Quick Employment tutorial

November 2018

1 Dependencies

1. python 3

   Any version of python 3 should work. Tests are run under 3.6.1

2. numpy

   `pip3 install numpy`

3. pandas

   `pip3 install pandas`

4. sklearn

   `pip3 install sklearn`

5. xgboost

   `pip3 install xgboost`

   or check this webpage https://xgboost.readthedocs.io/en/latest/build.html

6. QGIS

   https://www.qgis.org/en/site/forusers/download

   Please download version 2.18

2 Overview

QGIS is a geographic information system that manipulates maps. It is the software where we visualize our data and conduct map-related calculations. We can create or modify maps using QGIS. There are two types of maps: vector files (also called shapefiles, i.e. files with `.shp` ending) and raster files (`.tif` files etc). Vector files are consisted of points, lines or polygons and raster files are made up from pixels. Quick Employment code could take inputs both from shapefiles and raster files.

Xgboost is a widely used implementation of boosted decision trees. A quick introduction to boosted trees and xgboost could be found in the following links.
https://xgboost.readthedocs.io/en/latest/tutorials/model.html
https://xgboost.readthedocs.io/en/latest/tutorials/index.html

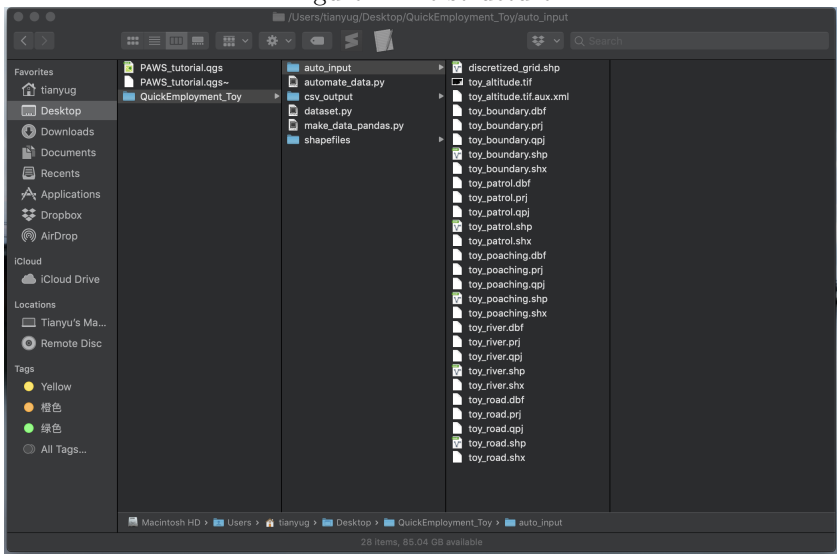
We use Xgboost to train our prediction model.

3 Quick Employment Code Tutorial

3.1 file structure

Please download and unzip `QuickEmployment_Toy.zip` file. `auto_input` is the folder containing all the input map files. `shapefiles` is the folder we store intermediate `.shp` files. `csv_output` is the folder we store intermediate `.csv` files. Note: `automate_data.py` and `make_data_pandas.py` have been modified from the github repo. Please use the code attached with this tutorial.

Figure 1: file structure

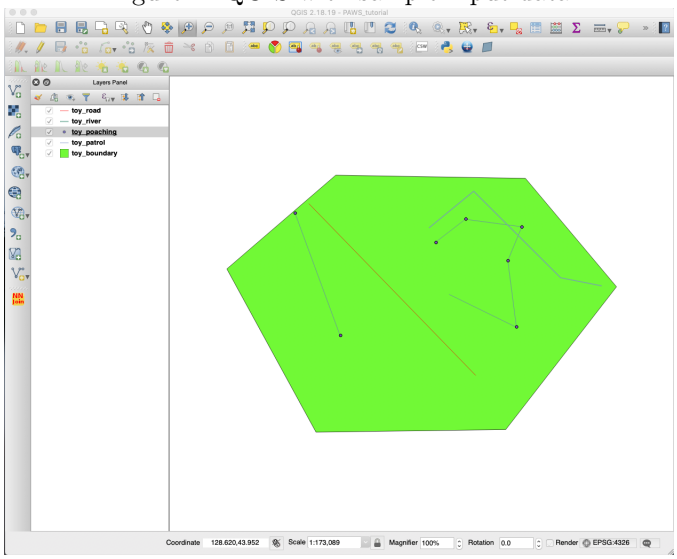


3.2 Quick Look at QGIS

After downloading QGIS, we can use it to take a look at the sample data and get ourselves familiar with the environment. Please start a new QGIS project. Go to `auto_input` folder, right click each .shp file and select `open with QGIS`. Note the color scheme might be different. The color of a layer could be changed by right clicking the layer in the left layer panel and selecting `properties`.

`toy_boundary` is the boundary of the toy wildlife conservation area.
`toy_patrol` is the past patrol route.
`toy_poaching` contains points of locations where poaching activities were found.
`toy_road` and `toy_river` are features we use to make predictions.
Note: Please note that `toy_poaching` points only exist in places that have been patrolled (covered by `toy_patrol`). Our algorithm relies on this fact to make predictions. (Modifications can be made to relax this restriction if we only have poaching data without patrolling data. Please contact us if you need this accommodation). Also note patrol data could be several lines (in this case) or a set of points.

Figure 2: QGIS with sample input data



3.3 Using QGIS to Preprocess Map Data

Now we are ready to start running the code. Open `automate_data.py` with a text editor and change the corresponding file paths in Module 0 to locations of corresponding folders on your computer (as in Figure 3). Note: We split the conservation area into rectangular cells and each cell is then treated as a data point. `gridWidth` and `gridHeight` are width and height of each grid cell. The units are CRS dependent. We recommend trying bigger values such as 0.1 as a starting point with fewer number of cells and then gradually decrease cell size to get an appropriate number of cells to work with. The cell size could be treated as a hyperparameter to tune for predictions. For each cell, we are able to compute "is" features (e.g. if a cell has river passing through) and "dist" features (e.g. distance from a cell to the nearest river). This is specified by `dist_layers` and `int_layers`. Please include patrol data and poaching data in

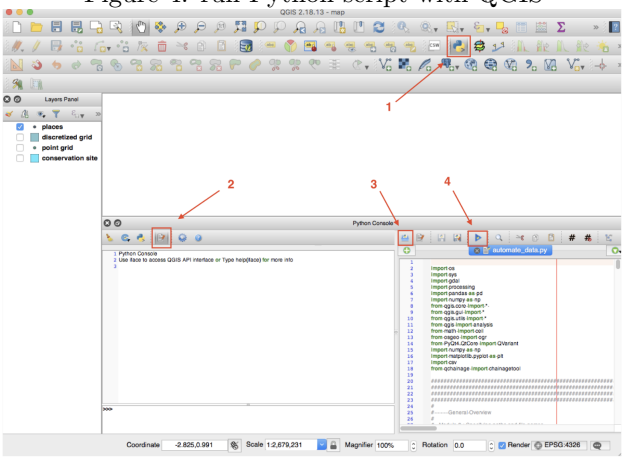
both lists.

Figure 3: change file paths in `automate_data.py`

```
#
#   Module 0 : Specifying paths and file names
#
#   Important : use shapefiles of the same CRS
#
#   The current entries are placeholders/examples
#

#might be in meters, might be roughly in longitude/latitude depending on CRS
gridWidth = 0.01
gridHeight = 0.01

#set lowerleft and upper right coordinates of discretization if necessary
#set to [-1, -1] and [-1, -1] if you don't want to customize the boundary
#and simply want to use the tightest square discretization
lowerleft = [-1, -1]
upperright = [-1, -1]

#specify the name of the boundary file, you do not need to specify the path to this file as
#long as all input shapefiles are in the same path described by input_path
boundary_file = "toy_boundary.shp"

#import paths
input_path = "/Users/tianyug/Desktop/QuickEmployment_Toy/auto_input/"
output_path_shapefiles = "/Users/tianyug/Desktop/QuickEmployment_Toy/shapefiles/"
output_path_excel = "/Users/tianyug/Desktop/QuickEmployment_Toy/csv_output/"

#specify layers we want to find distances from
dist_layers = ['toy_patrol.shp','toy_poaching.shp','toy_road.shp','toy_river.shp']

#specify layers we want to find intersections with
int_layers = ['toy_patrol.shp','toy_poaching.shp','toy_road.shp']

#specify raster files
raster_layers = ['toy_altitude.tif']
```

QGIS contains its own distribution of Python. We will use QGIS's Python to run `automate_data.py`.

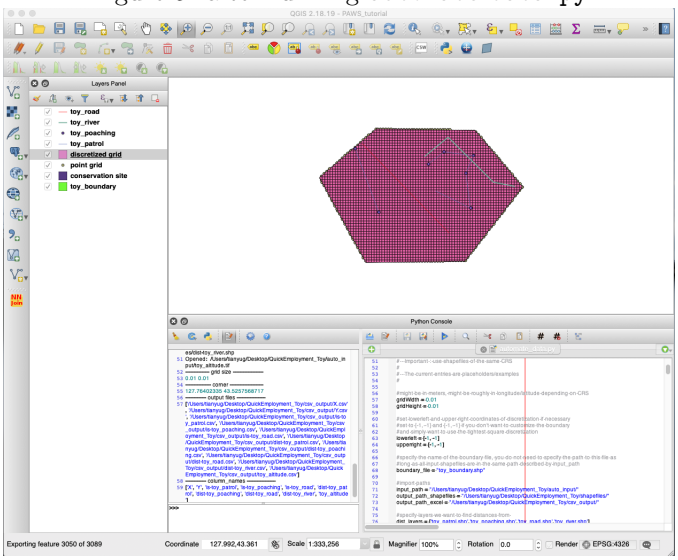
Figure 4: run Python script with QGIS



Please click 1 to open a python console. Click 2 to open QGIS's text editor. Then click 3 to load `automate_data.py` into QGIS. Finally click 4 to run `automate_data.py`. It may take hours for QGIS to run depending on the number of cells.

After QGIS is done, it should look like Figure 5

Figure 5: after running `automate_data.py`



3.4 Applying Xgboost to Make Predictions

After running `automate_data.py`, the python console should print out something like the following lines.

```
--------------- grid size -------------------
0.01 0.01
---------------- corner ---------------------
127.76402335 43.5257568717
------------- output files ------------------
['/Users/tianyug/Desktop/QuickEmployment_Toy/csv_output/X.csv', ...]
------------- column_names ------------------
['X', 'Y', 'is-toy_patrol', ...]
Finished
```

Now open `make_data_pandas.py` in a text editor. Copy the list under output files to `files`. Copy the list under column_names to `columns_names` in module -1, as shown in Figure 6.

Figure 6: module -1 for `make_data_pandas.py`

```
31  ##############################################################################
32
33  #
34  #   Module -1 : Processing output files from automate_data.py
35  #
36  #
37  #
38
39  files = ['/Users/tianyug/Desktop/QuickEmployment_Toy/csv_output/X.csv', '/Users/tianyug/Desktop/
40
41  column_names = ['X', 'Y', 'is-toy_patrol', 'is-toy_poaching', 'is-toy_road', 'dist-toy_patrol',
42
```

Run `make_data_pandas.py` by opening a new terminal window, cd to the corresponding directory, and running `python3 make_data_pandas.py`. Note: now we are running our own Python. The program would terminate at line 84 becuase of `sys.exit()` and generate a file called `final.csv`. `final.csv` contains all data and features generated from maps from `auto_input`.

Select features we would like to use and put column names of the corresponding features in `selected_features` (line 112). Copy grid size and corner to corresponding variables in module 0 as in Figure 7.

Figure 7: module 0 for `make_data_pandas.py`

```
92  #
93  #   Module 0 : Specifying paths and file names
94  #
95  #   The following are placeholders/examples
96  #
97
98  # name of excel sheet containing all the features and labels for conservation 1 and 2
99  fn1 = "final.csv"
100 fn2 = "final.csv"
101
102 # name of text file output for probabilistic predictions of
103 # each grid cell in conservations 1 and 2
104 qgis_file_in1 = "predictions1.txt"
105 qgis_file_in2 = "predictions2.txt"
106 # raster file of probabilistic predictions
107 qgis_file_out1 = "predictions_heatmap1.asc"
108 qgis_file_out2 = "predictions_heatmap2.asc"
109 # specify which features to use from final.csv feature spreadsheet
110 selected_features = ["is-toy_road",
111                      "normal-dist-toy_road",
112                      "normal-toy_altitude"
113                     ]
114 # specify which feature symbolizes where patrolling occurs
115 patrol = 'is-toy_patrol'
116 # specify which feature symbolizes where poaching occurs
117 poaching = 'is-toy_poaching'
118
119 # represents the coordinates of the left bottom corner for
120 # conservation site 1 (longitude and latitude if working with WGS84)
121 xcorner1 = 127.76402335
122 ycorner1 = 43.5257568717
123 # represents the coordinates of the left bottom corner for
124 # conservation site 2 (longitude and latitude if working with WGS84)
125 xcorner2 = 127.76402335
126 ycorner2 = 43.5257568717
127
128 # define the grid sizes (discretization levels) for each conservations ite
129 # which should match from the automate_data.py script
130 gridDim1 = 0.01
131 gridDim2 = 0.01
132
```

Now comment out line 84 in `make_data_pandas.py` (`sys.exit()`) to run the rest of the program. (Note: module -1 of `make_data_pandas.py` could be separated as a new program to avoid commenting and uncommenting line 84).

3.5 Checking the prediction results

After running `make_data_pandas.py`, the main folder should look like Figure 8.

Now open `predictions_heatmap1.asc` using QGIS. Right click the layer and open `properties` menu. Select your favorite color scheme for the prediction heatmap as shown in Figure 9 and Figure 11

After hitting `apply` and `ok`, the prediction heatmap should look like Figure 11

Figure 8: after running `make_data_pandas.py`

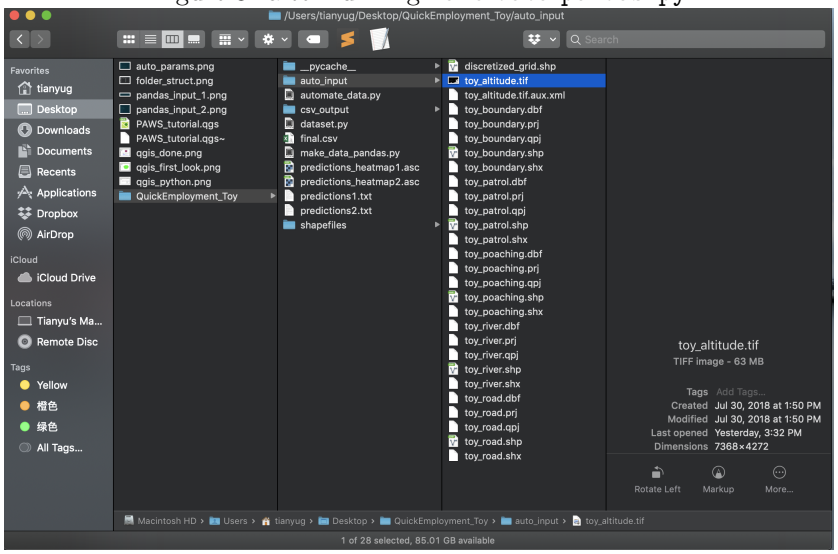
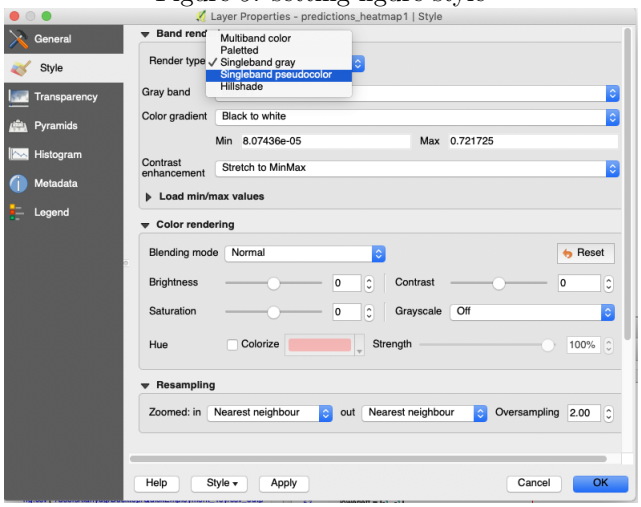


Figure 9: setting figure style

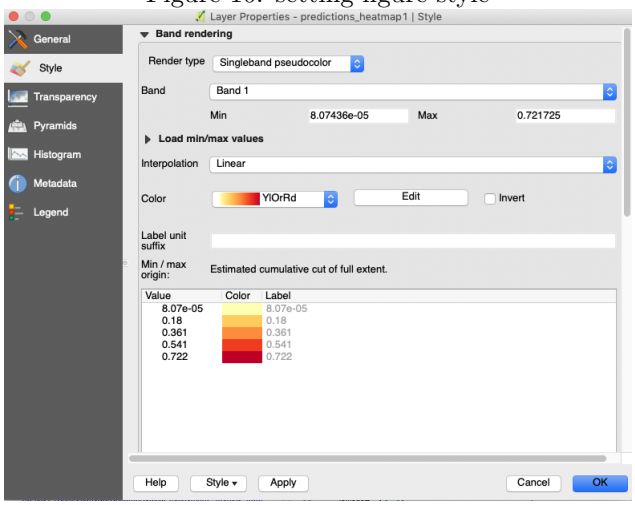


3.6 Some notes

3.6.1 Parameter tuning

There are many hyperparameters to tune for this model. We can tune cell size as mentioned before. We can also tune the Xgboost parameters in `make_data_pandas.py` in line 373 and 374 as shown below. More notes on parameter tuning can be found on Xgboost's website. We provide the functionality to conduct cross-

Figure 10: setting figure style



validation. Uncomment line 568 in `make_data_pandas.py` (`classify_familiar_trial()`) to run cross-validation. Metrics will be printed out in python console.

```
373: param = {'max_depth': 10, 'eta': 0.1, 'silent': 1, 'objective': 'binary:logistic'}
374: num_round = 1000
```

3.6.2 Transfer learning

You may have noticed, in Figure 7, module 0 in `make_data_pandas.py`, we have two sets of variables, i.e. `fn1`, `fn2`, etc. The purpose for this design is to support prediction for a different conservation site where no patrolling and poaching data are available, using the model we have trained using the known conservation site. If you do not need this feature, simply comment out line 572 (`prep_qgis(qgis_file_in2, qgis_file_out2, gridDim2, xcorner2, ycorner2, df_alldata2)`).

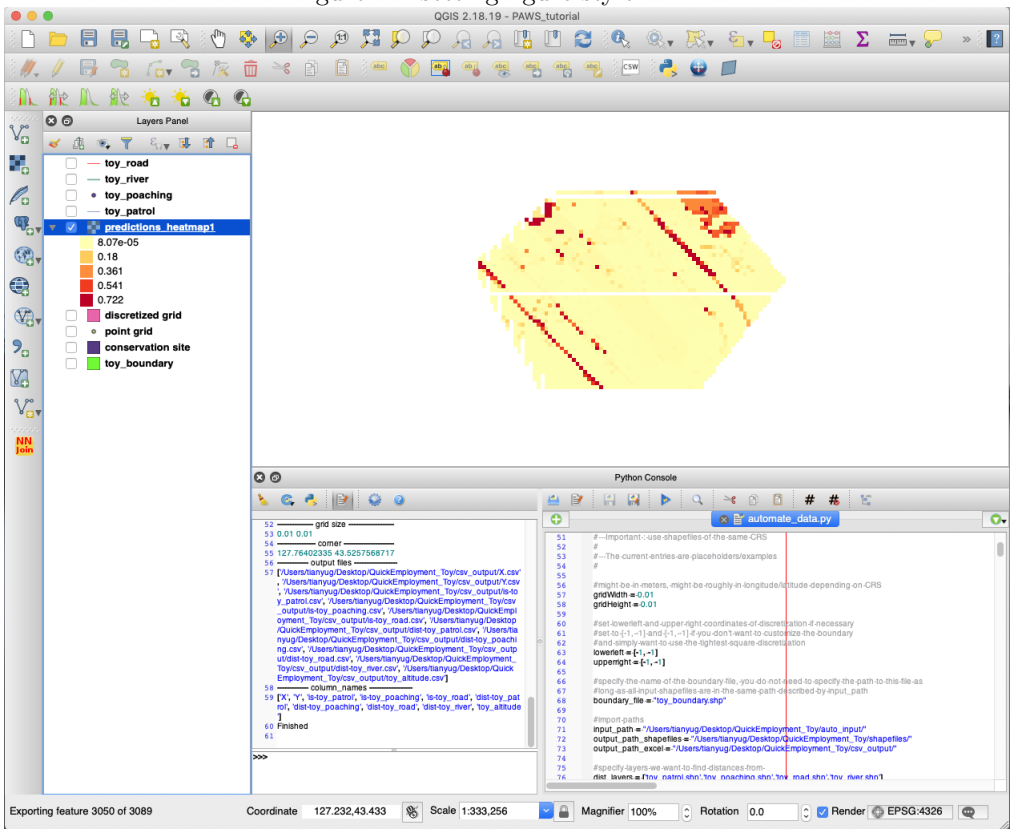
3.6.3 Some known problems of the code

In some rare cases, the is-features may not be able to be computed correctly and end up with every value to be 0. If this case happens, please contact us and we can work around this problem.
The prediction heatmap may have some areas without any value. This depends on the shape of the boundary of the conservation area. We are trying to fix this problem.

3.6.4 Our words

Thank you for your interests in the tool we developed! This code is still under development and you may encounter errors when following this tutorial. Please

Figure 11: setting figure style



do not hesitate to contact us if you need help. We will try our best to be as supportive as possible.