioning primary key and combining columns.

The basic format for `fields` is as shown below:

```
...
Enter columns [format = name, type, (optional) size]:

count, int
name, char, 40
year, int
...
```

where `name` represents name of the column and `type` represents the type of data present in the column. The following can be used to describe the data type:

```
pk-auto: Auto generated primary key starting from 1
pk-[char,int,double]: primary key with data type
char: strings
int: integers
double:floats/decimals
ct-[int,double,char]:Cross tab data
skip: used to skip the column in database
```

`pk-auto` is used to create an additional column of type int which acts as a primary key with values starting from 1. While `pk-[char,int,double]` is used to make a primary key from existing columns of the table having data type of char/int/double.

The Smith et al. Masses of Mammals `mammal-masses` dataset script includes this type of functionality.

```
...
   "name": "mammal-masses",
  "resources": [
      {
          "dialect": {
              "missingValues": [
                  -999
              ],
              "header_rows": 0
          },
          "name": "MammalMasses",
          "schema": {
              "fields": [
                  {
                      "name": "record_id",
                      "type": "pk-auto"
                  },
                  {
                      "name": "continent",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "status",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "sporder",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "family",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "genus",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "species",
                      "size": "20",
                      "type": "char"
                  },
                  {
                      "name": "log_mass_g",
                      "type": "double"
                  },
                  {
                      "name": "comb_mass_g",
                      "type": "double"
                  },
                  {
                      "name": "reference",
```

(continues on next page)

(continued from previous page)

```
                    "type": "char"
                }
            ]
        },
        "url": "http://www.esapubs.org/Archive/ecol/E084/094/MOMv3.3.txt"
    }
  ],
  "retriever": "True",
  "retriever_minimum_version": "2.0.dev",
  "title": "Masses of Mammals (Smith et al. 2003)",
...
```

5.7 Restructuring cross-tab data

It is common in ecology to see data where the rows indicate one level of grouping (e.g., by site), the columns indicate another level of grouping (e.g., by species), and the values in each cell indicate the value for the group indicated by the row and column (e.g., the abundance of species x at site y). This is referred as cross-tab data and cannot be easily handled by database management systems, which are based on a one record per line structure. The Retriever can restructure this type of data into the appropriate form. In scripts this involves telling the retriever the name of the column to store the data in and the names of the columns to be restructured.

```
...
Add crosstab columns? (y,N): y
Crosstab column name: <name of column to store cross-tab data>
Enter names of crosstab column values (Press return after each name):

ct column 1
ct column 2
ct column 3
...
```

The Moral et al 2010 script. `mt-st-helens-veg` takes advantage of this functionality.

```
...
"name": "mt-st-helens-veg",
  "resources": [
      {
          "dialect": {
              "delimiter": ","
          },
          "name": "species_plot_year",
          "schema": {
              "ct_column": "species",
              "ct_names": [
                  "Abilas",
                  "Abipro",
                  "Achmil",
                  "Achocc",
                  "Agoaur",
                  "Agrexa",
                  "Agrpal",
                  "Agrsca",
                  "Alnvir",
                  "Anamar",
```

(continues on next page)

(continued from previous page)

```
                "Antmic",
                "Antros",
                "Aqifor",
                "Arcnev",
                "Arnlat",
                "Astled",
                "Athdis",
                "Blespi",
                "Brocar",
                "Brosit",
                "Carmer",
                "Carmic",
                "Carpac",
                "Carpay",
                "Carpha",
                "Carros",
                "Carspe",
                "Casmin",
                "Chaang",
                "Cirarv",
                "Cisumb",
                "Crycas",
                "Danint",
                "Descae",
                "Elyely",
                "Epiana",
                "Eriova",
                "Eripyr",
                "Fesocc",
                "Fravir",
                "Gencal",
                "Hiealb",
                "Hiegra",
                "Hyprad",
                "Junmer",
                "Junpar",
                "Juncom",
                "Leppun",
                "Lommar",
                "Luepec",
                "Luihyp",
                "Luplat",
                "Luplep",
                "Luzpar",
                "Maiste",
                "Pencar",
                "Pencon",
                "Penser",
                "Phahas",
                "Phlalp",
                "Phldif",
                "Phyemp",
                "Pincon",
                "Poasec",
                "Poldav",
                "Polmin",
                "Pollon",
```

(continues on next page)

(continued from previous page)

```
                    "Poljun",
                    "Popbal",
                    "Potarg",
                    "Psemen",
                    "Raccan",
                    "Rumace",
                    "Salsit",
                    "Saxfer",
                    "Senspp",
                    "Sibpro",
                    "Sorsit",
                    "Spiden",
                    "Trispi",
                    "Tsumer",
                    "Vacmem",
                    "Vervir",
                    "Vioadu",
                    "Xerten"
                ],
                "fields": [
                    {
                        "name": "record_id",
                        "type": "pk-auto"
                    },
                    {
                        "name": "plot_id_year",
                        "size": "20",
                        "type": "char"
                    },
                    {
                        "name": "plot_name",
                        "size": "4",
                        "type": "char"
                    },
                    {
                        "name": "plot_number",
                        "type": "int"
                    },
                    {
                        "name": "year",
                        "type": "int"
                    },
                    {
                        "name": "count",
                        "type": "ct-double"
                    }
                ]
            },
            "url": "http://esapubs.org/archive/ecol/E091/152/MSH_SPECIES_PLOT_YEAR.csv"
        },
...
```

5.8 Script Editing

Note: Any time a script gets updated, the minor version number must be incremented from within the script.

The JSON scripts created using the retriever CLI can also be edited using the CLI.

To edit a script, use the `retriever edit_json` command, followed by the script's shortname;

For example, editing the `mammal-life-hist` (Mammal Life History Database - Ernest, et al., 2003) dataset, the editing tool will ask a series a questions for each of the keys and values of the script, and act according to the input.

The tool describes the values you want to edit. In the script below the first keyword is citation, `citation ( <class 'str'> )` and it is of class string or expects a string.

```
dev@retriever:~$ retriever edit_json mammal-life-hist

  ->citation ( <class 'str'> ) :

  S. K. Morgan Ernest. 2003. Life history characteristics of placental non-volant␣
↪mammals. Ecology 84:3402

  Select one of the following for the key 'citation'

  1. Modify value
  2. Remove from script
  3. Continue (no changes)


  Your choice: 3

    ->homepage ( <class 'str'> ) :

    http://esapubs.org/archive/ecol/E084/093/


  Select one of the following for the key 'homepage':

  1. Modify value
  2. Remove from script
  3. Continue (no changes)


  Your choice: 3

    ->description ( <class 'str'> ) :

    The purpose of this data set was to compile general life history characteristics␣
↪for a variety of mammalian
    species to perform comparative life history analyses among different taxa and␣
↪different body size groups.


  Select one of the following for the key 'description':

  1. Modify value
  2. Remove from script
  3. Continue (no changes)


  Your choice: 3

    ->retriever_minimum_version ( <class 'str'> ) :
```

(continues on next page)

(continued from previous page)

```
    2.0.dev


  Select one of the following for the key 'retriever_minimum_version':

  1. Modify value
  2. Remove from script
  3. Continue (no changes)


  Your choice: 3

    ->version ( <class 'str'> ) :

    1.1.0


  Select one of the following for the key 'version':

  1. Modify value
  2. Remove from script
  3. Continue (no changes)


  Your choice: 3

    ->resources ( <class 'list'> ) :

    {'dialect': {}, 'schema': {}, 'name': 'species', 'url': 'http://esapubs.org/
↪archive/ecol/E084/093/Mammal_lifehistories_v2.txt'}


  1 .  {'dialect': {}, 'schema': {}, 'name': 'species', 'url': 'http://esapubs.org/
↪archive/ecol/E084/093/Mammal_lifehistories_v2.txt'}

  Edit this dict in 'resources'? (y/N): n
  Select one of the following for the key 'resources':

  1. Add an item
  2. Delete an item
  3. Remove from script
  4. Continue (no changes)
  ...
```

CHAPTER 6

Developer's guide

6.1 Required Modules

If you are installing from source you will need a Python 2.6+ installation and the following modules:

```
setuptools
xlrd
Sphinx
```

6.2 Setting up servers

You need to install all the database infrastructures to enable local testing.

```
SQLite
MySQL
PostgreSQL
MSAccess
```

You will also need the following modules:

```
mysqldb (MySQL)
psycopg2 (PostgreSQL)
pypyodbc (MS Access)
```

6.3 Style Guide for Python Code

Run `pep8` on the given file to make sure the file follows the right style. In some cases we do tend to work outside the `pep8` requirements. The compromise on `pep8` may be a result of enforcing better code readability. In some cases `pep` shows errors for long lines, but that can be ignored.

```
pep8 pythonfile.py
```

6.4 Testing

Follow these instructions to run a complete set of tests for any branch Clone the branch you want to test.

Two ways of installing the program using the setup tools.

we can either install from source as

```
$ pip install . --upgrade or python setup.py install
```

or install in development mode.

```
$  python setup.py develop
```

For more about installing refer to the python setuptools documentation.

you can also install from Git.

```
# Local repository
pip install git+file:///path/to/your/git/repo #  test a PIP package located in a␣
↪local git repository
pip install git+file:///path/to/your/git/repo@branch  # checkout a specific branch by␣
↪adding @branch_name at the end

# Remote github repository
pip install git+git://github.com/myuser/myproject  #  package from a github repository
pip install git+git://github.com/myuser/myproject@my_branch # github repository␣
↪Specific branch
```

6.4.1 Running tests locally

Services Used

Check the services' home pages in case you have to add the same capabilities to your master branch.

```
Travis
AppVeyor
readthedocs
codecov
```

links Read The Docs, codecov, AppVeyor and Travis

To run the tests you will need to have all of the relevant database management systems and associated modules installed (see `Setting up servers`). Create the appropriate permissions for the tests to access the databases. You can do this by running the following commands in MySQL and PostgreSQL and creating the .pgpass file as described below:

```
MySQL
-----
mysql -e "CREATE USER 'travis'@'localhost';" -uroot
mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'travis'@'localhost';" -uroot
mysql -e "GRANT FILE ON *.* TO 'travis'@'localhost';" -uroot

PostgreSQL
```

(continues on next page)

(continued from previous page)

```
----------
psql -c "CREATE USER postgres WITH PASSWORD 'Password12!'"
psql -c 'CREATE DATABASE testdb'
psql -c 'GRANT ALL PRIVILEGES ON DATABASE testdb to postgres'

Create .pgpass in your home directory:
localhost:*:testdb:postgres:Password12!
```

To run tests we use pytest. From the source top level directory, run

```
$   py.test
```

To run tests on a specific test category add the path of the test module to the end of the py.test command:

```
$   py.test ./test/test_retriever.py
```

This will only run test_retriever.py

6.4.2 Continuous Integration

The main GitHub repository runs test on both the Travis (Linux) and AppVeyor (Windows) continuous integration platforms.

Pull requests submitted to the repository will automatically be tested using these systems and results reported in the `checks` section of the pull request page.

6.5 Create Release

6.5.1 Start

1. **Run the tests**. Seriously, do it now.
2. In the *master* branch update the version number in `setup.py` (if it hasn't already been bumped)
3. Run `python version.py` (this will update `version.txt`)
4. Update the version number in `retriever_installer.iss` (if it hasn't already been bumped)
5. Update `CHANGES.md` with major updates since last release
6. Commit changes
7. Add a tag with appropriate version number, e.g. ,`git tag -a v1.8.0 -m "Version 1.8.0"`
8. Push the release commit and the tag

```
git push upstream master
git push upstream --tags
```

6.5.2 Linux

Building the DEB package does not work using conda. If conda is your main Python change 'python' in 'build.sh' to '/usr/bin/python' or otherwise Adjust the path to use the system Python.

1. **Run the tests** (unless you just ran them on the same machine)

2. Checkout master
3. Run `build.sh`

6.5.3 Windows

1. **Run the tests**. This helps makes sure that the build environment is properly set up.
2. Checkout master
3. Run `sh build_win`

6.5.4 Mac

1. **Run the tests**. This helps makes sure that the build environment is properly set up.
2. Checkout master
3. Run `build_mac`
4. Install the retriever for verification. Reference http://www.data-retriever.org/download.html

6.5.5 Pypi

1. *sudo python setup.py sdist bdist_wheel upload*

6.5.6 Cleanup

1. Bump the version numbers as needed. The version number are located in the `setup.py`, `retriever_installer.iss`, `version.txt` and `retriever/_version.py`

6.6 Mac OSX Build

Building the Retriever on OSX.

6.6.1 Python binaries

This build will allow you to successfully build the Mac App for distribution to other systems.

1. Install the Python 3 Installer (or Python 2 if you have a specific reason for doing so) from the Python download site.
2. Use pip to install any desired optional dependencies `pip install pymysql psycopg2 pyinstaller pytest` You will need all of these dependencies, for example pyinstaller, if you want to build the Mac App for distribution

6.6.2 Homebrew

Homebrew works great if you just want to install the Retriever from source on your own machine, but at least based on this recipe it does not support distribution of the Mac App to other versions of OS X (i.e., if you build the App on OS X 10.9 it will only run on 10.9)

1. Install Homebrew `ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)"`
2. Install Xcode
3. Install Python `brew install python`
4. Install the Xcode command line tools `xcode-select --install`
5. Make brew's Python the default `echo export PATH='usr/local/bin:$PATH' >> ~/.bash_profile`
6. Install xlrd via pip `pip install xlrd`. No `sudo` is necessary since we're using brew.
7. Clone the Retriever `git clone git@github.com:weecology/retriever.git`
8. Switch directories `cd retriever`
9. Standard install `pip install . --upgrade`

If you also want to install the dependencies for MySQL and PostgreSQL this can be done using a combination of homebrew and pip.

1. `brew install mysql`
2. Follow the instructions from `brew` for starting MySQL
3. `brew install postgresql`
4. Follow the instructions from `brew` for starting Postgres
5. `sudo pip install pymysql MySQL-python psycopg2`

`MySQL-python` should be installed in addition to `pymysql` for building the `.app` file since pymysql is not currently working properly in the `.app`.

6.6.3 Conda

- This hasn't been tested yet

6.7 Creating or Updating a Conda Release

To create or update a Conda Release, first fork the conda-forge retriever-feedstock repository.

Once forked, open a pull request to the retriever-feedstock repository. Your package will be tested on Windows, Mac and Linux.

When your pull request is merged, the package will be rebuilt and become automatically available on conda-forge.

All branches in the conda-forge/retriever-feedstock are created and uploaded immediately, so PRs should be based on branches in forks. Branches in the main repository shall be used to build distinct package versions only.

For producing a uniquely identifiable distribution:

- If the version of a package is not being incremented, then the build/number can be added or increased.

- If the version of a package is being incremented, then remember to return the build/number back to 0.

6.8 Documentation

We are using Sphinx and Read the Docs. for the documentation. Sphinx uses reStructuredText as its markup language. Source Code documentation is automatically included after committing to the master. Other documentation (not source code) files are added as new reStructuredText in the docs folder

In case you want to change the organization of the Documentation, please refer to Sphinx

Update Documentation

The documetation is automatically updated for changes with in modules. However, the documentation should be updated after addition of new modules in the engines or lib directory. Change to the docs directory and create a temporary directory, i.e. `source`. Run

```
cd  docs
mkdir source
sphinx-apidoc -f  -o ./source /Users/../retriever/
```

The `source` is the destination folder for the source rst files. `/Users/../retriever/` is the path to where the retriever source code is located. Copy the `.rst` files that you want to update to the docs direcotry, overwriting the old files. Make sure you check the changes and edit if necessary to ensure that only what is required is updated. Commit and push the new changes. Do not commit the temporary source directory.

Test Documentation locally

```
cd  docs  # go the docs directory
make html # Run

Note:
Do not commit the build directory after making html.
```

Read The Docs configuration

Configure read the docs (advanced settings) so that the source is first installed then docs are built. This is already set up but could be change if need be.

6.9 Collaborative Workflows with GitHub

Submiting issues

Categorize the issues based on labels. For example (Bug, Dataset Bug, Important, Feature Request and etc..) Explain the issue explicitly with all details, giving examples and logs where applicable.

Commits

From your local branch of retriever, commit to your origin. Once tests have passed you can then make a pull request to the retriever master (upstream) For each commit, add the issue number at the end of the description with the tag `fixes #[issue_number]`.

Example:

```
Add version number to postgres.py to enable tracking

```

(continues on next page)

(continued from previous page)

```
Skip a line and add more explanation if needed
fixes #3
```

Clean histroy

We try to make one commit for each issue. As you work on an issue, try adding all the commits into one general commit rather than several commits.

Use `git commit --amend` to add new changes to a branch.

Use `-f` flag to force pushing changes to the branch. `git push -f origin [branch_name]`

CHAPTER 7

Contributor Code of Conduct

7.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

7.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

Using welcoming and inclusive language Being respectful of differing viewpoints and experiences Gracefully accepting constructive criticism Focusing on what is best for the community Showing empathy towards other community members Examples of unacceptable behavior by participants include:

The use of sexualized language or imagery and unwelcome sexual attention or advances Trolling, insulting/derogatory comments, and personal or political attacks Public or private harassment Publishing others' private information, such as a physical or electronic address, without explicit permission Other conduct which could reasonably be considered inappropriate in a professional setting

7.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

7.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

7.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at ethan@weecology.org. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

7.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4.

CHAPTER 8

Retriever API

8.1 retriever package

8.1.1 Subpackages

retriever.engines package

Submodules

retriever.engines.csvengine module

class retriever.engines.csvengine.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for writing data to a CSV file.

abbreviation = 'csv'

auto_column_number = 0

create_db()
Override create_db since there is no database just a CSV file

create_table()
Create the table by creating an empty csv file

datatypes = {'auto': 'INTEGER', 'bigint': 'INTEGER', 'bool': 'INTEGER', 'char': 'T

disconnect()
Close the last file in the dataset

disconnect_files()
Close each file after being written

execute(*statement*, *commit=True*)
Write a line to the output file

executemany(*statement*, *values*, *commit=True*)
Write a line to the output file

format_insert_value(*value*, *datatype*)
Formats a value for an insert statement

get_connection()
Gets the db connection.

insert_limit = 1000

insert_statement(*values*)
Returns a comma delimited row of values

name = 'CSV'

required_opts = [('table_name', 'Format of table name', './{db}_{table}.csv')]

table_exists(*dbname*, *tablename*)
Check to see if the data file currently exists

table_names = []

to_csv(*sort=True*)
Export sorted version of CSV file

retriever.engines.download_only module

retriever.engines.download_only.**dummy_method**(*self*, **args*, ***kwargs*)

class retriever.engines.download_only.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for writing data to a CSV file.

abbreviation = 'download'

add_to_table(**args*, ***kwargs*)

auto_create_table(*table*, *url=None*, *filename=None*, *pk=None*)
Download the file if it doesn't exist

auto_get_datatypes(**args*, ***kwargs*)

auto_get_delimiter(**args*, ***kwargs*)

connect(**args*, ***kwargs*)

convert_data_type(**args*, ***kwargs*)

create_db(**args*, ***kwargs*)

create_db_statement(**args*, ***kwargs*)

create_table(**args*, ***kwargs*)

create_table_statement(**args*, ***kwargs*)

database_name(**args*, ***kwargs*)

disconnect(**args*, ***kwargs*)

drop_statement(**args*, ***kwargs*)

execute(**args*, ***kwargs*)

executemany(**args*, ***kwargs*)

extract_fixed_width(**args*, ***kwargs*)

extract_gz(**args*, ***kwargs*)

extract_tar(**args*, ***kwargs*)

extract_zip(**args*, ***kwargs*)

fetch_tables(**args*, ***kwargs*)

final_cleanup()
Copies downloaded files to desired directory

Copies the downloaded files into the chosen directory unless files with the same name already exist in the directory.

find_file(*filename*)
Checks for the given file and adds it to the list of all files

format_insert_value(**args*, ***kwargs*)

get_connection()
Gets the db connection.

get_ct_data(**args*, ***kwargs*)

get_ct_line_length(**args*, ***kwargs*)

get_cursor(**args*, ***kwargs*)

get_input(**args*, ***kwargs*)

insert_data_from_archive(**args*, ***kwargs*)

insert_data_from_file(**args*, ***kwargs*)

insert_data_from_url(*url*)
Insert data from a web resource

insert_raster(**args*, ***kwargs*)

insert_statement(**args*, ***kwargs*)

insert_vector(**args*, ***kwargs*)

load_data(**args*, ***kwargs*)

name = 'Download Only'

next(**args*, ***kwargs*)

register_files(*filenames*)
Identify a list of files to be moved by the download

When downloading archives with multiple files the engine needs to be informed of all of the file names so that it can move them.

required_opts = [('path', 'File path to copy data files', './'), ('subdir', 'Keep the

set_engine_encoding(**args*, ***kwargs*)

set_table_delimiter(**args*, ***kwargs*)

supported_raster(**args*, ***kwargs*)

table_exists(*dbname*, *tablename*)
Checks if the file to be downloaded already exists

table_name(**args*, ***kwargs*)

to_csv(**args*, ***kwargs*)

warning(**args*, ***kwargs*)

retriever.engines.jsonengine module

Engine for writing data to a JSON file

class retriever.engines.jsonengine.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for writing data to a CSV file.

abbreviation = 'json'

auto_column_number = 0

create_db()
Override create_db since there is no database just a JSON file

create_table()
Create the table by creating an empty json file

datatypes = {'auto': 'INTEGER', 'bigint': 'INTEGER', 'bool': 'INTEGER', 'char': 'T

disconnect()
Close out the JSON with a *n]}* and close the file.

Close all the file objects that have been created Re-write the files stripping off the last comma and then close with a *n]}*.

execute(*statement*, *commit=True*)
Write a line to the output file

executemany(*statement*, *values*, *commit=True*)
Write a line to the output file

format_insert_value(*value*, *datatype*)
Formats a value for an insert statement

get_connection()
Gets the db connection.

insert_limit = 1000

insert_statement(*values*)
Return SQL statement to insert a set of values.

name = 'JSON'

required_opts = [('table_name', 'Format of table name', './{db}_{table}.json')]

table_exists(*dbname*, *tablename*)
Check to see if the data file currently exists

table_names = []

to_csv(*sort=True*, *path=None*)
Export table from json engine to CSV file

retriever.engines.msaccess module

class retriever.engines.msaccess.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for Microsoft Access.

abbreviation = 'msaccess'

convert_data_type(*datatype*)
MS Access can't handle complex Decimal types

create_db()
MS Access doesn't create databases.

datatypes = {'auto': 'AUTOINCREMENT', 'bigint': 'INTEGER', 'bool': 'BIT', 'char':

drop_statement(*object_type*, *object_name*)
Returns a drop table or database SQL statement.

get_connection()
Gets the db connection.

insert_data_from_file(*filename*)
Perform a bulk insert.

insert_limit = 1000

instructions = 'Create a database in Microsoft Access, close Access,then \nselect your

name = 'Microsoft Access'

placeholder = '?'

required_opts = [('file', 'Enter the filename of your Access database', './access.mdb'

retriever.engines.mysql module

class retriever.engines.mysql.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for MySQL.

abbreviation = 'mysql'

create_db_statement()
Return SQL statement to create a database.

datatypes = {'auto': 'INT(5) NOT NULL AUTO_INCREMENT', 'bigint': 'BIGINT', 'bool':

get_connection()
Get db connection.

PyMySQL has changed the default encoding from latin1 to utf8mb4. https://github.com/PyMySQL/PyMySQL/pull/692/files For PyMySQL to work well on CI infrastructure, connect with the preferred charset

insert_data_from_file(*filename*)
Call MySQL "LOAD DATA LOCAL INFILE" statement to perform a bulk insert.

insert_limit = 1000

lookup_encoding()
Convert well known encoding to MySQL syntax

MySQL has a unique way of representing the encoding. For example, latin-1 becomes latin1 in MySQL. Please update the encoding lookup table if the required encoding is not present.

max_int = 4294967295

name = 'MySQL'

placeholder = '%s'

required_opts = [('user', 'Enter your MySQL username', 'root'), ('password', 'Enter yo

set_engine_encoding()
Set MySQL database encoding to match data encoding

table_exists(*dbname*, *tablename*)
Check to see if the given table exists.

retriever.engines.postgres module

class retriever.engines.postgres.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for PostgreSQL.

abbreviation = 'postgres'

auto_create_table(*table*, *url=None*, *filename=None*, *pk=None*)
Create a table automatically.

Overwrites the main Engine class. Identifies the type of table to create. For a Raster or vector (Gis) dataset, create the table from the contents downloaded from the url or from the contents in the filename. Otherwise, use the Engine function for a tabular table.

create_db()
Create Engine database.

create_db_statement()
In PostgreSQL, the equivalent of a SQL database is a schema.

create_table()
Create a table and commit.

PostgreSQL needs to commit operations individually. Enable PostGis extensions if a script has a non tabular table.

datatypes = {'auto': 'serial', 'bigint': 'bigint', 'bool': 'boolean', 'char': 'var

drop_statement(*objecttype*, *objectname*)
In PostgreSQL, the equivalent of a SQL database is a schema.

format_insert_value(*value*, *datatype*)
Format value for an insert statement.

get_connection()
Gets the db connection.

Please update the encoding lookup table if the required encoding is not present.

insert_data_from_file(*filename*)
Use PostgreSQL's "COPY FROM" statement to perform a bulk insert.

insert_limit = 1000

insert_raster(*path=None*, *srid=4326*)
Import Raster into Postgis Table Uses raster2pgsql -I -C -s <SRID> <PATH> <SCHEMA>.<DBTABLE> | psql -d <DATABASE> The sql processed by raster2pgsql is run as psql -U postgres -d <gisdb> -f <elev>.sql

insert_statement(*values*)
Return SQL statement to insert a set of values.

insert_vector(*path=None*, *srid=4326*)
Import Vector into Postgis Table

– Enable PostGIS (includes raster) CREATE EXTENSION postgis;

– Enable Topology CREATE EXTENSION postgis_topology;

– fuzzy matching needed for Tiger CREATE EXTENSION fuzzystrmatch;

– Enable US Tiger Geocoder CREATE EXTENSION postgis_tiger_geocoder; Uses shp2pgsql -I -s <SRID> <PATH/TO/SHAPEFILE> <SCHEMA>.<DBTABLE> | psql -U postgres -d <DBNAME>>>

The sql processed by shp2pgsql is run as psql -U postgres -d <DBNAME>>>

max_int = 2147483647

name = 'PostgreSQL'

placeholder = '%s'

required_opts = [('user', 'Enter your PostgreSQL username', 'postgres'), ('password',

spatial_support = True

supported_raster(*path*, *ext=None*)
Return the supported Gis raster files from the path

Update the extensions after testing if a given raster type is supported by raster2pgsql.

retriever.engines.sqlite module

class retriever.engines.sqlite.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for SQLite.

abbreviation = 'sqlite'

create_db()
Don't create database for SQLite

SQLite doesn't create databases. Each database is a file and needs a separate connection. This overloads‘create_db‘ to do nothing in this case.

datatypes = {'auto': ('INTEGER', 'AUTOINCREMENT'), 'bigint': 'INTEGER', 'bool': 'IN

fetch_tables(*dataset*, *table_names*)
Return sqlite dataset as list of pandas dataframe.

get_bulk_insert_statement()
Get insert statement for bulk inserts

This places ?'s instead of the actual values so that executemany() can operate as designed

get_connection()
Get db connection.

insert_data_from_file(*filename*)
Perform a high speed bulk insert

Checks to see if a given file can be bulk inserted, and if so loads it in chunks and inserts those chunks into the database using executemany.

insert_limit = 1000

name = 'SQLite'

placeholder = '?'

required_opts = [('file', 'Enter the filename of your SQLite database', './sqlite.db',

retriever.engines.xmlengine module

class retriever.engines.xmlengine.**engine**
Bases: *retriever.lib.engine.Engine*

Engine instance for writing data to a XML file.

abbreviation = 'xml'

auto_column_number = 0

create_db()
Override create_db since there is no database just an XML file.

create_table()
Create the table by creating an empty XML file.

datatypes = {'auto': 'INTEGER', 'bigint': 'INTEGER', 'bool': 'INTEGER', 'char': 'T

disconnect()
Close out the xml files

Close all the file objects that have been created Re-write the files stripping off the last comma and then close with a closing tag)

execute(*statement*, *commit=True*)
Write a line to the output file.

executemany(*statement*, *values*, *commit=True*)
Write a line to the output file.

format_insert_value(*value*, *datatype*)
Format value for an insert statement.

get_connection()
Get db connection.

insert_limit = 1000

insert_statement(*values*)
Create the insert statement.

Wrap each data value with column values(key) using _format_single_row <key> value </key>.

name = 'XML'

required_opts = [('table_name', 'Format of table name', './{db}_{table}.xml')]

table_names = []

to_csv(*sort=True*, *path=None*)
Export table from xml engine to CSV file.

retriever.lib package

Submodules

retriever.lib.cleanup module

class retriever.lib.cleanup.**Cleanup**(*function=<function no_cleanup>*, ***kwargs*)
Bases: future.types.newobject.newobject

This class represents a custom cleanup function and a dictionary of arguments to be passed to that function.

retriever.lib.cleanup.**correct_invalid_value**(*value*, *args*)
This cleanup function replaces missing value indicators with None.

retriever.lib.cleanup.**floatable**(*value*)
Check if a value can be converted to a float

retriever.lib.cleanup.**no_cleanup**(*value*, *args*)
Default cleanup function, returns the unchanged value.

retriever.lib.datapackage module

retriever.lib.datapackage.**clean_input**(*prompt=''*, *split_char=''*, *ignore_empty=False*, *dtype=None*)
Clean the user-input from the CLI before adding it.

retriever.lib.datapackage.**create_json**()
Creates datapackage.JSON script. http://specs.frictionlessdata.io/data-packages/#descriptor-datapackagejson
Takes input from user via command line.

Usage: retriever new_json

retriever.lib.datapackage.**delete_json**(*json_file*)
Delete the json file from the script write path's directories.

retriever.lib.datapackage.**edit_dict**(*obj*, *tabwidth=0*)
Recursive helper function for edit_json() to edit a datapackage.JSON script file.

retriever.lib.datapackage.**edit_json**(*json_file*)
Edit existing datapackage.JSON script.

Usage: retriever edit_json <script_name> Note: Name of script is the dataset name.

retriever.lib.datapackage.**get_contains_pk**(*dialect*)
Set contains_pk property.

retriever.lib.datapackage.**get_delimiter**(*dialect*)
Get the string delimiter for the dataset file(s).

retriever.lib.datapackage.**get_do_not_bulk_insert**(*dialect*)
Set do_not_bulk_insert property.

retriever.lib.datapackage.**get_fixed_width**(*dialect*)
Set fixed_width property.

retriever.lib.datapackage.**get_header_rows**(*dialect*)
Get number of rows considered as the header.

retriever.lib.datapackage.**get_nulls**(*dialect*)
Get list of strings that denote missing value in the dataset.

retriever.lib.datapackage.**get_replace_columns**(*dialect*)
Get the replace values for columns from the user.

retriever.lib.datapackage.**get_script_filename**(*shortname*)
Return the file name of a script.

File names have '_' while the script variable names have '-'.

retriever.lib.datapackage.**is_empty**(*val*)
Check if a variable is an empty string or an empty list.

retriever.lib.datasets module

retriever.lib.datasets.**dataset_licenses**()
Return set with all available licenses.

retriever.lib.datasets.**dataset_names**()
Return list of all available dataset names.

retriever.lib.datasets.**datasets**(*keywords=None*, *licenses=None*)
Search all datasets by keywords and licenses.

retriever.lib.datasets.**license**(*dataset*)
Get the license for a dataset.

retriever.lib.defaults module

retriever.lib.download module

retriever.lib.download.**download**(*dataset*, *path='./'*, *quiet=False*, *subdir=False*, *debug=False*)
Download scripts for retriever.

retriever.lib.dummy module

Dummy connection classes for connectionless engine instances

This module contains dummy classes required for non-db based children of the Engine class.

class retriever.lib.dummy.**DummyConnection**
Bases: object

close()

commit()

cursor()

rollback()

class retriever.lib.dummy.**DummyCursor**
Bases: *retriever.lib.dummy.DummyConnection*

retriever.lib.engine module

class retriever.lib.engine.**Engine**
Bases: future.types.newobject.newobject

A generic database system. Specific database platforms will inherit from this class.

add_to_table(*data_source*)
Adds data to a table from one or more lines specified in engine.table.source.

auto_create_table(*table*, *url=None*, *filename=None*, *pk=None*)
Create table automatically by analyzing a data source and predicting column names, data types, delimiter, etc.

auto_get_datatypes(*pk*, *source*, *columns*)
Determine data types for each column.

For string columns adds an additional 100 characters to the maximum observed value to provide extra space for cases where special characters are counted differently by different engines.

auto_get_delimiter(*header*)
Determine the delimiter.

Find out which of a set of common delimiters occurs most in the header line and use this as the delimiter.

connect(*force_reconnect=False*)
Create a connection.

connection
Create a connection.

convert_data_type(*datatype*)
Convert Retriever generic data types to database platform specific data types.

create_db()
Create a new database based on settings supplied in Database object engine.db.

create_db_statement()
Return SQL statement to create a database.

create_raw_data_dir(*path=None*)
Check to see if the archive directory exists and creates it if necessary.

create_table()
Create new database table based on settings supplied in Table object engine.table.

create_table_statement()
Return SQL statement to create a table.

cursor
Get db cursor.

database_name(*name=None*)
Return name of the database.

datatypes = []

db = None

debug = False

disconnect()
Disconnect a connection.

disconnect_files()
Files systems should override this method.

Enables commit per file object.

download_file(*url*, *filename*)
Download file to the raw data directory.

download_files_from_archive(*url*, *file_names=None*, *archive_type='zip'*, *keep_in_dir=False*, *archive_name=None*)
Download files from an archive into the raw data directory.

drop_statement(*object_type*, *object_name*)
Return drop table or database SQL statement.

execute(*statement*, *commit=True*)
Execute given statement.

executemany(*statement*, *values*, *commit=True*)
Execute given statement with multiple values.

extract_fixed_width(*line*)
Split line based on the fixed width, returns list of the values.

extract_gz(*archive_path*, *archivedir_write_path*, *file_name=None*, *open_archive_file=None*, *archive=None*)
Extract gz files.

Extracts a given file name or all the files in the gz.

extract_tar(*archive_path*, *archivedir_write_path*, *archive_type*, *file_name=None*)
Extract tar or tar.gz files.

Extracts a given file name or the file in the tar or tar.gz. # gzip archives can only contain a single file

extract_zip(*archive_path*, *archivedir_write_path*, *file_name=None*)
Extract zip files.

Extracts a given file name or the entire files in the archive.

fetch_tables(*table_names*)
This can be overriden to return the tables of sqlite db as pandas data frame. Return False by default.

final_cleanup()
Close the database connection.

find_file(*filename*)
Check for an existing datafile.

format_data_dir()
Return correctly formatted raw data directory location.

format_filename(*filename*)
Return full path of a file in the archive directory.

format_insert_value(*value*, *datatype*)
Format a value for an insert statement based on data type.

Different data types need to be formated differently to be properly stored in database management systems. The correct formats are obtained by:

1. Removing extra enclosing quotes
2. Harmonizing null indicators
3. Cleaning up badly formatted integers

4. Obtaining consistent float representations of decimals

get_connection()
This method should be overridden by specific implementations of Engine.

get_ct_data(*lines*)
Create cross tab data.

get_ct_line_length(*lines*)
Returns the number of real lines for cross-tab data

get_cursor()
Get db cursor.

get_input()
Manually get user input for connection information when script is run from terminal.

insert_data_from_archive(*url*, *filenames*)
Insert data from files located in an online archive. This function extracts the file, inserts the data, and deletes the file if raw data archiving is not set.

insert_data_from_file(*filename*)
The default function to insert data from a file. This function simply inserts the data row by row. Database platforms with support for inserting bulk data from files can override this function.

insert_data_from_url(*url*)
Insert data from a web resource, such as a text file.

insert_raster(*path=None*, *srid=None*)
Base function for installing raster data from path

insert_statement(*values*)
Return SQL statement to insert a set of values.

insert_vector(*path=None*, *srid=None*)
Base function for installing vector data from path

instructions = 'Enter your database connection information:'

load_data(*filename*)
Generator returning lists of values from lines in a data file.

1. Works on both delimited (csv module) and fixed width data (extract_fixed_width) 2. Identifies the delimiter if not known 3. Removes extra line endings

name = ''

pkformat = '%s PRIMARY KEY %s '

placeholder = None

required_opts = []

script = None

script_table_registry = {}

set_engine_encoding()
Set up the encoding to be used.

set_table_delimiter(*file_path*)
Get the delimiter from the data file and set it.

spatial_support = False

supported_raster(*path*, *ext=None*)
"Spatial data is not currently supported for this database type or file format. PostgreSQL is currently the only supported output for spatial data.

table = None

table_name(*name=None*, *dbname=None*)
Return full table name.

to_csv(*sort=True*, *path=None*)

use_cache = True

warning(*warning*)
Create a warning message using the current script and table.

warnings = []

write_fileobject(*archivedir_write_path*, *file_name*, *file_obj=None*, *archive=None*, *open_object=False*)
Write a file object from a archive object to a given path

open_object flag helps up with zip files, open the zip and the file

retriever.lib.engine.**file_exists**(*path*)
Return true if a file exists and its size is greater than 0.

retriever.lib.engine.**filename_from_url**(*url*)
Extract and returns the filename from the url.

retriever.lib.engine.**gen_from_source**(*source*)
Return generator from a source tuple.

Source tuples are of the form (callable, args) where callable(*args) returns either a generator or another source tuple. This allows indefinite regeneration of data sources.

retriever.lib.engine.**reporthook**(*tqdm_inst*, *filename=None*)
tqdm wrapper to generate progress bar for urlretriever

retriever.lib.engine.**skip_rows**(*rows*, *source*)
Skip over the header lines by reading them before processing.

retriever.lib.engine_tools module

Data Retriever Tools

This module contains miscellaneous classes and functions used in Retriever scripts.

retriever.lib.engine_tools.**create_file**(*data*, *output='output_file'*)
Write lines to file from a list.

retriever.lib.engine_tools.**create_home_dir**()
Create Directory for retriever.

retriever.lib.engine_tools.**file_2list**(*input_file*)
Read in a csv file and return lines a list.

retriever.lib.engine_tools.**final_cleanup**(*engine*)
Perform final cleanup operations after all scripts have run.

retriever.lib.engine_tools.**get_script_version**()
This function gets the version number of the scripts and returns them in array form.

retriever.lib.engine_tools.**getmd5**(*data*, *data_type='lines'*)
Get MD5 of a data source.

retriever.lib.engine_tools.**json2csv**(*input_file*, *output_file=None*, *header_values=None*)
Convert Json file to CSV.

Function is used for only testing and can handle the file of the size.

retriever.lib.engine_tools.**name_matches**(*scripts*, *arg*)
Check for a match of the script in available scripts

if all, return the entire script list if the exact script is available, return that script if no exact script name detected, match the argument with keywords title and name of all scripts and return the closest matches

retriever.lib.engine_tools.**reset_retriever**(*scope='all'*, *ask_permission=True*)
Remove stored information on scripts, data, and connections.

retriever.lib.engine_tools.**set_proxy**()
Check for proxies and makes them available to urllib.

retriever.lib.engine_tools.**sort_csv**(*filename*)
Sort CSV rows minus the header and return the file.

Function is used for only testing and can handle the file of the size.

retriever.lib.engine_tools.**sort_file**(*file_path*)
Sort file by line and return the file.

Function is used for only testing and can handle the file of the size.

retriever.lib.engine_tools.**xml2csv**(*input_file*, *outputfile=None*, *header_values=None*, *row_tag='row'*)
Convert xml to csv.

Function is used for only testing and can handle the file of the size.

retriever.lib.excel module

Data Retriever Excel Functions

This module contains optional functions for importing data from Excel.

class retriever.lib.excel.**Excel**
Bases: future.types.newobject.newobject

static cell_value(*cell*)
Return string value of an excel spreadsheet cell.

static empty_cell(*cell*)
Test if excel cell is empty or contains only whitespace.

retriever.lib.get_opts module

retriever.lib.install module

retriever.lib.install.**install_csv**(*dataset*, *table_name='./{db}_{table}.csv'*, *debug=False*, *use_cache=True*)
Install datasets into csv.

retriever.lib.install.**install_json**(*dataset*, *table_name='./{db}_{table}.json'*, *debug=False*, *use_cache=True*)
Install datasets into json.

retriever.lib.install.**install_msaccess**(*dataset*, *file='./access.mdb'*, *table_name='[{db} {table}]'*, *debug=False*, *use_cache=True*)
Install datasets into msaccess.

retriever.lib.install.**install_mysql**(*dataset*, *user='root'*, *password=''*, *host='localhost'*, *port=3306*, *database_name='{db}'*, *table_name='{db}.{table}'*, *debug=False*, *use_cache=True*)
Install datasets into mysql.

retriever.lib.install.**install_postgres**(*dataset*, *user='postgres'*, *password=''*, *host='localhost'*, *port=5432*, *database='postgres'*, *database_name='{db}'*, *table_name='{db}.{table}'*, *bbox=[]*, *debug=False*, *use_cache=True*)
Install datasets into postgres.

retriever.lib.install.**install_sqlite**(*dataset*, *file='./sqlite.db'*, *table_name='{db}_{table}'*, *debug=False*, *use_cache=True*)
Install datasets into sqlite.

retriever.lib.install.**install_xml**(*dataset*, *table_name='./{db}_{table}.xml'*, *debug=False*, *use_cache=True*)
Install datasets into xml.

retriever.lib.load_json module

retriever.lib.load_json.**read_json**(*json_file*, *debug=False*)
Read Json dataset package files

Load each json and get the appropriate encoding for the dataset Reload the json using the encoding to ensure correct character sets

retriever.lib.models module

Data Retriever Data Model

This module contains basic class definitions for the Retriever platform.

retriever.lib.repository module

Checks the repository for updates.

retriever.lib.repository.**check_for_updates**(*quiet=False*)
Check for updates to datasets.

This updates the HOME_DIR scripts directory with the latest script versions

retriever.lib.scripts module

retriever.lib.scripts.**SCRIPT_LIST**()
Return Loaded scripts.

Ensure that only one instance of SCRIPTS is created.

class retriever.lib.scripts.**StoredScripts**

get_scripts()

retriever.lib.scripts.**check_retriever_minimum_version**(*module*)
Return true if a script's version number is greater than the retriever's version.

retriever.lib.scripts.**get_script**(*dataset*)
Return the script for a named dataset.

retriever.lib.scripts.**open_csvw**(*csv_file*, *encode=True*)
Open a csv writer forcing the use of Linux line endings on Windows.

Also sets dialect to 'excel' and escape characters to ''

retriever.lib.scripts.**open_fr**(*file_name*, *encoding='ISO-8859-1'*, *encode=True*)
Open file for reading respecting Python version and OS differences.

Sets newline to Linux line endings on Windows and Python 3 When encode=False does not set encoding on nix and Python 3 to keep as bytes

retriever.lib.scripts.**open_fw**(*file_name*, *encoding='ISO-8859-1'*, *encode=True*)
Open file for writing respecting Python version and OS differences.

Sets newline to Linux line endings on Python 3 When encode=False does not set encoding on nix and Python 3 to keep as bytes

retriever.lib.scripts.**reload_scripts**()
Load scripts from scripts directory and return list of modules.

retriever.lib.scripts.**to_str**(*object*, *object_encoding=<open file '<stdout>'*, *mode 'w'>*)
Convert a Python3 object to a string as in Python2.

Strings in Python3 are bytes.

retriever.lib.table module

class retriever.lib.table.**Dataset**(*name=None*, *url=None*)
Bases: future.types.newobject.newobject

Dataset generic properties

class retriever.lib.table.**RasterDataset**(*name=None*, *url=None*, *dataset_type='RasterDataset'*, ***kwargs*)
Bases: *retriever.lib.table.Dataset*

Raster table implementation

class retriever.lib.table.**TabularDataset**(*name=None*, *url=None*, *pk=True*, *contains_pk=False*, *delimiter=None*, *header_rows=1*, *column_names_row=1*, *fixed_width=False*, *cleanup=<retriever.lib.cleanup.Cleanup object>*, *record_id=0*, *columns=[]*, *replace_columns=[]*, *missingValues=None*, *cleaned_columns=False*, ***kwargs*)
Bases: *retriever.lib.table.Dataset*

Tabular database table.

add_dialect()
Initialize dialect table properties.

These include a table's null or missing values, the delimiter, the function to perform on missing values and any values in the dialect's dict.

add_schema()
Add a schema to the table object.

Define the data type for the columns in the table.

auto_get_columns(*header*)
Get column names from the header row.

Identifies the column names from the header row. Replaces database keywords with alternatives. Replaces special characters and spaces.

clean_column_name(*column_name*)
Clean column names using the expected sql guidelines remove leading whitespaces, replace sql key words, etc.

combine_on_delimiter(*line_as_list*)
Combine a list of values into a line of csv data.

get_column_datatypes()
Get set of column names for insert statements.

get_insert_columns(*join=True*, *create=False*)
Get column names for insert statements.

create should be set to *True* if the returned values are going to be used for creating a new table. It includes the *pk_auto* column if present. This column is not included by default because it is not used when generating insert statements for database management systems.

values_from_line(*line*)

class retriever.lib.table.**VectorDataset**(*name=None*, *url=None*, *dataset_type='VectorDataset'*, ***kwargs*)
Bases: *retriever.lib.table.Dataset*

Vector table implementation.

retriever.lib.templates module

Datasets are defined as scripts and have unique properties. The Module defines generic dataset properties and models the functions available for inheritance by the scripts or datasets.

class retriever.lib.templates.**BasicTextTemplate**(***kwargs*)
Bases: *retriever.lib.templates.Script*

Defines the pre processing required for scripts.

Scripts that need pre processing should use the download function from this class. Scripts that require extra tune up, should override this class.

download(*engine=None*, *debug=False*)
Defines the download processes for scripts that utilize the default pre processing steps provided by the retriever.

process_archived_data(*table_obj*, *url*)
Pre-process archived files.

Extract the files from the archived source based on the specifications. Either extact a single file or the entire files.

process_spatial_insert(*table_obj*)

process_tables(*table_obj*, *url*)

process_tabular_insert(*table_obj*, *url*)

class retriever.lib.templates.**HtmlTableTemplate**(*title=''*, *description=''*, *name=''*, *urls={}*, *tables={}*, *ref=''*, *public=True*, *addendum=None*, *citation='Not currently available'*, *licenses=[{'name': None}]*, *retriever_minimum_version=''*, *version=''*, *encoding=''*, *message=''*, ***kwargs*)

Bases: *retriever.lib.templates.Script*

Script template for parsing data in HTML tables.

class retriever.lib.templates.**Script**(*title=''*, *description=''*, *name=''*, *urls={}*, *tables={}*, *ref=''*, *public=True*, *addendum=None*, *citation='Not currently available'*, *licenses=[{'name': None}]*, *retriever_minimum_version=''*, *version=''*, *encoding=''*, *message=''*, ***kwargs*)

Bases: future.types.newobject.newobject

This class defines the properties of a generic dataset.

Each Dataset inherits attributes from this class to define it's Unique functionality.

checkengine(*engine=None*)
Returns the required engine instance

download(*engine=None*, *debug=False*)
Generic function to prepare for installation or download.

matches_terms(*terms*)

reference_url()

retriever.lib.tools module

retriever.lib.tools.**open_csvw**(*csv_file*, *encode=True*)
Open a csv writer forcing the use of Linux line endings on Windows.

Also sets dialect to 'excel' and escape characters to ''

retriever.lib.tools.**open_fr**(*file_name*, *encoding='ISO-8859-1'*, *encode=True*)
Open file for reading respecting Python version and OS differences.

Sets newline to Linux line endings on Windows and Python 3 When encode=False does not set encoding on nix and Python 3 to keep as bytes

retriever.lib.tools.**open_fw**(*file_name*, *encoding='ISO-8859-1'*, *encode=True*)
Open file for writing respecting Python version and OS differences.

Sets newline to Linux line endings on Python 3 When encode=False does not set encoding on nix and Python 3 to keep as bytes

retriever.lib.tools.**to_str**(*object*, *object_encoding=<open file '<stdout>'*, *mode 'w'>*)

retriever.lib.tools.**walk_relative_path**(*dir_name*)
Return relative paths of files in the directory

retriever.lib.warning module

class retriever.lib.warning.**Warning**(*location*, *warning*)
Bases: future.types.newobject.newobject

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

D

E

F

G

H

I

S

T

U

V

W

X