 doi:10.20944/preprints201805.0470.v1

Article

Open-sourced remote sensing data management with the Irish Earth Observation (IEO) Python module

Guy Serbin [1] and Stuart Green [2,*]

[1] Environment, Soils, and Land Use Department, Teagasc Environmental Research Centre, Johnstown Castle, Co. Wexford, Y35 N2D6, Ireland; guy.serbin@teagasc.ie

[2] Agri-Food Business and Spatial Analysis Department, Teagasc National Food Research Centre, Ashtown, Dublin 15, Ireland; stuart.green@teagasc.ie

* Correspondence: guy.serbin@teagasc.ie; Tel.: +353-53-917-1201

Abstract: Many remote sensing analytical data products are most useful when they are in an appropriate regional or national projection, rather than globally based projections like Universal Transverse Mercator (UTM) or geographic coordinates, i.e., latitude and longitude. Furthermore, leaving data in the global systems can create problems, either due to misprojection of imagery because of UTM zone boundaries, or because said projections are not optimised for local use. We developed the open-source Irish Earth Observation (IEO) Python module to maintain a local remote sensing data library for Ireland. This pure Python module, in conjunction with the IEOtools Python scripts, utilises the Geospatial Data Abstraction Library (GDAL) for its geoprocessing functionality. At present, the module supports only Landsat TM/ETM+/OLI/TIRS data that have been corrected to surface reflectance using the USGS/ESPA LEDAPS/ LaSRC Collection 1 architecture. This module and the IEOtools catalogue available Landsat data from the USGS/EROS archive, and includes functions for the importation of imagery into a defined local projection and calculation of cloud-free vegetation indices. While this module is distributed with default values and data for Ireland, it can be adapted for other regions with simple modifications to the configuration files and geospatial data sets.

Keywords: Remote sensing, Python, data management, Landsat, open-source

---

1. Introduction

The recent availability of free, calibrated Earth observation (EO) data from NASA, USGS, and ESA archives, including those from MODIS, Landsat, and the Sentinel satellites, has enabled the development of local EO data libraries that can consist of hundreds or thousands of scenes. These local libraries require optimisation of data and cataloguing in order to maximise their utility. For each data library, a number of questions need to be answered. Firstly, the spatial extent of each library, and appropriate scenes, needs to be determined. Once that has been determined, a database of available scenes and associated metadata needs to be established and updated as new scenes are acquired. Each library needs to download, ingest, and properly archive and catalogue needed scenes in a manner that is most useful for analytical exploitation. Data catalogues also require metadata on data quality, including spatiotemporal accuracy.

Local and regional remote sensing analyses are often most useful when data are in local projections that are optimised for global positioning systems (GPS) and make use of recent, more accurate reference spheroids [1]. There are a couple of reasons for this: accompanying local and regional data sets are often in a local projection that best describes the local landscape, and is less sensitive to geodetic changes due to plate tectonics. Secondly, remote sensing data are often distributed in global projection systems, which may distort local geographic features. The Irish Transverse Mercator projection [1] was created to more accurately map the Republic of Ireland and Northern Ireland, and is based upon the European Terrestrial Reference System (ETRS89).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

Commercial and open-source image server software do exist for the management of geospatial data archiving and serving, e.g., ArcGIS Image Server (commercial; ESRI, Redlands, CA, USA), Erdas Apollo (commercial; Hexagon Geospatial, Norcross, GA, USA), GeoServer [2], and Jagwire (commercial; Harris Geospatial Solutions, Inc., Boulder, CO, USA). However, these products are not automatically configured for discovery of available, calibrated data from external data providers, nor for ordering and ingesting them into local archives without additional scripting and configuration. Furthermore, hardware and software costs and maintenance requirements often make the use of these software packages prohibitive, particularly where local analysis requirements may only require a lightweight data management solution.

With these considerations in mind, we have created the new open source Irish Earth Observation (IEO) Python module and the associated IEOtools scripts for the management of local Landsat 4 – 8 imagery archives. While the defaults for this module and the IEOtools scripts are for Ireland, including both the Republic and Northern Ireland, they can easily be adapted to other local projections. These can also be incorporated with any software, whether commercial or open-source, that provides support for Python scripting. The aim of this paper is to describe these and show how they can be of utility for the remote sensing community.

2. Considerations for a projection-specific geoprocessing module

2.1 Decision to use a specific projection

As part of various remote sensing projects, we decided to create a local library of Landsat 4 – 8 TM/ ETM+/ OLI/TIRS imagery, starting from 1982 and onwards. We identified Landsat World Reference System 2 [WRS-2; 3] tiles that intersected with Ireland's land masses (Figure 1) to facilitate queries of available scenes. Initially, data were kept in the Universal Transverse Mercator (UTM) projections, which spanned UTM Zones 29 and 30 North. While UTM is convenient for global distribution of some satellite data products, e.g., Landsat, it caused problems with data projection where scene centres lay immediately to the east of the UTM 29N/ 30N boundaries, as seen in Figure 2. In this Figure, a Landsat scene from WRS-2 Path 206 Row 22, with a scene centre just to the east of the boundary between UTM Zones 29N and 30N, was misprojected westward over the Atlantic Ocean, just to east of the boundary between UTM Zones 28N and 29N. We discovered that warping of these scenes into ITM fixed this issue, and minimised potential geometric errors that might have occurred by warping data into a different UTM Zone.

2.2 ENVI format

We decided to utilise the ENVI raster data format (Harris Geospatial, Boulder, CO, USA) for raster data products, due to the flexibility of the ENVI header (*.hdr, HDR) format. Each HDR is a text file that contains required and optional metadata tags [4], and accompanies the main data file. Required tags include data type, lines and samples, number of bands, and data ordering (e.g., band sequential, band interleaved by line or pixel). Optional tags can include metadata like spectral data, image classification scheme class names, colour tables, projection, solar acquisition geometry, and scene acquisition time metadata. Because these tags are optional, potentially useful metadata are often missing, but this format enables us to easily add in what we deem necessary. Additionally, this format enables us to create new metadata tags where needed.

2.3 LEDAPS and LaSRC surface reflectance products

Landsat 4 – 7 Surface Reflectance [LEDAPS; 5,6] and Landsat 8 Surface Reflectance Code [LaSRC; 7] Collection 1 Level 2 products available from the US Geological Survey's (USGS) Earth Resources Observation and Science (EROS) Center's Sciences Processing Architecture (ESPA) include surface reflectance (SR), thermal infrared (TIR), a pixel quality assurance (QA) layer, and possible indices. The pixel QA layer is based upon the Fmask algorithm [8], and differs between LEDAPS and LaSRC products. While both pixel QA products include bit values for clear land, water, cloud shadow, snow/ ice, and clouds, they also include bit values for cloud confidence levels

[6,7]. The LaSRC pixel QA layer differs in that it also includes bit values for cirrus cloud confidence and pixels affected by terrain occlusions [7]. Our data importation algorithms are optimised to import the SR, TIR, and pixel QA products. For all products, created ENVI HDR metadata include product-specific descriptions including the 21 character Landsat scene identifier, and data acquisition time. For the SR and TIR products, missing sensor-specific band wavelength, full-width half maximum (FWHM), and band names were added to the final products. Legacy support is included for pre-Collection 1 LEDAPS and LaSRC Fmask products, and for these missing class names and a class colour scheme are added to aid in visualisation.

2.4 New metadata

The text-based nature of ENVI HDR files has been used by other software packages like SPIRITS [9] to maximise the effectiveness of remote sensing data. Because every HDR contains simple metadata tags and values, customised metadata can be created to improve the usefulness of data products. This capability is also exploited in IEO, via the optional "parent rasters" tag. This tag was created as a diagnostic means of tracing the input data used to create an output raster product, and help identify issues with raster processing code or raster geometry. Within IEO, it is implemented during the ingestion process for stacked Landsat imagery and vegetation index (VI) products.

2.5 File extensions

The ENVI file format is very flexible with respect to data file extensions, and can automatically handle numerous ones provided that they are accompanied by a HDR. As such, ENVI data files can include *.img, *.dat, *.envi, and other file extensions, or none at all. For IEO, we've chosen the *.dat extension for a few reasons. Firstly, ENVI data files historically used the *.dat extension, and certain software packages, like ArcGIS Desktop (ESRI, Redlands, CA, USA), have preferred this extension for data importation. Secondly, the *.img extension has been historically been associated with the Erdas Imagine (Hexagon Geospatial, Norcross, GA, USA) format.

We also make use of the virtual raster (VRT, *.vrt) format for the rapid mosaicking of Landsat scenes in different WRS-2 Rows in the same Path that are from the same date. While this format does help conserve disk space, it has limited capabilities with respect to raster metadata.

3. The code and data files

The IEO Python module and IEOtools scripts were written in open-source Python 3, with the most recent code written using the Anaconda Python (Anaconda, Inc., Austin, TX, USA, https://www.anaconda.com/) distribution, but is designed to also be Python 2.7 compliant. The IEO and IEOtools code includes open-source Python code from other sources. We adapted code from Erickson et al. [10] for vector and raster data processing. Methods for handling and saving ENVI data types were used from Boggs [11]. Code was also created with the aid of Python online documentation [12].

3.1 The IEO Python module

The IEO module is designed to install as a Python egg file via *python setup.py install*. The installation process will create a configuration file, *ieo.ini*, unless one is adapted from *sample_ieo.ini*. It will also create a default file structure for a Landsat archive and a metadata catalogue, unless these already exist. It will also attempt the installation of a number of prerequisite libraries, if not already installed, specifically the Geospatial Data Abstraction Library (GDAL; http://www.gdal.org/), NumPy (http://www.numpy.org/), NumExpr (https://numexpr.readthedocs.io/en/latest/index.html), and the Pillow implementation of the Python Imaging Library (PIL, https://pillow.readthedocs.io) Python modules.

The IEO module 1.x versions have been designed mainly to support 30 meter Landsat 4 – 8 TM/ ETM+/ OLI/TIRS imagery at 30 m resolution. As such, Landsat MSS and 15 m panchromatic imagery

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

are not currently supported, though support may be added to future versions. Additionally Sentinel-2 data are not supported, but are planned for version 2.0 and higher. The main IEO module, ieo.py, contains a number of functions designed to handle LEDAPS/ LaSRC files from both Landsat Collection 1 (starting with version 1.1.1) and Pre-Collection 1 datasets (https://landsat.usgs.gov/what-are-naming-conventions-landsat-scene-identifiers). With IEO version 1.1.1, users may choose to utilise to 40 character Landsat Product Identifier rather than the 21 character Scene Identifier [6,7] for output filenames, though the default is set to the latter.

The module is broken down into a number of sections. The first section imports required libraries and configuration data. This includes module variables available to external Python code which call IEO, and are detailed in Table 1. While many of these values are set in the *ieo.ini* configuration file, they are mutable, should user requirements dictate a change in one or more values between scripts, e.g., a case where a user wants to manage two different archives in two different projections using this software. The next section belongs to a number of uncategorised functions that are utilised by other functions:

1. logerror(): error function logging.
2. extract_xml(): XML tag value extraction.
3. get_landsat_fileparams(): extract Landsat satellite number, WRS Path, WRS Row, acquisition year, acquisition day of year, and acquisition date (YYYYMMDD) from a Landsat Scene Identifier.
4. makegrid(): by default, this function will create the AIRT, provided that a shapefile with the outline of Ireland is provided (*makegrid(inshape = <path to Ireland shapefile>)*). However, it is designed to flexibly create tile grids, including settings for tile dimensions and numbers in the south – north and west – east directions. It should be noted that currently there is a maximum of 676 tiles in the west – east direction, but this can easily be increased in the future versions with minimal code changes. It can be used to create grids that intersect any shapefile, as seen in Figure 3 for the Upper Caragh, Owenroe, and Kealduff sub-catchments in Co. Kerry. Each grid tile also has a tile name, where letters denote the location starting with "A" in the west and "1" in the south. For the AIRT, the southwesternmost tile that could be created is named "A01", whereas for a grid like the one portrayed in Figure 3, it would be "AA001".
5. makeparentrastersstring(): this function takes a list of filenames that are used to create an output raster, and formats the ENVI HDR string.

The next section contains functions specifically for scene reprojection and accuracy assessment:

1. reproject(): this function will warp data from its native projection to the locally-defined projection, by default ITM. It does so by calling an external function, *gdalwarp*, using *subprocess.Popen*. The output will be in ENVI format.
2. checkscenelocation(): this function checks the relative spatial accuracy of Landsat scenes that were warped to the local projection, rejecting any warped scene whose scene centre is over 50 km from the generic scene centre in the WRS-1 or WRS-2 locally-projected polygons that are defined in *ieo.WRS1* and *ieo.WRS2*, respectively. The details of the shapefiles are discussed in further detail later in this text and in Table 2.

The following section contains functions for Landsat import and VI calculations:

1. envihdracqtime(): this function will read the acquisition time, if present, from an ENVI header file.
2. maskfromqa(): this function creates a binary memory mask from the Pixel QA layer using the Boolean module variables described in Table 3. Like with the variables described in Table 1, these are also mutable for scripts calling the IEO module, though unlike those, default values are hard-coded in *ieo.py*. The created output data mask will have values of 1 for good pixels, and 0 for bad.
3. calcvis(): this function calculates the Normalised Difference Vegetation Index [NDVI; 13] and Enhanced Vegetation Index [EVI; 14] for clear land pixels from Landsat data.
4. EVI(): this function is called by *calcvis()* to calculate EVI, and returns a two-dimensional double integer (*numpy.int16* type) array.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

5. NDIndex(): this function calculates a two-band normalised difference index (Band A – Band B)/ (Band A + Band B), and returns a two-dimensional double integer (*numpy.int16* type) array. It is called by *calcvis()* to calculate NDVI.
6. importESPA(): this function ingests new LEDAPS/ LaSRC data from the USGS/EROS/ESPA, either in gzipped TAR file (tar.gz) or decompressed formats, provided that said data are in GeoTIFF or ENVI formats. HDF formats are not supported. It decompresses the data, virtually stacks them where necessary, warps the data into the local projection, and calculates VIs. It then cleans up intermediate files, and archives the original gzipped TAR file to *ieo.archdir*.
7. ESPAreprocess(): this function adds a Landsat scene identifier string as a new line to text file to be uploaded to ESPA for processing.

Then there is a section for file compression and decompression utilities:

1. unzip(): opens *.zip files.
2. maketarfile(): creates a gzipped TAR file in *ieo.archdir* of all *.dat and *.hdr files in a directory. The tar.gz is named based upon the Landsat scene identifier string found in the compressed files.
3. untarfile(): this extracts the contents of a tar.gz to disk.

3.1.1. ENVIfile.py submodule

The ENVIfile.py submodule writes data and metadata in ENVI file format. The bulk of this module is a class called *ENVIfile()*. The class contains three subclasses: *file*, *header*, and *colorfile*. The *file* subclass contains a function for processing class parameters that relate to raster data dimensionality and type. The header subclass contains two functions- *ENVIfile.header.readheader()* reads in existing ENVI header metadata into class attributes, and *ENVIfile.header.prepheader()* prepares class attributes that are used for output header metadata. The *colorfile* subclass writes colorfiles (*.clr) text files containing RGB values for classification values to aid in their display in GIS software.

The attributes of this class include output raster data (*data*). By default, it contains a dictionary (hash) called *headerdict*, with a nested dictionary of default ENVI header file tags with *None* values, *headerdict['default']*. This dictionary is then populated automatically populated with subdictionaries containing supported data types, e.g., Landsat TM, Pixel QA, NDVI, and EVI, which are cloned from the default subdictionary, and populated with the appropriate metadata for each data type, along with a subdictionary containing codes to differentiate different Landsat data types. These values are the used to create *ENVIfile.header* subclass attributes via *ENVIfile.header.getdictdata()*. The *headerdict* can also be called from external Python code in order to flexibly create HDR files, though this is most efficiently handled the using *ENVIfile.header* subclass attributes.

3.1.2. Included data files

Included with the IEO module are a number of shapefiles that reside in the shapefiles subdirectory of *ieo.catdir*. By default, four shapefiles in ITM are included, as summarised in Table 2. The first consists of Sentinel-2 tiles [15] for Ireland, Landsat WRS-1 and 2 Path/Row tiles [16], and finally a new polygon data set, the All-Ireland Raster Tile (AIRT) system, as seen in Figure 1. AIRT was developed to better analyse time series of Landsat data from different but overlapping Landsat paths, and is utilised by other Python modules which call IEO.

Also included is a text file called *badlist.txt*, which contain dates for which Landsat imagery were found to have contained either serious geometric or radiometric issues, an example of which can be seen in Figure 4. The scenes included in this file include some that were geometrically corrected to pre-Collection 1 Level-1 Terrain Corrected (L1T, https://landsat.usgs.gov/landsat-processing-details) standards, the spatial accuracy for which ideally should be within a pixel (https://landsat.usgs.gov/geometry). The dates in this file are in *YYYYJJJ* format, whereby *YYYY* denotes the year and *JJJ* the day of year. The problematic dates were determined by manual observation of derived data sets, often through the use of the *parent rasters* metadata stored in derived ENVI products. This file resides in the Landsat subdirectory of *ieo.catdir*.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

3.2. IEOtools Python scripts

IEOtools (https://github.com/Teagasc/IEOtools) are a collection of Python scripts which call the IEO module, and do the bulk of the importation work for the local library. The majority of the tools are designed to utilise the USGS/EROS Earth Science Processing Architecture (ESPA) for handling LEDAPS/ LaSRC data, although one is for creating VRT files following data ingestion. It is recommended that these be downloaded and installed in a directory path that is convenient for the end user to execute them. They are designed to be used in conjunction with the USGS/EROS/ESPA website (https://espa.cr.usgs.gov/) and the ESPA Bulk Downloader software (https://github.com/USGS-EROS/espa-bulk-downloader), but at present do not exploit the USGS/EROS ESPA API (https://github.com/USGS-EROS/espa-api). These scripts are described in terms of their order of use:

1. *updateshp.py*: This script will query Landsat scene metadata for WRS-2 scenes of interest, and save new data available from the USGS, including the scene footprint in the local projection and metadata. These will be saved in *ieo.landsatshp*, which will be created if it doesn't already exist on disk. The specific scenes queried are determined by settings in the accompanying *updateshp.ini* file, either by specified ranges in the "pathrowvals" line, or from *ieo.WRS2* if "useWRS2 = Yes" is set. The script is designed to exploit two different modes of data ingestion. The default method utilises JSON-based queries, which target a specific Minimum Bounding Rectangle (MBR), though a legacy option allows for the use of large XML global metadata files that are available from Landsat metadata service (https://landsat.usgs.gov/download-entire-collection-metadata). The second option is not recommended as it requires downloading several large files containing metadata for every available Landsat acquisition globally. This script requires that the user have a USGS/EROS Registration System (ERS, https://ers.cr.usgs.gov/register/) account to query data. It will download scene thumbnails to the *ieo.catdir/Landsat/Thumbnails* directory and save their local file locations under the "Thumb_JPG" field. It also will identify Landsat scenes that were ingested by IEO in *ieo.srdir*, *ieo.btdir*, *ieo.fmaskdir*, *ieo.pixelqadir*, *ieo.ndvidir*, and *ieo.evidir*, and save their file path locations to the appropriate polygons' feature metadata under the "LEDAPS", "BT", "Fmask", "Pixel_QA", "NDVI", and "EVI" fields, respectively. It does not have any required command line switches, but will query the user for their ERS username and password if these are not supplied. Full detail on these command line options is available by using the –*help* switch.
2. *makeESPAproclist.py*: This script will check the available scenes in the shapefile from the previous script, and existing surface reflectance data in *ieo.srdir*. It then creates a list of new scenes to download, excluding scenes that cannot be processed by LEDAPS due to low sun elevation angles (< 15°) or lack of ancillary data (see https://landsat.usgs.gov/landsat-surface-reflectance-high-level-data-products). The script's behaviour can be modified a number of command line options, including which scenes to process by WRS-2 Path, Row, maximum cloud cover, etc. Full detail on these command line options is available by using the –*help* switch.
3. The list produced in step 2 is uploaded to https://espa.cr.usgs.gov/, and the following products should be requested:
    a. Input metadata
    b. Surface reflectance
    c. Brightness temperature
    d. Pixel QA
    e. ENVI format (IEO functions also will work with GeoTiff, but not HDF or NetCDF).
4. Once the order is complete, processed scenes are downloaded using the ESPA Bulk Downloader software (https://github.com/USGS-EROS/espa-bulk-downloader).

 doi:10.20944/preprints201805.0470.v1

5. *newespaimport.py*: scans for new available scenes in *ieo.ingest*, identifies missing data, and then calls *ieo.importespa()* to ingest the new data into the local archive. Its behaviour can be modified via command line switches, use *–help* for more detail.
6. *makevrts.py*: This script creates VRTs for ingested data, and saves file path locations and metadata to CSV files in *ieo.catdir/Landsat*. Its behaviour can be modified via command line switches, use *–help* for more detail.

3.3. Adaptability for other country, regional, or state plane grids

While IEO and IEOtools configuration files and geodatabase contain defaults for Ireland, these can easily be adapted for other grid systems with the following modifications prior to IEO installation. Firstly, the end user will have to acquire geospatial data sets, e.g., an outline of the area/ region/ country for which they desire to adapt the module, and Landsat WRS-1, WRS-2 [16], and Sentinel-2 [15] generic scene footprint data. They then will need to create subsets of the required WRS-1, WRS-2, or Sentinel-2 tiles. These will need to be warped into a recommended local projection. The next step is to create a custom *ieo.ini* file, which can be adapted from the *sample_ieo.ini* file that is provided in the installation's *config* subdirectory. It is important to populate this file with accurate values, as the defaults are all for Ireland, and said values will affect the behaviour of both IEO and IEOtools. Upon installation, the new *ieo.ini* will be installed inside the IEO module egg file. After installation, the user should create a new national/ regional tile system using *ieo.makegrid()*, ideally by using extents that have been buffered out past the extent of the regional shapefile (recommended minimum of 300 m buffer, but up to user), and tiles that can fit 60 m × 60 m pixels, for compatibility Landsat MSS, TM/ ETM+/ OLI/TIRS, and Sentinel-2 data. Within IEOtools, only one script, *updateshp.py*, has custom settings that are configurable in *updateshp.ini*. If a different region is used, then it is important to update this configuration file, as the defaults are also for Ireland.

4. Conclusions and future development

The open source Python IEO module and IEOtools scripts allow for easy, efficient management of large Landsat data archives. The IEO module can be used by external Python code to manage and access large volumes of locally-stored LEDAPS/ LaSRC-corrected surface reflectance data, while providing means to assess data quality, e.g., account for possible geometric errors by data exclusion. Thus, this module can help extend the capabilities of any software, including commercial server or workstation packages, which can utilise Python scripting. While the defaults for this software are for Ireland, it is designed to work with any GDAL-supported coordinate system with simple modifications to configuration files, and is even flexible enough to accommodate for multiple libraries using multiple projections.

Future versions of IEOtools will incorporate the USGS/EROS ESPA API (https://github.com/USGS-EROS/espa-api) to streamline the ordering of scenes. We intend for future versions of this software to include support for Landsat MSS, Sentinel-1, Sentinel-2, EO-1 Hyperion and ALI, and other past, current, and future sensors. We invite those who make use of this software to help contribute to this project and make it a more useful product for the geospatial community.

Author Contributions: Conceptualization, G.S.; Methodology, G.S.; Software, G.S.; Validation, G.S.; Formal Analysis, G.S.; Investigation, G.S.; Resources, G.S. and S.G.; Data Curation, G.S. and S.G.; Writing-Original Draft Preparation, G.S.; Writing-Review & Editing, G.S. and S.G.; Visualization, G.S.; Supervision, G.S. and S.G.; Project Administration, G.S.; Funding Acquisition, S.G.

Funding: This research was funded as part of the CForRep project.

Acknowledgments: We also gratefully acknowledge the assistance of Mark Pelletier of Inuteq LLC in reviewing the code and contributing a function for XML tag parsing. Karen O'Neill and Daire Ó hUallacháin are acknowledged for supplying the Upper Caragh, Owenroe, and Kealduff sub-catchments shapefile of Co. Kerry with data provided from the KerryLIFE Project (LIFE 13 NAT/IE/000144) and the Irish Environmental Protection Agency.

 doi:10.20944/preprints201805.0470.v1

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. 1. Ordnance Survey Ireland; Ordnance Survey of Northern Ireland *New map projections for Ireland*; Dublin, Ireland, 16 February 2001, 2001; p 11.
2. 2. Open Source Geospatial Foundation Geoserver. http://geoserver.org/ (11 April 2018),
3. 3. NASA, *Landsat 7 science data users handbook*. National Aeronautics and Space Administration: Greenbelt, MD, 2011.
4. 4. Harris Geospatial Solutions ENVI header files. http://www.harrisgeospatial.com/docs/ENVIHeaderFiles.html (14 February 2018),
5. 5. Masek, J.G.; Vermote, E.F.; Saleous, N.E.; Wolfe, R.; Hall, F.G.; Huemmrich, K.F.; Feng, G.; Kutler, J.; Teng-Kui, L., A Landsat surface reflectance dataset for North America, 1990-2000. *Geoscience and Remote Sensing Letters, IEEE* **2006**, *3*, 68-72.
6. 6. U.S. Geological Survey, *Landsat 4-7 surface reflectance (LEDAPS) product guide*. Department of the Interior: Sioux Falls, SD, USA, 2018.
7. 7. U.S. Geological Survey, *Landsat 8 surface reflectance code (LaSRC) product guide*. Department of the Interior: Sioux Falls, SD, USA, 2017.
8. 8. Zhu, Z.; Woodcock, C.E., Object-based cloud and cloud shadow detection in Landsat imagery. *Remote Sensing of Environment* **2012**, *118*, 83-94.
9. 9. Eerens, H.; Haesen, D., Spirits software for the processing and interpretation of remotely sensed image time series user's manual. European Commission Joint Research Centre Ispra, Italy, 2016; p 392.
10. 10. Erickson, J.; Daniel, C.; Payne, M. The Python GDAL/OGR Cookbook. http://pcjericks.github.io/py-gdalogr-cookbook/index.html (15 February 2018),
11. 11. Boggs, T. Spectral python. http://pydoc.net/Python/spectral/0.17/spectral.io.envi/ (15 February 2018),
12. 12. Python Software Foundation Python documentation. https://docs.python.org (15 February 2018),
13. 13. Rouse, J.W., Jr.; Haas, R.H.; Deering, D.W.; Schell, J.A. *Monitoring the vernal advancement and retrogradation (green wave effect) of natural vegetation*; Texas A&M University: College Station, TX, October 1973, 1973; p 93.
14. 14. U. S. Geological Survey *Landsat surface reflectance-derived spectral indices product guide*; U.S. Geological Survey: Sioux Falls, SD, USA, December 2017, 2017; p 31.
15. 15. European Space Agency Sentinel-2 data products. https://sentinel.esa.int/web/sentinel/missions/sentinel-2/data-products (10 April 2018),
16. 16. U.S. Geological Survey Path/Row shapefiles. https://landsat.usgs.gov/pathrow-shapefiles (12 February 2018),

Tables

Table 1. IEO module variables that are available to external scripts and modules and defined in *ieo.ini*. Values shown here are defaults for Ireland, but can be modified prior to installation for local projections and needs.

Variable	Description	Comments
ieo.NTS	All-Ireland Raster Tiles	Resides in ieo.catdir/shapefiles
ieo.WRS1	Generic Ireland Landsat 1-3 WRS-1 scene polygons	Resides in ieo.catdir/shapefiles
ieo.WRS2	Generic Ireland Landsat 4-8 WRS-2 scene polygons	Resides in ieo.catdir/shapefiles
ieo.Sen2tiles	Generic Ireland Sentinel-2 tile polygons	Resides in ieo.catdir/shapefiles
ieo.landsatshp	Landsat shapefile containing local archive inventory	Resides in ieo.catdir/Landsat
ieo.logdir	Directory for error logs	Use ieo.logerror() for creating logs.
ieo.catdir	Catalogue for archive inventory and metadata	
ieo.srdir	Landsat LEDAPS atmospherically corrected surface reflectance directory	
ieo.btdir	Landsat brightness temperature directory	
ieo.fmaskdir	Directory containing Fmask cloud/ shadow masks	
ieo.pixelqadir	Directory containing Pixel QA layers	
ieo.evidir	Directory for clear land Enhanced Vegetation Index (EVI) data	
ieo.ndvidir	Directory for clear land Normalised Difference Vegetation Index (NDVI) data	
ieo.ingestdir	Directory for new LEDAPS/ LaSRC data from USGS/ESPA prior to ingest	Contains *.tar.gz files
ieo.archivedir	Directory for archiving original LEDAPS/ LaSRC data from ESPA after ingest	Contains *.tar.gz files
ieo.badlandsat	File path to *badlist.txt* containing problematic scenes	
ieo.prjstr	String containing EPSG code in "EPSG:XXXX" format for local projection	"ESPG:2157" for ITM
ieo.prj	Python OSR Spatial Reference object	
ieo.projacronym	Local projection acronym for filenames	Format: "ITM", with no whitespace
ieo.useProductID	Use Landsat Collection 1 Product Identifier rather than Scene Identifier	Default = False

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

Table 2. Default polygon vector layers contained in the shapefiles subdirectory of *ieo.catdir*. With the exception of AIRT, all shapefiles were subsetted and warped from geographic latitude and longitude in a WGS-84 datum to Irish Transverse Mercator (ITM) projection. WRS denotes World Reference System, and all shapefiles used were for descending orbits [16]. * denotes a product that was warped to ITM.

Layer name	Description	Source
Ireland_Sentinel2_tiles_ITM	Sentinel 2 tiles that touch Ireland	[15]
Ireland_WRS1_Landsat_1_3_ITM	Landsat WRS-1 Path/Row scenes	[16]
Ireland_WRS2_Landsat_4_8_ITM	Landsat WRS-2 Path/Row scenes	[16]
AIRT	All-Ireland Raster Tile grid, 29 ×30 km grid	IEO module

Table 3. IEO module variables that control the behaviour of the *maskfromqa()* function and their default values as hard-coded in *ieo.py*. These can be modified within scripts which call either this function or *calcvis()*.

Variable	**Description**	**Comments**
ieo.qaland	Include land pixels from Pixel QA layer	Default = True
ieo.qawater	Include water pixels from Pixel QA layer	Default = False
ieo.qasnow	Include snow/ ice pixels from Pixel QA layer	Default = False
ieo.qashadow	Include cloud shadow pixels from Pixel QA layer	Default = False
ieo.qausemedcloud	Include medium confidence cloud pixels from Pixel QA layer	Default = False
ieo.qausemedcirrus	Include medium confidence cirrus cloud pixels from Pixel QA layer	Default = True, Landsat 8 only
ieo.qausehighcirrus	Include high confidence cirrus cloud pixels from Pixel QA layer	Default = True, Landsat 8 only
ieo.qauseterrainocclusion	Include terrain occluded pixels from Pixel QA layer	Default = False, Landsat 8 only

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

Figures

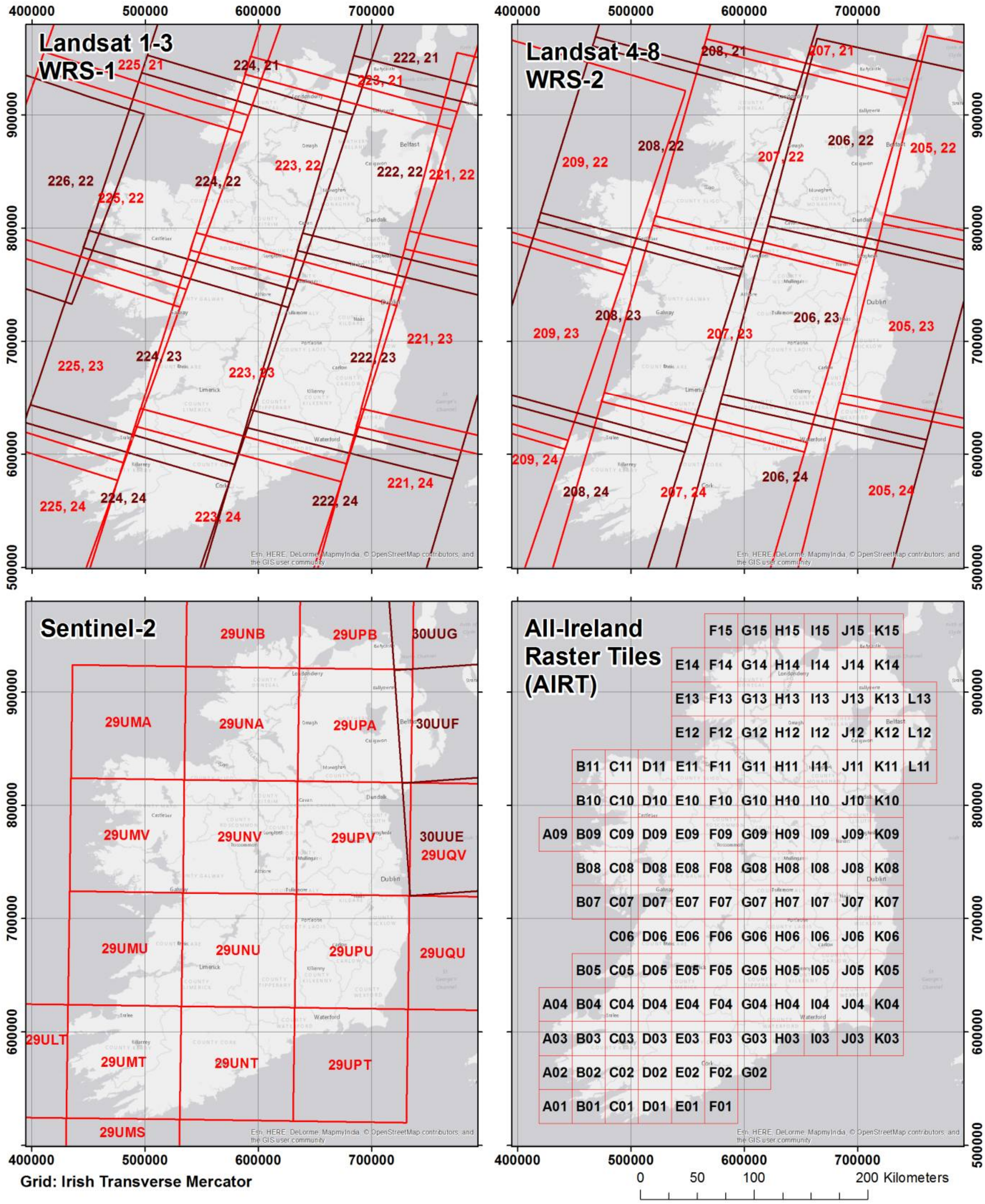

Figure 1. Polygon vector data sets distributed with the IEO geodatabase, in Irish Transverse Mercator projection. Upper left and right: Landsat 1 – 3 WRS-1 and 4 – 8 WRS-2 Path/ Row combinations, respectively. Lower left: Sentinel-2 tiles. Lower right: the All-Ireland Raster Tile (AIRT) system.

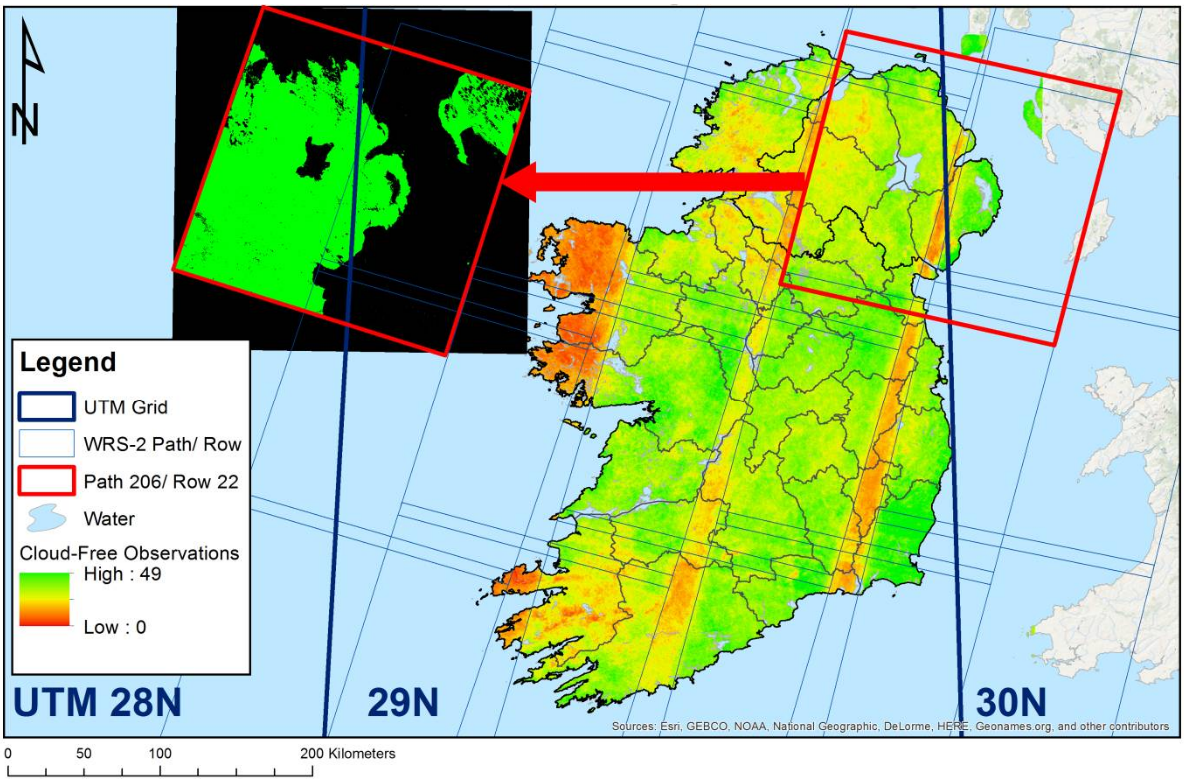

Figure 2. Severe geolocation errors that can occur when using data from differing UTM zones.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2018 doi:10.20944/preprints201805.0470.v1

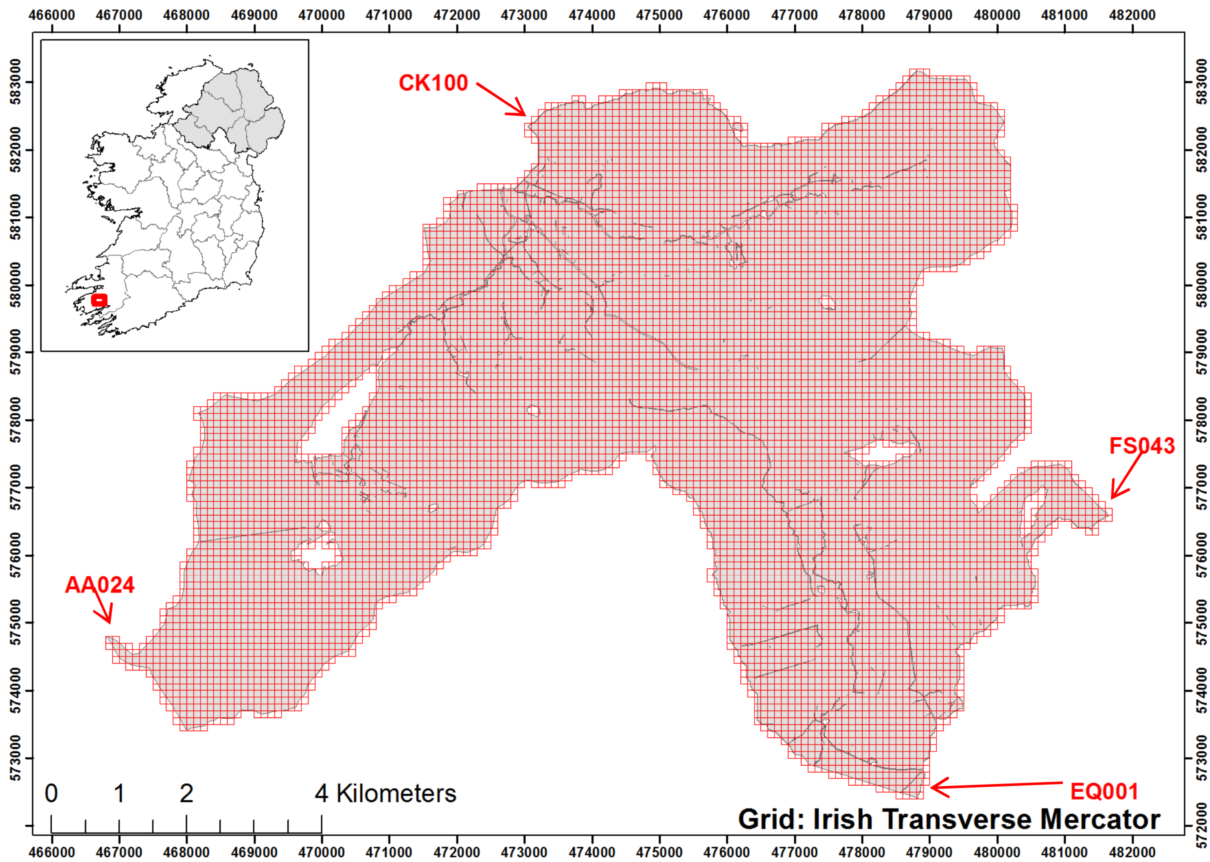

Figure 3. A 149 × 108 tile grid of the Upper Caragh, Owenroe, and Kealduff sub-catchments in Co. Kerry that was created using the *makegrid()* function. Each tile measures 100 m × 100 m. Red letter and number combinations denote individual tile identifiers.

 doi:10.20944/preprints201805.0470.v1

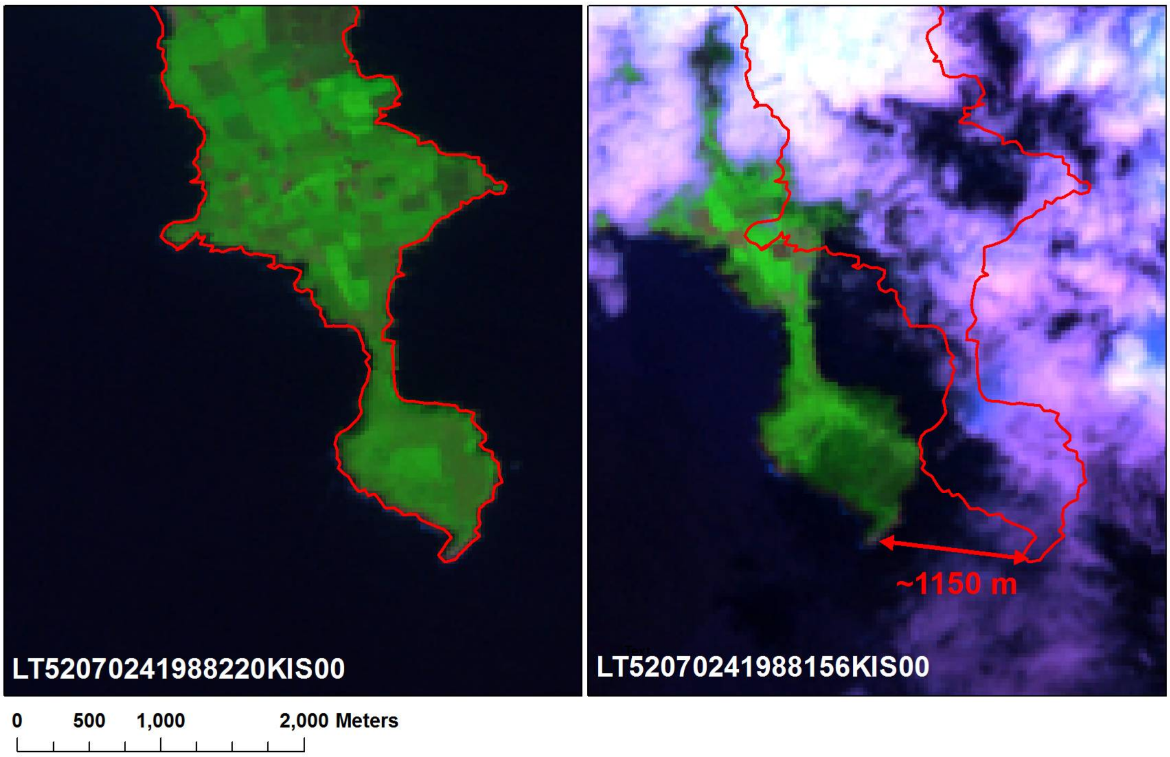

Figure 4. Examples of correct (left) and incorrect (right) geometric corrections found in Landsat Level-1 Terrain Corrected (L1T, https://landsat.usgs.gov/landsat-processing-details) scenes of Downmacpatrick, Co. Cork. The scene on the right is excluded in the *badlist.txt* file.