**NASA DEVELOP National Program**



NASA Jet Propulsion Laboratory

*Fall 2015*

New Mexico Water Resources II

Automating the Daily Delivery of Priestley-Taylor Jet Propulsion Laboratory (PT-JPL) Evapotranspiration Data to the New Mexico Office of the State Engineer for Enhanced Water Resource Management

 **Technical Report**

Rough Draft – November 3, 2015

Trevor McDonald (Project Lead)

Gregory Halverson

Joshua Fisher, NASA Jet Propulsion Laboratory (Science Advisor)

Greg Moore, NASA Jet Propulsion Laboratory (Science Advisor)

Previous Contributors:

Sol Kim

Agustin Muniz

Trevor McDonald

# I. Abstract

As New Mexico is experiencing some of the most severe drought in the US, equipping water resource management with evapotranspiration data becomes increasingly vital. Knowledge of rangeland conditions is necessary for decisions regarding cattle management, emergency response for rapid rangeland and farmland deterioration, fire management risk decisions, and determining drought severity. New Mexico land managers and decision-makers currently assess rangeland conditions using spatially-limited *in situ* spot checks which provides limited information. Additionally, weekly Normalized Difference Vegetation Index (NDVI) and evapotranspiration products for New Mexico counties are not widely distributed nor easily accessible. By providing an automated, streamlined, non-proprietary evapotranspiration product to the New Mexico Office of the State Engineer, New Mexico decision makers will have easy access to critical evapotranspiration data which will drive water resource decision making and drought assessment. To create the evapotranspiration product, we utilized the Moderate Imaging Spectroradiometer sensor on NASA’s satellites Aqua and Terra to retrieve six MODIS land and four MODIS atmosphere datasets.

**Keywords**

Evapotranspiration, Drought, Remote Sensing, Water Resources, Land Management, Emergency Response, MODIS, NCEP, PT-JPL

# II. Introduction

Equipping water resource management with evapotranspiration data becomes increasingly vital, as the entire state of New Mexico is experiencing some of the most severe drought in the US. Knowledge of rangeland conditions is necessary for decisions regarding water resource management, cattle management, emergency response for rapid rangeland and farmland deterioration, fire management risk decisions, and determining drought severity. This project delivered a high-resolution evapotranspiration product to New Mexico decision-makers and addressed two NASA national application areas: water resources and agriculture. Water resource managers can use the daily delivered evapotranspiration as a tool to better understand where limited water resources are and where they can afford to reduce water use. Rangeland and Agriculture communities benefit from evapotranspiration data because it can be used to quantitatively answer whether or not a crop is under or over watered.

New Mexico land managers currently assess rangeland conditions using spatially limited *in situ* spot checks, which provides limited information. Additionally, weekly evapotranspiration products for New Mexico counties are not widely distributed nor easily accessible.

By providing an automated, streamlined, non-proprietary evapotranspiration product to the New Mexico Office of the State Engineer, New Mexico decision makers will have easy access to critical evapotranspiration data which will drive water resource decision making. The analysis will start January 2000 and conclude in October 2015.

# III. Methodology

The datasets required for the PT-JPL ET model consist of fourteen level-2 datasets, including six Moderate Resolution Imaging Spectroradiometer (MODIS) land, four MODIS atmosphere, and four from the National Centers for Environmental Prediction (NCEP) in order to provide high-resolution continuous data over land for any given day. The six MODIS land imagery datasets acquired from USGS include MCD12Q1, MCD43B2, MCD43B3, MOD11\_L2, MOD13A1, and MOD15A2. The four MODIS atmosphere imagery datasets acquired from LAADS include MOD04\_L2, MOD05\_L2, MOD06\_L2, and MOD07\_L2. The four NCEP acquired from NCEP datasets include air.2m.gauss, shum.2m.gauss, tmin.2m.gauss, and uwnd.10m.gauss.

We downloaded data from their respective archive centers and uploaded it into the Jet Propulsion Laboratory institutional supercomputing machine. This process will be automated and occur on a daily basis to check for new data. The daily check provides near real-time functionality by automatically retrieving and processing additional data seamlessly without the need for manual interaction. We validated the data and stored it in a pre-processing location where the conversion process can access it. Validation will confirm that all datasets have been downloaded completely from their given data source.

Following the acquisition and/or extraction of the required datasets, a few MODIS atmospheric datasets must be processed in preparation for the PT-JPL ET model, as it requires data in a tile-by-tile basis where each tile is ten by ten degrees at the equator. The MODIS atmospheric datasets that required re-gridding and tiling are MOD04, MOD05, MOD06, and MOD07. These are all level-2 MODIS datasets that come in swath format. We initially retrieved these datasets from NASA’s level-1 and Atmosphere Archive and Distribution System (LAADS), then stored them in a pre-processing database.

The PT-JPL ET model calculates on a tile-by-tile basis to allow for parallel computing. To accomplish this transformation, the data goes through three different steps: gridding on sinusoidal projection, stitching appropriate granules together, and finally subsetting output tiles.

The primary tool used to accomplish the three-step transformation is the Hierarchical Data Format – Earth Observing System (HDF-EOS) to GeoTIFF Conversion Tool (HEG). The first step uses a Bourne Shell script to grid the swath data onto a sinusoidal projection. The shell script takes the pre-processing database and performs a swath-to-grid function, swtif, to every hierarchical data format (hdf) file that is from the same day. The parameters include: HDF as the output file type, nearest neighbor resampling type, Sinusoidal projection with a pixel size of 4633.1271653m (corresponding to 5km resolution). For every file input, the output is a new hdf file that is gridded in sinusoidal projection and an ASCII file that contains metadata.

Once all files for a given day are converted to sinusoidal grid format, another shell script determines the domain of each Sinusoidal tile grid that overlaps Earth using a text file that contains the bounding coordinates. When the tile’s domain is determined, the newly created output files containing metadata are searched to find granules that have any overlap with the given tile. If it overlaps the tile’s domain, it is recorded for use as an input file. When all the files that contain overlap with the tile are collected they are then stitched together using HEG to create one mosaicked file.

The last step is taking the mosaicked file and creating a subset of the tile’s domain and saving it to a post-processing database with the remainder of the datasets required. Note, all datasets entering the PT-JPL ET model must have a standardized projection and data format of MATLAB**.**

The PT-JPL ET model produces a global output of ET on a tile-by-tile basis in MATLAB format. In order for these products to beeligible for the front-end pipeline and end-user interface, several processes are required. Each MATLAB tile must be converted to GeoTIFF, merged to create a global extent, projected and moved to a user defined GeoServer directory. Additionally, the Apache web server and Apache Tomcat support our web interface, giving GeoServer the ability to provide a tiled web mapping service for our PT-JPL ET products. Furthermore, each GeoTIFF is input by cURL commands to GeoServer and must follow the naming convention ({DATASETname\_DATEyyyymmdd}) in order for the time query to be functional in the web interface. The features implemented for our web interface address past accessibility issues as well as allowing the user to query past dates through the time dimension alongside its statistical analysis.

# IV. Results & Discussion

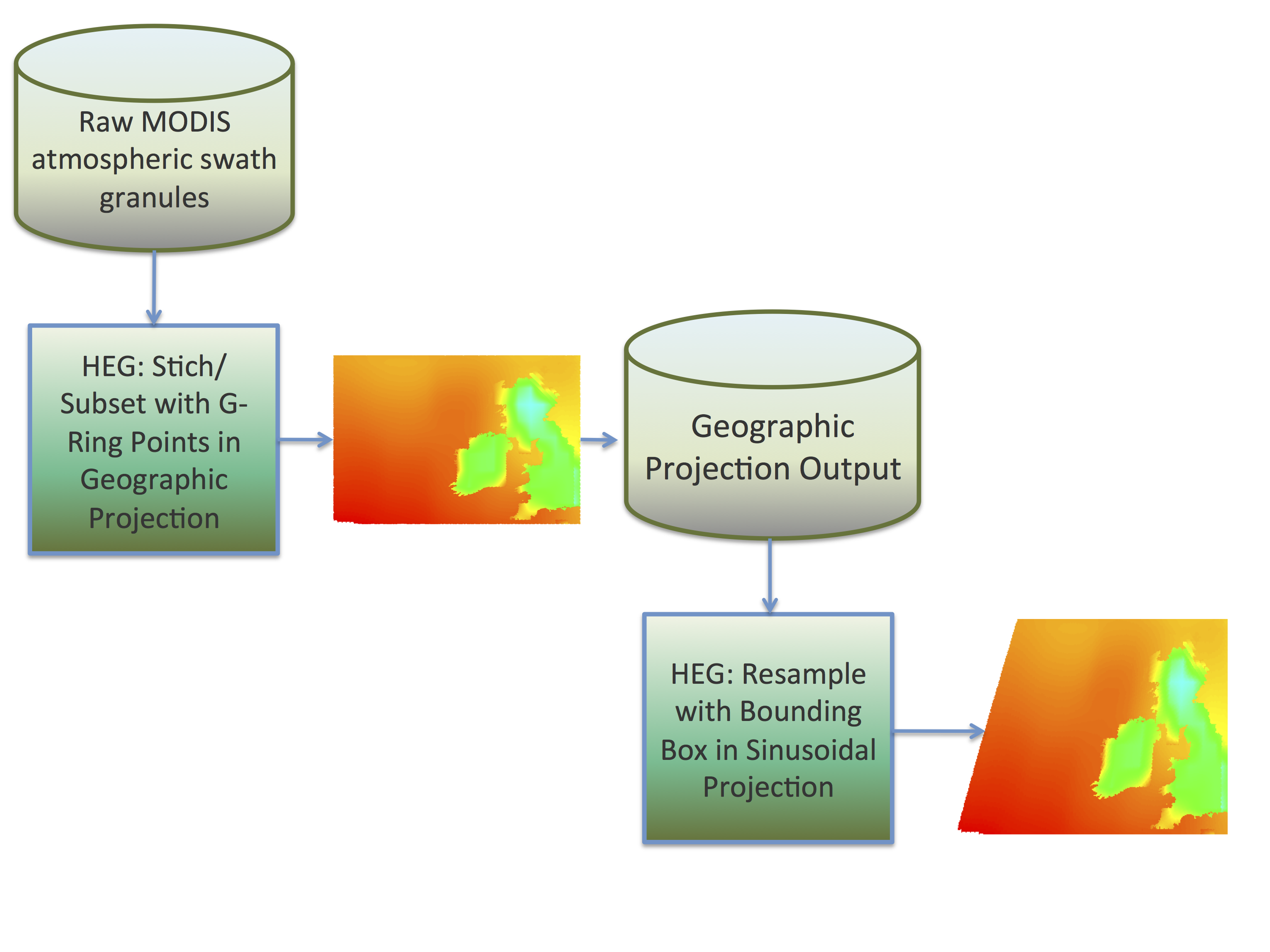
* GDAL
* HEG
* .M
* .TIF
* GDAL
* CURL
* Leaflet
* OpenLayers

Data Sources:

* NCEP
* USGS
* LAADS

**ET Data Cycle**

**Figure 1.** A visual representation of the automated data pipeline architecture. The pipeline initially requests the data, acquires it and sends some data for processing. Once processing is done, all data is fed into an ET PT-JPL model to create ET outputs. The output will be post-processed and hosted on a web server.



**Figure 2.** A schematic of the two-step re-gridding process. Raw MODIS swaths are stitched and subsetted before being projected in sinusoidal projection. The program used was the HDF to GeoTiff program or HEG.

# V. Conclusions

In the future each of these components will be completed, coordinated and automated. In order to achieve automation a master script conducting each module is required to ensure streamlined daily products and dissemination to the web interface. Moreover, we have coordinated with our partner, John Longworth of the New Mexico Office of the State Engineer to determine future functions and quantitative measures that will allow them a streamlined path to creation actionable items based off of our data analytics.

In conclusion, this project represents the development of a new paradigm that seeks to improve land management and policy through the use of NASA satellite data. The scientific community has made significant strides in modeling ecological phenomena using satellites but has failed to bridge the gap between these scientific achievements and their fullest potential by informing land managers and policy makers. Our project aims to be proof that such a paradigm can be successful in enhancing land and resource decision making.

# VI. Acknowledgments

Science advisors Joshua Fisher, Greg Moore, Manish Verma, Munish Sikka, and Laura Jewel provided insight and expertise that greatly assisted the research. Nick Rousseau, Brittany Zajic, Gwen Miller, Christine Rains, and Daniel Jensen assisted throughout the project guiding us through deliverables and answering research questions.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

This material is based upon work supported by NASA through contract NNL11AA00B and cooperative agreement NNX14AB60A.

# VII. References

Fisher, Joshua B., Kevin P. Tu, and Dennis D. Baldocchi. "Global Estimates of the Land–atmosphere Water Flux Based on Monthly AVHRR and ISLSCP-II Data, Validated at 16 FLUXNET Sites." *Remote Sensing of Environment* 112.3 (2008): 901-19. Web.

*HDF-EOS to GeoTIFF Conversion Tool*. Computer software. N.p., n.d. Web. <http://newsroom.gsfc.nasa.gov/sdptoolkit/HEG/HEGHome.html>.

*HDF-Java*. Computer software. *HDF Group*. Vers. 2.11. N.p., n.d. Web. <https://www.hdfgroup.org/>.

Hendrix, Valerie, Lavanya Ramakrishnan, Youngryel Ryu, Catharine Van Ingen, Keith R. Jackson, and Deborah Agarwal. "CAMP: Community Access MODIS Pipeline." *Future Generation Computer Systems* 36 (2014): 418-29. Web.

*Level 1 and Atmosphere Archive and Distribution System*. N.p., n.d. Web. <https://ladsweb.nascom.nasa.gov/>.

Moore, Greg. "Rn/ET Download Product Tools and Interfaces." (2014): 1-7. Web.

Python Software Foundation. Python Language Reference, version 2.7.6. Available at <http://www.python.org>

Wolfe, R.e., D.p. Roy, and E. Vermote. "MODIS Land Data Storage, Gridding, and Compositing Methodology: Level 2 Grid." *IEEE Transactions on Geoscience and Remote Sensing* 36.4 (1998): 1324-338. Web.

# VIII. Content Innovation

We implemented the data pipeline architecture to facilitate the unsupervised near real-time retrieval, processing of MODIS data and visualization of global ET via our web interface. We wrote the pipeline that consists of a data retrieval component in Python. The Python code invokes Wget commands. These Wget commands perform the heavy lifting of actually downloading the data. Wget is a non-interactive utility that retrieves content over the web. To have the download component execute regularly, a cron job is setup to invoke the Python code on a daily basis. The Cron software utility is a time-based job scheduler in Unix-like operating systems.

We have used HDF-EOS to GeoTIFF Conversion Tool (HEG) to regrid MODIS Level 2 atmospheric datasets into Sinusoidal Tile Grid format. A combination of three features within HEG is used to accomplish the task: regrid/projection, stitching, and subsetting.

We enabled end-users improved access and distribution methods for PT-JPL ET data products through our web interface. The processes coordinating the daily modeled ET output to our web interface include format conversion from MATLAB to GeoTiff, GDAL merge and project, CURL commands to GeoServer and the tiled web service pushing data to the access point, the web map. Additionally, the time dimension is available for users to query desired dates interactively within the interface.

# IX. Appendices

**Automated MODIS/NCEP Data Product Download Tool**

**NASA Jet Propulsion Laboratory, Pasadena CA**

**Agustin Muniz**

**August 2015**

**1 INTRODUCTION**

wget is used under the hood of the data fetch objects to perform the data retrieval.

**1A SETUP**

$PIPELINE\_PATH is referred to in this document as the root directory of the datafetchpipeline directory.

To setup the download component on your machine you can clone a copy of the git repository.

For versions of git 1.7+ use the following command in your terminal:

git clone https://github.com/munizas/datafetchpipeline.git

For older git versions use the following command in your terminal:

git clone git://github.com/munizas/datafetchpipeline.git

Once the repository has been cloned to your machine you need to edit the following configuration files that are placed in the config folder.

**config.json**

Edit the paths accordingly in this file

**config.py**

Edit the config\_path variable in the Config class to point to the correct path

**emailcred.json**

Edit this file to contain the correct login information for you email account to send notifications.

**emails.json**

Edit this file to contain the email address that you want to notify as part of the email service.

**data.json**

Specify your datasets here.

**preprocessproducts.json**

Specify which datasets need to be further processed by the conversion component down the pipeline.

Lastly, to setup run.py to be executed daily you can create a cron job using the following commands:

In your terminal, to create a cron job type the following

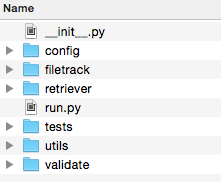
**crontab –e**

this will open a file to which you will add the following line to execute run.py daily:

@daily $PIPELINE\_PATH/run.py

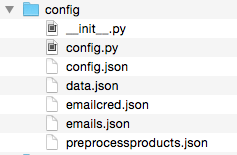
save your edits and now run.py will be executed daily at midnight. Feel free to adjust the time to your liking.

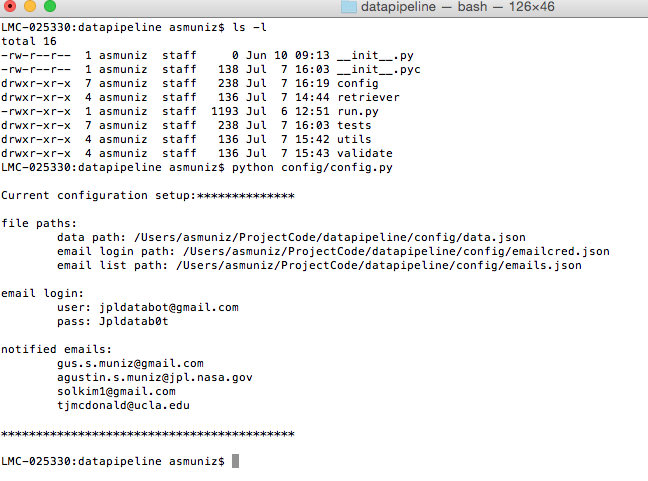
**2 ROOT DIRECTORY**



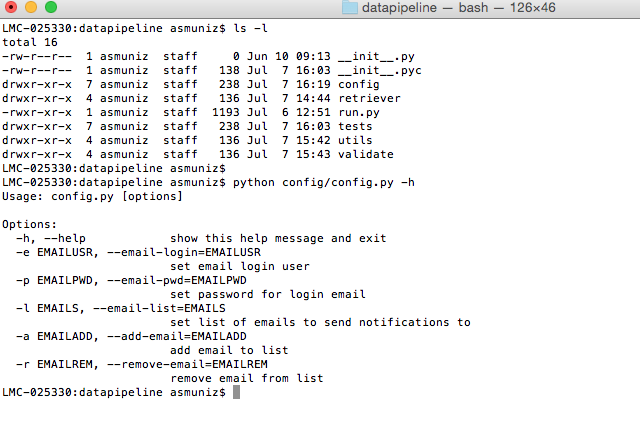
Above is the top-level view of the file structure. The driver of the download tool is run.py, which sets up data fetch objects, log validators and email. To automate the download process you can setup a cronjob on Linux/Unix-like operating systems to invoke run.py daily or on a Windows machine setup a task with TaskScheduler.

**3 CONFIG**

The config folder contains configuration information that is needed to run the tool properly. The config.py module contains the Config class, which is used to initialize data structures. The configuration data is stored in JSON format. data.json stores the information for each data product to be downloaded. emailcred.json stores the login name and password for the email account that is used to send daily update emails. emails.json stores all of the email address that are to be notified of the daily downloads. config.json stores the paths to the configuration files. preprocessproducts.json stores which datasets will eventually need to be ran through pre-process components. You can directly edit these JSON files to add, remove or change any data. This manual editing is prone to typing errors so a command line utility is provided to help with this task.

config.py can be invoked to display the current configuration.

config.py can also be invoked with flags to change the email login/password as well as adding and removing emails from the notification list.



For example, to set the list of notified emails the following command will do just that:

$ python config/config.py

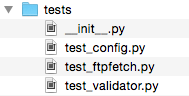
gus.s.muniz@gmail.com,testemail@yahoo.com

Notice that the emails are separated by a single comma with no spaces in between.

**4 RETRIEVER**

Macintosh HD:Users:asmuniz:Desktop:Screen Shot 2015-07-07 at 3.22.59 PM.pngThe retriever folder contains the datafetch.py module, which houses the fetch classes (FtpFetch, USGSFetch, NERSCFetch). These classes perform the actually work of retrieving data through their fetch methods. Currently there are three fetch classes. FtpFetch is used to retrieve data from any ftp site. USGSFetch is for retrieving data specifically from e4ftl01.cr.usgs.gov and NERSCFetch is for pulling data from portal.nersc.gov. To see how these classes are used take a look at run.py.

Log files are generated by wget and are stored in a logs directory, which is located in the user specified data save directory.

****

**5 TESTS**

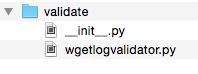
This folder contains unit tests that were used during development.

**6 UTILS**

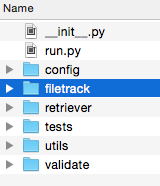
**Macintosh HD:Users:asmuniz:Desktop:Screen Shot 2015-07-07 at 3.44.39 PM.png**

The GmailSend class is located in the emailutils.py module. GmailSend is used to prepare and send an email through the gmail service.

**7 VALIDATE**

****WgetLogValidator is an object that is used to parse through wget log files and make sure files were properly downloaded.

**8 FILETRACK**



The filetrack folder is used to store all of the daily preprocess.txt files. The preprocess.txt files contain the paths to the downloaded data for the given day. These text files are used as an interface to any processing component that needs to work on the downloaded data. The text files provide location of the data.

**9 NEXT STEPS**

**(08/03/15)**



Add additional validation to downloaded datasets such as checksum.

**Documentation for Regridding Procedure**

**Sol Kim**

**NASA Jet Propulsion Laboratory, Pasadena CA**

**Written for LINUX system**

**Tested and Compatible on Zodiac**

**August 2015**

**INTRODUCTION**

This document goes over step-by-step on how to get the regridding software and scripts set up on zodiac. Regridding is performed on raw unprojected MODIS Atmospheric swaths into sinusoidal tiles. Format is left in .hdf through entire process. Regridding at the current stage has several issues which can be found in the “CURRENT ISSUES” section. Command line codes will be denoted with a “>” symbol. For this document, MAC OSX (UNIX based) was used as local and zodiac (LINUX based) as the remote.

For your reference:

MODIS Swath gridding and mapping:

<http://modis-atmos.gsfc.nasa.gov/MOD04_L2/grids.html>

MODIS Sinusoidal Tile Grid:

<http://modis-land.gsfc.nasa.gov/MODLAND_grid.html>

The regridding occurs in two steps due to issues with the HEG program running into errors.

1. For each sinusoidal tile, raw MODIS swaths are stitched together that overlap the tile. The stitched area is subsetted according to the sinusoidal tile’s G-Ring points. Finally, the stitched and subsetted area is projected with geographic projection. Errors occur if you try to go directly to sinusoidal projection.
2. The geographically projected tiles are then reprojected in sinusoidal projection and bounding boxes are applied according to the sinusoidal tile’s bounding box.

The G-Ring points and bounding box coordinates are found here as text files: <http://modis-land.gsfc.nasa.gov/MODLAND_grid.html>

**INSTALLING HEG**

1) Download HDF-EOS to GeoTIFF Conversion Tool (HEG) onto your local computer: <http://newsroom.gsfc.nasa.gov/sdptoolkit/HEG/HEGHome.html>

* zodiac requires the hegLNX version for the LINUX platform

2) Secure copy this .tar.gz file into your zodiac home directory:

>scp hegLNXv2.12.tar.gz [USER@zodiac.jpl.nasa.gov:~/](mailto:USER@zodiac.jpl.nasa.gov:~/)

3) From your home directory in zodiac, uncompress the tar.gz file

>tar –xzvf hegLNXv2.12.tar.gz

4) Run the install script with:

>./install

* Follow the instructions that the install script asks and provide the information required
* Java 1.7 or greater is required. On zodiac you may need to run the following commands to load the java module before installing:

>module avail

\*This brings up a list of modules available on zodiac. Find Java and load the module with:

>module load java/jre-7u7

5) Create a script to set some environment variables to allow HEG to run. From your home directory:

>cd heg/bin

>cat HEG

You will have to add these environment variables to variable script. I run heg from C shell (csh) so I set my variables with C shell syntax. My script is below:

#! /bin/csh

setenv MRTDATADIR /home/kimsol/heg/data

setenv PGSHOME /home/kimsol/heg/TOOLKIT\_MTD

setenv HEGUSER SOL

setenv LD\_LIBRARY\_PATH /home/kimsol/heg/bin

setenv OMP\_NUM\_THREADS 1

Save this script on zodiac and run it with:

>csh

>source FILENAME.sh

If you want to set these variables automatically everytime you log on to zodiac, you can edit your .bashrc file in your home directory to export these environment variables.

If you run into issues not covered here follow the install instructions on: <ftp://edhs1.gsfc.nasa.gov/edhs/HEG_Tool/Version2.12/516eed001_HEG_UsersGuide_v2.12_Rev01.pdf>

**INSTALLING HDF SOFTWARE**

1. HDF software and instructions for installation can be found here: <https://www.hdfgroup.org/release4/obtain.html>

* Download the Linux platform version of HDF4
* Only need HDF4 for the ncdump utility
* Installation instructions are clear on the website

DOWNLOADING MODIS ATMOSPHERIC DATASETS

1. Find a MODIS Atmospheric data set on (MOD0#\_L2, # = 4-7): <ftp://ladsweb.nascom.nasa.gov/allData/51/>

2. From zodiac make a directory to store your MODIS data.

>mkdir MOD06\_2004\_01

>cd MOD06\_2004\_01

3. From your new directory, ftp the directory of data you want. For this exercise I use MOD06\_L2/2004/001 as the directory of data.

>ftp -i <ftp://ladsweb.nascom.nasa.gov/allData/51/MOD06_L2/2004/001/>

(the –i flag allows you to avoid the prompt confirming each file)

>mget \*.hdf

(grab every file ending with .hdf)

**STEP ONE: STITCHING/SUBSETTING SWATH INTO GEOGRAPHIC PROJECTION**

1. You will need to secure copy these files into your home directory with all your MODIS data. These files can be found on the New Mexico Ag & Water Flash Drive.

* gring\_coord.txt
* template\_sw.prm

2. You will need to secure copy gring\_stitch\_subset.sh into the directory with your MODIS data. This script is on the flash drive.

3. Copy ncdump utility from the HDF4 software package into your home directory. “cd” into where the utilities are stored (the bin directory under the hdf directory) then “cp” the file into MOD06\_2004\_01.

4. Open the gring\_stitch\_subset.sh file and make a few edits according to your data set.

* Edits will be made to
  + “dir\_path” line 99 according to your current directory where the MODIS data is stored
  + “obj\_name” line 184 according to your MODIS data set
  + “field\_name” line 185 according to the field you are extracting from your data set
  + the directory path on line 200 according to where the subset\_stitch\_swath utility is stored. Should only have to change out “kimsol” with your user
* These edits can be made in an automated fashion by reading the metadata but I have not yet gotten to that yet. This can be easily fixed!

5. Open the template\_sw.prm file and make a few edits

* Only the OUTPUT\_FILENAME and OUTPUT\_STITCHED\_FILENAME need to be edited to where you want the output to be stored. Should only have to change the home directory by replace “kimsol” with your user. Keep in mind that “obj\_name,” “field\_name,” “iv”, and “ih” will be replaced by the gring\_stitch\_subset.sh script.

6. Make the directories where the output will be stored according the template\_sw.prm file.

7. In the directory with the MODIS data, run the gring\_stitch\_subset.sh script

>./gring\_stitch\_subset.sh

8. As the code is running, you should see a flow of outputs from the script. Check your output files and see that the files are being created successfully. Performing a visual check of the data at this stage is helpful.

**STEP TWO: PROJECTING INTO SINUSOIDAL**

1. You will need to secure copy these files into your home directory with all your MODIS data. These files can be found on the New Mexico Ag & Water Flash Drive.

* tile\_coord.txt
* template\_sin.prm

2. You will need to secure copy geo\_sin.sh into the directory with your STEP ONE outputs. This script is on the flash drive.

3. Open the geo\_sin.sh file and make a few edits according to your data set.

* Edits will be made to
  + the directory path on line 68 according to where the resample utility is stored. Should only have to change out “kimsol” with your user
* These edits can be made in an automated fashion by reading the metadata but I have not yet gotten to that yet. This can be easily fixed!

4. Open the template\_sw.prm file and make a few edits

* Only the INPUT\_FILENAME and OUTPUT\_FILENAME need to be edited to where you want the output to be stored. Should only have to change the home directory by replace “kimsol” with your user.

7. In the directory with the STEP ONE outputs, run the geo\_sin.sh script

>./geo\_sin.sh

8. As the code is running, you should see a flow of outputs from the script. Check your output files and see that the files are being created successfully. Performing a visual check of the data at this stage is helpful.

**VISUAL CHECK**

1. The program I found to be most compatible opening up HEG outputs on a Mac is HDFView. The download link and install procedure can be found here:

<https://www.hdfgroup.org/products/java/release/download.html#download>

I used this program locally on my mac so I downloaded the mac OS X version but they also have a Linux version if you wanted to visually check them within zodiac so you do not have to go through transferring files between local and remote.

2. Once the program is installed, open it up. You can drag any kind of hdf file into the program to have it show up on the left side of the HDFView screen. Under your file’s folder, there should be a “Data Fields” directory with a list of data fields available on that file. If it is a HEG output, it should only have one field. To visually see the file, right click the field and choose “Open As.” From here you have two option, both of which are helpful, “Spreadsheet” view or “Image” view. “Spreadsheet” gives you all the values within your data field. It is helpful to open this one up first to determine three items for “Image” view:

-Minimum value (approximate)

-Maximum value (approximate)

-Invalid value

3. Next, open the data field as an image and replace the “Valid Range” min and max with the values you got from the spreadsheet. Also replace the invalid value with the one you got from the spreadsheet. Most I have seen are “-32768” or “-999.” “Rainbow” provides a good palette choice. Click “Ok” and an image should be shown!

**Documentation for ET Data Cycle**

**Gregory Halverson**

**NASA Jet Propulsion Laboratory, Pasadena, CA**

**October 2015**

**INTRODUCTION**

The ET Data Cycle is a Python program that integrates the Automated MODIS/NCEP Data Product Download Tool, Regridding Procedure, PT-JPL ET Algorithm, and ET Front-End components into an automated system. The Automated MODIS/NCEP Data Product Download Tool has been incorporated and expanded into a Python infrastructure for the ET data cycle. The Regridding Procedure has been converted into Python as the first of several necessary preprocessing directives to prepare ancillary data for the PT-JPL ET Algorithm.

The PT-JPL ET Algorithm is under development in MATLAB as part of the ECOSTRESS mission, and is unavailable as a component in a prototype for the ET Data Cycle in the second term of this project. The second half of the second term of this project focused on designing the ET Front-End through communication with the New Mexico Office of the State Engineer. Future work on this project will incorporate the PT-JPL ET Algorithm and ET Front-End into the ET Data Cycle as an automated system for daily delivery of evapotranspiration data.

**ENVIRONMENT**

The testing environment for this project uses the Anaconda 2.3.0 (64-bit) distribution of Python 2.7.10 and GDAL 1.11.2 on SUSE Linux Enterprise Server 11.

An installation BASH script for Anaconda Python can be obtained for free from <https://www.continuum.io/downloads>.

WGET or download and secure-copy **Anaconda2-2.4.0-Linux-x86\_64.sh** to your home directory. Use **chmod +x** to make the file executable if necessary, and call the script from the BASH command line. Add **export PATH=$HOME/bin:$HOME/anaconda/bin:$PATH** to your .bashrc script so that Python and other software installed are run from the home directory.

Compiling GDAL without root privileges on SUSE Linux requires the installation of multiple dependencies in the home directory.

Following the “OCI without installing Oracle” directions on the GDAL website, the first dependency is Instant Client: <https://trac.osgeo.org/gdal/wiki/BuildingOnUnix>. Make sure to use **--prefix=$HOME** in the **configure** stage of this and all dependencies.

GDAL on Linux also depends on sqlite3. WGET or download and secure-copy **sqlite-autoconf-3070603.tar.gz** from <https://www.sqlite.org/download.html>.

Unpack the sqlite3 source code using **tar –zxvf autoconf-3070603.tar.gz.** CD to the **autoconf-3070603** directory.

Use **./configure --prefix=$HOME --with-pic; make; make install** to install sqlite3 to the home directory with shared library support.

GDAL requires the HDF4 library to read the NASA HDF-EOS2 data format used by MODIS.

WGET or download and secure copy **hdf-4.2.11.tar.gz** from <https://www.hdfgroup.org/ftp/HDF/releases/HDF4.2.11/src/>.

Unpack HDF4 with **tar –zxvf hdf-4.2.11.tar.gz**. CD to the **hdf-4.2.11** directory.

Use **./configure --prefix=$HOME --with-pic; make; make install** to install HDF4 to the home directory with shared library support.

WGET or download and secure-copy **gdal-1.11.2.tar.gz** from <https://trac.osgeo.org/gdal/wiki/DownloadSource>.

Unpack GDAL with **tar –zxvf gdal-1.11.2.tar.gz**. CD to the **gdal-1.11.2** directory.

Use **./configure --prefix=$HOME --with-sqlite3=$HOME --with-hdf4=$HOME --with-pic** to install GDAL with HDF4 support to the home directory.

Finally, use **pip install pygdal** to install the Python bindings for GDAL.

**DATA REQUEST**

The Data Request module interprets requests for ancillary data inputs to the PT-JPL ET model. Ancillary data requests are read from JSON-encoded text files in the **ancillary\_definitions** directory into Python dictionary data structures. The filename convention for ancillary data definitions is **[ancillary data name].definition.json**.

For example:

**surface\_temperature.definition.json**

{

  "name": "surface\_temperature",

  "product\_name": "MOD06\_L2",

  "object\_name": "mod06",

  "field\_name": "Surface\_Temperature",

  "preprocessor\_directives": ["regrid", "reproject"],

  "regrid": "modis\_land\_gring",

  "reproject": "WGS84"

}

The ancillary data definitions reference the data product and NETCDF object and field name from which to extract the data. Preprocessor directives needed to transform the data into a format compatible with the PT-JPL ET model are also listed in the ancillary data definitions.

At the end of the second term of this project, the ancillary data inputs to the PT-JPL ET model are still being defined and are subject to change. This system is being designed with the requirement that ancillary data for the PT-JPL ET model may be requested from a variety of sources and transformed into any resolution and projection necessary for future versions of the model.

**DATA ACQUISITION**

The Data Acquisition module retrieves data specified by the Data Request module. The **product\_name** field of the ancillary data definitions specifies the source data products for the Data Acquisition module to retrieve.

**DATA SOURCES**

The source of data for a requested data product is resolved using definitions stored in the **product\_definitions** directory with the filename convention of **[product name].definition.json**.

For example:

**MOD06\_L2.definition.json**

{

"name": "MOD06\_L2",

"description": "MODIS/Terra Cloud",

"host": "LAADS",

"extensions": ["hdf", "xml"],

}

The product definitions are read from JSON-encoded text files to Python dictionary data structures. The **host** field of the product definitions directs the Data Acquisition model to the LAADS, USGS, or NCEP servers.

The NERSC source of MODIS data used in the Summer 2015 project provided an archive of atmospheric datasets until 2013. It was replaced by the Level 1 and Atmosphere Archive and Distribution System (LAADS) in Fall 2015, which hosts MOD04\_L2, MOD05\_L2, MOD06\_L2, and MOD07\_L2 daily to the present day.

**PROTOCOL INTERFACE**

To accommodate the need for changing data sources, two modules were written, FTP\_interface and HTTP\_interface. These modules can be used to traverse filesystems hosted using FTP and HTTP protocols. The Protocol Interface modules are used by the Host Interface modules to access the LAADS, USGS, and NCEP data sources.

**HOST INTERFACE**

In order to monitor the datasets and dates available on the data sources, three modules were written under **data\_acquisition.host\_interface**: LAADS, USGS, and NCEP. These modules can be used to determine which datasets are available on each site, what dates are available for these datasets, and filter available dates against a given range or provide the most recent data for a relative date.

The Host Interface modules are used to access the data sources specified by the product definitions. Host objects are passed to the fetch objects to build lists of files to download for each project.

**FETCH OBJECTS**

The fetch objects from the Data Product Download Tool have been reconfigured for the new data sources. They can be found under the **data\_acquisition.retriever** module as LAADS, USGS, and NCEP. The new versions of the fetch objects now take date ranges as parameters. This will allow separate processes to run for the daily data cycle and for calculating historic PT-JPL ET products. The fetch objects now build lists of individual files to download and pass their addresses to the download manager in order to have more control and feedback in the data retrieval process.

The fetch objects are tailored for each data source. The fetch objects take a host object to interface a data source corresponding to the host referenced in the product definitions, the product name referenced by the ancillary data definitions, the location to store the requested data, and a date range for the requested data. The host object is used to search for files corresponding to the product name and date range specified, and the addresses of these files are passed to the Download Manager.

**DOWNLOAD MANAGER**

The Download Manager wraps the functionality of WGET and stores its output as log files that can be analyzed by the WgetLogValidator module. The Download Manager takes a file address on a remote server and stores the file in a specified directory on the local machine. WGET called in the environment to perform the file transfer because of its download validation capabilities. The log files produced by these calls to WGET are collected and analyzed into a download report by the WgetLogValidator module and can be emailed to interested parties specified in **config/emails.json**.

**PREPROCESSING**

The source data for the ET algorithm comes in diverse resolutions and projections. So a preprocessing phase in the data cycle is necessary to homogenize the data before passing it to MATLAB. This module uses GDAL to access the layers of data contained within the HDF format used by MODIS.

The Preprocessing module uses the NETCDF object and field names specified in the ancillary data definitions to extract the requested data from the retrieved data. A variety of preprocessor directives are interpreted and executed from the ancillary data definitions.

The **gring\_stitch\_subset.sh** and **reproject\_sinusoidal.sh** BASH scripts from the Regridding Procedure in the Summer 2015 project have been converted into the Python scripts **gring\_stitch\_subset.py** and **reproject\_sinusoidal.py**, and absorbed into the **preprocessing.HEG\_interface** module of the ET Data Cycle as the initial preprocessing directives available.

These scripts wrap the functionality of the HDF to EOS Conversion Tool for converting the swaths of MODIS level 2 atmospheric products into MODIS land tiles to match the current development requirements of the PT-JPL ET algorithm. The coordinates of retrieved MODIS atmospheric swaths are compared to a table of g-ring coordinates to determine which swaths are passed to the HEG tool to produce each MODIS land tile.

Testing of the Regridding Procedure both in BASH and in Python resulted in many tiles being rejected by HEG, especially when positive and negative coordinates are mixed when tiles overlap the meridian. Current development of this project focuses on resolving this problem, potentially by replacing HEG calls with GDAL. Other preprocessing directives will need to be defined as well to allow future versions of the PT-JPL ET algorithm to request data from a variety of sources in a consistent format.

**MATLAB INTERFACE**

This module will be the core of the ET data cycle. Once ancillary data have been acquired, reprojected, and resampled, they will be converted into MATLAB matrices for parallel processing on the Zodiac supercomputer. Future work on this project will connect the ancillary data request back-end in Python to the PT-JPL ET algorithm in MATLAB.

**POSTPROCESSING**

The output of the ET algorithm will be converted into GeoTIFF files for display on the GeoServer.

**GEOSERVER INTERFACE**

This module will communicate with the Evapotranspiration Front-End. The output of the ET algorithm will be converted into GeoTIFF files for display on the GeoServer.

**Documentation for ET Front-End**

**NASA Jet Propulsion Laboratory, Pasadena CA  
Trevor J McDonald**

**August / October 2015**

**1 ACQUIRING PT-JPL ET DATA FROM ZODIAC**

a. # Login Zodiac account to retrieve PT-JPL ET product

**ssh -Y trevorjm@zodiac.jpl.nasa.gov**

**--> INSERT PASSWORD <--**

b. # cd to location of files

**cd /nobackupNFS/rnet/gmoore/RnET/results/PTJPL/LEptJPL/2004/153**

c. # Secure Copy .mat files to designated location

**scp -r trevorjm@zodiac.jpl.nasa.gov:/nobackupNFS/rnet/gmoore/RnET/results/PTJPL/LEptJPL/2004/153 /Users/trevorjm/Desktop/**

**--> INSERT PASSWORD <--**

\*\* # A folder named 153 on your Desktop should contain all .mat files from file path: /nobackupNFS/rnet/gmoore/RnET/results/PTJPL/LEptJPL/2004/153

**2 RUN ET\_GEOTIFF.M**

# This MATLAB script will convert each tile to .tif

# Example data found at:

/nobackupNFS/rnet/gmoore/RnET/results/PTJPL/LEptJPL/2004/153

# Assign path for input files from the ET algorithm output .mat files

cd /Users/trevorjm/Desktop/153;

# Assign path:

tifFileName = ['/Users/trevorjm/Desktop/tests/test2', '/', fileName(1:end-4)];

**3 MERGE OUTPUT .TIF(S) WITH GDAL MERGE TO PRODUCE GLOBAL EXTENT**

# cd into directory where output tiffs are (from MATLAB)

# Be in the directory of converted .mat files to .tif

**gdal\_merge.py -o output.tif $(ls \*.tif)**

**4 DEFINE & PROJECT SINGLE .TIF WITH GDAL WARP**

# Template: gdalwarp -s\_srs EPSG:4326 -r near -te -20037508.34 -20037508.34

20037508.34 20037508.34 output.tif testa.tif

# Be in the directory of converted .mat files to .tif

**gdalwarp -s\_srs EPSG:54008 -t\_srs EPSG:3857 -r near -te -20037508.34 -20037508.34 20037508.34 20037508.34 globaltest.tif globaltest1B.tif**

**5 PREPARING DATA FOR GEOSERVER WITH CURL COMMANDS, MOVE TO DIRECTORY**

# Move merged & projected .tif to GeoServer

\*\* Move the output file to the designated location that you will call in your 4b1 cURL command \*\*

# I.E.

"file:///Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles/NM.shp"

# Template mv ~/Desktop/output.tif

/Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles

**6 USE CURL COMMANDS TO IMPORT AND ENABLE IN GEOSERVER**

a1. # Add Workspace for Vector format

# From terminal this will create worksspace in GeoServer called "Bound"

# Tomcat & GeoServer must be open and logged on

\*\* This procedure creates a workspace where other vector files can be stored. It only needs to be completed once then called when adding vector files in the future. \*\*

**curl -v -u admin:NMwater2015\_geoserver -XPOST -H "Content-type: text/xml" -d "<workspace><name>Vector</name></workspace>" http://localhost:8090/geoserver/rest/workspaces**

a2. # Add Workspace for Vector format

# From terminal this will create worksspace in GeoServer called "Raster"

# Tomcat & GeoServer must be open and logged on

\*\* This procedure creates a workspace where other raster files can be stored. It only needs to be completed once then called when adding vector files in the future. \*\*

**curl -v -u admin:NMwater2015\_geoserver -XPOST -H "Content-type: text/xml" -d "<workspace><name>Raster</name></workspace>" http://localhost:8090/geoserver/rest/workspaces**

\*\* Validation: Confirming the creation of the workspace \*\*

**curl -v -u admin:NMwater2015\_geoserver -XGET -H "Accept: text/xml" \**

**http://localhost:8090/geoserver/rest/workspaces/Bound**

## OR ##

**curl -v -u admin:NMwater2015\_geoserver -XGET -H "Accept: text/xml" \**

**http://localhost:8090/geoserver/rest/workspaces/Raster**

b1 . Add Store & Layer

## dataStores = vector

## coverageStores = raster

## wmsStores = cascaded WMS server

# Vector = datastores

# From terminal this will add a store & layer to GeoServer

# Tomcat & GeoServer must be open and logged on

# Be sure that you have moved the file to the designated location I.E. /Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles/

\*\* # This procedure can be conducted for vector files to be added to the 'Vector' workspace, datastore NM\_shape (or any other).

**curl -u admin:NMwater2015\_geoserver -v -XPUT -H 'Content-type: text/plain' –d**

**"file:///Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles/NM.shp" http://localhost:8090/geoserver/rest/workspaces/Vector/datastores/NM\_shape/external.shp**

# Raster = coveragestores

# From terminal this will add a store & layer to GeoServer

# Tomcat & GeoServer must be open and logged on

# Be sure that you have moved the file to the designated location I.E. /Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles/

\*\* # This procedure can be conducted for raster files to be added to the 'Raster' workspace, coveragestore 'Global'.

**curl -u admin:NMwater2015\_geoserver -v -XPUT -H 'Content-type: text/xml' -d "file:///Applications/GeoServer.app/Contents/Java/data\_dir/data/shapefiles/globaltest1b.tif" http://localhost:8090/geoserver/rest/workspaces/Raster/coveragestores/Global/external.geotiff**

**7 USE CURL COMMANDS TO ADD STYLE**

# Add Style for Raster

**curl -u admin:NMwater2015\_geoserver -XPOST -H 'Content-type: text/xml' -d '<style><name>Global1B</name><filename>"file:///usr/local/geoserver/data\_dir/data/shapefiles/Global1B-wm2.sld.xml"</filename></style>' http://localhost:8090/geoserver/rest/styles**

# Execute this request to upload the file Global1B-wm2.sld.xml

**curl -u admin:NMwater2015\_geoserver -XPUT -H 'Content-type: application/vnd.ogc.sld+xml' -d @Global1B-wm2.sld.xml http://localhost:8090/geoserver/rest/styles/Global1B**

# Global1B-wm2.sld.xml

**<?xml version="1.0" encoding="ISO-8859-1"?>**

**<StyledLayerDescriptor version="1.0.0" xmlns="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"**

**xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:schemaLocation="http://www.opengis.net/sld http://schemas.opengis.net/sld/1.0.0/StyledLayerDescriptor.xsd">**

**<NamedLayer>**

**<Name>gtopo</Name>**

**<UserStyle>**

**<Name>dem</Name>**

**<Title>Simple DEM style</Title>**

**<Abstract>Classic elevation color progression</Abstract>**

**<FeatureTypeStyle>**

**<Rule>**

**<RasterSymbolizer>**

**<Opacity>1.0</Opacity>**

**<ColorMap>**

**<ColorMapEntry color="#FFFFFF" quantity="-1" label="nodata" opacity="0.0" />**

**<ColorMapEntry color="#FFFFFF" quantity="0.0000000001" label="nodata" opacity="0.0" />**

**<ColorMapEntry color="#8c510a" quantity="0.1" label="0.1>" />**

**<ColorMapEntry color="#d8b365" quantity="112" label="112" />**

**<ColorMapEntry color="#f6e8c3" quantity="224" label="224" />**

**<ColorMapEntry color="#d1e5f0" quantity="336" label="336" />**

**<ColorMapEntry color="#67a9cf" quantity="448" label="448" />**

**<ColorMapEntry color="#2166ac" quantity="560" label="560+" />**

**</ColorMap>**

**</RasterSymbolizer>**

**</Rule>**

**</FeatureTypeStyle>**

**</UserStyle>**

**</NamedLayer>**

**</StyledLayerDescriptor>**

# Once these procedures are successfully conducted the style, “Global1B” can be inserted as a parameter for the WMS layer as shown in step 8.

**8 IMPLEMENT IN HTML SCRIPT WITH URL & OTHER PARAMETERS**

# This is a snippet from the greater HTML code that implements a WMS layer from GeoServer

**var test = L.tileLayer.wms("http://localhost:8090/geoserver/Raster/wms", {**

**layers: 'Raster:ImgMos',**

**time: current\_date**

**format: 'image/png',**

**opacity: 0.9,**

**transparent: true**

**style: Global1B**

**}).addTo(map);**