Patch identification and tracking routine - October 2017 by Laura Crews

This series of scripts identifies and tracks patches of model cells with properties meeting user-input criteria (for example, patches with water temperature above 5°C). The scripts should be called in the following order:

1.) patch_grid.m (Matlab): creates masks of cells that meet or do not meet patch criteria.

Input: default is an .nc file containing all days of data with properties interpolated onto a constant z-grid. Script will need to be adjusted to iterate through multiple .nc files or to depth interpolate.

User must set: Properties and thresholds to be used in patch identification

Output: Daily .csv files to be read by patch_identification.py

2.) patch_identificaion.py (Python): identifies and tracks patches.

Input: Daily .csv patch grid files created by patch_grid.m

User must set:

1. *Minimum area and maximum radius* for patch identification. As currently written, patches may never exceed the maximum radius. They must initially be larger than the minimum area, but can later shrink below this value.
2. *Tracking parameters*: Tracking works by limiting the change in patch area and center location from one frame to the next. MAX_TRAVEL is the maximum distance a patch can move as a percent of its radius. MAX_SIZECHANGE is the maximum percent change in area between frames. Increase MOVE_WEIGHT and SIZE_WEIGHT to assign more importance to minimizing movement or minimizing size change.

Important notes:

1. Patch grid file names will be printed to console when .csv files created by patch_grid.m are loaded. Review printout to check that they are loaded in the correct order and that no unexpected or hidden files are also loaded.
2. If earliest patch grid day does not correspond to the first day in the .nc file (you want to start later in the .nc file), adjust Line 160 (currentDay = fileNumber + 1). This line determines the day number saved to each patch. Note that you must add one because Python is 0-indexed and Matlab is 1-indexed (so fileNumber = 0 corresponds to .nc day = 1).

Other requirements: Must import patchClass.py (included in the code I sent you) to define a patch object and save information about it.

Output: Python list of patch objects. This information will be written to .csv files by save_patches.py

3.) save_patches.py (Python): saves information about patches into .csv files to be read into matlab.

Input: Python list of patch objects identified in patch_identification.py

User must set:

1. Minimum number of days a patch persists before its information is saved.
2. The size of the model domain (i.e. the dimensions of the patch grid files created by patch_grid.m).

Important note: Determining which cells are within the patch is slow. There's probably a more efficient way to do this. If you are only interested in patch location and size (can be saved quickly), comment out the code in Lines 51-60.

Output:

1. 'Summary' .csv files for each identified patch. Columns are day within the .nc file, center location, patch area, and patch velocity (distance moved from the previous frame).
2. 'Mask' .csv files for each identified patch for each day. List of coordinates of all points within the patch. This will be made into a mask by patch_data.m

4.) patch_data.m (Matlab): Creates a Matlab data structure with information about each patch. Uses .nc file to calculate additional properties of the patch (i.e. interior temperature).

Input: Patch 'summary' and 'mask' files created by save_patches.py

User must set:

1. Model cell grid size to convert patch area and velocity from units of cells^2 and cells/day to units of km^2 and km/day.
2. Additional patch properties to be calculated.

Optional scripts to be called:

1. testOverlap.m plots the patch_grid file used to identify patches with the identified patch contour overlaid. This ensures that patches have been identified properly. If these two contours do not align, it is likely due to a day-indexing error (need to adjust currentDay = fileNumber + 1 in patch_identification.py)
2. makeSampleMask.m creates a mask of a random subset of points within the patch. Using a smaller number of points makes the calculation of patch properties faster for any properties that require vertical calculations (i.e. the depth of maximum temperature within the water column).

Output: Matlab data structure in which each row contains all the data for a single patch. Columns are:

1. List of days the patch was active
2. List of center locations (x, y) and (lon, lat) on each day patch was active
3. List of patch areas on each day patch was active
4. List of patch velocities (distance moved from previous day) on each day patch was active
5. List of masks of cells within the patch on each day patch was active
6. Lists of averages of patch properties (properties selected by user) on each day patch was active

5.) patch_animation.m (matlab): Optional script that creates animation of patch tracking.

Input: patches data structure created by patch_data.m.

User must adjust: Property to be plotted (temperature, etc), other plotting values (bathymetry contours, color axis, etc).

Other information: Calls patchesActiveToday.m to create a mask of all patches active on a single day