**Ongoing methods documentation**

**ABANDONED LAND**

**In R**

* Use 2012 data from LUCAS
* Used GPS lat and long data points
* Filtered for…
  + U112 and D10, D20: Abandoned agricultural land
  + U410 B, C, D, E, F classes: Abandoned areas
    - B – Cropland
    - C – Woodland
    - D – Shrubland
    - E – Grassland
    - F – Bare land and lichens/moss
* For abandoned points
  + Set the lat and long as coordinates
  + Defined the CRS: "+proj=longlat +ellps=WGS84 +datum=WGS84 +init=epsg:3857"
    - This is the one that’s specific to GEE, because the dataset didn’t specify and just said WGS84
  + Transform into spatial points dataframe
* To set extent as border of Latvia
  + Get data from rworldmapxtra and specify Latvia
  + Set coordinates as lat and long
  + Set same CRS
  + Transform to spatial points dataframe
  + Transform to raster
* Set extent of the spatial points dataframe to raster
* Save as shapefile
* Create another dataset to help with classification – this is just points used for forestry (U120)
  + This is helpful because for a classification you need two classes for it to work
  + Also, forestry will have quite similar reflectance as agricultural points so will help make the classification better

**In GEE**

* Load shapefile as table asset – load the shp, dbf, and shx but not prj because GEE has a projection
* Do this for forestry points too
* Also, for the border (unclear if this is necessary as of now)
* For both forestry and abandoned points:
  + Set points as a feature collection variable and then add it as a layer to view the points
  + Add satellite imagery and filter for a year (start and end date at the beginning and end of a year)
    - Filter this by the border (unclear if this is necessary)
  + Create a cloud free composite to improve satellite imagery
    - Could be improved – the way I did this was very basic
  + Add composite as a layer
  + Draw polygons around 70 of each point: select the pixel that the point falls in, look around the pixel and look for any other pixels of the same colour – any pixels that are the same, even connected by a diagonal, select as one polygon (so it will be one point). If unsure at all if they are the same, leave the ones that you’re not sure about out
    - First 25 of each
    - Then 50 of abandoned, 25 forestry
    - 50 each
    - 70 abandoned, 50 forestry
    - 70 each
  + Choose the bands for training – chosen based on what was done in this tutorial https://geohackweek.github.io/GoogleEarthEngine/05-classify-imagery/
    - Blue B1
    - Green B2
    - Red B3
    - Near infrared B4
    - Shortwave infrared 1 B5
    - Shortwave infrared 2 B7
  + Train the classification on the polygons, looking at those specific bands
  + Set the palette and display the classification
  + Get the accuracy and error of the classification
    - Resubstitution error: error rate obtained from the training data
      * This is optimistic and not a good indicator of performance on future data
    - To get more accurate error – split the data into training and testing
      * This assumes the training and the test data are representative samples of the whole dataset – this becomes truer with larger datasets so should be fine here
      * Training dataset is 70% of data, testing is 30%
      * Train the classifier with the training dataset and classify with testing set
      * Get the error and accuracy as a confusion matrix - <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>
      * Export this as a CSV (doesn’t work otherwise because it runs out of time)
      * Potential for cross-validation?? Using CORINE data?
  + Export classification as image, feature collection and map