**(.py file modified from original to protect proprietary business data structures and methods)**

**Data munging instructions**

* Get the LPK file from ESRI
  1. LPK is an ESRI proprietary data compression format, but under the hood it is basically just 7-Zip and you can use it to unzip it.
  2. Open the .*lpk* and navigate until you find a directory that has files like “esriinfo”, “v10”, “v107”, etc. Typically the largest number will be what you are looking for as it is the most recent data, so extract it. The directory will contain a .*lyr* file and a sub-directory with the .*gdb* extension. This is the geo-database with the data you want and it is an industry standard.
* Convert the GDB to a Shapefile
  1. First, you will need the GDAL/OGR packages. GDAL is a translator library for working with raster and vector geospatial data. OGR Simple Features Library is a C++ open source library with CLI tools allowing read access to a variety of vector file formats importantly including in this case ESRI shapefiles. There is a great deal of support for GDAL/OGR as it is commonly used in GIS software projects, particularly if you are developing with python. Here is a good guide for installation on the Ubuntu platform. If you are developing in a Windows environment, stop what you are doing immediately and begin the process of learning how to develop in a Linux one. ;)

<https://mothergeo-py.readthedocs.io/en/latest/development/how-to/gdal-ubuntu-pkg.html>

Don’t forget to export the environmental variables for the compiler.

* 1. Use the ogr2ogr CLI tools to convert from *.gdb* to *.shp*. To do this I ran the following command in the terminal from the v107 directory containing the .*gdb* and .*lyr* files.

***Ogr2ogr -f “ESRI Shapefile” zip\_code\_data zip\_poly.gdb***

This will convert and dump the contents of the defined file (zip\_poly.gdb) into a new sub directory named “zip\_code\_data”. You will find the data in the v107 directory (or whatever it is named in your case).

* + Dump the data into a GeoJSON format
  1. Open a browser and navigate to <https://mapshaper.org/>. Import the .*dbf*, .*shp*, and if available .*shx* and .*prj* files. It will take some time to parse these files, but when it is done you will have polygons displayed.
  2. Select the simplify option at the top and check the “prevent shape removal” box. Select the algorithm you prefer (in this case, I selected “Visvalingam/weighted area”), and click “apply”. After a little bit of time calculating, you will get a slider bar up top. Zoom into an area where you have some familiarity with what the polygons should look like (in my case, I zoomed into the Seattle area and viewed the zip codes there) and then drag the slider to the right until you have simplified it sufficiently to your needs. In my case, I dragged the slider to about 30%. Once you have done this, it is important that you click “repair” as there will be several line intersections leftover from the smoothing algorithm. Typically, mapshaper will be able to successfully repair the intersections, otherwise you may need to find a different degree of simplification and simplify from there. It is preferable to not have to manually correct these intersections.
  3. Export the data by clicking “export”, which should open an export menu. For the file format select “GeoJSON”, then confirm by clicking export in that menu. After some time processing the data, you will be provided a prompt to name and download the data (I chose zip\_poly.json). You will need to open the .*json* and confirm if the coordinates are sufficiently low in resolution. If they are not, delete the file and re-simplify at a different ratio. I did this several times before settling on an appropriate polygon resolution.

**Instructions for running ZipZap**

Once the data has been properly munged, open up the .py file and provide the file path to your zip data in the main procedure call at the bottom. If you are running test data, comment out the prod calls and un-comment the test calls, and vice versa if you are running prod data.