Tutorial for `pylis`

`pylis` is a set a functions that I have been developing to make working with LIS output data easier. In this quick tutorial I give an overview of the functions you may find useful for your MSc thesis.

```
# enable loading in the necessary scripts
# use here the path where the pylis folder is located

import sys
sys.path.append("/data/leuven/340/vsc34049/PhD/scripts")
```

Reading in data

The function `lis_cube()` from the `pylis.readers` module reads in data that is stored in the `SURFACEMODEL` folder of the LIS output. The main inputs are:

- `lis_dir`: where is the LIS output stored? This is the path provided in the `lis.config` file, i.e., don't include `"SURFACEMODEL"`;
- `lis_input_file`: the LDT output with the LIS domain. We need this for the latitude and longitude (these variables are masked over water in the LIS output);
- `var`: the variable for which to read in the data cube. Make sure to use the correct suffix (`_inst` or `_tavg`);
- `start`: first date of the data cube;
- `end`: last data of the data cube.

Let's read in the soil moisture data cube.

```
from pylis import readers

# we will need the lis_input file to obtain the latitude and longitude
lis_input_file = "/dodrio/scratch/projects/2022_200/project_output/rsda/vsc34049/nu-wrf/OL/lis_

# read in soil moisture data cube over US
dc_sm = readers.lis_cube(
    lis_dir = "/dodrio/scratch/projects/2022_200/project_output/rsda/vsc34049/output/amandine_
    lis_input_file = lis_input_file,
    var = "SoilMoist_inst",
    start = "01/01/2002",
    end = "31/12/2019"
)
```

The resulting object is an `xr.DataArray` with in this case 4 dimensions: time, soil layer, and x- and y-direction of the grid. For a regular latitude-longitude grid, `x` and `y` will have a one-to-one correspondence with the longitude and latitude coordinates respectively, but this is not the case for this particular output, which is on a Lambert conformal grid. Hence, the `lat` and `lon` coordinates themselves are also two-dimensional on the `xy`-grid.

```
dc_sm
```

```
xarray.DataArray    (time: 6574, layer: 4, x: 159, y: 261)

nan nan nan nan nan nan ... 0.2801 0.2798 0.2798 0.2826 0.2845 0.2873
```

```
▾ Coordinates:
    lon      (x, y)          float32  -120.8 -120.6 ... -67.69 -67.46
    lat      (x, y)          float32  22.8 22.84 22.87 ... 47.3 47.25
    layer    (layer)           int64  1 2 3 4
    time     (time)   datetime64[ns]  2002-01-01 ... 2019-12-31
▸ Indexes:  (2)
▾ Attributes:
    description :  LIS model output
    variable :     SoilMoist_inst
```

You can always use `?` to find out more about a function.

```python
from pylis import readers
?readers.lis_cube
```

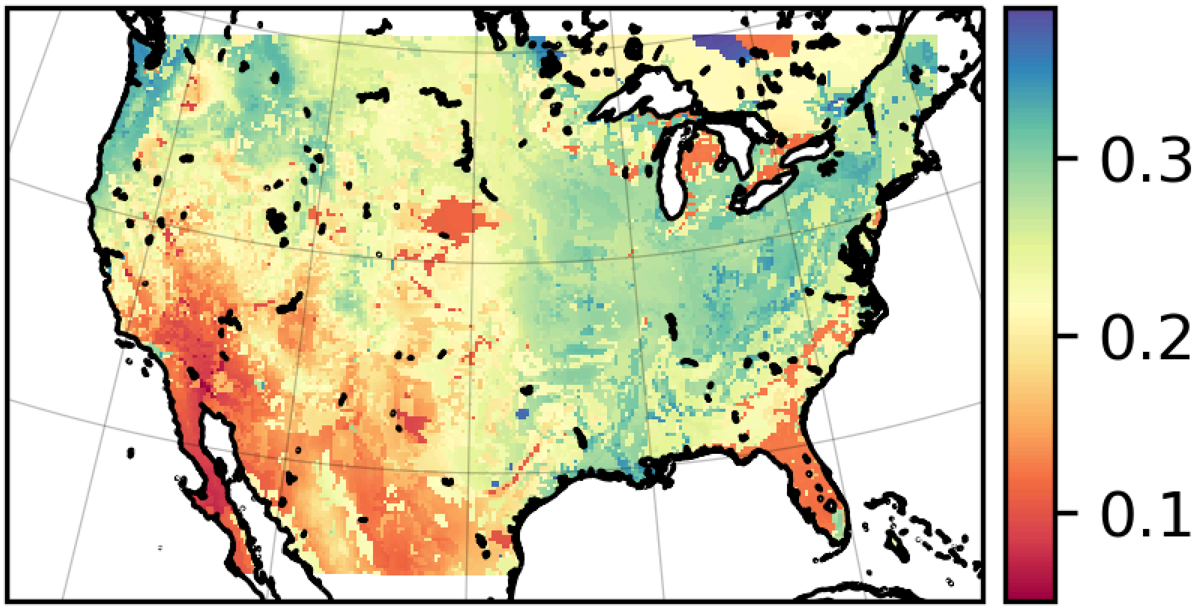
```
Signature:
readers.lis_cube(
    lis_dir,
    lis_input_file,
    var='SoilMoist_inst',
    h=0,
    d='01',
    start='01/04/2015',
    end='31/12/2020',
    freq='1D',
)
Docstring:
Read data cube of LIS model output

:param str lis_dir: parent directory of the LIS output
:param str lis_input_file: path to LIS input file containing the lat/lon information
:param str var: which variable to read in the data cube for
:param int h: UTC time at which LIS outputs (e.g. daily outputs at 12 UTC: h = 12)
:param str d: domain (in filename)
:param str start: start of the data cube (format "DD/MM/YYYY")
:param str end: end of the data cube (format "DD/MM/YYYY")
:param str freq: temporal resolution of the output
File:      /data/leuven/340/vsc34049/PhD/scripts/pylis/readers.py
Type:      function
```

Plotting data

The module `pylis.visualization` has a function `map_imshow` which takes input in the `xr.DataArray` format and creates a map. For example, we can plot the mean soil moisture of the surface layer as follows:

```python
from pylis import visualization as vis
import cartopy.crs as ccrs

vis.map_imshow(
    dc_sm.sel(layer = 1).mean(dim = "time"),
    projection = ccrs.LambertConformal(central_longitude = -98.3, central_latitude = 38.5),
    cmap = "Spectral",
    xmax = None,
    dpi = 300
)
```

```
/data/leuven/340/vsc34049/miniconda/envs/default_env/lib/python3.8/site-
packages/cartopy/mpl/geoaxes.py:1597: UserWarning: The input coordinates to pcolormesh are
interpreted as cell centers, but are not monotonically increasing or decreasing. This may lead
to incorrectly calculated cell edges, in which case, please supply explicit cell edges to
pcolormesh.
  X, Y, C, shading = self._pcolorargs('pcolormesh', *args,
```



Note that I had the `projection` argument here to the same one as the one of the LIS output data. For a regular lat-lon grid, you would choose `ccrs.PlateCarree()`. An overview of all projections is listed here: https://scitools.org.uk/cartopy/docs/latest/reference/projections.html.

The `xmax` option was set to None because the function defaults to a crop around Europe otherwise.

Simple calculations

There is a useful function to quickly compute the root-zone soil moisture from a soil moisture data cube with 4 layers:

```
from pylis.help import root_zone

dc_sm_rootzone = root_zone(dc_sm)
```

This data cube has now 'lost' its `layer` dimension:

```
dc_sm_rootzone
```

```
xarray.DataArray    (time: 6574, x: 159, y: 261)

nan nan nan nan nan nan ... 0.2657 0.2658 0.266 0.2677 0.2688 0.2705

▾ Coordinates:
    lon        (x, y)        float32  -120.8 -120.6 ... -67.69 -67.46
    lat        (x, y)        float32  22.8 22.84 22.87 ... 47.3 47.25
```

```
layer     ()       int64 1
time      (time)   datetime64[ns] 2002-01-01 ... 2019-12-31
```

▸ Indexes: (1)

▸ Attributes: (0)

Let’s read in the LAI as well:

```
from pylis import readers

# we will need the lis_input file to obtain the latitude and longitude
lis_input_file = "/dodrio/scratch/projects/2022_200/project_output/rsda/vsc34049/nu-wrf/OL/lis_

# read in soil moisture data cube over US
dc_lai = readers.lis_cube(
    lis_dir = "/dodrio/scratch/projects/2022_200/project_output/rsda/vsc34049/output/amandine_u
    lis_input_file = lis_input_file,
    var = "LAI_inst",
    start = "01/01/2002",
    end = "31/12/2019"
)
```

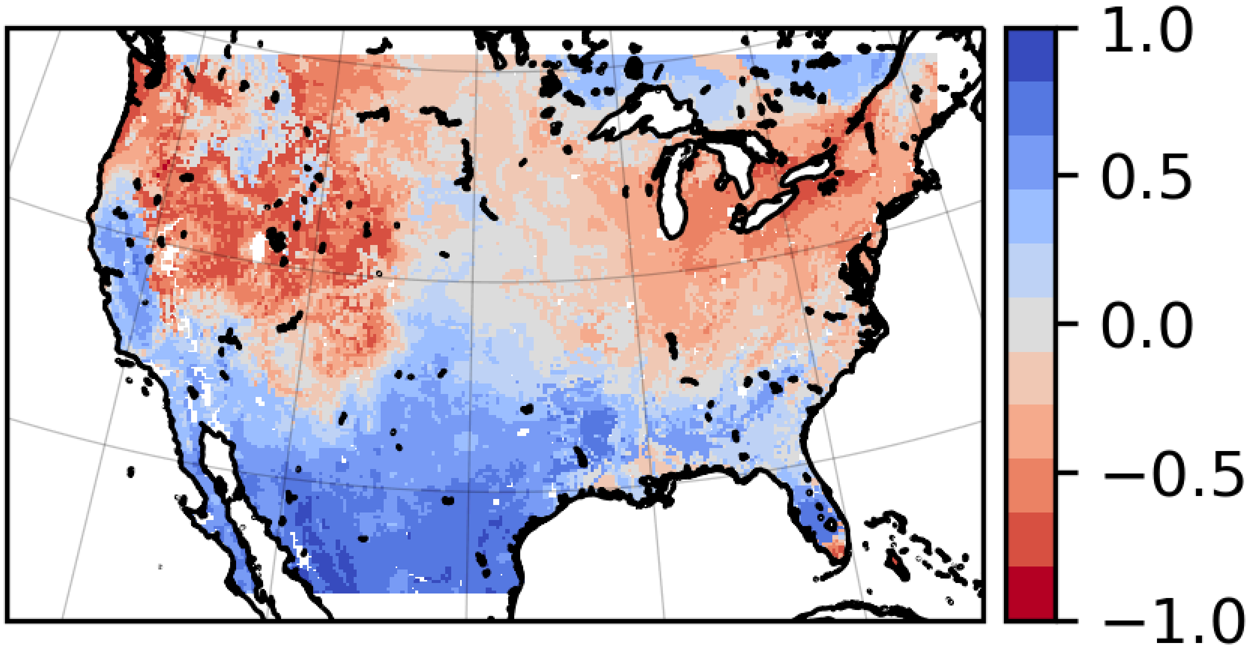
We can compute simple statistics such as the correlation between the root-zone soil moisture and the LAI through xarray’s built-in functionalities:

```
import xarray as xr

corr_lai_rzsm = xr.corr(dc_sm_rootzone, dc_lai, dim = "time")
```

```
/data/leuven/340/vsc34049/miniconda/envs/default_env/lib/python3.8/site-
packages/numpy/lib/nanfunctions.py:1878: RuntimeWarning: Degrees of freedom <= 0 for slice.
  var = nanvar(a, axis=axis, dtype=dtype, out=out, ddof=ddof,
/data/leuven/340/vsc34049/miniconda/envs/default_env/lib/python3.8/site-
packages/numpy/lib/nanfunctions.py:1878: RuntimeWarning: Degrees of freedom <= 0 for slice.
  var = nanvar(a, axis=axis, dtype=dtype, out=out, ddof=ddof,
```

The following map shows regions of positive correlation in blue and negative correlation in red.

```
from pylis import visualization as vis
import matplotlib.pyplot as plt
import cartopy.crs as ccrs

vis.map_imshow(
    corr_lai_rzsm,
    projection = ccrs.LambertConformal(central_longitude = -98.3, central_latitude = 38.5),
    cmap = plt.get_cmap("coolwarm_r", 11),
    vmin = -1, vmax = 1,
    xmax = None,
    dpi = 300
)
```



1.0
0.5
0.0
−0.5
−1.0