# Fall Army Worm LCS processing steps

There are a number of separate processing steps that use bash and python scripts. These are described below.

## Pre-requisites (as tested):

- Ubuntu 18.04
- Python 3.7.10
- Conda 4.10.3 (https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html)
- cdo toolkit (https://code.mpimet.mpg.de/projects/cdo) [To install `apt install cdo`]
- nco toolkit (http://nco.sourceforge.net/) [To install `apt install nco`]

A conda virtual environment can be used to set up most of the python pre-requisites using the `environment.yml` provided (`conda env create -f environment.yml`).

Notes:

- The first line of the `yml` file sets the environment name (`lcs`).
- To create conda environment `conda env create -f environment.yml`.
- To activate environment `conda activate lcs`.
- To deactivate environment `conda deactivate`.
- To remove conda environment `conda remove --name lcs`

Total disk space for demo: 2 GB (600MB)

Total processing time for demo: 30 minutes (including data download time)

## Repository Contents

```
faw
├── data
│   ├── demo_results
│   │   ├── NI_mainland_notempsfc_120hr_Nov-Apr.png
│   │   ├── NI_mainland_notempsfc_Nov-Apr.png
│   │   └── NI_mainland_sfctemp24_120hr_Nov-Apr.png
│   ├── aus_mainland_coast.png
│   ├── aus_qld.png
│   ├── coast_filled.png
│   ├── coast.png
│   ├── landmass.bmp
│   └── NI_coast.png
├── source
│   ├── base.py
│   ├── environment.yml
│   ├── faw_connections.py
│   ├── faw_era5_coast.py
│   ├── faw_era5_extract.sh
│   ├── faw_era5_lcs.py
│   ├── faw_heatmap.py
```

```
│   ├── faw_image_processing.py
│   ├── faw_met.py
│   ├── map_of_fle.py
│   ├── map.py
│   ├── metric_to_angular.py
│   └── met_utils.py
├── LICENSE
└── README.md
```

## Demo

Uploaded code had been setup to run a smaller subset of data than used in the manuscript (single month of data rather than 5 years). Instructions to reproduce the complete analysis are included in the main script of each processing step.

Note: All data is expected to be stored in the folder `~/faw_lcs` (created by step 1 if does not exist). This location can be altered by altering scripts appropriately.

## Processing steps

1. Download hourly reanalysis era5 met data from CDS (Climate Data Store)
    - 950mb wind velocities and temperature and 2m surface temperature
    - Source code: `faw_met.py`
    - Requirements:
        - CDS account (self-registration at https://cds.climate.copernicus.eu/user/register?destination=%2F%23!%2Fhome)
        - CDS API key (available from CDS User Profile). See instructions (https://cds.climate.copernicus.eu/api-how-to)
        - Python package: cdsapi
    - Years downloaded: 2015-2019
    - Adjustable settings:
        - `region`: [-10, 100, -70, 210] (sub domain centred on New Zealand and Eastern Australia)
        - `output_folder`: file destination. Will be created if does not exist.
    - Output: netcdf files (two 950mb file per year (one at 1degree resolution and 6 hour forecast, the other at 0.25 degree resolution and 1 hour forecast) and one T2 file per year (at 0.25 degree resolution and 1 hour forecast)) in output folder
    - Notes:
        - 1 degree, 6 hour forecasts are used to generate ftle field (step 3). Size approx. 60MB per year.
        - 0.25, 1 hour forecasts are used to define biological constraints on extracted lcs (step 7). Size of 950mb wind is approx. 5.3 GB per year. Size of surface temperature is approx 1.8 GB per year.
        - Downloaded 950mb temperatures are not been used in current implementation (but still included in downloaded data)
        - Region downloaded needs to be larger than domain to analyze.
        - Time to download demo data (dependent on internet connection and server) - approx. 15 minutes.

    - Download of full dataset will take approx. 1.5 hours per year.

2. Generate angular wind velocity files required for LCS

    - Source code: `faw_era5_extract.sh`, `metric_to_angular.py`
    - Requirements:
        - fbriol LCS package lagrangian (`conda install -c fbriol lagrangian`)
        - NCO toolkit (http://nco.sourceforge.net/) - Ubuntu 18.04 LTS installation `apt install nco` [v4.7.2-1]
        - CDO toolkit (https://code.mpimet.mpg.de/projects/cdo) - Ubuntu 18.04 LTS installation `apt install cdo`
    - Inputs:
        - era5 met data (step 1)
    - Adjustable settings:
        - Set `base_fldr` and `in_fldr` variables to folder containing downloaded met data.
        - Set `res` string variable to match downloaded resolution *100 of step 1 (e.g. res="100" for res=1.0)
        - Set `every_h` variable to match h of step 1 ie. `every_h=1`
        - Set `p` variable to match pressure extracted in step 1 (i.e. `p=950`)
        - Set `ndays` variable to number of days of year to process (`ndays=365`)
        - For demo, set `demo` variable to "true" to only processes year 2015.
        - For reproducability, set `demo` variable to "false" to process all data.
    - Process:
        - Interpolates 1deg 950mb wind data to 0.25deg (using `nco` and `cdo`)
        - Extracts yearly 950mb wind data data into separate netcdf files for each hour (1452 files per year)
        - Adds date attribute to U and V wind component variables, removes time dimension and reorders dimensions (to format required for fbriol lagrangian)
        - Generates a list_950mb.ini file as required by fbriol lagrangian
        - Executes fbriol/metric_to_angular.py to convert metric wind velocities to angular velocities.
    - Intermediary output:
        - `template.nc` and `template_renamed.nc` are temporary netcdf files used to define interpolated netcdf structure
        - `era5_metric/yyyy/extract_h.nc` netcdf files (yyyy = year, h=hour index 0-1459)
        - Note: File size approx 2.4GB per year
    - Output:
        - `era5_angular/yyyy/extract_h.nc` netcdf files (yyyy = year, h=hour index)
        - `list_950mb.ini` file (one per year)
        - Note: File size approx 2.4GB per year
    - Notes:
        - `metric_to_angular.py` has been copied from console scripts supplied with the lagrangian package.
        - if location of generated netcdf is changed after generation then paths in `list_950mb.ini` will need to be altered before they can be used in step 3.
        - intermediate output (era5_metric) can be removed.
        - Time to process: 5 minutes per year

3. Generate backward ftle fields and LCS images

   - Source code: `faw_era5_lcs.py`, `map_of_fle.py`, `map.py`
   - Package requirements: scipy, matplotlib, lagrangian, netCDF4, numpy
   - Inputs:
     - angular velocity netcdf (step 2)
   - Process:
     - Generate backward ftle fields using fbriol ftle executable
     - Generate png image highlighting LCS
     - Generate animation of png images
   - Output:
     - backward ftle field netcdf (1452 per year being one every 6 hours)
     - png image highlighting lcs (1452 per year)
     - avi animation of images (1 per year)
   - Notes:
     - `map_of_fle.py` has been copied from console scripts supplied with the lagrangian package.
     - location of angular velocity net cdf can be changed within process() method by setting `wind_velocities_config` parameter.
     - location of output can be set in process() argument `output_folder`
     - Approximately 5 minutes to generate 32 images (5 hours to generate 1 year of images)

4. Generate coastline image and landmass mask from LCS images

   - Source code: `faw_era5_coast.py`, `base.py`
   - Package requirements: PIL, cv2, skimage, matplotlib
   - Inputs:
     - single LCS png image (from step 3)
   - Process:
     - Run `python faw_era5_coast.py` to generate `coast.png`
     - Using a graphics editor, fill areas of `coast.png` representing land masses where LCS should be excluded (because insects would land rather than continue their flight) to generate a `landmass.bmp`. Fill colour should be red (RGB=#FF0000)
   - Outputs:
     - Coastlines of Australia and New Zealand (`coast.png`) at reduced image size (200x200)
     - Image of landmass areas (`landmass.bmp`) at reduced image size (200x200)
   - Negligible time to process or disk space required.
   - Example images are provided (`faw\data\`), copy to output folder (`~/faw_lcs`) to use.

5. Reprocess LCS images ready for line-following algorithm

   - Source code: `faw_image_processing.py`, `base.py`
   - Package requirements: PIL, cv2, skimage, matplotlib
   - Inputs:
     - LCS images (step 3)
     - landmass.bmp (step 4)
   - Outputs:
     - Folder `lcs_output` of 200x200 lcs images.

- To run (within lcs environment): `python faw_image_processing.py`
- Note: Time to process < 5 minutes per year, disk space approx 6MB per year.

6. Create images representing starting and ending locations for insect travel.

- Using `coast.png` image (from step 4) create two separate images; one representing starting coastline for insect travel; the second representing destination coastline.
- For paper, two distinct starting coastlines were created (mainland Australia (`aus_mainland.png`), and north-eastern Australia (`aus_qld.png`)) and a single destination coastline being North Island, New Zealand (`NI_coast.png`).
- Examples of these images are provided (`faw\data\`), copy to output folder (`~/faw_lcs`) to use.

7. Find connections between Australia and New Zealand and apply biological constraints

- Source code: `faw_connections.py`, `base.py`, `met_utils.py`
- Requirements:
- Inputs:
  - Reprocessed LCS images (step 5)
  - Source coastline image (step 6 - either `aus_mainland.png` or `aus_qld.png`)
  - Destination coastline image (step 6 - `NI_coast.png`)
  - 0.25 degree, 1 hour forecasts of 950mb wind and temperature
  - 0.25 degree, 1 hour forecasts of 2m surface temperature
- Process:
  - Use a line-following algorithm to find all distinct lcs paths originating near source coast line (Australia) and terminating in destination coastline (North Island, New Zealand)
  - For each connecting path determine length, time to traverse (based on wind speed at 950mb), and surface temperature at start.
- Outputs:
  - json file recording inputs (not used)
  - image highlighting distinct quickest paths connecting source (Australia) and destination (New Zealand) within each LCS image
  - csv (`connections.csv`) of FAW suitable pathways containing:
    - filename of connecting image
    - date_from, date_to of LCS image
    - length of quickest path in km
    - length of shortest path in km (not used)
    - ratio of shortest path length to straight-line distance (not used)
    - ratio of quickest path length to straight-line distance (not used)
    - time to traverse quickest path
    - time to traverse shortest path (not used)
    - average temperature along quickest path (not used)
    - max surface temperature at start of quickest path
- To run (within lcs environment): `python faw_connections.py`. Processing time and disk space required is negligible.

8. Generate heatmap of FAW suitable pathways

- Source code: `faw_heatmap.py`

- Package requirements: matplotlib, numpy
- Inputs:
  - Australia-NZ pathways (step 7 - images and `connections.csv`)
  - `coast.png` and `coast_filled.png`
- Steps:
  - Merge all yearly connections.csv files into a single file
  - Filter possible connections based on date and applying constraints on max travel time and minimum starting surface temperature.
  - Six combinations are analysed:
    - North-Eastern Australia, with no constraints, May-Oct time frame.
    - Mainland Australia, with no constraints, Nov-Apr time frame.
    - North-Eastern Austrlia, with max travel time of 120 hours, May-Oct time frame.
    - Mainland Australia, with max travel time of 120 hours, Nov-Apr time frame.
    - North-Eastern Australia, with max travel time of 120 hours and starting temperature less than 24 degrees, May-Oct time frame.
    - Mainland Australia, with max travel time of 120 hours and starting temperature less than 24 degrees, Nov-Apr time frame.
- To run (within lcs environment): `python faw_heatmap.py`. Processing time and disk space required is negligible.
- Note: For demo data (8 days in November 2015) only the Australia mainland combinations will have results (as North-Eastern Australis combinations are between May-Oct and demo data does not cover this period). Expected Mainland Australia outputs for demo data are provided (`faw/data/demo_results`).