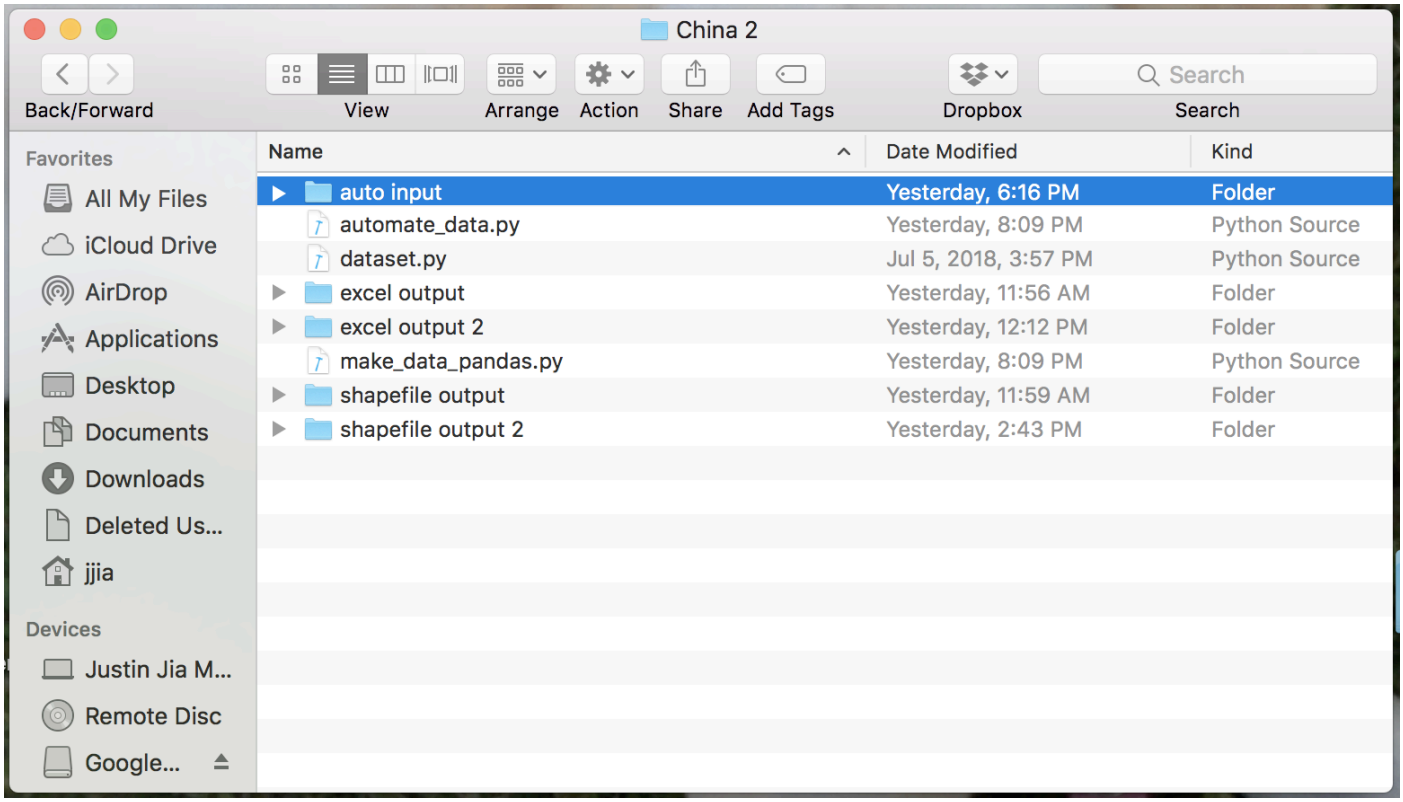
Deploying PAWS

0.INSTALL DEPENDENCIES. Several libraries for Python are required, in addition to the use of QGIS.

1.Getting your environment ready. Our code assumes you are working in a directory organized below. Download the Python files 'automate_data.py', 'dataset.py', and 'make_data_pandas.py' from our Github and create the other folders for now (naming is irrelevant as long it is kept consistent in code).



2.Getting the data. Users will benefit much more greatly with their own specialized, localized data. However, there are also several online sources that provide useful information.

- https://extract.bbbike.org - Contains easily accessibly and abundant worldwide information, conveniently obtainable by providing coordinates. We suggest this as a starting point.
- http://www.diva-gis.org/datadown - Another source of diverse files, however more often at a lower resolution
- http://sedac.ciesin.columbia.edu/data/set/groads-global-roads-open-access-v1 - Maintains animal density information, but also not at a high resolution
- https://guides.library.upenn.edu/globalgis - Directory to more public resources if the above are not sufficient.

3.Basic manipulation of the data. We offer some helpful, simple steps for using QGIS

3.1. DEM to slope raster layer. Raster-> 'Terrain Analysis' -> DEM. Specify the output directory and name, and you have obtained a slope layer from an altitude layer.

3.2. Breaking up some files. In our experience, there are shapefiles such as 'places.shp' from the first website given in Step 2 that is a larger subset of other desired features, such as residency levels ranging from hamlets and villages to cities and towns. In these cases, here are the steps we suggest.

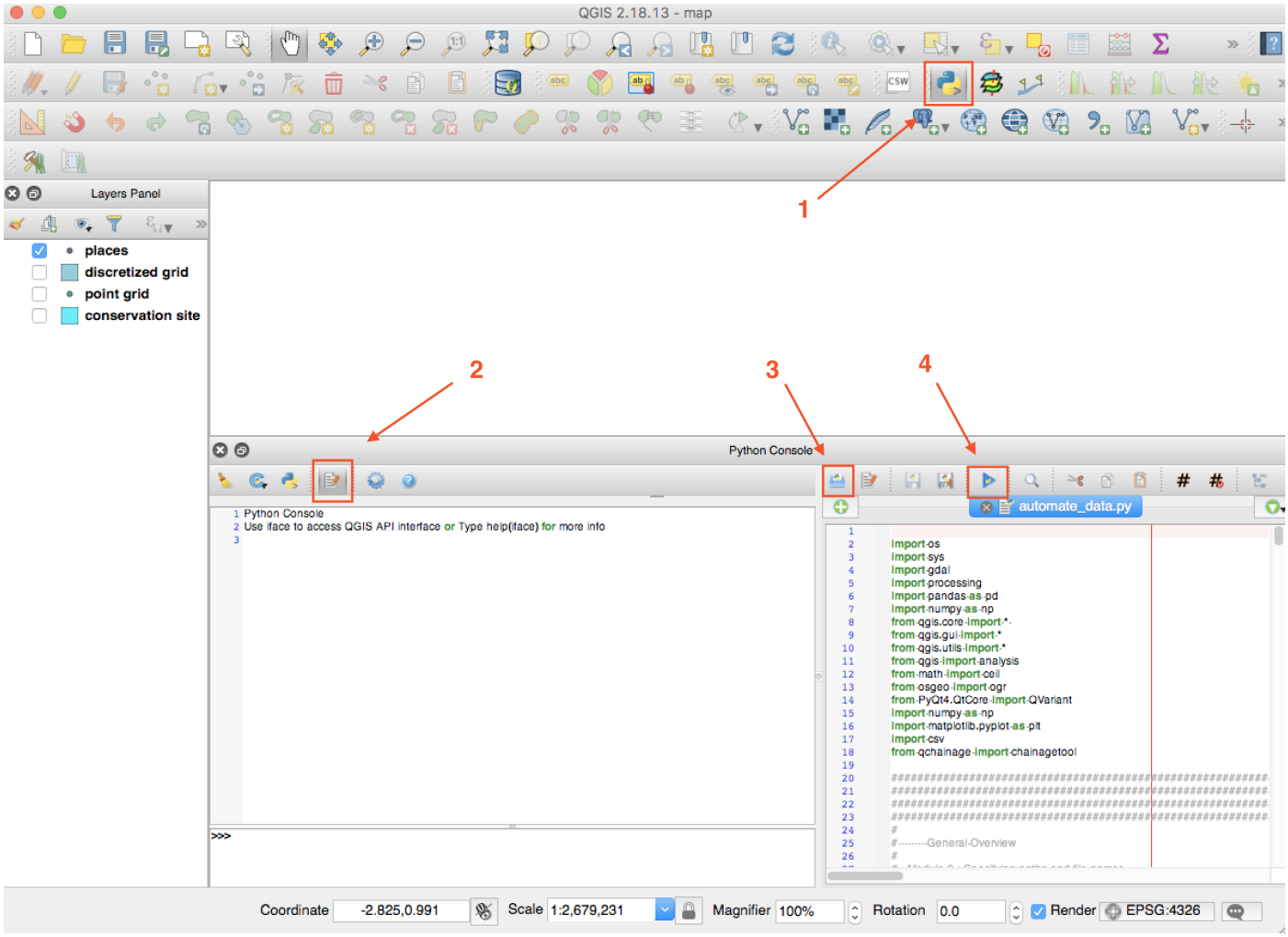
Drag the shapefile into QGIS, right click -> Open Attributes Table -> Select/filter Features (or Ctrl + F). Find the right feature to filter through, type in the desired value for that feature, and select the features

with those values. Then right click on the layer -> Save As -> specify the output directory and name and you have exported the desired subset of the layer.

4.Specifying the data. Open up 'automate_data.py' and navigate to Module 0 near the top. There, you must specify shapefiles/raster/tif files that will provide data that you want to gather, such as whether or not a given grid cell contains a river, how far that grid cell's center is from a village, the elevation at the center, etc.

We expect that all input files, either individually provided or obtained in Steps 2 or 3, be placed in the directory 'auto input'. In this module you will also need to specify the paths to these directories. If you do not know how to get this, simply drag the folder into terminal and the output should specify. Other specifications include the level and extent of discretization. More details are found in the file.

5.Using QGIS to generate the data. Open the Python Console (1), show the editor (2), import the downloaded 'automate_data.py' script (3), and finally run the script (4). This may take a long time depending on the desired level of discretization, but QGIS will keep running even though it may appear frozen. Annotations for progress are printed out. Execute this process twice if you have both a labeled and unlabeled conservation site to predict for. Make sure you define different output directories in Module 0 of 'automate_data.py' so the data is distinguishable.



6.Preparing to make predictions. We will assume you have an unlabeled conservation site to predict for. If you do not, ignore all the steps for it and comment out the last line of code in 'make_data_pandas.py'. Again, in Module 0, specify the names of the excel sheets produced from Step 5. Either copy and relabel both in the bigger directory or specify the paths to the 'excel output' directories. Specify what you want the prediction text and raster files to be called, and then enumerate the features you want to use for training. Note that features beginning with 'norm-' are the normalized versions that should be used. Also specify the grid corners and cell size (the latter should be consistent with Step 2). Once editing Module 0 is done, simply run the script.

Note there are a couple main functions in 'make_data_pandas.py'. The first, `classify_familiar_tri`al, can be used to test the classifier on the labeled conservation site, and some hyper parameters can be tuned to increase the performance if desired. The second, `main_poaching_predict`, actually trains on all the data on the first conservation site, and produces predictions for both regions in text files. The third, `prep_qgis`, converts these text files to raster files.

7.Rinse and repeat. If the predictions and performances seem weak or nonsensical, perhaps tune some parameters in `classify_familiar_tri`al or gather more data. Expanding feature dimensionality need not require executing the whole of 'automate_data.py' from scratch. We can simply specify only the new features to calculate and generate csv files for, and subsequently augment these files onto the resulting 'final.csv' data sheet.

8.Reach out to us! By providing this tool, we hope for more opportunities to collaborate with onsite workers at various conservation sites to continuously improve our algorithm. Being that this is our publically available code, please do reach out to us so we can provide further complexity beyond this release, for example by incorporating patrol officers' impressions of the threat levels across the site or by simply providing us more data to use. Help us prevent illegal poaching across the world.