System Installation (Deployment) Manual

The code for our project is separated into two git repositories: the waterway datasets and the interactive map user interface.

Waterway datasets

- **Location**: https://github.com/swin-waterways/datasets
- **Contents**: This repository contains the original waterway dataset files that are merged, as well as a script to merge them together.

Merging and updating datasets

`update_datasets.py` is the all-in-one Python script to merge datasets together, and update some datasets by downloading them.

- **Usage**: This script can be called from the command line (`python update_datasets.py`) or imported by another Python script.
  - *Note*: A virtualenv may be required when using Linux

Requirements

- Python 3.10 or later
- aiohttp library (`pip install aiohttp`)
- pandas library (`pip install pandas`)

Additional functions

The script contains some work-in-progress functions that are commented out. If there is a use for these functions, the code can be continued to improve upon.

Command line usage

To use the script from the command line, change into the parent folder of the repository, then run:

```
./update_datasets.py
```

The following options can be passed when calling the script from the command line:

-h, –help show this help message and exit

-bf BOM_CONFIG_FILE, –bom-config-file BOM_CONFIG_FILE

JSON file containing pybomwater configuration (default: datasets/bom.json)

-dsf DATASETS_FILE, –datasets-file DATASETS_FILE

JSON file with list of datasets (default: datasets/datasets.json)

-dwf DELWP_DATASETS_FILE, –delwp-datasets-file DELWP_DATASETS_FILE

JSON file with list of DELWP datasets (default: datasets/delwp_datasets.json)

-hf HEADERS_FILE, –headers-file HEADERS_FILE

JSON file with list of default URL headers (default: datasets/headers.json)

-i, –interpolate Interpolate missing data (default: False)

-of OUTPUT_FILE, –output-file OUTPUT_FILE

JSON file with list of output arguments (default: datasets/output.json)

-m, –metadata-only Only output site metadata (default: False)

-s {none,basin,location,measurement}, –split-level {none,basin,location,measurement}

Where to split the data into multiple CSV files (default: none)

-t {download,output-csv,output-json}, –tasks {download,output-csv,output-json}

List of tasks to run (default: None)

Python usage

To use the script from another Python session, make sure `update_datasets.py` is in the folder you are currently working in, then run:

```
import update_datasets
```

in your Python script or from the Python console.

The following functions can be used when importing the `update_datasets` module:

```
create(datasets, output_dir)
# Create directories if they do not exist
## datasets = imported JSON file with list of dataset
    locations
## output_dir = output files directory

async download_url(session, url, filename)
# Internal function to asynchronously download data from URL
# Used by download_urls
## session = aiohttp session
## url = url to download from
## filename = file to save dataset to
# Returns: response

async download_urls(datasets, headers)
# Download datasets from URLs in datasets.json
## datasets = imported JSON file with list of datasets to
    download
## headers = imported JSON file with dictionary of headers to
    set when downloading files

format_time(dfs)
# Format Time column in a list of datasets
## dfs = list of DataFrames for which to format time
# Returns: list of DataFrames

merge(datasets)
# Merge other datasets (unused)
## datasets = imported JSON file with list of datasets to
    merge
# Returns: DataFrame

merge_delwp(delwp_datasets, interpolate=False,
    metadata_only=False, split_level='none')
# Merge DELWP datasets
## delwp_datasets = imported JSON file with dictionaries
    containing parameters and basin names
## interpolate = interpolate missing data (default: False)
## metadata_only = only output site metadata (default: False)
## split_level = where to split the data into multiple
    DataFrames (default: 'none')
    # options: 'none', 'basin', 'location', 'measurement'
# Returns: datasets: list of DataFrames, metadata_df:
    DataFrame

output_csv(df, output_file, quiet=False)
# Output to one CSV file
## df = DataFrame to be outputted
```

```
## output_file = file to output CSV data to
## quiet = suppress console output (default: False)

output_csv_files(dfs, output_dir)
# Output to multiple CSV files
## dfs = list of dictionaries with format
    {"name": "", "value": ""} # to output
    # name = file to output CSV data to
    # value = DataFrame to be outputted
## output_dir = directory to output files to

output_json(data, output_file)
# Output to one JSON file
## data = list to output as JSON
## output_file = file to write JSON data to

separate_time(df)
# Separate Time column from Date column
# Used by merge_delwp
## df = DataFrame with Date column
# Returns: DataFrame
```

Folders

The list of folders in this repository include:

- **bom**: Datasets from the BOM
- **datasets**: JSON configuration files for `update_datasets.py`
- **delwp**: Datasets from the DELWP (now DECCA)
- **geofabric**: Shapefiles representing river basins and river catchment areas
  - *Not used but could be useful for further research*
- **mdba**: Datasets from the MDBA
- **output**: Default output folder for `update_datasets.py`
- **wmis**: Newer datasets from DELWP, in a slightly better format

Datasets folder structure

The folder structure for datasets is as follows:

- **Source of data**: The organisation it came from
- **River/Basin**: The river/basin it is from
- **Location**: The location name
- ***Variable**: Optionally: The variable being recorded*

DELWP Output

The output from the `merge_delwp` function contains the following variables:

- **Rainfall**: The sum of the total rainfall for each hour at a point.
- **Flow/Height**: The mean (average) flow/height of the river for each hour at a point.

Date and Time values are outputted with the format **`%Y-%m-%d`** in the *`Date`* column and **`%H`** in the *`Time`* column

Interactive map user interface

- **Location**: https://github.com/swin-waterways/interactiveMapUI
- **Contents**: This repository contains the code and data required to run the interactive map user interface.

`create_app.py` is the Python script that runs the backend server for the map user interface.

- **Usage**: Run `python create_app.py` from the command line, then open the web address shown in the terminal output in your browser.
  - *Note*: A virtualenv may be required when using Linux

Requirements

- Python 3.9 or later
- flask library (`pip install flask`)
- folium library (`pip install folium`)
- matplotlib library (`pip install matplotlib`)
- numpy library (`pip install numpy`)
- pandas library (`pip install pandas`)