Department of Interior

U.S. Geological Survey

Algorithm Description Document

Land Change Monitoring Assessment and Projection (LCMAP)

Python Implementation of Continuous Change Detection (pyccd)

Version 1.0

January 2017

Brian Davis

Song Guo

Kelcy Smith

Contents

Executive Summary

The Continuous Change Detection (CCD) algorithm developed at the Center for Remote Sensing, Department of Geography and Environment, Boston University is under consideration as a primary application demonstrating the value of Land Change Monitoring Assessment and Projection. The U.S. Geological Survey (USGS) Earth Resources Observation and Science Center (EROS) has translated the original algorithm to open source platforms. This document provides a high level overview of the code characteristics and functions, and will be updated as the CCD algorithm becomes more clearly understood in the context of its performance as an open source capability.

Document History

Document Version	Publication Date	Change Description
0.1	July 22, 2015	First draft
1.0	July 29, 2015	Final draft for USGS review
1.1	August 11, 2015	Updated after developer review
1.2	August 31, 2015	Added Classification module information
1.3	September 24, 2015	Added updated reference
1.4	January 5, 2015	pyccd implementation

Introduction

The Continuous Change Detection and Classification (CCDC) algorithm was developed at the Center for Remote Sensing, Department of Geography and Environment, Boston University as a robust methodology for identifying when and how a land surface location changes through time. Every clear observation in a time series of Landsat data is used to model reflectance values to determine whether and when a change has occurred. Subsequent algorithms can further classify pixel locations to indicate what land cover types were observed before and after a detected change.

The original implementation of the CCD algorithm, written in Matrix Laboratory (MATLAB), has since been translated into an open source package as C code (C-CCD). Subsequently, the algorithm has also been ported to the python language to become the primary application used to test the development and implementation of a real-time system to support the U.S. Geological Survey (USGS) Land Change Monitoring Assessment and Projection (LCMAP) initiative.

This document provides a high level overview of the code characteristics and functions, and will be updated as the pyccd algorithm becomes more clearly understood in the context of its performance as an open source science function.

Dependencies

Due to the necessity to identify change based solely on spectral responses, pyccd requires a certain level of data consistency. Processing is dependent on the following conditions.

- All scenes are projected to a uniform mapping grid
- Scenes are spatially clipped to include the maximum coverage within a path/row time series
- Input data is retrieved interactively through the LCMAP Information Warehouse and Data Store (IW+DS) Applications Programming Interface (API)
- No cloud, cloud shadow, or snow pixels are present
- No Landsat data prior to 1984 is included (Thematic Mapper (TM), Enhanced Thematic Mapper Plus (ETM+), and Operational Land Imager (OLI)/Thermal Infrared Sensor (TIRS) only)
- http or web-gis access to the LCMAP API is assumed

Inputs

The current pyccd acquires inputs from the USGS EROS Science Processing Architecture (ESPA), which provides the following.

- Landsat Level 2 Surface Reflectance for TM, ETM+, and OLI
- Landsat Level 2 Brightness Temperature for TM, ETM+ and OLI
- Landsat C version of Function of Mask (CFmask) for TM, ETM+, and OLI

Preprocessing includes:

- ESPA data loaded as tiles to the IW+DS, accessible through the API
- Reprojection and spatial subsetting to gridded tiles

Outputs

The output from pyccd for each location in the data stack includes the coefficients used to model a time series and detect change. The variables include the following.

- Date of the time series model start
- Date of the time series model end
- Date of observed trend deviation (change)
- Coefficients for each time series model for each spectral band
- Position of each time series model (x/y location in the tile/grid)
- Probability (0-100) that a pixel has changed
- Number of "good" observations used for model estimation
- Quality of the model estimation (based on what model and process is used)
- Magnitude of change (difference between model prediction and observation for each spectral band)

In order to provide products adequate for the evaluation of pyccd results, classification can be applied to pyccd outputs and ancillary data (including: aspect, change number, DEM, Fmask_stat, MPW, slope, and sinks) to generate annual maps as itemized below:

- Land Cover
- Land Cover Confidence
- Land Cover Condition
- Timing of Land Cover Change

- Change Magnitude
- Procedure Quality Assurance (QA)

Prototype Code

The source algorithm software was delivered by its principal investigators from Boston University. The MATLAB-based code and associated Wiki are advertised as available on the following Web sites.

https://code.google.com/p/ccdc
https://github.com/prs021/ccdc

The pyccd implementation encompasses only the change detection aspect of the algorithm suite contained within the above repositories. Pre-processing and land cover classification algorithms are comprised of separate code sets. The completed python versions of the change detection function, pseudo-code, and documentation are publicly available for review and comment on the following Web site.

https://github.com/USGS-EROS/lcmap-pyccd

The classification process is currently distributed as a stand-alone module which executes on the output from the preceding processes. It will be integrated into LCMAP infrastructure separately.

Verification Methods

The MATLAB code and its results have been published successfully by the principal investigators. Their methodologies are described in the references listed in this document.

Results from the USGS python code were verified against outputs from the original Matlab source to ensure process translations produced the same results as the original algorithm implementation. pyccd is peer-reviewed by Landsat and EROS Science developers, and external collaborators. Subsequent revisions will be included in the repository. Scientific validation of pyccd is underway.

Maturity

In the context of the USGS adaptation of the National Oceanic and Atmospheric Administration (NOAA) Climate Data Record (CDR) Maturity Matrix

(http://www1.ncdc.noaa.gov/pub/data/sds/cdr/Guidelines/Maturity_Matrix_Template.xlsx), pyccd falls within the early research stage. The pyccd maturity definition is listed below in **Table 1**, and is described without regard to the potentially higher maturity levels in the heritage MATLAB CCD implementation.

Criteria	Level	Definition
Software Readiness	3	Moderate code changes expected
Metadata	1	Little or none
Documentation	2	Draft Algorithm Description Document (ADD) in progress
Product Validation	2	Underway
Public Access	2	Limited data availability to develop familiarity
Utility	2	Limited or ongoing

Table 1: pyccd Product Maturity Matrix

User Services

All higher level Landsat products, services and interfaces are supported by User Services staff at USGS EROS. Any questions, comments, or concerns are welcomed through the Landsat "Contact Us" online correspondence form. Please indicate "Surface Reflectance Data Request" as the topic of regard. Electronic mail can also be sent to the customer service address included below, with the same indication of topic.

> USGS User Services
> http://landsat.usgs.gov/contactus.php
> mailto:custserv@usgs.gov?subject=Surface Reflectance Data

User support is available Monday through Friday from 8:00 a.m. – 4:00 p.m. Central Time. Inquiries received outside of these hours will be addressed during the next business day.

Pre-Processing of Observations

The general workflow of pyccd begins with the assumption that tiled/gridded time series observations are available from the IW+DS API. All data ordering, pre-processing and loading into the LCMAP infrastructure must be completed before the necessary observations can be presented to pyccd. To accomplish the necessary processing, USGS EarthExplorer (https://earthexplorer.usgs.gov) can used to generate a list of scenes in the path/row and temporal range of interest. The scene list is submitted to the ESPA on-demand interface (https://espa.cr.usgs.gov) to order processing of Landsat data. The data is then fed into the

appropriate LCMAP utilities for loading into the IW+DS to create Analysis Ready Data (ARD). Characteristics of ARD include:

- Pixel Alignment
- Consistent Projection
- Standardized Extents (tiles)
- Standard Level 2 Products Suite
- Eliminates Swath Overlap
- Adds Side-lap, (Temporal Density)
- User-defined Area of Interest (AOI)
- ARD Named Future Standard Landsat Product

The data stored in the IW+DS takes advantage of the side-lap of the World Reference System-2 (WRS-2) path/row definition to obtain increased temporal density for time series observations. For the Conterminous U.S. (CONUS), an increase of nearly 40% temporal density can be expected, using the definitions from the Landsat 7 Data Users Handbook included in **Table 2**, and the approximations for CONUS in Table 3.

Table 5.1 Image Sidelap of Adjacent Swaths	
Latitude (degrees)	**Image Sidelap (%)**
0	7.3
10	8.7
20	12.9
30	19.7
40	29.0
50	40.4
60	53.6
70	68.3
80	83.9

Table 2: Table 5.1 of Landsat 7 Data Users Handbook

CONUS Extents	**Percent Sidelap**
TX/FL $\approx 24^0$ N	$\approx$ 15.62%
WA/ME $\approx 36^0$ N	$\approx$ 20.28%
Avg. $\approx 30^0$ N	$\approx$ 19.70%

Table 3: Average Percent Sidelap for CONUS

Figure 2 presents a graphical depiction of the increased spatial and temporal density of ARD.

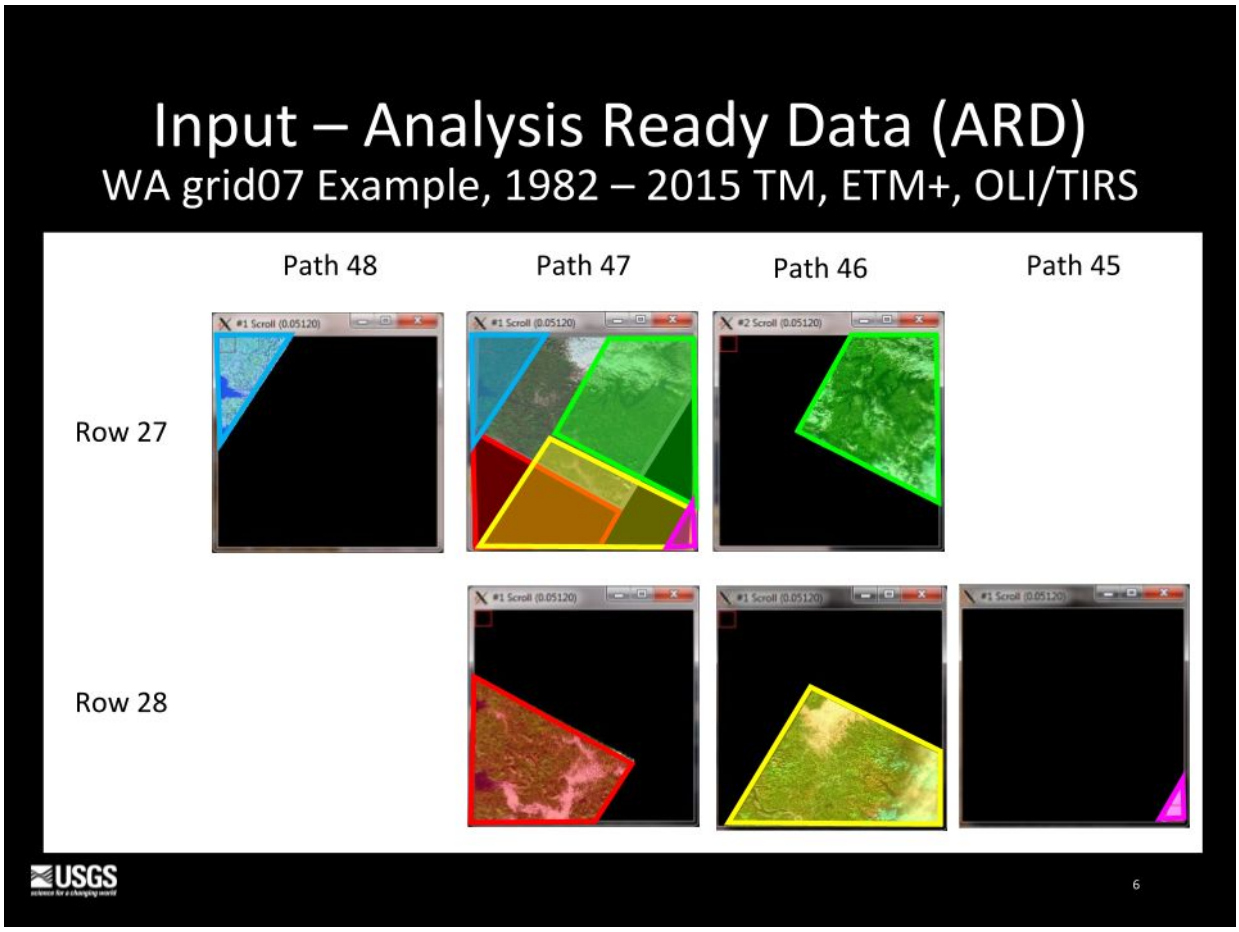

Figure 2: Graphical Depiction of ARD Temporal Density

Table 4 lists the Landat Surface Reflectance (SR) and Thermal bands used as input the the pyccd algorithm, and pre-processed to ARD specifications.

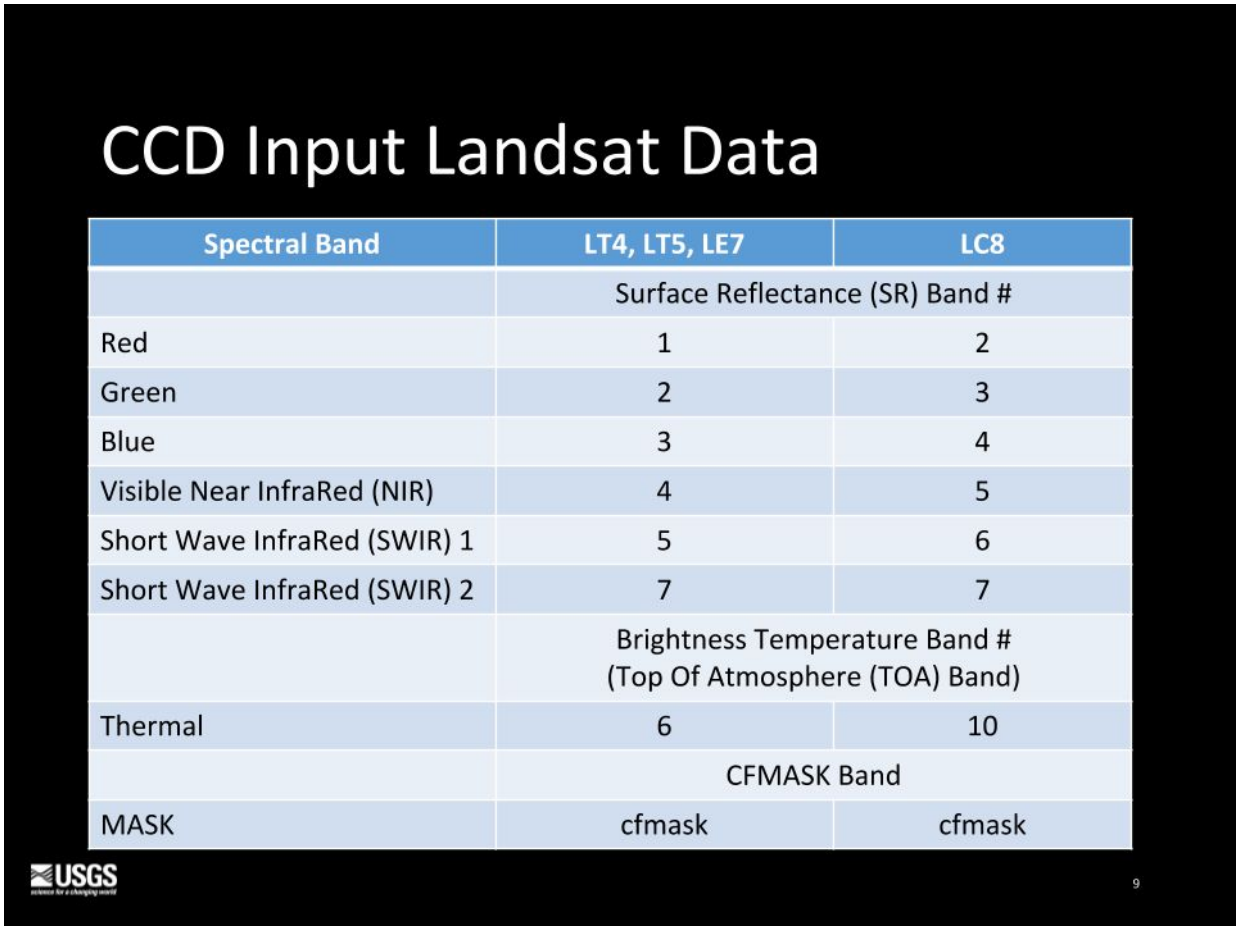

Table 4: pyccd Input Landsat Data

Terminology

To aid in the description of the processing steps, the following generic terms are introduced to clarify definitions. These terms are intended to avoid the jargon of software engineers, which may not be understood by earth science researchers, and, conversely, avoid the jargon of earth science researchers, which may be misinterpreted by software engineers.

Model	regression fit of points to a curve
Variogram	Variogram is a description of the spatial continuity of the data, usually the squares of the differences.Madogram is instead the abs of the diffs, and the “distance” between points is ignored.
MEOW	Minimum Expected Observation Window (number of coefficients * 3)
Tmask	time-series mask, outliers determined by regression fit using green (SWIR1) band

Window	current sliding array of X # number of observations (model window, peek window, fit window)
Fit Window	observations that are used to fit a regression model to, subset of the model window
Model Window	observations that are currently represented by a fitted regression model
Peek Window	observations that are currently being analyzed / under statistical scrutiny
Stable Model	generalized fit 4 coefs (min) from LASSO Regression
Detection Bands	subset of spectral bands used to determine outliers and detect change: red, green, NIR, SWIR1, SWIR2
LASSO Regression	Initial first attempt at fitting points to curve, and subsequent start detection, change detection
Robust Regression	more general purpose curve fitting for identifying outliers
Residual	predicted value from a curve fit - actual value, used as a measure of error
Persistent Processing Mask	Application of Tmask, cfmask, etc., to mask, not eliminate data from input arrays

Additionally, explanation of the mathematical models used to calculate medians and curve fitting will increase understanding of the processing procedure descriptions. Figure 4 depicts the variogram used to filter outliers in the observation data. A variogram is a generral description of the spatial continuity of the data, usually the squares of the differences of adjacent values. A specific type of variogram, a madogram, is instead the abs of the diffs. In the case of pyccd the distance between points is irrelevant, and not included in the simplified calculations.

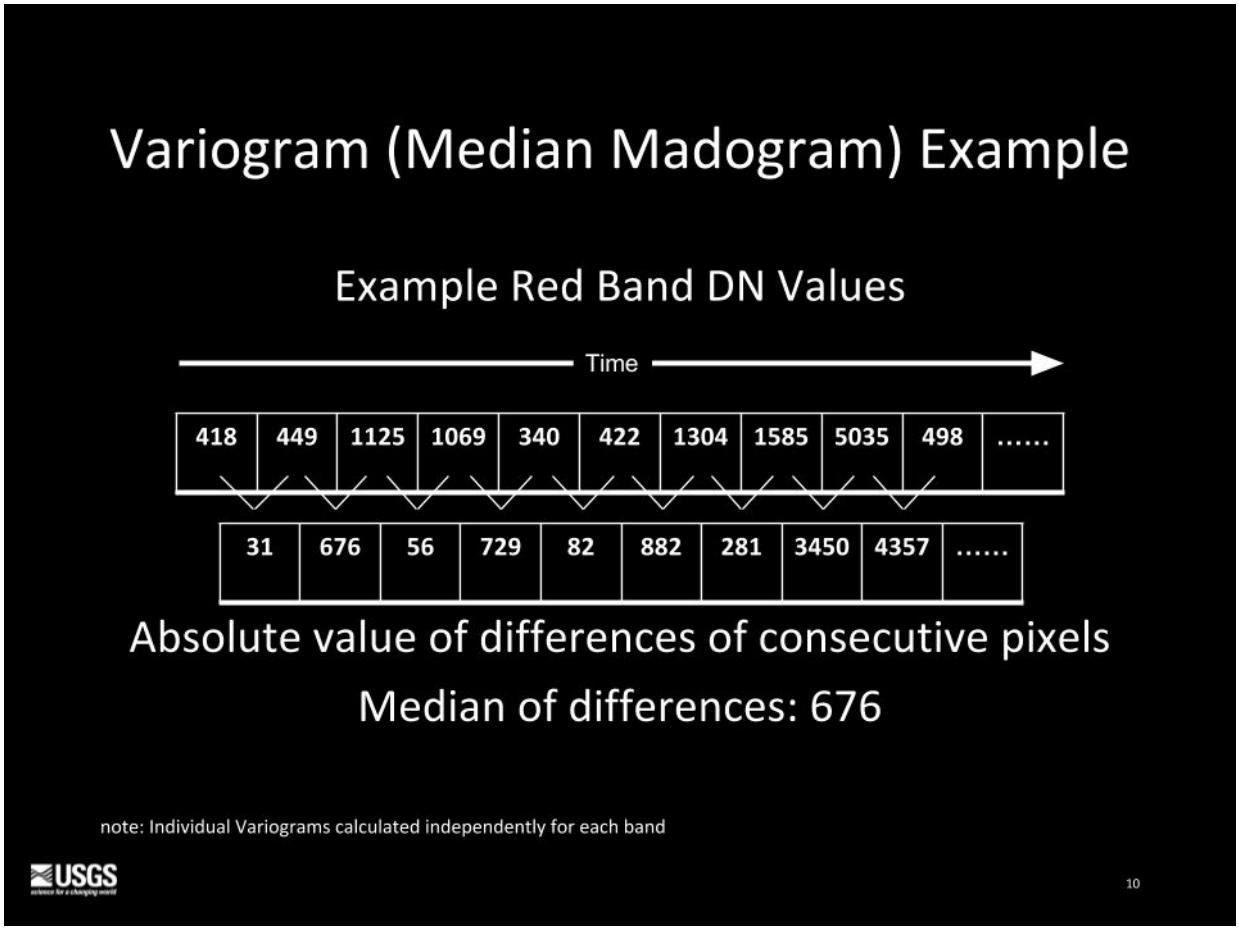

Figure 3: Variogram Example.

Least Absolute Shrinkage and Selection Operator (LASSO) regression is mathematical model used to do curve fitting of points. LASSO regression is used because the python module used is a standard all python applications can adhere to. Calculations are documented and consistent across platforms and applications. LASSO regression gives much better output than traditional step-wise regression, requiring fewer parameters and can be automated to a large extent. It is also referred to as "regression with regularization". This means the size of coefficients are penalized, in an effort to avoid both underfitting or overfitting a curve to the observations. **Figure 4** lays out the LASSO Regression formula.

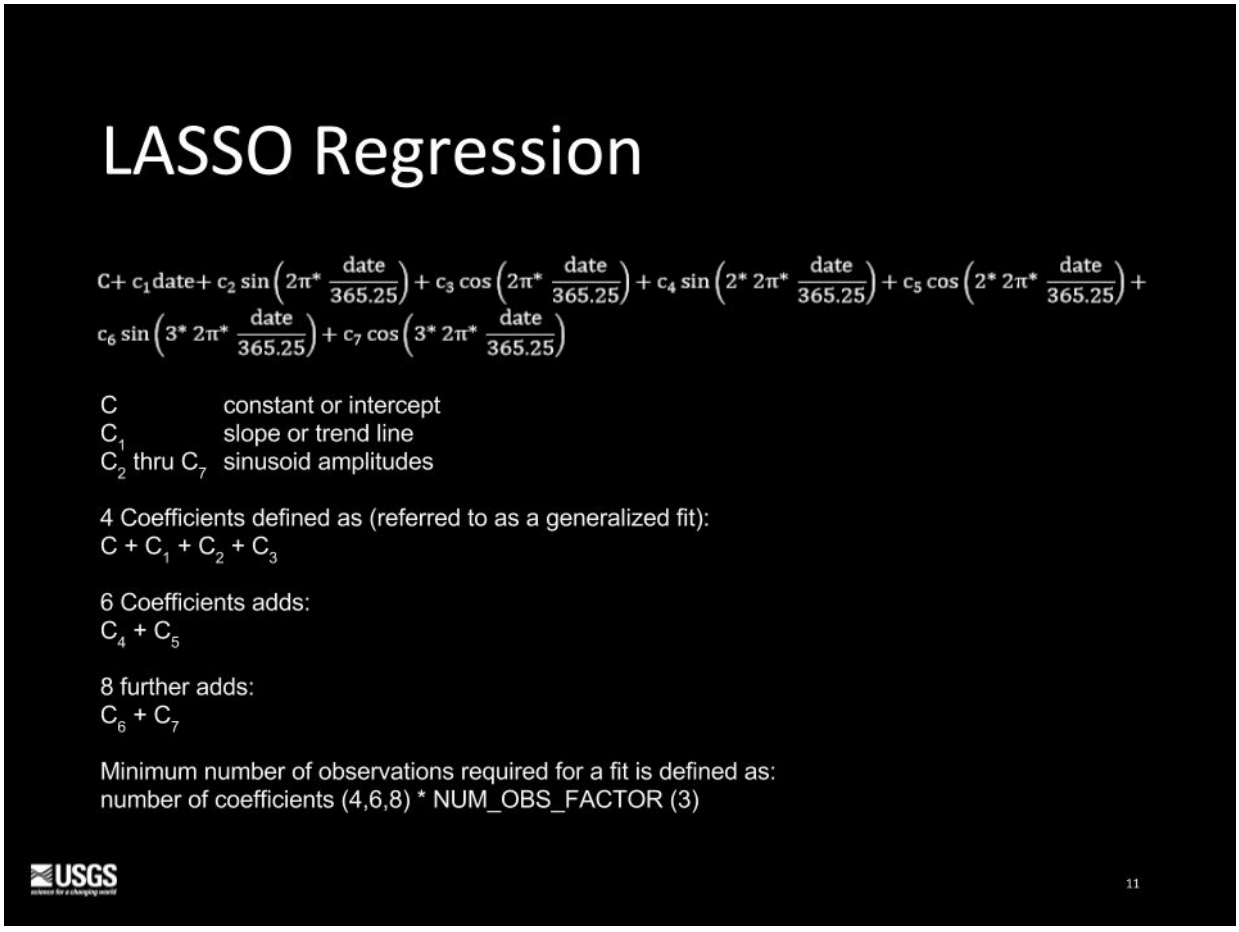

Figure 4: LASSO Regression

The Robust regression used by pyccd is a Ordinary Least Squares (OLS) implementation, also a standard python module which is well documented and platform-independent. This equation, laid out in **Figure 5**, is used to determine outliers from a previously-defined model curve, because it is a more general purpose curve-fitting algorithm, and more appropriate for determining an acceptable distance away from an established curve. It also incorporates a consideration for the temporal span of the points.

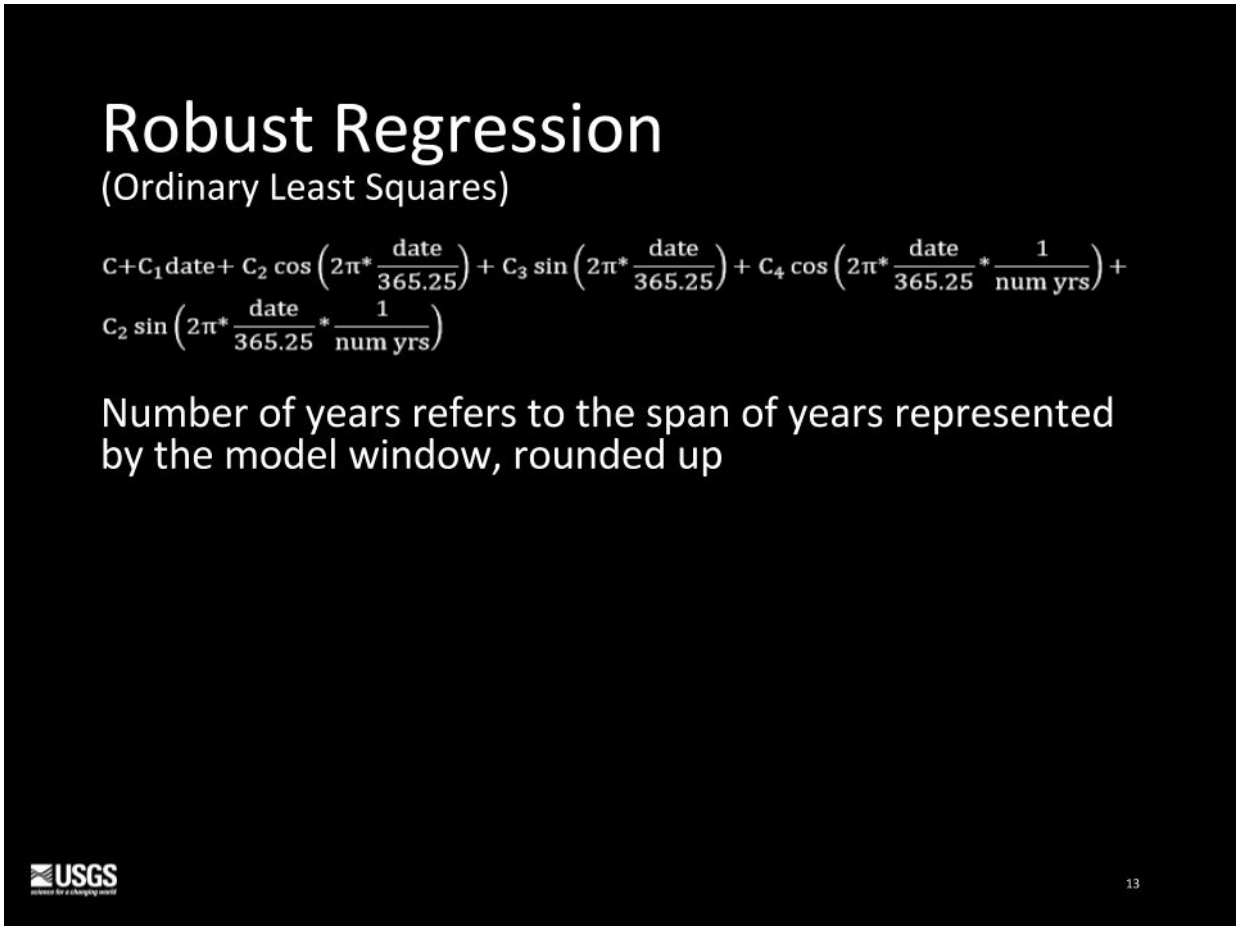

Figure 5: Robust Regression

The time series observations are filtered for clouds, cloud shadows, and snow using "Tmask," a Robust regression model, which generates time series curves of the green and shortwave infrared bands for the pixels CFmask determined are clear. Tmask model results are compared to observations to remove outliers CFmask may have missed. The pseudo-code for Tmask is include here:

use the Robust Regression method to fit curves for the green and SWIR1 bands, for a given observation window

If any residual is > the Tmask Threshold (4.89 * variogram value),
mark the observation as an outlier and update the Persistent Processing Mask

Exception Processing Procedures

Two processing scenarios result in the inability to determine automated change detection: persistent snow; and insufficient clear observations. After determining either of these two cases, processing is completed after the processing the valid observations.

1. If > 75% of valid data observations are snow, then the observation set is considered permanent snow. For this minimal set of observations, perform a generalized curve fit (if > 12 valid observations) or from median values (if < 12 points).
2. if insufficient clear observations, mask observations exceeding median of the green band + 400, then do a generalized curve fit for the remaining observation set.

The pseudo-code for determining which processing scenario to use is very simple:

```
if the percentage of clear pixels < clear threshold:
        if the percentage of snow > snow threshold:
                Persistent Snow Procedure
        else:
                Insufficient Clear Procedure
else:
        Standard Procedure
```

Standard Processing Procedure

If enough clear observations exist in the time series (greater than 25% of the total) to possibly detect change, some initial change detection setup is performed.

Mask out duplicate observations using first of two swath overlap observations

Sort temporally

Convert thermal values from degrees Kelvin to Celsius:
value = value * 10 – 27315

Create mask(s) of values that fall outside of acceptable ranges, which are considered saturated if > acceptable maximum :

Reflectance acceptable range:	0 -> 10,000
Thermal acceptable range:	-9,320 -> 7,070

Create Variogram (madogram) for each band

Figure 1 illustrates the major steps in standard processing executed by pyccd when sufficient clear observations have been determined. The standard processing steps depicted in **Figure 1** are described in more detail in the subsequent narrative, examples, and graphics.

Pseudo-code from the code repository will also be incorporated to aid description of processing details.

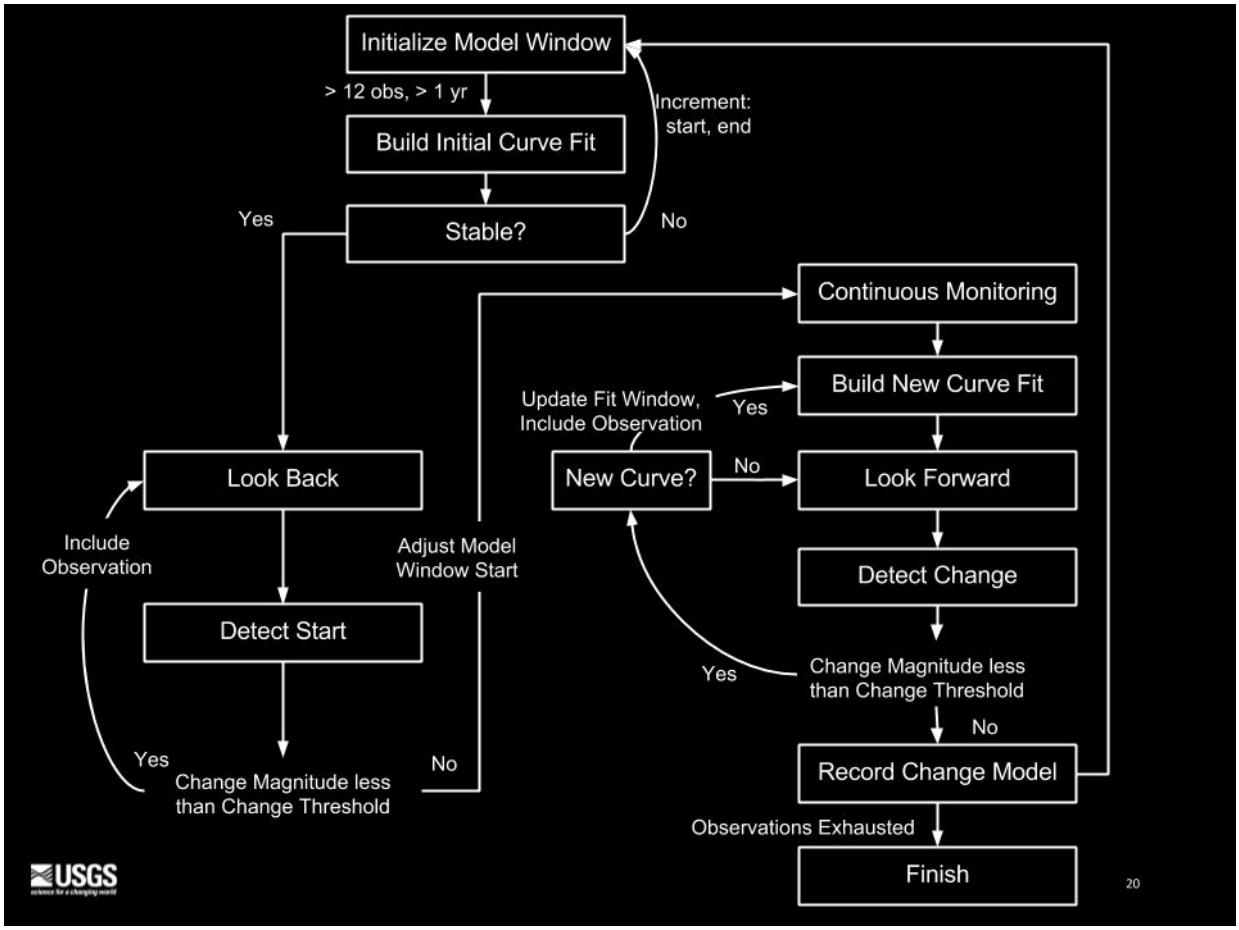

Figure 1: pyccd Work Flow

Initialize Model Window

initialize first model window
    (start is set to previous break point; window size set to contain 12 observations)
while there are clear observations:
    mask outliers in current window as defined by tmask
    if not enough observations (12) and minimum temporal span (1 year) in the window:
        increase window size by 1 continue to the next observation

Build Initial Curve Fit

perform LASSO Regression using 4-coefficient model

Stable?

for each of the detection bands (r, g, n, s1, s2)

model slope coef * diff (end – start dates)

check_vals = (|slope| + |first model residual| + |last model residual|) / MAX

(where MAX = max (variogram, model RMSE))

if summation of (check_vals)2 < change threshold:

stable

else:

increment model window start and end by 1

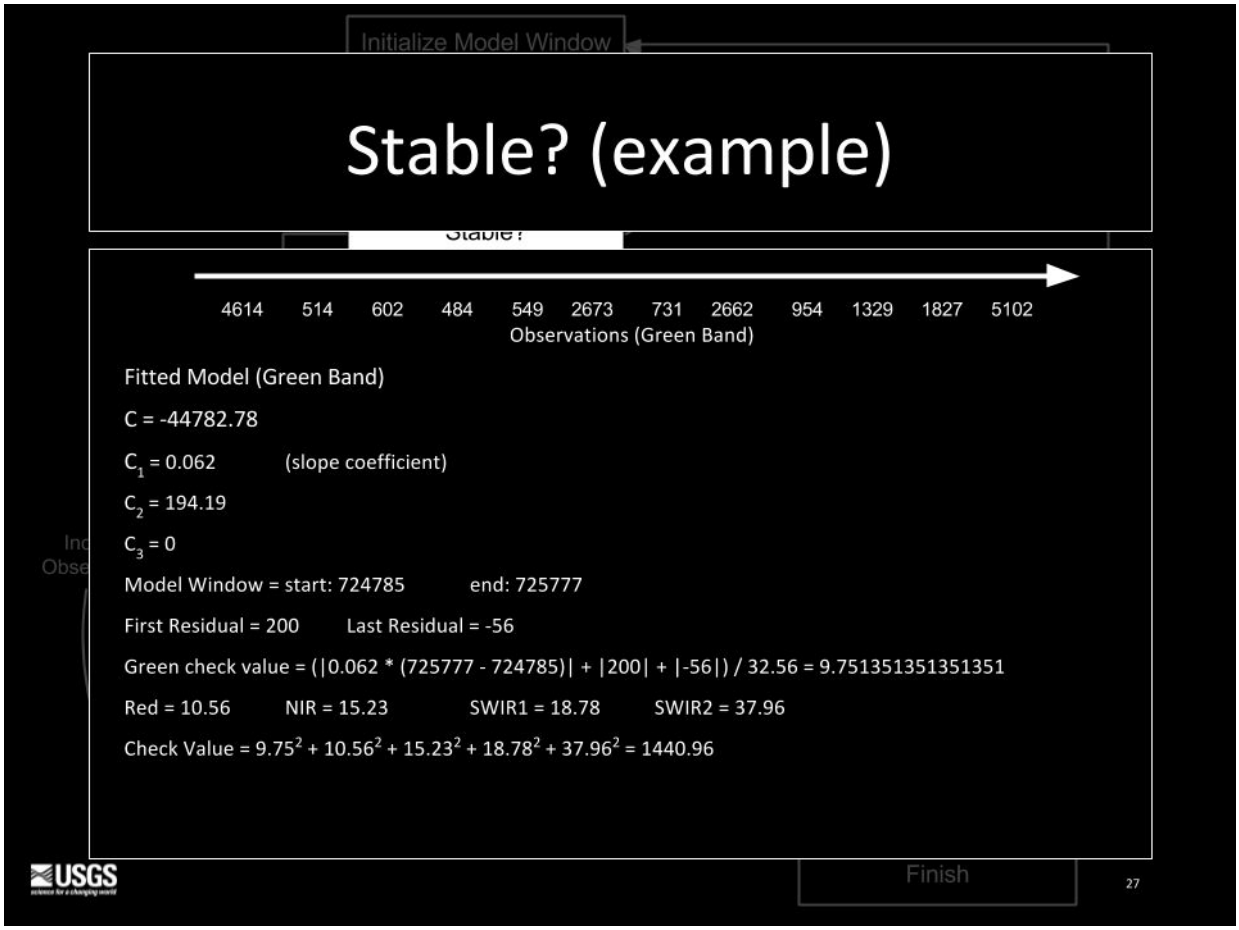

Figure 6: Stable Model Example

Look Back

for each observation preceding current model window start and end of previous model window:

If change magnitude < change threshold:

add observation to current model window

else:

proceed with continuous monitoring

note: when look-back is complete, if this is the very first model window, and there 6 or more unused observations, do a general curve fit on them

Detect Start

using the observations in the Peek Window, calculate a change magnitude per observation:

for each of the detection bands (r, g, n, s1, s2):

difference magnitude = residuals / MAX

(where MAX = max (variogram, model RMSE))

change magnitude = sum (difference magnitude ^2)

if the minimum (change magnitude) > change threshold:

change detected

if the first change magnitude > outlier threshold:

update Persistent Processing Mask

else:

include the observation in the Model Window

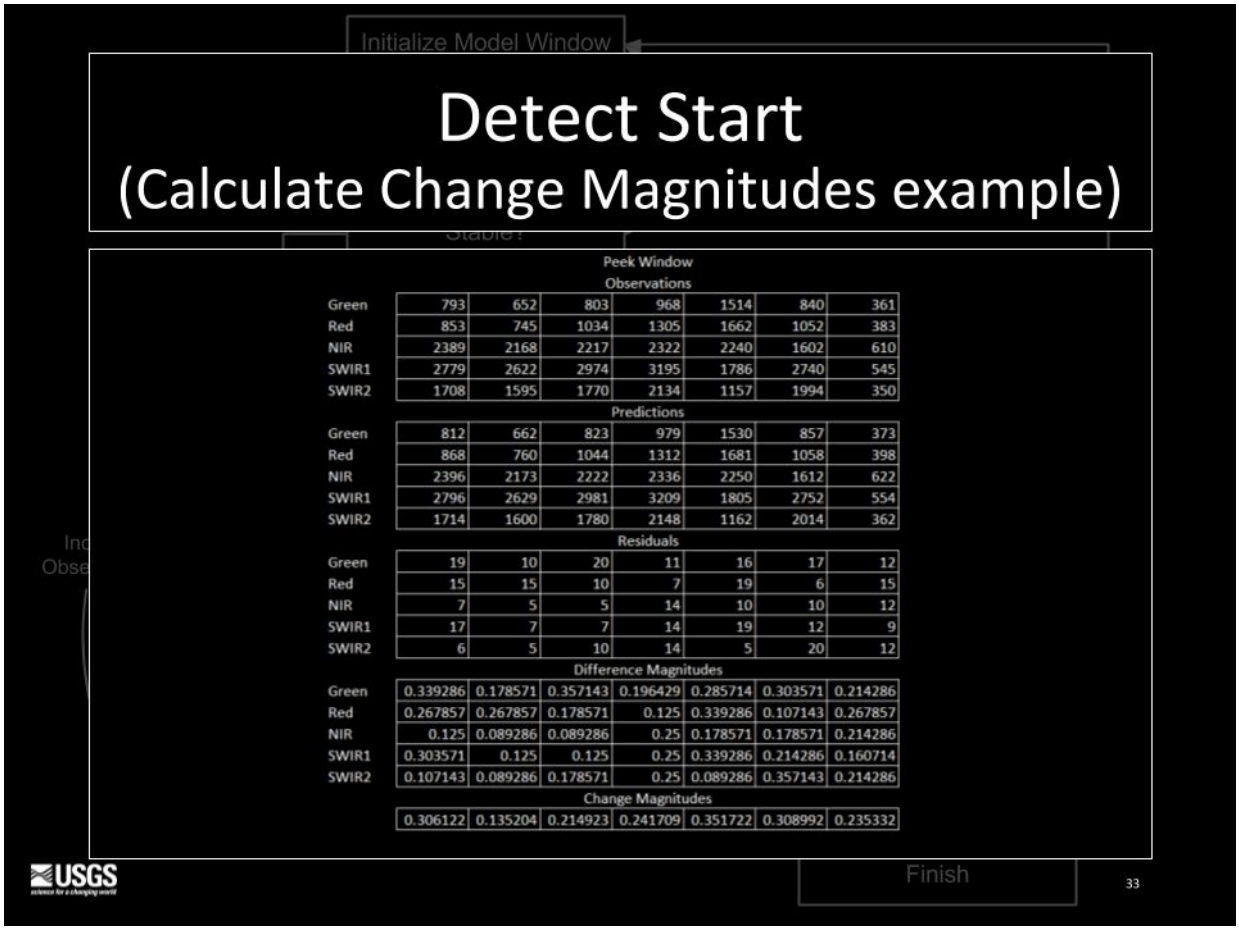

Figure 7: Detect Start Example Calculations

Change Magnitude less than Change Threshold

if Change Magnitude less than Change Threshold:

include observation

continue with Look Back

else:

adjust Model Window start

proceed with Continuous Monitoring

Continuous Monitoring

loop through remaining observations until change is detected or observations are exhausted

Build New Curve Fit

utilize LASSO Regression with a set number of coefficients based on the number of observations encompassed by the model window

Number of Coefficents per Model Window size:

# coefficents	observation count*
4	12
6	18
8	24

*number of coefficient times the NUM_OBS_FACTOR (3)

Look Forward

Look Forward is similar to Look Back, but assessing observations to the right of the existing model curve (forward in time)

Detect Change

using the observations in the Peek Window

if the Model Window > 24 observations:

model RMSE used is recalculated using the closest 24 observations temporally to the Peek Window

calculate a change magnitude per observation

for each of the detection bands (r ,g, n, s1, s2):

difference magnitude = residuals / MAX

(where MAX = max (variogram, model RMSE))

change magnitude = sum (difference magnitude ^2)

if the minimum (change magnitude) > change threshold:

change detected

if the first change magnitude > outlier threshold:

update persistent processing mask

else:

include the observation in the model window

Change Magnitude less than Change Threshold

if Change Magnitude less than Change Threshold

proceed with New Curve?

else:

Record Change Model

New Curve?

Record Change Model

start
end
break
per band coefficients
per band RMSE's
per band median of the last used Peek Window residuals
number of observations
number of fitted coefficients
change probability

Finish

if the number of remaining observations is < MEOW size but >= Peek Size:

fit a generalized model is for remaining observations

send/save output results

exit

References - CCDC

Zhu, Z., Woodcock, C. E., Holden, C., & Yang, Z., Generating synthetic Landsat images based on all available Landsat data: Predicting Landsat surface reflectance at any given time. *Remote Sensing of Environment*, *162*, 67-83, 2015, doi:10.1016/j.rse.2015.02.009

Zhu, Z. & C.E. Woodcock, Continuous change detection and classification of land cover using all available Landsat data, *Remote Sensing of Environment*, 144, 152-171, 2014, doi:10.1016/j.rse.2014.01.011

Zhu, Z., C.E. Woodcock, & P. Olofsson, Continuous monitoring of forest disturbance using all available Landsat imagery, *Remote Sensing of Environment*, 122, 75-91, 2012, doi:10.1016/j.rse.2011.10.030

References - ARD

https://drive.google.com/drive/folders/0B5wwu78kdogOfmZIWGI1S2dNRE5LeDVNeGxDU1phUndyTHpMbUVuNFdFNTZBV2RsbVhqMjQ

https://drive.google.com/a/doi.gov/file/d/0B5wwu78kdogONnN5N1BGTjFvdGM/view?usp=sharing

http://landsat.gsfc.nasa.gov/landsat-7-science-data-users-handbook/

References - Mathematical Models

http://www.gammadesign.com/gswinhelp/other_autocorrelation_measures/madograms.htm

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Acronyms

Acronym	Description
ADD	Algorithm Description Document
AOI	Area Of Interest
API	Applications Programming Interface
ARD	Analysis Ready Data
BIP	Band Interleaved by Pixel
CCD	Continuous Change Detection
CDR	Climate Data Record
CFmask	C version of Function of Mask
CSV	Comma Separated Values
DN	Digital Number
DOI	Department of Interior
EROS	Earth Resources Observation and Science Center
ESPA	EROS Science Processing Architecture

ETM+	Enhanced Thematic Mapper Plus
GeoTIFF	Georeferenced Tagged Image File Format
IW+DS	Information Warehouse & Data Store
LASSO	Least Absolute Shrinkage and Selection Operator
LCMAP	Land Change Monitoring Assessment and Projection
MATLAB	MATrix LAboratory
MEOW	Minimum Expected Observation Window
NIR	Near InfraRed
NOAA	National Oceanic and Atmospheric Administration
OLI	Operational Land Imager
RFC	Random Forest Classifier
RIRLS	Robust Iteratively Reweighted Least Squares
RMSE	Root mean square error
SR	Surface Reflectance
SWIR	Short Wave InfraRed
TOA	Top Of Atmosphere
TM	Thematic Mapper
USGS	U.S. Geological Survey