LSDS-1327
Version 1.0

Department of the Interior
U.S. Geological Survey

LANDSAT
BURNED AREA (BA)
ALGORITHM DESCRIPTION DOCUMENT (ADD)

Version 1.0

March 2019

USGS
science for a changing world

LANDSAT
BURNED AREA (BA)
ALGORITHM DESCRIPTION DOCUMENT (ADD)

March 2019

Document Owner:

GAIL SCHMIDT (Affiliate) Digitally signed by GAIL SCHMIDT (Affiliate) Date: 2019.03.05 15:07:20 -07'00'

Gail Schmidt Date
Software Engineer
SGT, Inc.

Approved By:

KAREN ZANTER Digitally signed by KAREN ZANTER Date: 2019.03.06 10:07:58 -06'00'

Karen Zanter Date
LSDS CCB Chair
USGS

EROS
Sioux Falls, South Dakota

Executive Summary

This Landsat Burned Area (BA) Algorithm Description Document (ADD) defines the algorithm used for the generation of Burn Probability and Burn Classification bands, which are contained within the Landsat Level 3 BA Science Product created at the U.S. Geological Survey (USGS) Earth Resource Observation and Science (EROS) Center.

This document is under Land Satellites Data System (LSDS) Configuration Control Board (CCB) control. Please submit changes to this document, as well as supportive material justifying the proposed changes, via Change Request (CR) to the Process and Change Management Tool.

Document History

Document Number	Document Version	Publication Date	Change Number
LSDS-1327	Version 1.0	March 2019	CR 14718

Contents

List of Figures

List of Tables

Section 1 Introduction

1.1 Background

The Landsat Burned Area (BA) algorithm identifies burned areas in temporally-dense time series of Landsat image stacks to produce the Landsat Level 3 Burned Area Science Product. The algorithm is based on the Landsat Burned Area Essential Climate Variable algorithm (Hawbaker and others 2017) with modifications to handle Landsat Analysis Ready Data (ARD) and incorporate Landsat 8 imagery, and is refined to use spectral indices developed specifically for mapping burned areas. The BA algorithm requires the Level-2 Pixel Quality Assessment (QA) mask, Brightness Temperature (BT), and Surface Reflectance (SR) as input, in addition to several static inputs described in Section 2.2. The algorithm creates products using Landsat 4-5 Thematic Mapper (TM), Landsat 7 Enhanced Thematic Mapper Plus (ETM+) and Landsat 8 Operational Land Imager (OLI) / Thermal Infrared Sensor (TIRS) data.

1.2 Purpose and Scope

The primary purpose of this document is to provide technical details and information on the creation and how the BA algorithm works.

1.3 Document Organization

This document contains the following sections:

- Section 1 introduces BA.
- Section 2 provides technical details on the inputs and outputs of BA algorithm.
- Section 3 provides details on the BA algorithm procedure.
- Appendix A contains a list of acronyms.
- The References section contains a list of reference documents and supporting webpages.

Section 2 Burned Area Algorithm Details

2.1 Burned Area Details

The BA algorithm involves multiple steps: pre-processing of the input data stacks (and preparing for the modeling), modeling burned area probabilities for each acquisition, and generation of the annual composites (future addition TBD).

From the Landsat Analysis Ready Data (ARD) tiles, the algorithm makes use of predictors derived from individual Landsat ARD tiles, lagged reference conditions, and change metrics between the tile and reference predictors.

The algorithm output products are the BA tiles, which consist of burn probabilities and a binary burn classification. Annual composites are also generated including: the maximum burn probability across all Landsat tiles in a year (burn probability), the count of tiles a pixel was classified as burned (burn count), and the Julian date of the first Landsat tile a pixel was classified as burned (burn date).

The Landsat Burned Area algorithm is implemented as a series of python scripts that are executed in sequence with a bash shell script (pBAMA_v2_ARD_yeti.sh). The python scripts and shell script were written for the USGS's YETI high-performance computing system, running on a Linux system. The BA algorithm involves the following steps, each performed by an individual python script, in the order that they are executed within pBAMA_v2_ARD_yeti.sh:

1. Preprocessing (preprocessing.py)
2. Converting Pixel QA to an Fmask-like band (qa2fmask.py)
3. BA Processing
    a. Subsetting the ARD tile stack for the current year (subset_stack.py)
    b. Copying the files associated to subset stack (subset_stack_copy_files.py)
    c. Generating the burn probabilities (sample_and_predict_open_by_block.py)
    d. Generating binary burn classification (threshold_stack_by_scene.py)
    e. Writing XML metadata (scene_EarthExplorer_metadat.py)
    f. Writing FGDC metadata (scene_FGDC_metadata)
    g. Generating PNG browse files (scene_visuals.py)
    h. Packaging the Burned Area product (scene_EE_pacjkage.py)

2.2 Algorithm Inputs

Table 2-1 lists the Inputs of the BA algorithm.

Description	Purpose
Landsat ARD tile stack of Surface Reflectance (SR) and Brightness Temperature (BT)	Used to create the spectral indices and tasseled cap variables for generating the burn probabilities

Description	Purpose
Level-2 Pixel QA Mask (PIXELQA)	Used to create an fmask-like band
Level-3 ecoregion raster file	Used to create variables that are input to the model
Shapefile for Landsat ARD	Used to crop the path/row, state, and ecoregion data to the current tile extents
Shapefile of Worldwide Reference System-2 (WRS-2) path/rows	Used to generate metadata
Shapefile of states in the region	Used to generate metadata

Table 2-1. Burned Area Algorithm Inputs

2.3 Burned Area Outputs

Table 2-2 lists the outputs of the BA algorithm.

Description	Units	Type
Burn Probability	Percent/Flag	8-bit Unsigned Integer
Burn Classification	Flag	8-bit Unsigned Integer

Table 2-2. Burned Area Algorithm Outputs

2.4 Burned Area Flowchart

Figure 2-1 shows the overall flow diagram of the Burned Area algorithm.

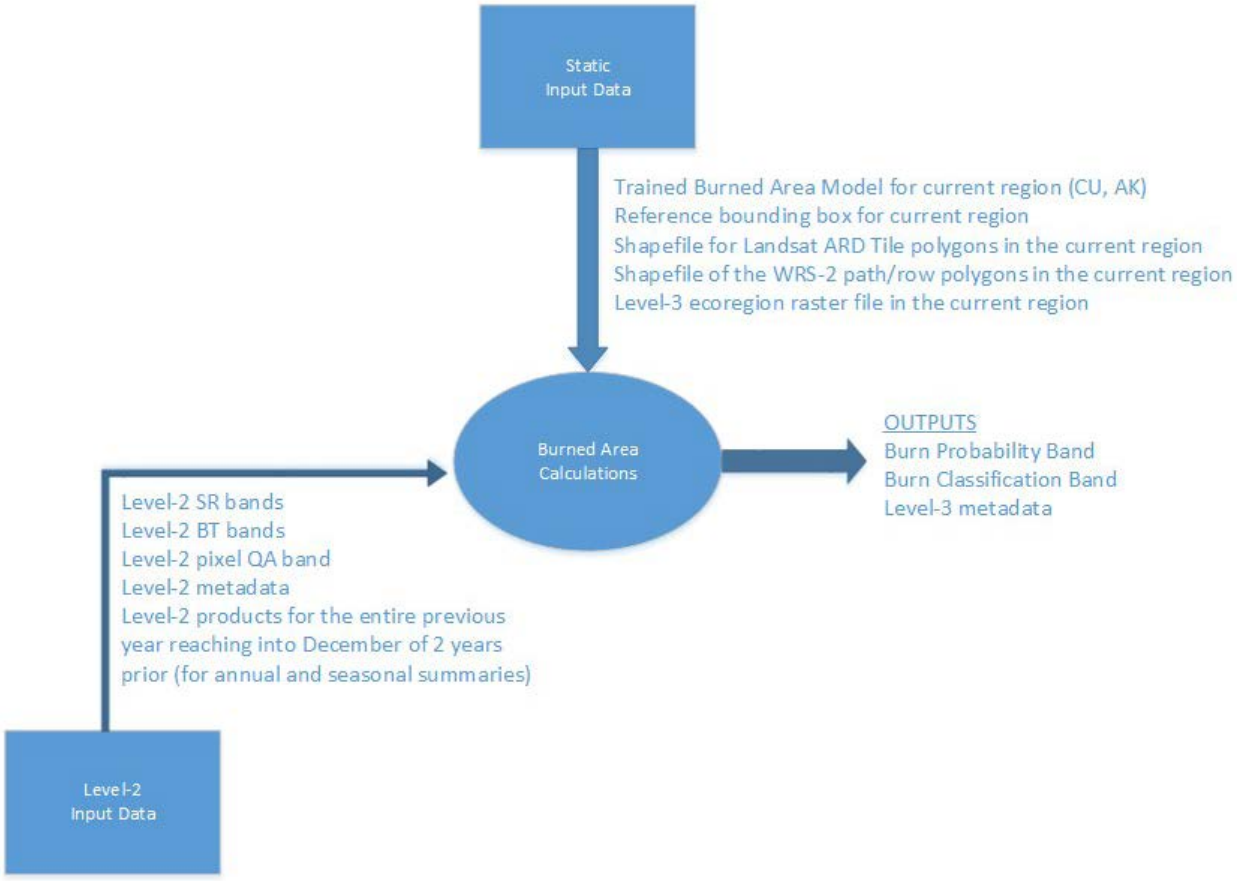

Figure 2-1. Burned Area Algorithm Flow Diagram

Section 3 Burned Area Algorithm Procedure

3.1 Preprocessing (preprocessing.py)

Inputs:

1. Landsat ARD tile stack of SR and BT .tar packages for the desired date ranges, including data for the previous 3 years (for lagged reference conditions)
2. Temporary directory for untarring and processing the data (optional)
3. Output directory for writing the preprocessed results
4. Name of .csv stack file for storing information about the ARD tile stack
5. Region to be processed (CU)
6. Horizontal tile number to be processed
7. Vertical tile number to be processed
8. Reference bounding box for the current region (static file)
9. Shapefile for the Landsat ARD Tile polygons in the current region (static file – used to crop the path/row, state, and ecoregion data to the current tile extents)
10. Shapefile for the path/row polygons, used to generate metadata.
11. Shapefile for the states in the regions, used to generate metadata.
12. Level-3 ecoregion raster file in the current region (static file-used for input to the models)

Outputs:

1. Multi-band GeoTIFF file (8 bands) with TM/ETM+ bands 1-7 and an fmask-like band for each acquisition (stored in the {output_dir}/lndsr directory)
2. Original XML metadata file for each acquisition (stored in the {output_dir}/lndsr_metadata directory)
3. ARD tile polygon for the current ARD tile (stored in the {output_dir}/vectors directory)
4. Path/row polygons for the current ARD tile (stored in the {output_dir}/vectors directory)
5. State polygons for the current ARD tile (stored in the {output_dir}/vectors directory)
6. Ecoregion shapefile cropped to the current ARD tile (stored in the {output_dir}/rasters directory)
7. stack.csv metadata file which contains a list of all the XML files, associated metadata, and an output field to designate whether or not they will get processed (stored in the output directory)
8. bb.csv bounding box file which contains the west, north, east, and south bounding box coordinates for this tile (stored in the output directory)

Description:

The preprocessing sets up the output subdirectories for the various outputs which will be generated for the particular geographic region as well as the specific ARD Tile. Projection information for the tile is obtained and set for output products. The bounding box coordinates for this tile are determined and written to the bb.csv file – these are used to ‘snap’ the cropped ecoregion raster to the Landsat data. All of the output raster files (ecoregions) and vectors (path/row polygons, state polygons, and tile polygons) are cropped and written to the output raster or vector directory, respectively.

Next the list of .tar files within the input directory is generated. The Brightness Temperature file (band 6 for TM / ETM+ and band 10 for TIRS) is extracted from each BT.tar package into a temporary directory, and the pixel QA, Surface Reflectance bands, and XML metadata files are extracted from each SR.tar package. The OLI band names are renamed to match the TM/ETM+ band names based on their spectral range (e.g. the near-infrared OLI band 5 is renamed to band 4). The pixel QA band is converted to an fmask-like band (see Section 3.2 for more information). Then all bands are stacked into a single multiband GeoTIFF file, in order of SR_band1, SR_band2, SR_band3, SR_band4, SR_band5, BT_band6, SR_band7, and fmask-like QA band. The multiband GeoTIFF file is written with compression to the output directory. The temporary directory and contents are removed.

The last step of preprocessing is to loop through all the XML metadata files, extract the metadata, and write the metadata to a CSV file, which contains key information about each acquisition in the stack, such as satellite, sensor, acquisition date, percent cloud cover, percent snow and ice, percent fill, and the geometric Root Mean Square Error (RMSE). If the RMSE is above the maximum RMSE or the percent cloud cover is greater than the maximum cloud cover, then the acquisition is not processed. Currently the default is to exclude images with > 10 m RMSE and > 80% cloud cover. If the date of the acquisition is prior to 1984, then the acquisition is written to the CSV file but flagged to not be processed for burned area probabilities. After processing has completed, if the CSV file exists in the output directory, then it indicates that the preprocessing completed successfully.

This approach, in general, describes the “stack processing” of historic temporal stacks of data (e.g. 1984-2017).

3.1.1 Forward Processing

Once the historic data has been processed, the new data will also need to be processed as part of “forward processing”. The forward processing code will only process new acquisitions. The code checks the preprocessing output directory. If an acquisition doesn’t exist in that directory, then the acquisition is processed as described above and written to a multiband GeoTIFF file. The associated XML metadata file is written to the metadata directory. The stack.csv file will contain metadata for each of the acquisitions in the ARD tile stack, however the only acquisitions flagged for processing in the CSV file will be the new acquisitions. The forward processing code also expects that the ARD tile polygon and the ecoregion shapefile are available as part of the initial stack processing.

NOTE: All the data products written to the output directory are used in full as part of the BA processing. Forward processing and full stack processing will utilize data in this directory. Do NOT rerun pre-processing unless new files have been added to the input tile stack, otherwise the output stack.csv file will specify all products with a ‘0’ in the output column for not processing.

3.2 Converting Pixel QA to fmask-like band (qa2fmask.py)

Inputs:
1. Pixel QA band

Outputs:
1. Fmask-like band (unsigned 8-bit integer)

Description:
The qa2fmask routine reads the pixel QA band (PIXELQA), then loops through each pixel in the QA band to convert them to values that were previously used in the fmask band.

```
If pixel_qa is fill
     Fmask = nodata (255)
Else if pixel_qa is clear and (pixel_qa cloud confidence <= 1 or
pixel_qa cirrus confidence <= 1)
     Fmask = 0
Else if pixel_qa is cloud and (pixel_qa cloud confidence >= 1 or
pixel_qa cirrus confidence >= 1)
     Fmask = 3
Else if pixel_qa is cloud shadow
     Fmask = 4
Else if pixel_qa is snow
     Fmask = 2
Else if pixel_qa is water
     Fmask = 1
Else if pixel_qa is terrain occluded
     Fmask = nodata (255)
Else
     Fmask = 0
```

The data are written out as unsigned 8-bit integers to a GeoTIFF file using DEFLATE compression. The geolocation information from the input pixel QA band is used for this band and the fill value is written as 255. The file extension is _CFMASK.tif.

3.3 BA Processing

Inputs:
1. Multi-band GeoTIFF file (8 bands) with TM/ETM+ bands 1-7 and an fmask-like band for each acquisition (stored in the {input_dir}/lndsr directory)
2. Original XML metadata file for each acquisition (stored in the {input_dir}/lndsr_metadata directory)
3. ARD tile polygon for the current ARD tile (stored in the {input_dir}/vectors directory)
4. Ecoregion shapefile cropped to the current ARD tile (stored in the {input_dir}/rasters directory)

5. stack.csv metadata file which contains a list of all the XML files, associated metadata, and an output field to designate whether or not they will get processed (stored in the input directory)

Outputs:
1. Burn probability image for each acquisition {unsigned 8-bit integer}
2. Binary burn classification image for each acquisition {unsigned 8-bit integer}
3. ESPA XML metadata file for each acquisition
4. FGDC XML metadata file for each acquisition
5. PNG browse products for each acquisition (Landsat 7-4-2, burn probability, burn classification)

Description:
Processing of the burned area probabilities and classifications occurs one year at a time. The processing script loops through each year, from the specified start year to end year, and determines the input and output directories and files, based on datasets from the preprocessing. For year in the date range, using an annual time increment, the following steps are implemented:

1. Subset the stack.csv file to the stack needed for the current year
2. Copy the files associated to this subset stack for the ARD tile to the temporary processing directory
3. Generate the burn probabilities
4. Generate binary burn classifications from the burn probabilities
5. Write XML metadata file for the burn probabilities and burn classification images
6. Write FGDC metadata for the burn probabilities and burn classification images
7. Generate PNG browse files for each acquisition showing the raw Landsat 7-4-2, burn probability, and the burn classification
8. Package the burn probability and burn classification products in a .tar file
9. Copy the results to the destination folder

3.3.1 Subsetting the ARD tile stack for the current year (subset_stack.py)

Inputs:
1. stack.csv metadata file which contains a list of all the XML files, using a ‘1’ in the output column to specify if the file should be processed
2. Start date for the metadata subset
3. End date for the metadata subset
4. Sensor (optional, all sensors are included in the subset if not specified)

Outputs:
1. Subset stack.csv which only contains the metadata needed to process burn probabilities from start date to end date (with lags applied), as well as the specified sensor (default is to process all the sensors). Acquisitions will be specified for burned area processing based on the output field value.

Description:
This script subsets the full temporal stack.csv file into a subset based on the specified start date, end date, and sensor (optional). The input stack.csv file is read. The start year and end year are pulled from the start and end dates, respectively. A time lag of 3 years is subtracted from the start year to provide enough data for the lagged reference conditions and change metrics.

If the sensor is not specified, then the sensor is not used in the subsetting and all files are processed. A mask is set up based on the acquisition date, time lag, and sensor in the stack.csv file. Any records that meet the start/end date are flagged with an output field value of 1, indicating these acquisitions will actually be processed for burn probabilities. Otherwise the output value is set to 0 and the acquisition is only used for generating the summaries and metrics. The original stack.csv is then subset using the date range, time lag, and sensor.

3.3.2 Copying the files associated to subset stack (subset_stack_copy_files.py)

Inputs:
1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a '1' in the output directory to specify files that need to be processed
2. Directory location of the multiband GeoTIFF files, metadata files, raster, and vector files for the ARD tile stack

Outputs:
1. Directory to copy the subset of files to for burned area processing

Description:
This script copies the ecoregion raster file to the output directory. It then opens the subset stack file and reads the names of the acquisitions in the subset. Any files which exist in the destination directory which don't apply to the subset stack are removed (the same output directory is used for processing year after year of data and therefore previous years of data will have already been copied). The missing multiband GeoTIFF file and XML metadata files for the subset stack are copied from the input lndsr and lndsr_metadata directories to the same subdirectories in the output directory.

3.3.3 Generating the burn probabilities (sample_and_predict_open_by_block.py)

Inputs:
1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a '1' in the output directory to specify files that need to be processed
2. Directory location of the multiband GeoTIFF files, metadata files, raster, and vector files for the ARD tile stack
3. Shelf file with gradient-boosted regression burn classifier object (produced from training the GBR model)
4. Colormap template for the burn probability images

5. Number of rows and columns of blocks in pixels to process per CPU (tiled processing in a multi-threaded environment)
6. Number of CPUs for processing (tiled processing in a multi-threaded environment)

Outputs:
1. Burn probability for each of the acquisitions to be processed for the current subset stack {Byte}

Description:
This script reads the input parameters and validates the input files are available. It reads the Gradient Boosted Regression Model (GBRM) file, which includes the tree classifier and the predictors, which currently are nbrt1 (normalized burn ratio with thermal band), nbrt1_mean_3yr, vi6t (spectral index with bands 4 and 6), vi45 (spectral ratio with band 4 and 5), vi45_mean_3yr, nbrt1_sd_3yr, ndvi (normalized difference vegetation index), and ndvi_mean_3yr. It then sets up the 258 stack variables split into Input, Stack, Output, and other (generated from the inputs), many of which are deleted when running the GBRM in sampling mode. These variables are available for training the model, if needed.

Ultimately the only variables which are kept for modeling (keep_predictors) are the Input (raw bands, fmask, and ecoregion file), Stack (year, julian day, sensor), and any variables defined in the GBRM shelf variable list as well as associated difference, mean, and standard deviation variables. This stack is then sorted. A stack of unused predictors is identified as any of the variables from the initial stack which are not included in the keep_predictors stack. These unused variables are deleted from the initial stack, as are the ‘eco_l2’ and ‘eco_l3’ variables. The output burn probability response variable (BP) is added to the final stack of variables. Now the stack is ready for the modeling process. The GBRM modeling is run in a multi-threaded environment and the data is processed in blocks, based on the size of the image and the specified lines/samples to be processed per CPU. The modeling first opens the ecoregion datasets to determine the image size. Each of the blocks are processed independently. The stack-related variables (i.e. year, sensor) are populated for the block. The ecoregion data is then read for the block, and from that file, the eco_l1, eco_l2, and eco_l3 stack variables are populated. Each of the Landsat bands are loaded into the stack next, including the fmask band. In this section, the fmask values are changed from 3 to 253 for clouds; 2 to 252 for snow/ice; 4 to 254 for cloud shadows; and 1 to 251 for water. This allows the fmask values to be preserved in the burn probability outputs. From the current variables (ecoregion, bandX, and fmask), a processing mask is created to skip the nodata and unclear pixels. If pixel for any of the bands is fill, then the pixel is masked from processing. Furthermore, if the fmask is non-zero (water, snow/ice, clouds, cloud shadows), then the pixel is also masked from processing. The stack is then “shrunk” to drop images with no useful data.

With the remaining data/variables in the stack, the spectral index (e.g. NDVI-like) variables are computed using the bands specified when the variable was defined. The tasseled cap (function type of TC) variables are computed next (i.e. TCbrightness, TCgreenness, and TCwetness). The SAVI, EVI, spectral ratios, MIRBI, BAI, NBRT1,

GEMI are all calculated. For the spectral indices and the tasseled cap variables, if any of the values are less than -1 or greater than 1, then are set to -1.0 or 1.0, respectively. Next the annual/lagged summaries for the bands and the spectral indices are calculated. Then the difference predictors (difference between one variable and another) are calculated. And, the relative difference predictors (relative difference between one variable and another) are calculated. Finally the GBRM burn probabilities are computed. The predictor variables are rearranged to match the order of the training data, then the model is run on the predictor variables for each scene in the stack. The output probabilities are multiplied by 100.0 to percentages. The burn probabilities are then combined with the fmask values (251-255) and written to the output file. Finally, all the input/output files are closed and memory is cleaned up.

The last step of the processing, after all the blocks have been completed, is to copy the burn probability color template for the current burn probability acquisition file.

3.3.4 Generating binary burn classification (threshold_stack_by_scene.py)

Inputs:

1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a '1' in the output directory to specify files that need to be processed
2. Input directory for the burn probabilities for this ARD tile
3. Output directory for the burn classifications for this ARD tile
4. Colormap file for the burn classification images

Outputs:

1. Binary burn classification based on the thresholding of burn probabilities, burn thresholds, region seeds, and flood filling of the burn areas {unsigned 8-bit integer}

Description:

This script reads the input stack CSV file and keeps the files that are flagged with a '1' in the output column. For each file in the filtered stack, the burn probability (BP) and output burn classification (BC) filenames are determined. The flood-fill algorithm is then used for identifying the burn scars.

The algorithm first masks any burn probability values greater than 100% to be 0 – these are clouds, shadows, water or snow/ice pixels. The bp_regions and bp_seeds arrays are defined as the same shape as the burn probability image and initialized to zero. For each pixel, if the burn probability is greater than or equal to the seed probability threshold (96%), then flag it as a seed pixel in bp_seeds. Next, small seeds are removed using a seed size threshold of 22 pixels (which is approximately 5 acres). This helps clean up false positives but may also remove small fires.

If there are any seed pixels in the bp_seeds image, then region properties are obtained from the first pixel in each region. Regions are defined by pixels with a connectivity of 1. Now loop through each region, pulling the area (number of pixels) for the region and the line/sample coordinates of the region. If the size of the region is greater than or equal to

the seed size threshold (still 22 pixels ~ 5 acres), then the region is filled in with surrounding pixels of lower probability. Thus, if any of the surrounding pixels are between 71% and 100% burn probability, then they are added as burn pixels in the region.

The binary burn classification is generated from pixels that were flagged in the burn regions. The fmask information is added for pixels that were not classified as burned. The new burn classification image is written to the burn classification and the burn classification color file is copied to the output directory.

3.3.5 Writing XML metadata (scene_EarthExplorer_metadata.py)

Inputs:

1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a '1' in the output directory to specify files that need to be processed
2. Directory for the burn probabilities for this ARD tile
3. Directory for the burn classifications for this ARD tile
4. Acquisition-based metadata template

Outputs:

1. Acquisition level XML metadata files for both burn probability and burn classification files for each of the acquisitions to be processed for the current subset stack.

Description:

This script finds the acquisitions in the subset stack that are to be output for the forward processing. The script uses the input XML file for the SR product as the starting point for the output acquisition-based metadata, then appends the burn probability and burn classification bands to the original XML file. The new XML file is written to both the burn probability directory and burn classification directory.

A change in processing for the forward processing only writes the XML file, using _BA.xml as the extension, to the burn probability directory. There isn't a need to duplicate this XML file in multiple directories.

3.3.6 Writing FGDC metadata (scene_FGDC_metadata)

Inputs:

1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a '1' in the output directory to specify files that need to be processed
2. Directory for the burn probabilities for this ARD tile
3. Directory for the burn classifications for this ARD tile
4. FGDC metadata template

Outputs:
1. Federal Geographic Data Committee (FGDC) metadata files for both burn probability and burn classification for each of the acquisitions to be processed for the current subset stack.

Description:
This script finds the acquisitions in the subset stack that are to be output for the forward processing. The script uses an FGDC template of the BA products as the starting point for the output FGDC metadata, then populates placeholders within the template file. Placeholders include the product ID, burned area category filename, burned area probability filename, region, states, ARD tile, scene date, source start/end date, production date, software version, bounding box in lat/long, bounding box in projection coordinates, number of lines/samples, and projection parameters.

A change in processing for the forward processing no longer writes the state information to the FGDC metadata file.

3.3.7 Generating PNG browse files (scene_visuals.py)

Inputs:
1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a ‘1’ in the output directory to specify files that need to be processed
2. Directory location of the multiband GeoTIFF files for this ARD tile
3. Directory for the burn probabilities for this ARD tile
4. Colormap file for the burn probability images
5. Directory for the burn classifications for this ARD tile
6. Colormap file for the burn classification images
7. Directory for output PNG images

Outputs:
1. Quick-look product (containing plots for raw Landsat bands 7-4-2, burn probability, and burn classification) for each of the acquisitions to be processed for the current subset stack.

Description:
This script finds the acquisitions in the subset stack that are to be output for the forward processing. The script uses the burn probability color map to create an RGB burn probability plot. Similarly, it uses the RGB burn classification color map to create the quick-look plot for the burn classification. Next bands 7, 4, 2 are used from the input multi-band GeoTIFF to generate a false color plot. All three are plotted on a PNG quick-look product with associated title, subtitles, and axis information.

3.3.8 Packaging the Burned Area product (scene_EE_package.py)

Inputs:

1. Subset stack.csv (from subset_stack) which only contains the metadata for start date to end date, using a ‘1’ in the output directory to specify files that need to be processed
2. Directory for the burn probabilities for this ARD tile
3. Directory for the burn classifications for this ARD tile
4. Directory for output tar packages

Outputs:

1. Tar packages containing the FGDC metadata, XML metadata, burn probability GeoTIFF (and associated color map), burn classification GeoTIFF (and associated color map) for each of the acquisitions to be processed for the current subset stack.

Description:

This script finds the acquisitions in the subset stack that are to be output for the forward processing. The script determines the name of the FGDC metadata, XML metadata, burn probability files, and burn classification files, then tar them together into one tar package for the acquisition. The tar file is written to the output directory.

Appendix A Acronyms

ADD	Algorithm Description Document
ARD	Analysis Ready Data
BA	Ball Aerospace and Technologies Corporation
BT	Brightness Temperature
CCB	Configuration Control Board
CR	Change Request
EROS	Earth Resource Observation and Science
ETM+	Enhanced Thematic Mapper Plus
LSDS	Land Satellites Data System
OLI	Operational Land Imager
PIXELQA	Pixel QA Mask
RMSE	Root Mean Square Error
SR	Surface Reflectance
TM	Thematic Mapper
USGS	U.S. Geological Survey
WRS-2	Worldwide Reference System-2

References

Hawbaker, T.J., Vanderhoof, M.K., Beal, Y.J., Takacs, J.D., Schmidt, G.L., Falgout, J.T., Williams, B.R., Fairaux, N.M., Caldwell, M.K., Picotte, J.J., Howard, S.M., Stitt, S., and Dwyer, J.L., 2017, Mapping burned areas using dense time-series of Landsat data: Remote Sensing of Environment, v. 198, p. 504–522, at http://dx.doi.org/10.1016/j.rse.2017.06.027.