wikirepo Documentation

Release 1.0.0

andrewtavis

Dec 28, 2021

CONTENTS:

|quality|

Python based Wikidata framework for easy dataframe extraction

```
pip install wikirepo
```

```
git clone https://github.com/andrewtavis/wikirepo.git
cd wikirepo
python setup.py install
```

```
import wikirepo
```

CHAPTER

ONE

DATA - MODULES

The `data` directory comprises all functions for the wikirepo data querying process. Included modules are those for the query process itself and needed utility functions.

1.1 query

The `data.query` module provides a function that calls and combines data from Wikidata

Note: the purpose of this module is for a *wikirepo.data.query()* function call

Functions

- `wikirepo.data.query.query()`

1.2 data_utils

The `data.data_utils` module provides utility functions for querying data.

Functions

- `wikirepo.data.data_utils._get_fxn_idx()`
- `wikirepo.data.data_utils._get_dir_fxns_dict()`
- `wikirepo.data.data_utils._check_data_assertions()`
- `wikirepo.data.data_utils._get_max_workers()`
- `wikirepo.data.data_utils.incl_dir_idxs()`
- `wikirepo.data.data_utils.gen_base_df()`
- `wikirepo.data.data_utils.assign_to_column()`
- `wikirepo.data.data_utils.gen_base_and_assign_to_column()`
- `wikirepo.data.data_utils.assign_to_cols()`
- `wikirepo.data.data_utils.gen_base_and_assign_to_cols()`
- `wikirepo.data.data_utils.query_wd_prop()`
- `wikirepo.data.data_utils.query_repo_dir()`
- `wikirepo.data.data_utils.interp_by_subset()`
- `wikirepo.data.data_utils.sum_df_prop_vals()`

- wikirepo.data.data_utils.split_col_val_dates()
- wikirepo.data.data_utils.count_df_prop_vals()

1.3 lctn_utils

The data.lctn_utils module provides functions for querying locations.

Functions

- wikirepo.data.lctn_utils.lctn_to_qid_dict()
- wikirepo.data.lctn_utils.qid_to_lctn_dict()
- wikirepo.data.lctn_utils.incl_lctn_lbls()
- wikirepo.data.lctn_utils.incl_lctn_ids()
- wikirepo.data.lctn_utils.lctn_lbl_to_qid()
- wikirepo.data.lctn_utils.qid_tp_lctn_lbl()
- wikirepo.data.lctn_utils.depth_to_col_name()
- wikirepo.data.lctn_utils.depth_to_cols()
- wikirepo.data.lctn_utils.depth_to_qid_col_name()
- wikirepo.data.lctn_utils.depth_to_qid_cols()
- wikirepo.data.lctn_utils.find_qid_get_depth()
- wikirepo.data.lctn_utils.get_qids_at_depth()
- wikirepo.data.lctn_utils.iter_set_dict()
- wikirepo.data.lctn_utils.gen_lctns_dict()
- wikirepo.data.lctn_utils.derive_depth()
- wikirepo.data.lctn_utils.merge_lctn_dicts()
- wikirepo.data.lctn_utils.find_key_items()

Classes

- wikirepo.data.lctn_utils.LocationsDict

1.4 time_utils

The data.time_utils module provides utility functions for querying time information.

Functions

- wikirepo.data.time_utils.interval_to_col_name()
- wikirepo.data.time_utils.truncate_date()
- wikirepo.data.time_utils.truncate_date_col()
- wikirepo.data.time_utils.incl_intervals()
- wikirepo.data.time_utils.make_timespan()
- wikirepo.data.time_utils.latest_date()

- wikirepo.data.time_utils.earliest_date()
- wikirepo.data.time_utils.truncated_latest_date()
- wikirepo.data.time_utils.truncated_earliest_date()

1.5 wd_utils

The data.wd_utils module provides utility functions for accessing and storing Wikidata information.

Functions

- wikirepo.data.wd_utils.check_in_ents_dict()
- wikirepo.data.wd_utils.load_ent()
- wikirepo.data.wd_utils.is_wd_id()
- wikirepo.data.wd_utils.prop_has_many_entries()
- wikirepo.data.wd_utils.get_lbl()
- wikirepo.data.wd_utils.check_stget_propr_similarity()
- wikirepo.data.wd_utils.get_prop_id()
- wikirepo.data.wd_utils.get_prop_lbl()
- wikirepo.data.wd_utils.get_prop_val()
- wikirepo.data.wd_utils.prop_has_qualifiers()
- wikirepo.data.wd_utils.get_qualifiers()
- wikirepo.data.wd_utils.get_prop_qualifier_val()
- wikirepo.data.wd_utils.get_val()
- wikirepo.data.wd_utils.get_prop_t()
- wikirepo.data.wd_utils.get_prop_start_t()
- wikirepo.data.wd_utils.get_prop_end_t()
- wikirepo.data.wd_utils.format_t()
- wikirepo.data.wd_utils.get_formatted_prop_t()
- wikirepo.data.wd_utils.get_formatted_prop_start_t()
- wikirepo.data.wd_utils.get_formatted_prop_end_t()
- wikirepo.data.wd_utils.get_prop_timespan_intersection()
- wikirepo.data.wd_utils.get_formatted_prop_start_end_t()
- wikirepo.data.wd_utils.prop_start_end_to_timespan()
- wikirepo.data.wd_utils.get_prop_timespan()
- wikirepo.data.wd_utils.dir_to_topic_page()
- wikirepo.data.wd_utils.check_for_pid_sub_page()
- wikirepo.data.wd_utils.t_to_prop_val_dict()
- wikirepo.data.wd_utils.t_to_prop_val_dict_dict()

Classes

- wikirepo.data.wd_utils.EntitiesDict

1.6 upload (WIP)

wikirepo.data.upload will be the core of the eventual wikirepo upload feature. The goal is to record edits that a user makes to a previously queried or baseline dataframe such that these changes can then be pushed back to Wikidata. With the addition of Wikidata login credentials as a wikirepo feature (WIP), the unique information in the edited dataframe could then be uploaded to Wikidata for all to use.

The same process used to query information from Wikidata could be reversed for the upload process. Dataframe columns could be linked to their corresponding Wikidata properties, whether the time qualifiers are a point in time or spans using start time and end time could be derived through the defined variables in the module header, and other necessary qualifiers for proper data indexing could also be included. Source information could also be added in corresponding columns to the given property edits.

Pseudocode for how this process could function follows:

In the first example, changes are made to a *df.copy()* of a queried dataframe. pandas is then used to compare the new and original dataframes after the user has added information that they have access to.

```
import wikirepo
from wikirepo.data import lctn_utils, wd_utils
from datetime import date

credentials = wd_utils.login()

ents_dict = wd_utils.EntitiesDict()
country = "Country Name"
depth = 2
sub_lctns = True
timespan = (date(2000,1,1), date(2018,1,1))
interval = 'yearly'

lctns_dict = lctn_utils.gen_lctns_dict()

df = wikirepo.data.query()
df_copy = df.copy()

# The user checks for NaNs and adds data

df_edits = pd.concat([df, df_copy]).drop_duplicates(keep=False)

wikirepo.data.upload(df_edits, credentials)
```

In the next example *data.data_utils.gen_base_df* is used to create a dataframe with dimensions that match a time series that the user has access to. The data is then added to the column that corresponds to the property to which it should be added. Source information could further be added via a structured dictionary generated for the user.

```
import wikirepo
from wikirepo.data import data_utils, wd_utils
from datetime import date

```

(continues on next page)

(continued from previous page)

```
credentials = wd_utils.login()

locations = "Country Name"
depth = 0
# The user defines the time parameters based on their data
timespan = (date(1995,1,2), date(2010,1,2)) # (first Monday, last Sunday)
interval = 'weekly'

base_df = data_utils.gen_base_df()
base_df['data'] = data_for_matching_time_series

source_data = wd_utils.gen_source_dict('Source Information')
base_df['data_source'] = [source_data] * len(base_df)

wikirepo.data.upload(base_df, credentials)
```

Put simply: a full featured wikirepo.data.upload function would realize the potential of a single read-write repository for all public information.

CHAPTER

TWO

DATA - PROPERTY DIRECTORIES

The `data` directory comprises all functions for the wikirepo data querying process. Included sub-directories containing those modules that link wikirepo queries to their Wikidata metadata.

2.1 climate

`data.climate` is a directory of modules connecting wikirepo to corresponding Wikidata features for climate properties.

Modules included in wikirepo/data/climate follow, with full details of needed property parameters being shown in the linked source codes:

2.2 demographic

`data.demographic` is a directory of modules connecting wikirepo to corresponding Wikidata features for demographic properties.

Modules included in wikirepo/data/demographic follow, with full details of needed property parameters being shown in the linked source codes:

2.3 economic

`data.economic` is a directory of modules connecting wikirepo to corresponding Wikidata features for economic properties.

Modules included in wikirepo/data/economic follow, with full details of needed property parameters being shown in the linked source codes:

2.4 electoral_results

`data.electoral_results` is a directory of modules connecting wikirepo to corresponding Wikidata features for electoral_results properties.

Modules included in wikirepo/data/electoral_results follow, with full details of needed property parameters being shown in the linked source codes:

2.5 geographic

`data.geographic` is a directory of modules connecting wikirepo to corresponding Wikidata features for geographic properties.

Modules included in wikirepo/data/geographic follow, with full details of needed property parameters being shown in the linked source codes:

2.6 institutional

`data.institutional` is a directory of modules connecting wikirepo to corresponding Wikidata features for institutional properties.

Modules included in wikirepo/data/institutional follow, with full details of needed property parameters being shown in the linked source codes:

2.7 political

`data.political` is a directory of modules connecting wikirepo to corresponding Wikidata features for political properties.

Modules included in wikirepo/data/political follow, with full details of needed property parameters being shown in the linked source codes:

2.8 misc

`data.misc` is a directory of modules connecting wikirepo to corresponding Wikidata features for miscellaneous properties.

Modules included in wikirepo/data/misc follow, with full details of needed property parameters being shown in the linked source codes:

CHAPTER

THREE

MAPS (WIP)

wikirepo/maps is a further goal of the project, as it combines wikirepo's focus on easy to access open source data and quick high level analytics.

Query Maps

As in wikirepo.data.query, passing the `locations`, `depth`, `timespan` and `interval` arguments could access GeoJSON files stored on Wikidata, thus providing mapping files in parallel to the user's data. These files could then be leveraged using existing Python plotting libraries to provide detailed presentations of geographic analysis.

Upload Maps

Similar to the potential of adding statistics through wikirepo.data.upload, GeoJSON map files could also be uploaded to Wikidata using appropriate arguments. The potential exists for a myriad of variable maps given `locations`, `depth`, `timespan` and `interval` information that would allow all wikirepo users to get the exact mapping file that they need for their given task.

CHAPTER

FOUR

UTILS

The `utils` module provides utility functions for general operations.

Functions

- *wikirepo.utils._make_var_list()*
- *wikirepo.utils._return_given_type()*
- *wikirepo.utils.try_float()*
- *wikirepo.utils.round_if_int()*
- *wikirepo.utils.gen_list_of_lists()*
- *wikirepo.utils.check_str_similarity()*
- *wikirepo.utils.check_str_args()*

wikirepo.utils.**_make_var_list**(*var*)
Allows for a one line check for if a variable is a list.

wikirepo.utils.**_return_given_type**(*var*, *var_was_str*)
Allows for a one line return or list or string variables.

wikirepo.utils.**try_float**(*string*)
Checks if a string is a float.

wikirepo.utils.**round_if_int**(*val*)
Rounds off the decimal of a value if it is an integer float.

wikirepo.utils.**gen_list_of_lists**(*original_list*, *new_structure*)
Generates a list of lists with a given structure from a given list.

wikirepo.utils.**check_str_similarity**(*str_1*, *str_2*)
Checks the similarity of two strings.

wikirepo.utils.**check_str_args**(*arguments*, *valid_args*)
Checks whether a str argument is valid, and makes suggestions if not.

CHAPTER

FIVE

CONTRIBUTING TO WIKIREPO

Thank you for your consideration in contributing to this project!

Please take a moment to review this document in order to make the contribution process easy and effective for everyone involved.

Following these guidelines helps to communicate that you respect the time of the developers managing and developing this open source project. In return, and in accordance with this project's code of conduct, other contributors will reciprocate that respect in addressing your issue or assessing patches and features.

5.1 Using the issue tracker

The issue tracker for wikirepo is the preferred channel for *bug reports*, *features requests* and *submitting pull requests*.

5.2 Bug reports

A bug is a *demonstrable problem* that is caused by the code in the repository. Good bug reports are extremely helpful - thank you!

Guidelines for bug reports:

1. **Use the GitHub issue search** to check if the issue has already been reported.
2. **Check if the issue has been fixed** by trying to reproduce it using the latest `main` or development branch in the repository.
3. **Isolate the problem** to make sure that the code in the repository is *definitely* responsible for the issue.

Great Bug Reports tend to have:

- A quick summary
- Steps to reproduce
- What you expected would happen
- What actually happens
- Notes (why this might be happening, things tried that didn't work, etc)

Again, thank you for your time in reporting issues!

5.3 Feature requests

Feature requests are more than welcome! Please take a moment to find out whether your idea fits with the scope and aims of the project. When making a suggestion, provide as much detail and context as possible, and further make clear the degree to which you would like to contribute in its development.

5.4 Pull requests

Good pull requests - patches, improvements and new features - are a fantastic help. They should remain focused in scope and avoid containing unrelated commits. Note that all contributions to this project will be made under the specified license and should follow the coding indentation and style standards (contact us if unsure).

Please ask first before embarking on any significant pull request (implementing features, refactoring code, etc), otherwise you risk spending a lot of time working on something that the developers might not want to merge into the project. With that being said, major additions are very appreciated!

When making a contribution, adhering to the GitHub flow process is the best way to get your work merged:

1. Fork the repo, clone your fork, and configure the remotes:

```
# Clone your fork of the repo into the current directory
git clone https://github.com/<your-username>/<repo-name>
# Navigate to the newly cloned directory
cd <repo-name>
# Assign the original repo to a remote called "upstream"
git remote add upstream https://github.com/<upsteam-owner>/<repo-name>
```

2. If you cloned a while ago, get the latest changes from upstream:

```
git checkout <dev-branch>
git pull upstream <dev-branch>
```

3. Create a new topic branch (off the main project development branch) to contain your feature, change, or fix:

```
git checkout -b <topic-branch-name>
```

4. Commit your changes in logical chunks, and please try to adhere to Conventional Commits. Use Git's interactive rebase feature to tidy up your commits before making them public.

5. Locally merge (or rebase) the upstream development branch into your topic branch:

```
git pull --rebase upstream <dev-branch>
```

6. Push your topic branch up to your fork:

```
git push origin <topic-branch-name>
```

7. Open a Pull Request with a clear title and description.

Thank you in advance for your contributions!

5.5 License

5.6 Change log

CHAPTER

SIX

CHANGELOG

wikirepo tries to follow semantic versioning, a MAJOR.MINOR.PATCH version where increments are made of the:

- MAJOR version when we make incompatible API changes
- MINOR version when we add functionality in a backwards compatible manner
- PATCH version when we make backwards compatible bug fixes

CHAPTER

SEVEN

WIKIREPO 1.0.0 (DECEMBER 28TH, 2021)

- Release switches wikirepo over to semantic versioning and indicates that it is stable

CHAPTER

EIGHT

WIKIREPO 0.1.1.5 (MARCH 28TH, 2021)

Changes include:

- An src structure has been adopted for easier testing and to fix wheel distribution issues
- Code quality is now checked with Codacy
- Extensive code formatting to improve quality and style
- Fixes to vulnerabilities through exception use

CHAPTER

NINE

WIKIREPO 0.1.0 (FEB 23RD, 2021)

First stable release of wikirepo

Changes include:

- Full documentation of the package
- Virtual environment files
- Bug fixes
- Extensive testing of all modules with GH Actions and Codecov
- Code of conduct and contribution guidelines

CHAPTER

TEN

WIKIREPO 0.0.2 (DEC 8TH, 2020)

The minimum viable product of wikirepo:

- Users are able to query data from Wikidata given locations, depth, time_lvl, and timespan arguments
- String arguments are accepted for Earth, continents, countries and disputed territories
- Data for greater depths can be retrieved by creating a dictionary given initial starting locations and going to greater depths using the contains administrative territorial entity property
- Data is formatted and loaded into a pandas dataframe for further manipulation
- All available social science properties on Wikidata have had modules created for them
- Estimated load times and progress are given
- The project's scope and general roadmap have been defined and detailed in the README

CHAPTER

ELEVEN

PROJECT INDICES

- genindex

INDEX

Symbols

C

G

R

T