# User manual for SMOS processing

## Scripts -overview

#### Python

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| SMOSGridM5\_32bit\_newftp.py | FTP\_flags | Download from FTP, process soil moisture flags |
| SMOSGridW\_mh\_sysargv.py | Raster Analyzer | Prepare raster for calculations based on type of analysis |
| SMOS\_BL\_mholicka.py | Statistics Analyzer | Process Statistics |
| automapper\_setup.py | Automapper Setup | Find the necessary files for mapping |
| Maps\_Automator\_mh.py | Automapper Execute | Map the needed files |

#### Batch

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| Run.bat | Full execute | To run all scripts based on supplied parameters |
| Setup.bat | setup | To setup the working environment as needed |

Please note that all scripts are documented, so any additional info will be in the scripts.

## General Notes

* Not every variable used in the script will be outlined in this User Manual. Only those variables that should be changed by the user are described, or any variables that are very important.
* This process is modular and relative. Each script has its input and output folders hardcoded as per file structure. **If you change one, you need to change it in all scripts.** The user expects to see these variables:
  + Input folder
  + Output folder
  + Any intermediate folder(s)
  + Test/Done : This is a variable that was used to do dry – runs or to skip very long processes if they were already done.

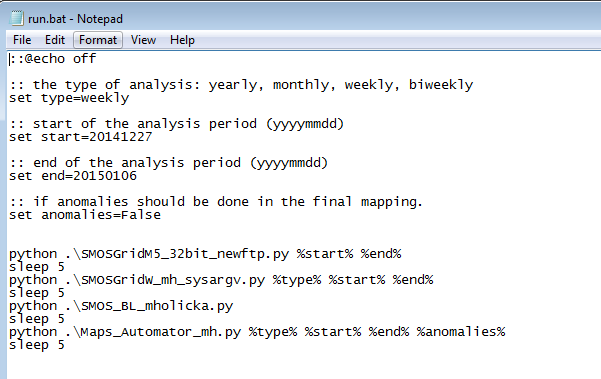
## File Structure

* There is no need to have any specific file structure. The Setup.bat file will create files as needed, and every script will make and create files and directories as it needs.

Please see the .gif included in this folder (Setup.gif)

### Running Options

-Batch Mode: Utilizing run.bat, and replace variables as needed.



- Single Mode: Run each script individually, according to instructions from each script.

**Script order:**

1. FTP\_Flags : this will make some folders
2. Raster Analyzer : This will use the folders generated from the FTP\_Flags.
3. Statistics Analyzer:
4. Automapper
   1. Automapper Setup
   2. Automapper Execute

The user can see the final result of this process (The map) in /main/Automapper/Outputs.

## Individual Script Documentation

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| SMOSGridM5\_32bit\_newftp.py | FTP\_flags | Download from FTP, process soil moisture flags |

**Major Variables**

|  |  |  |
| --- | --- | --- |
| Name | Purpose | Type and Example |
| fDate | Start date of processing (yyyymmdd) | String 20150101 |
| lDate | Start date of processing (yyyymmdd) | String 20150102 |
| Period | How many days to move by after processing | String 1 |
| strMode | Do the files need to be downloaded and processed from FTP. | String “y” or” n” |

**Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Input (s) | Output |
| filemaker | Make a full file path | * path (String) * wantedpath (String) | None |

How it works: path slowly becomes wanted path by constantly os.mkdir recursively. Wantedpath gets shorter each time, and path becomes that.

|  |  |
| --- | --- |
| Path   * Blank * / * /path/ * /path/to * /path/to/file | Wantedpath   * /path/to/file * path/to/file * to/file * /file * Blank |

**Parts of the script**

1. FTP processing
   1. Based on start and end dates, downloads .ZIP Files
   2. Converts each .dbl file to.txt
   3. Makes folders as needed (TXT,TIF)
2. TIF Processing
   1. Based on start and end date, makes a raster based on the txt files from previous step
   2. Flags
      1. SNOW\_MIX,SNOW\_WET,SNOW\_DRY,FROST,RAIN
      2. If any of these are true, Soil\_moisture becomes -999 (NoData)
3. Saving
   1. Each tif is stored based on the date , all .tifs are in 64 bit depth
   2. Each tif is converted to 32 bit float pixel depth
   3. Tifs are saved in the appropriate folder.

**GIF**

FTP\_Flag.gif shows the folders.

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| SMOSGridW\_mh\_sysargv.py | Raster Analyzer | Prepare raster for calculations based on type of analysis |

**Major Variables**

|  |  |  |
| --- | --- | --- |
| Name | Purpose | Type and Example |
| year\_to\_do | If yearly: which year or years to process | String yyyy or yyyy\_yyyy |
| Month\_to\_do | If monthly | String yyyy\_mm or yyyy\_mm\_yyyy\_mm |
| Time\_to\_do | Weekly/Biweekly processing | String yyyy\_mm\_dd or yyyy\_mm\_dd\_ yyyy\_mm\_dd |

**Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Input (s) | Output |
| filemaker | Make a full file path | * path (String) * wantedpath (String) | None |
| First\_Iso | To get the date that January 4th is on. | * Year(string) | Datetime object |
| Iso\_greg | To get the gregorian date from an ISO date | * Year\_iso(int) * Week\_iso(int) * Day\_iso(int) | Int. |
| Compare | To make the appropriate files and run the SMOS\_Compare script as needed | * Dict(Ordered Dicitonary) * Type(String) | None |
| SMOS\_Compare | Based on the list of files for that specific time period, it averages it all out and makes a new raster. | * List (Ordered Dict) * Year (int) * Mnth\_week(int) * Type(string) | None |
| Input\_handler(Only used in batch mode) | To convert the batch input as needed based on type of analysis | * Start(string) * End(string) * Type(string) | String |

**Parts of the script**

* File Selector – this is a complicated process and is best looked at in the script itself.

|  |  |
| --- | --- |
| Type |  |
| Yearly | based on the start and end years, the dicitonary is populated. No checking of consecutivness occurs |
| Monthly | The months in each year are seperated in each year. No checking of consecutivness occurs |
| Weekly | The weeks in each year are stored. They are ISO weeks, which are explained in user manual. Consecutivness is checked, so that each week starts on a monday , ends on Sunday and has all 7 days. If any day is missing,the week is ignored and the user is notified of this. |
| Biweekly | Based on the weeks, the biweeks are stored in thier respective years. Once again, these are ISO, and the weeks must be perfectly consecutive. This means that week x must be fully complete and week x+1 must be the next monday , or exactly 1 day after the end of the last week.If it not,then the biweek is ignored and execution skips to the beginning of the next biweek. |
| REPORTING | For monthly and yearly, there is no reporting of individual dates, and it will only notify if the month or year is missing.For weekly and biweekly, the user is notified of the missing week(s) and then the fact that biweek is incomplete.I would have liked to do an individual reporting system, but time was not on my side. |

* File Processor – Based on the type , it uses the functions compare and SMOS\_compare to take an average of each pixel in the stack. The rasters are turned into a Numpy Array and are stacked on each other such that the pixel at [i][j] are perfectly on top of each other. Each stack is averaged per pixel and the result is saved.
  + For example, a full week will have 7 rasters in the stack, so each pixel is averaged and saved
* File Saving- Each raster is saved as needed in the appropriate directory. This is according to the year,and then type.
  + Example: if the averaged raster is called : GlobAv\_SMUDP2\_OPER\_620\_2015\_Week\_14\_WGS84\_P6\_32b
    - It is a weekly, in the year 2015, so the output will be stored in :
      * Main/Outputs/GridW\_BL/weekly/2015
  + Same for :
    - * Main/Outputs/GridW\_BL/[yearly,monthly,biweekly]/2015

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| SMOS\_BL\_mholicka.py | Statistics Analyzer | Process Statistics |

**Major Variables**

* Apart from the normal variables, there are no other variables.

**Functions**

* This script has no functions, it is all just one large loop

**Parts of the script**

1. Getting master list:
   1. The list of all rasters from the previous script based on analysis type is populated and sorted based on the date.
2. Array Manipulation – Stats
   1. Based on the rules, the pixel is manipulated. Please note that this only applies to non-empty pixels

|  |
| --- |
| The pixel is not blank: the pixel in the final array adds the value to it, and the avg goes up by 1 to account for this. |
| If the pixel has no value, it is assigned the value of that pixel from the raster. |
| If the pixel value is greater than the current pixel value in the final array, it becomes that greater value. |
| If the max is still 101, than the max becomes the value of the pixel |
| If the pixel value is greater than the current max, the current max becomes that pixel's value. |
|  |

1. Baselines are established: This is used in the later parts, but as of the writing of this User Manual, the Baselines are not used in the next scripts.
2. Raster Creation

|  |
| --- |
| * 1. Percentile Raster |
| * 1. Percent Average raster |
| * 1. Difference from Average raster |
| * 1. LTDiff (this is the long-term difference than the average.) |

1. Raster Saving
   1. Based on the type of analysis that Is being done, the rasters are saved as needed. Please consult the tree section of this User Manual for more info and a sample run.

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| automapper\_setup.py | Automapper Setup | Find the necessary files for mapping |

NOTE: is actually never ran manually, and it is instead imported to the automapper.

**Major Variables**

None, please see the variables section for an explanation to the normal variables used throughout the scripts.

**Functions**

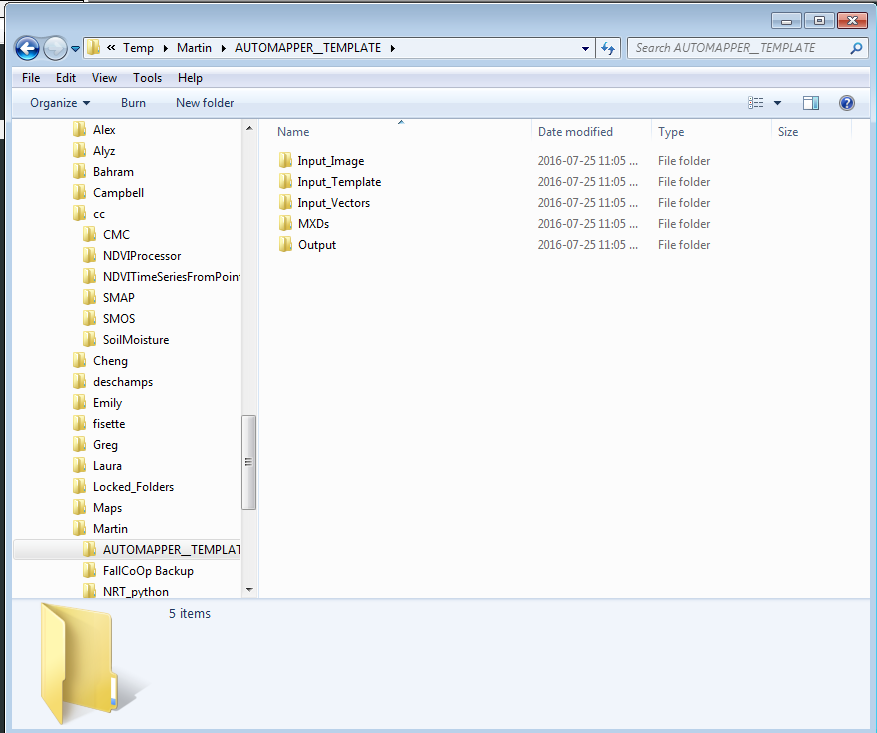
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Input (s) | Output |
| filemaker | Make a full file path | * path (String) * wantedpath (String) | None |
| Weekfinder | To get the ISO date of the week or biweek | * date (int) * type (String) | String |
| Filefinder | To get the needed files for mapping | * Start(String) * End(String) * Type (String * Loc(String) * Anom (Bool) | OrderedDict |
| Run | To setup the automapper for a clean run | * Start(String) * End(String) * Type (String * Loc(String) * Anom (Bool) | OrderedDict |

**Parts of the script**

1. Finding the files
   1. This relies on the function filefinder, which will find the paths to the files to be mapped.
   2. Filefinder Explanation : Futher explantion is in comments.
      1. Note: Filefinder is used only for weekly and biweekly, as monthly is already in a folder.
      2. If the years are the same, the dates are extracted and the dictionary is populated.
      3. If the Years are not the same for start and end, all years are populated to the end of that year until the end year, which is only to the date.
      4. Anomalies
         1. If the mapping is of the anomalies, the appropriate files are pathed to and included to the dictionary.
2. Setting up automapper
   1. This section is explained in detail following “Parts of the script”
      1. The needed files are copied to where they need to be.

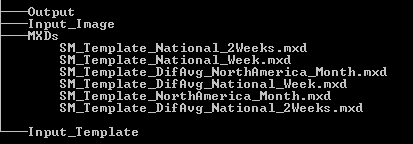
## Automapper Setup

Taking a look at the setup of the directory Automapper, we see the following.



|  |  |
| --- | --- |
| Name | What it contains |
| Input\_Image | The image that will be mapped (raster) |
| Input template | The template that will be used for the map |
| Input\_Vectors | Needed vectors for the map |
| MXDs | All types of MXDs that can be used. |
| Output | The final MXD and .pdf. |

Input\_Vectors are stable and should not be moved or changed, the complete list of all vectors can be found in the appendix.



**MXDs**

Based on the type of analysis (Monthly, Weekly,. Biweekly) and if Anomalies are to be mapped, the appropriate template is used.

Sample Run of Automapper\_Setup

* Input image : Raster.tif
* Specifications : Weekly and not interested in anomalies.

1. Previous inputs are all erased and the folders are remade.
2. The input image is copied to the Input\_Image dir based on the path.
3. The template for this setup (SM\_Template\_National\_Week.mxd) is copied from MXDs to Input\_Template dir.
4. Using the vectors and the template, the resulting map is made.
   1. Both the MXD and the .pdf are stored in the Output directory.

|  |  |  |
| --- | --- | --- |
| Name | Alias for user manual | Function |
| Maps\_Automator\_mh.py | Automapper Execute | Map the needed files |

**Major Variables**

|  |  |
| --- | --- |
| Name | Purpose |
| Mapsetup | Holds the input dictionary from the setup script |
| Type – String | What kind of analysis this is. |
| Anom- Boolean | Are anomalies being mapped? |
| Img\_list | The completed dictionary as a re-used variable |

**Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Input (s) | Output |
| filemaker | Make a full file path | * path (String) * wantedpath (String) | None |
| iso\_year\_start | To get the date that January 4th is on. | * Iso\_year(string) | Datetime object |
| iso\_to\_gregorian | To get the gregorian date from an ISO date | * Iso\_year(int) * Iso\_week (int) * Iso\_day(int) | Int. |

**Parts of the script**

1. Run the setup script to ensure a smooth run, and get the needed file pathings.
2. Map Making
   1. Declaring the different variables to be used in the map
   2. Changing the data sources
      1. The raster in Input\_Image is used instead of the default.
   3. Changing the dates
      1. Based on the date and type of analysis, the mxd is updated.
   4. Saving
      1. The pdf and MXD is stored.