Satellite Data Interrogation and Plotting Program

Instructions, Installation, and Usage

Dev: Dr. T. Connor Nelson

Org: Cooperative Institute for Research in the Atmosphere (CIRA) – Colorado State University and NOAA/NWS Operations Proving Ground

Contact: tconnorn@colostate.edu

Date: 08/23/2021

Version: 1.0.0

Introduction

The Satellite Data Interrogation and Plotting (hereafter called 'Satellite') program is a set of python scripts that can be used to grab GOES–16/17 satellite data online from AWS storage and plot a series of RGBs including: GeoColor (aka, "True Color"), Day Cloud Phase (DCP), Daytime Convection (DayConv), Daytime Microphysics (DayMicro), or Airmass. Future iterations will include more complex or additional types of RGBs. This code could be used to plot single-band satellite data; however, extensive simplification of it would need to occur (i.e., removing the RGB creation, normalization, stacking, etc.).

This code was optimized for use on a Windows Machine (v10) and runs on Python 3; however, this code can work for other os (e.g., Linux, Mac OS) with minor changes to directory syntax[1]. There are two modes that the code operates on: single file mode or batch mode. Single file mode, not surprisingly, creates RGBs for single files. Batch mode creates a series of plots over a time range. These are discussed below. These instructions assume you already have a working knowledge of coding (i.e., how to open a file or run a program from the command line) and at least some familiarity with Python.

Pre-installation

This document assumes that you already have a working version of Python 3 on your machine and you know how to use it. You will need to install the following additional libraries either via anaconda or pip:

- Pandas
- Numpy
- Matplotlib
- netCDF4
- cartopy
- xarray
- GOES

For plotting DayMicro, you will also need (not needed for other RGBs):

- pyorbital
- datetime
- pyspectal

For *most* things…to install via anaconda, launch your anaconda powershell and type either:

>> conda install 'packagename'

Example:

>> conda install numpy

Or

[1] Note that directory paths in Windows OS use the '\\' delimitator as opposed to the '/' delimitator for Linux or Mac OS.

>> conda install -c conda-forge 'packagename'

Example:

>> conda install -c conda-forge pyspectral

To install via pip, launch powershell or anaconda powershell and type:

>> pip install 'packagename'

Example:

>> pip install xarray

Installation

Once you have anaconda and all necessary packages installed, there are two ways you can install the program:

Method 1: Zip file

1. Download the Satellite.zip file
2. Move the zip file to the directory you want to run the code
    a. Say you want to run it on your desktop, move it from 'Downloads' to your 'Desktop'
3. Unzip the file
4. You are ready to go!

Method 2: Git pull

1. Open your anaconda powershell or favorite terminal
2. Change directory to where you want to run the code
    a. Say you want to run it on your desktop, cd to your 'Desktop'
3. Type in the following git command:
   >> git clone https://github.com/QGChi2/Satellite.git
4. You should see a new directory called 'Satellite' created with all necessary code inside.

Updates to code

Periodic updates may be made to the code or new versions may be released.

If you used ***Method 1*** to install (above), you can update the code by deleting and re-downloading the program (but make sure you do NOT delete any plots/data you have made!!!!).

If you used ***Method 2*** to install (above), you can more robustly update the code using git as shown below:

1. Via command line, change directory to where you have the Satellite program saved. This will hereafter be called the 'Satellite' directory. For example:
   >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//'
2. Complete a git pull:
   >> git pull
3. You should see the updated files populate the directories

Obtaining satellite data

Congratulations! If you have made it this far, then I have not screwed up too bad…yet. The first step to see if everything works is to try and download some satellite data from the GOES AWS webserver.

4. To start, open your terminal, anaconda poweshell, or whatever is your code-running interface.
5. Change directory to where you have the Satellite program saved. This will hereafter be called the 'Satellite' directory. For example:
   >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//'
6. The code will automagically read the directory you are in, so no need to edit code to change directory paths…unless you are working with a Linux/Mac OS.
   a. If you are working with a Linux/Mac OS, you will need to change the slash marks in the directory paths to be congruent with those OS (see footnote 1)
7. Open the GOES_data_grab.py file in your favorite flavor of text editor (e.g., Notepad++, vi, Wordpad, etc.)
8. Change the date and time range (in UTC) that you want to grab
9. The default channels to grab are Ch: 1, 2, 3, 5, and 13, which are needed to build the DCP and GeoColor RGBs. If you want other channels, you can modify the 'ch' variable.
10. Save the file and run the code with python in the Satellite directory[2]:
    >> python GOES_Data_grab.py
11. This script should grab data from the GOES AWS server and save them in a repository (called REPO) in a folder based on the date (YYYYMMDD)
    a. Example:
       'C://Users//Connor Nelson//Desktop//Python_code//Satellite//REPO//20210517'
    b. Note: it may take 5–10 min to run the data extraction, depending on how many files you are grabbing and your internet speed!
12. You will need to re-run this code for all other dates or time ranges needed.
13. The files have a weird architecture, which is explained in Beginners_Guide_to_GOES-R_Series_Data.pdf in the Satellite directory

[2] Note: you can also run by typing >> python, hit enter, and copy and paste the code into the terminal/powershell

Batch mode

Batch mode is used to create CONUS wide or zoomed-in PDFs of various RBGs over a range of times on any given date. This code will use the DCP file as an example, but a similar process can be used for the other RGBs. Zoomed-in views are currently only available for the days of the 2021 Operations Proving Ground Severe Weather Experiment (May 17–20; May 24–27). The zoom-ins on these dates focus in on the NWS County Warning Areas used on each day. To run this:

1. To start, open your terminal, anaconda poweshell, or whatever is your code-running interface.
2. Change directory to where you have the Satellite program saved. You will see a directory called "Code" that have the batch python scripts (e.g., GOES_DCP_RGB.py). This will hereafter be referred to as the "Batch" directory.
   >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//Code//'
3. The code will automagically read the directory you are in, so no need to edit code to change directory paths…unless you are working with a Linux/Mac OS.
   a. If you are working with a Linux/Mac OS, you will need to change the slash marks in the directory paths to be congruent with those OS (see footnote 1)
4. Open the GOES_DCP_RGB.py file in your favorite flavor of text editor (e.g., Notepad++, vi, Wordpad, etc.)
5. In the user modification section, change the date to be the date you want this to run on (YYYYMMDD)
   a. NOTE: current version will run this for ALL times on that date, future versions will allow for a time range. Also….make sure you have EQUAL numbers of files per band (i.e., if you have 36 files of band 2 on May 17th 2021, make sure you have the same number for band #13).
6. Save the file and run the code with python in the Batch directory
7. This script takes a LONG time (e.g., 30 min/1 hr of data files). But will save plots in a Plots directory, by date, RGB type, and Zoom/Full.
   a. Example(s):
      >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//Code//Plots//20210517//DCP//Full//'

      >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//Code//Plots//20210517//DCP//Zoom//'
8. Files will be output for every scan time on each date
9. Samples of what the plots should look like will be in:
   …//Satellite//Code//Plots//Example_output//'

Single-file mode

Single-file mode is used to create CONUS wide or zoomed-in PDFs of various RBGs at a single date and time. Basically, it is a one-shot of code if you just want a single image and not > 10 images. Zoomed-in views are currently only available for the days of the 2021 Operations Proving Ground Severe Weather

Experiment (May 17–20; May 24–27). The zoom-ins on these dates focus in on the NWS County Warning Areas used on each day. Data for a single file is saved locally in the Single_file_plotting directory. To run this on files you obtained either on your own or from the GOES_data_grab.py script, you will need to copy and paste the desired files into the Single-file directory and open the desired script (e.g., GOES_DCP_RGB.py) and modify the file names of file1, file2, and file3. The comment section above these tells you what channel goes where in the file names. To run this code do the following:

1. To start, open your terminal, anaconda poweshell, or whatever is your code-running interface.
2. Change directory to where you have the Satellite program saved. You will see a directory called "Single_file_plotting" that have the Single-file python scripts (e.g., GOES_DCP_RGB.py). This will hereafter be referred to as the "Single-file" directory.
   >> cd 'C://Users//Connor Nelson//Desktop//Python_code//Satellite//Single_file_plotting//'
3. The code will automagically read the directory you are in, so no need to edit code to change directory paths…unless you are working with a Linux/Mac OS.
   a. If you are working with a Linux/Mac OS, you will need to change the slash marks in the directory paths to be congruent with those OS (see footnote 1)
4. You will need to obtain GOES-16/17 data at a fixed date/time and the correct channels for the desired RGB and store them locally in the Single-file directory. For example, let's say we wanted the GOES-16 DCP RGB for 17 May 2021 at 16:59:00 UTC and already have those files downloaded (see Obtaining Satellite Data section above), we need to move into the Single-file directory the following files:
   a. **Channel 2:** OR_ABI-L2-CMIPC-M6C02_G16_s20211371656164_e20211371658537_c20211371659018.nc
   b. **Channel 5:** OR_ABI-L2-CMIPC-M6C05_G16_s20211371656164_e20211371658537_c20211371659007.nc
   c. **Channel 13:** OR_ABI-L2-CMIPC-M6C13_G16_s20211371656164_e20211371658549_c20211371659038.nc
5. **Note:** The naming on these files is odd…the date is provided as YYYYJJJHHMMsss where: YYYY= year, JJJ=3 digit calendar day, HH=hour (UTC), MM=minutes, sss=seconds in decimals (e.g., 65.3 seconds = 653).
6. Open the desired script (e.g., GOES_DCP_RGB.py) and modify the file names to match the correct channels for the files you put in the Single-file directory at step 4.
7. Run the code with python in the Batch directory
8. This script will save plots locally in the Single-file directory. If you want to change the save location, you will need to open the desired file (e.g., GOES_DCP_RGB.py) and manually change the save location (e.g., I want to save the files to my desktop, so I change output1=root to something like output1='C://Users//Connor Nelson//Desktop//')

Other (possibly) important sections of code

GOES Remapping

For all intents and purposes, GOES-16/17 data is saved on all online repositories as a netCDF file with its own coordinate system. In this coordinate system, latitude and longitude (y and x, respectively) are

defined as radians from the view of the satellite. Basically, the satellite is up in the sky at some height 'Z' looking towards the earth and gets information over a whole range of look angles. Our job is converting those radian angles to canonical latitude, longitude that we are accustomed to working with via remapping approximation (aka reprojection). For more information about this process, please see the following link (Hrisko 2018). In both the Single-file and Batch directories, there is a python script called "GOES_LL_Conv" that contains the function "lat_lon_reproj". This function completes the reprojection for you by simply providing it the path to the file (i.e., "root") and the name of the file (i.e., "file"). It will obtain the relevant metadata, complete the conversion, and output not only the latitude and longitude, but also useful information such as the channel number, wavelength, units, etc. For the most part, you shouldn't need to be worried about modifying this code, unless you are trying to improve the remapping, but it's sometimes nice to know what is going on under-the-hood of the car.

Solar Angle Correction

Sometimes, we need to apply an additional correction to the satellite data to account for the interaction of the sun with particles in the atmosphere. This type of correction is common for the IR channels, especially the Shortwave Window (Channel 7), where solar angle plays a large role in the intensity detected by the satellite. The solar angle correction is currently implemented in the DayMicro code for Channel 7. The code uses the 'Calculator' function from the pyspectal library and corrects the data using a calculated zenith angle and data from Channel 13 ("Clean" IR longwave). Again, for the most part, you shouldn't need to be worried about modifying this code, unless you are trying to improve the remapping, but it's sometimes nice to know what is going on under-the-hood of the car.

Resize function

One last piece of 'under-the-hood' code that is useful is the rebin (aka resize) function. The resize function is available in almost all of the RGB plotting code and ensures that all of the individual R, G, and B components have the same resolution, which is generally: 1500x2500. Common channels that need to be resized are: 2, 3, and 5.