Detailed instructions for constructing the mining data

Santiago Saavedra Mauricio Romero*

July 14, 2021

Contents

1 Introduction **2**

2 Inputs **3**
2.1 Files required . . . 3
2.2 R packages required . . . 4

3 Preparing the satellite data **4**
3.1 Identify the list of images available . . . 4
3.2 How to make an order of images . . . 5
3.3 Preparing the files for ML . . . 6
3.3.1 How to make bulk downloads . . . 7
3.3.2 Unzip images . . . 7
3.3.3 . . . 8
3.3.4 Topographic correction . . . 8
3.3.5 Clouds cleaning and clouds filling . . . 9
3.3.6 Remove tree . . . 10

4 Training the model **10**
4.1 Creating input data for model . . . 10
4.2 Train model . . . 11
4.3 Rasters to a data frame . . . 12

5 Predicting mining **13**
5.1 Run predictions . . . 13

*Sergio Perilla did the bulk effort for writing this document

1 Introduction

This document explains how to use the codes from repository MiningPredictions/src to generate the machine learning predictions of mining using satellite imagery. Items marked with * represent files unique to Colombia that might be hard to find for other places. Before starting it is important to define the appropriate projection for the area of study.

```
1  # For Colombia it is:
2  # +proj=utm +zone=18 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0;0;0
```

It is also important to define fixed units of analysis called "pathrows": blocks or squares on which the satellite images are divided. For additional info consult https://www.usgs.gov/land-resources/nli/landsat/landsat-shapefiles-and-kml-files. The following picture shows the coverage of pathrows in the Colombian surface.

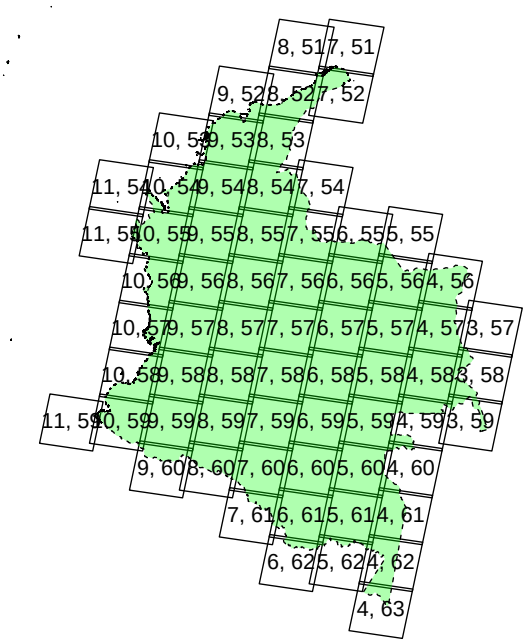

Figure 1: Scenes (path,row) from LANDSAT 7 covering Colombia

2 Inputs

2.1 Files required

1. DataNubes/copy/DEM: Digital Elevation Model is a model performed, not for us, which predict the slope and elevation of each pixel. These will be used for the topographic correction of the satellite images, because the amount of color can vary on the slope of the pixels. They are obtained with these steps:
   (a) Go to https://ssl.jspacesystems.or.jp/ersdac/GDEM/E/4.html
   (b) Create an account
   (c) Select shape of the focus area (Colombia)
   (d) Download all the tiles
   (e) Unzip all in the DataNubes/DEM folder
2. DataNubes/copy/Municipios.shp shapefile of Colombian municipalities. These are the national administrative units at which we will create our measures of mining[1].
3. *DataNubes/copy/Etter EcosistemasGenerales.shp shapefile of the ecosystems of Colombia. This will be used for the prediction model, because the model was confusing deserts with mines.
4. DataNubes/copy/TITULOS FINALB.shp shapefile of the legal mining titles of Colombia. This will be used for deciding the legality of mines. This file was created from two similar versions of the titles one from Ideam and other from the Mining Cadastre.
5. *DataNubes/copy/Minerals.shp shapefile of the resources available in the subsoil in Colombia. We digitized the map from page 12 of the PDF Agencia Nacional Minera (2013)
6. DataNubes/ManualDownloads/YES contains .zip folders of images manually classified as mines. These will be used to train the prediction model. We had a Research Assistant manually inspecting point coordinates of the Mining Census 2010, and drawing the exact shape of mines.
7. DataNubes/ManualDownloads/NO contains .zip folders of images manually classified as NO mines. These will be used to train the prediction model. We had a Research Assistant manually inspecting point coordinates of the Mining Census 2010, and drawing the exact shape of non-mined areas.
8. DataNubes/tree contains rasters files that indicate the year of deforestation, tree cover and gain-loss for each pixel. In section 3.3.6 you can find the link for access to raw data from Hansen.

[1]You can find an updated shapefile of Colombian municipalities in DataNubes/copy/Municipios-New.shp. The main difference is that the oldest one does not include some *corregimientos municipalities* as municipalities. All of our results were/are constructed with the oldest one.

2.2 R packages required

The package rgdal, rgeos, raster, devtools, and teamlucc, are necessary to perform the analysis. The following lines of code install them if they have not been installed yet.

```
# rgdal
if(!require(rgdal)){
install.packages("rgdal")
}
# rgeos
if(!require(rgeos)){
install.packages("rgeos")
}
# raster
if(!require(raster)){
install.packages("raster")
}
# devtools
if(!require(devtools)){
install.packages("devtools")
require(devtools)
}
# teamlucc
if(!require(teamlucc)){
install_github("mauricioromero86/teamlucc")
}
```

3 Preparing the satellite data

3.1 Identify the list of images available

We need to know the list of all images available for our study area in the entire study period, before selecting the ones to download and process.

1. Go to http://earthexplorer.usgs.gov/ and log in
2. In the first tab *Search criteria* select the area of interest and the period of study. Select your area drawing your polygon through clicks, is not possible to upload a shapefile.
3. Inside the tab *Search criteria*. In *Data Range* select your period of study. And in *Result Options* choose the total number of images to return, ideally 500 images per year.
4. In the second tab *Data Sets* select Landsat Collection 1 Level-2 (On-Demand) > Landsat 7 ETM+ C1 Level-2
5. In the third tab *Additional Criteria* select the following characteristics

(a) *Land Cloud Cover* and *Scene Cloud Cover* select your desired level of cloud cover. A smaller number means less clouds, so more visible area.

(b) *Collection Category* select Tier 1

(c) *Data Type Level-1* select Level 1TP

(d) *Scan Line Corrector* select SLC-off (2003-present)

6. In the fourth tab *Results*, click on *Click here to export your results* button, select Current Results and CSV. **Note**: You must be logged in to download and order scenes

7. You will receive an email with a link to download a compressed file. It contains a dataset that describes all available scenes. In our case, we have this file on "DataNubes/copy/Order_name.csv"

3.2 How to make an order of images

Once we know all the images we need to select. We wish to download, depending on cloud cover, date or other criteria. Note that an order can have a maximum of 500 images.

1. Run /DataNubes/ecosystems/src/scripts/make_anual_orders.R to create .txt files by year, with a list of indentifiers for the imagery to download.
Inputs:

   - *mainPath*: folder where the function ee_read.R is located in /ecosystems/src/scripts
   - *folder_metadata_file*: folder from file downloaded in the previous section with the scenes available. (In our case: /DataNubes/copy/Sherlock/orders/Raw)
   - *file_name*: file's name (In our case: LSR_LANDSAT_ETM_C1_291367)
   - *years*: is the vector of years for which you want to produce orders (In our case: 2017)
   - *name*: name of .txt files (In our case: Order 2018)
   - *output_dir*: desired folder to save .txt files (In our case: /DataNubes/copy/Sherlock/orders)

   Output is .txt files with the codes of the images to download in each order. (In our case DataNubes/copy/Order 2018.txt)

2. Go to http://espa.cr.usgs.gov/, log-in and click Order Data.

3. In scene list upload the .txt file of the order.

4. In Level-2 Products select only surface reflectance.

5. In Customization Options select format Geotiff and Reproject Products (in our case, Projection: Universal Transverse Mercator and UTM Zone: 18 North)

6. Click Submit

Note: Sometimes while submitting, some image ids from the .csv, are not available in the USGS Earth Explorer. Then, you should drop manually these ids from your .txt file.

Wait a couple of days to receive an email that the order is ready. A typically downloaded file, say *LE07009056201808070 1T1-SC20181207174006.tar.gz*, has the following characteristics:

- *LE07* indicates the image is from Landsat 7 Enhanced Thematic Mapper Plus
- *009056* is the pathrow the image covers
- *20180807* is the acquisition date expressed in year, month, day
- *01* Collection Number(01)
- *T1* Landsat scenes from Tier 1 (the highest available data quality)
- *SC20181207174006* is the order number USGS gives you
- *.tar.gz* indicates the compression mode

3.3 Preparing the files for ML

Run funcion_CodigoCompleto_s.R to create the final set of imagery for ML. The images for ML are images without clouds and negative values.

Inputs

- *DataNubes/Input*: Raw Landsat imagery downloaded in previous step
- *Order_2018_L07.csv*: Data frame with imagery characteristics the csv from the section 3.1.7
- *DataNubes/copy/DEM*: that contains rasters with the slope for each pixel by pathrow. Imagery from section 2.1.1.

Output

- *DataNubes/final_s*: rasters without clouds or negative values, one raster brick by pathrow.

The following steps are applied for each pathrow:

1. Download compressed files

3.3.1 How to make bulk downloads

So far, we know how to select the list of scenes and make an order of it. In this section, we explain how to download scenes. The bulk download uses a python code, which requires a special python version and cloning from espa-bulk-downloader.git repository [2].

The following lines from the code: funcion_CodigoCompleto_s.R calls the code download_espa_order.py to download compressed files.

```
system("python path_download_espa_order/download_espa_order.py
-e email
-o orden_id_to_download
-d path_download_folder
-u username
-p password
-i http://espa.cr.usgs.gov")
```

The list below describe the inputs from function system() to download imagery

- *email* that you used to log-in in section 3.1.1
- *orden_id_to_download* order id that USGS gives to you when you submit an order in section 3.2.6
- *path_download_folder* path where you will save the raw data. (In our case: DataNubes/Input)
- *username* that you used to log-in in section 3.1.1
- *password* that you used to log-in in section 3.1.1

2. Unzip raw imagery

3.3.2 Unzip images

Due to the big size of the satellite imagery, the files are double zipped. Then, the code funcion_CodigoCompleto_s.R uses the function extract_espa.R to unzip all images by pathrow.

```
espa_extract_sergio(in_folder, out_folder)
```

[2]In this webpage https://github.com/USGS-EROS/espa-bulk-downloader, you can find the tutorial for python and espa-bulk-downloader repository installation

where the **input** for the function is:

- *in_folder* where the compressed Landsat imagery was downloaded. (In our case: DataNubes/Input)
- *out_folder* where the uncompressed Landsat imagery was saved. (In our case: DataNubes/extract_new_PPPRRR)

The **output** of the function is a folder with raw imagery saved in DataNubes, where PPP is the path and RRR is the row of the image. Note that each folder inside of extract_new_PPPRRR, are different scenes for the same pathrow.

3. Choose how many images are necessary to clean clouds
3.3.3

4. Apply topographic correction for the selected images

3.3.4 Topographic correction

The imagery from each pathrow must be corrected topographically because color can vary given the slope of the pixel. Then, the function topographic_corr() from teamlucc package corrects this problem in the images. For this correction, you should have the Slope aspect for pathrow and image characteristics (i.e Sun Elevation and Sun Azimuth). The code should have the following structure:

```
topographic_corr(base, slopeaspect, sunelev=Sun.Elevation, sunazimuth=Sun.Azimuth,
DN_min=0, DN_max=10000)
```

where the **input** for the function are:

- *base* the image to correct
- *slopeaspect* is a RasterBrick with two layers. The first layer should be the slope, the second layer should be the aspect.
- *sunelev* is the sun elevation in degrees of the image to correct
- *sunazimuth* is the sun azimuth in degrees of the image to correct
- *DN_min* is the minimum allowable pixel value after correction (values less than DN_min are set to NA)
- *DN_max* is the maximum allowable pixel value after correction (values less than DN_max are set to NA)

The **output** of the function is a topocorrected RasterBrick for pathrow PPPRRR

5. Cloud cleaning and cloud filling

3.3.5 Clouds cleaning and clouds filling

The mining predictions are performed on the values from band 1 to band 7[3]. To apply the machine learning model is necessary to remove pixels with clouds, and fill them with a value from the same location on a cloud-free day. The function cloud_remove() from teamlucc package, uses one of several different algorithms to fill clouds in a Landsat image. For this correction, you should have more than one image by pathrow, and the cloud mask[4] of the image with fewer clouds. The code should have the following structure:

```
cloud_remove(base_tc, fill_tc, base_cloud_mask_try, DN_min=0, DN_max=10000,
algorithm="teamlucc", verbose = T)
```

where the **inputs** for the function are:

- *base_tc* is the cloudy image (base image) as a Raster*
- *fill_tc* is the clear image as a Raster* to use for filling base_tc
- *base_cloud_mask* is the cloud mask as a RasterLayer, with each cloud patch assigned to an unique integer code.
- *DN_min* is the minimum valid DN value (default of 0)
- *DN_max* is the maximum valid DN value (default of 10000)
- *algorithm* must be "teamlucc" or "simple". (In our case, we use "teamlucc" because is faster than "simple")

The **output** of the function is a RasterBrick for pathrow PPPRRR, without clouds and replaced with past land values.

The cloud filling takes a lot of time. We find that performing the cloud cleaning by block is faster than cleaning clouds in the whole pathrow. Also, it reduces the probability of error due to the algorithm used. The blocks are small squares that constitute the whole pathrow. Function SplitRas() create the small blocks for the *base_tc*, *fill_tc* and *base_cloud_mask*. The code has the following structure:

```
SplitRas(raster, ppside, folder, name)
```

where the **inputs** for the function are:

- *raster* is the RasterLayer to be splitted
- *ppside* is the number of cuttings on the x and y axis
- *folder* is the output folder to save the squares
- *name* is a character that label the square depending on the raster

The **output** of the function are $ppside^2$ RasterBricks that split the pathrow PPPRRR

[3]Excluding band 6 not processed for the Landsat 7

[4]The cloud mask is an image that comes with each scene after you unzip the compressed file.

3.3.6 Remove tree

Finally we remove pixels covered with trees in the Landsat imagery in the respective year. The tree images come from Hansen (2018)[5]. The code processTreeImages.R create pathrows that indicate whether a pixel is covered with trees or not.
Inputs

- **DataNubes/tree**: A folder with raw Hansen imagery
- **pathextent**: A folder, in DataNubes, with rasters which have the desired projection for the output rasters (In our case: DataNubes/trainTestLabels_proj/2010)

Outputs

- **DataNubes/tree_Proc**: A folder with rasters which indicate tree cover by pathrow

Later, you can run scripts/removeTreeYEAR.R that remove tree pixels from the processed Landsat imagery to the respective year. YEAR takes the values from 2004 to 2017.
Inputs

- **DataNubes/final_s** folder with all cleaned images from the section 3.4

Outputs

- **DataNubes/noTree** folder that saves all the imagery without trees

4 Training the model

4.1 Creating input data for model

You can find this codes into the folder ecosystems/src/scripts

1. Run createManualCorrectionMasksScript.R to transform the images of manually classified YES mines/NO mines into usable rasters by pathrow (In our case: DataNubes/labelCorrections/2010).
2. * Run createEcosystemMasks.R to transform the shapefile of ecosystems from Colombia, into usable rasters by pathrow. These rasters identify each kind of ecosystem with a number per pixel (In our case: DataNubes/ecosystemsMask).

   The last codes receive the following input and produce the output:
   Input:

   - *Shapefile*: The specific shapefile to rasterize
   - *Rasters* that indicate the outlines of each pathrow.

[5]You can find the raw images here https://earthenginepartners.appspot.com/science-2013-global-forest/download_v1.4.html?fbclid=IwAR2SuCjZbvzjuHzTnX4uLrnnpZ-3kLJbo0muYEZLYwslzfRnswK8VYLR7IU

Output:

- *Rasters* by pathrow that contain the values of the variable to rasterize. These rasters were saved in DataNubes/ShapeMask, where ShapeMask can be manual labelCorrections, titulosMask, minerpot, or EcosystemMask.

3. Run createTrainTestMasks.R to randomly separate the country into 75% of pixels for training and 25% for testing. The randomization is done in each square.

   This code receives the following input and produce the output:
   Input:

   - *Shapefile*: The shapefile with the country of analysis. In our case *COL_adm0* shapefile

   Output:

   - *Rasters* by pathrow that indicate if one pixel was assigned for training or testing. These rasters were saved in DataNubes/trainTestLabels_proj/2010

4. Run createLossyearMasksYEAR.R to transform Hansen imagery on how long ago a pixel was deforested, into usable rasters by pathrow. YEAR takes the values 2015, ..., 2017; for the rest of years 2004, ...2014 run createLossyearMasks.R.
   This code receive the following input and produce output:
   Input:

   - *DataNubes/tree*: The raw rasters from Hansen on loss year

   Output:

   - *DataNubes/lossyear*: Rasters by pathrow and year that indicate how long ago a pixel was deforested. These rasters were saved in DataNubes/lossyear

4.2 Train model

1. Run trainallm.R to train and test the mining prediction model.

   (a) Inputs are your available imagery to train the model. In our case:
   - **manualLabel**: rasters of pixels manually classified as mined/ non-mined according to the Census high resolution inspection.
   - **OSMcorrections**: rasters of mines according to Open Street Map.
   - **noTree**: rasters with the satellite band measurements after removing forest pixels.
   - **lossyear**: rasters with info on how long ago a pixel was deforested.
   - **ecosystemsMasks**: rasters with the type of ecosystem.
   - **traintestLabel**: raster indicating whether the pixel will be used for training or testing.

(b) Outputs

- **alltraining**: Matrix with the training pixels information.
- **alltesting**: Matrix with the testing pixels information.
- **modelname.Rda** is the mining prediction model. It will be used in the predictions section. In our case, we choose 100 binary decision trees so that it can be applied to other years.

4.3 Rasters to a data frame

The code dataframeFromRaster.R create a data frame from the pathrows(rasters) inside some specified folders. In this data frame, the rows are values from each pixel in each pathrow, and the columns are the number of RasterLayers inside pathrow through the folders. The code has the following structure:

```
dataframeFromRaster(filename, inputFolders, varnames)
```

Input:

- *filename* is the pathrow name as a character
- *inputFolders* specifies the paths where are located all the Masks to transform to data frame. In our case: DataNubes/Mask where Mask takes the values: final_s, lossyear, noTree, predictionPost, verifAndres_2poly, verifLuisF_2poly, verifSergio_2poly.
- *varnames* is a vector that specifies the name for the columns. In our case: band1-6, lossyear, prediction, mineAndres, mineLuisF, mineSergio.

The **output** of the function is a dataframe with the specified variables

(a) Outputs

(b) Descriptive outputs

i. **.tex files**
   - *tpr_opt* contains the true positive rate
   - *ppv_opt* contains the number of pixels truly predicted as mines
   - *opt_thresh* this file indicates the optimal threshold to declare a pixel as mined
   - *mat_conf* shows the confusion matrix for the estimated model
   - *fpr_opt* contains the false positive rate

ii. **.pdf file**
   - *ROC*: In this file was plotted the ROC curve for trained model

5 Predicting mining

5.1 Run predictions

i. Run ResultsbyMunicipio2004.R to create rasters with mining predictions (if you want to change ML model, you should change it in wraper_resbymun.R)

Inputs

- **DataNubes/noTree**: RasterBricks with 6 color bands, without; clouds, trees, and negative values.
- **DataNubes/ecosystemMask**: RasterBricks indicating the ecosystem for each pixel
- **DataNubes/lossyear**: Rasters with information about loss year
- **Rf_ALL.Rda**: .Rda with the ML model trained in the previous section

Output

- **PredictionPost imagery**: Rasters clasifing pixels as mined or not, saved in the PredictionPost folder.

- Run createTFINALBMasks.R to transform the shapefile of mining titles into usable rasters by pathrow, indicating whether each pixel has a legal title (In our case: DataNubes/titulosMasks). Then we classify each mining prediction as legal/illegal.
- * Run createMinerpotMasks.R to transform the shapefile of geological resources in the subsoil, into usable rasters by pathrow, information indicating which pixels have mineral resources underground (In our case: DataNubes/minerpots). We use this to classify a mining prediction as gold, coal or other mineral.

References

Agencia Nacional Minera (2013). Exploring opportunities: the answer is colombia. http://www.anm.gov.co/sites/default/files/Documentos/exploringopportunities.pdf.