Workshop 105 – Ballroom E

June 2024

Retrieval and Application of On Demand Global Field-scale Actual Evapotranspiration Data Since 1982

-

Creating Aggregations of ET using Python scripts

Gabriel B. Senay[1], Stefanie Kagone[2], Gabe Parrish[3],

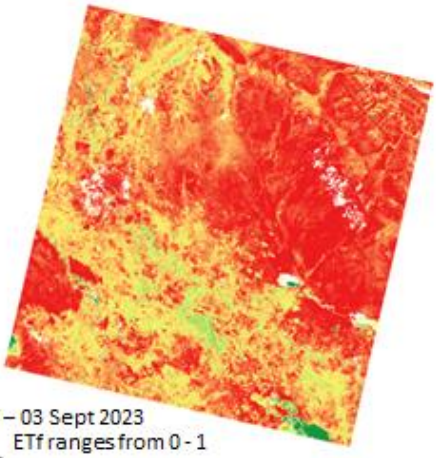
Kul Khand[1], and Jordan Dornbierer[4]

[1]U.S. Geological Survey (USGS)
Earth Resources Observations and Science (EROS) Center,
[2]ASRC Federal, [3]Innovate!, Inc., [4]KBR, Inc., contractors to USGS EROS Center, Sioux Falls, SD 57198, USA. Work performed under USGS Contract 140G0124D0001.

Introduction

The following document will provide guidance on a workflow from raw ETf Landsat scenes to create monthly/annual/seasonal ETa summaries from the USGS Landsat SSEBop ET data obtained from the ESPA website, including best practices of how to work with the data. Further insight into the functioning of the module can be gained by reviewing the code and the documentation on the USGS GitLab repository page. First we will discuss “what is an overpass ET image?” and how to use gap filling and interpolation to create an daily ET image from the overpass ET image.

What is an Overpass ET image?



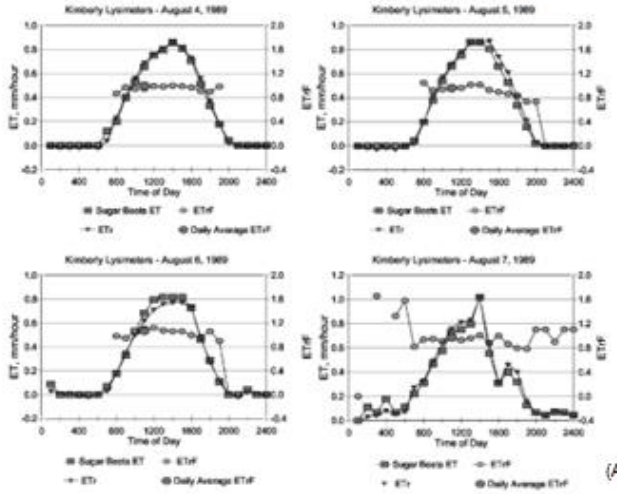
ETf – 03 Sept 2023
ETf ranges from 0 - 1

- Snapshot in time of ET
- ETf instant ~ ETf all day long
- ETf instant * ETo daily = ETa Daily

USGS

45

- Overpass ETf → Daily ETf? That's the beauty of ETf



(Allen et al., 2007)

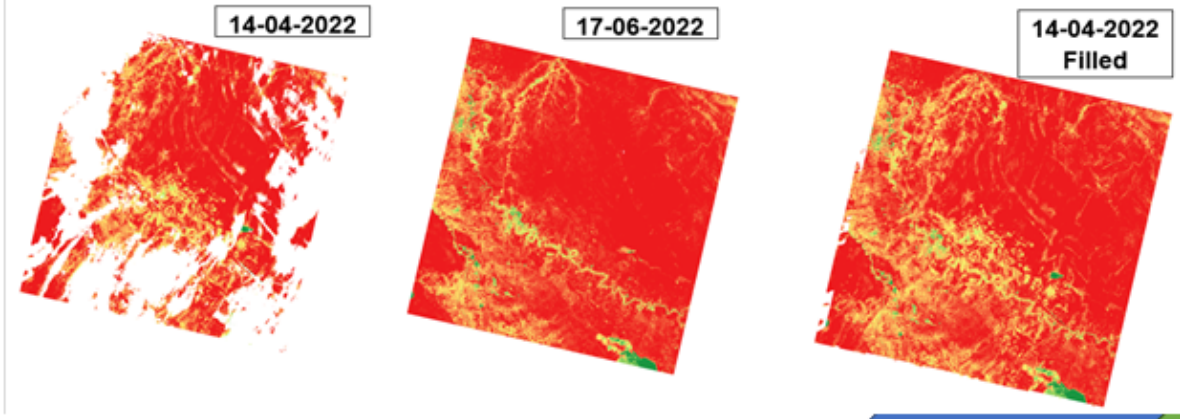
USGS

Fig. 2. Hourly ET and ET_rF for sugar beet crop versus time [based on lysimeter observations by Wright (1982), USDA-ARS, Kimberly, Idaho] for a series of four days in August (ET_rF for the 24 h period is the larger circle plotted at 11:00, which is satellite overpass time)

46

Gapfilling the ETf rasters

- Holes in rasters from clouds, other issues…We fill them with images from before/after



47

From overpass to daily ETf

- We interpolate ETf from overpasses…
- But how to Interpolate? linear, spline, other?
- ETf is analogous to a crop coefficient

ETo/ETr * Kc = ETc
ETo/ETr * ETf = ETa

Very Similar!

USGS

(Allen et al., 2007)

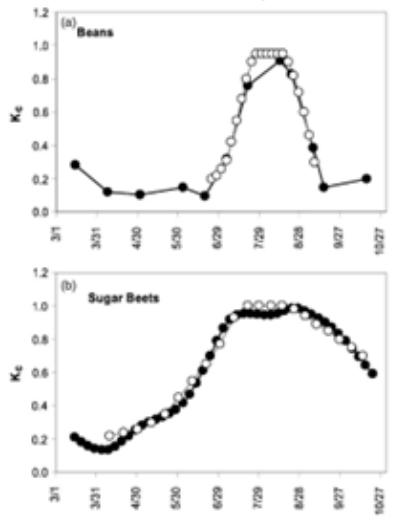
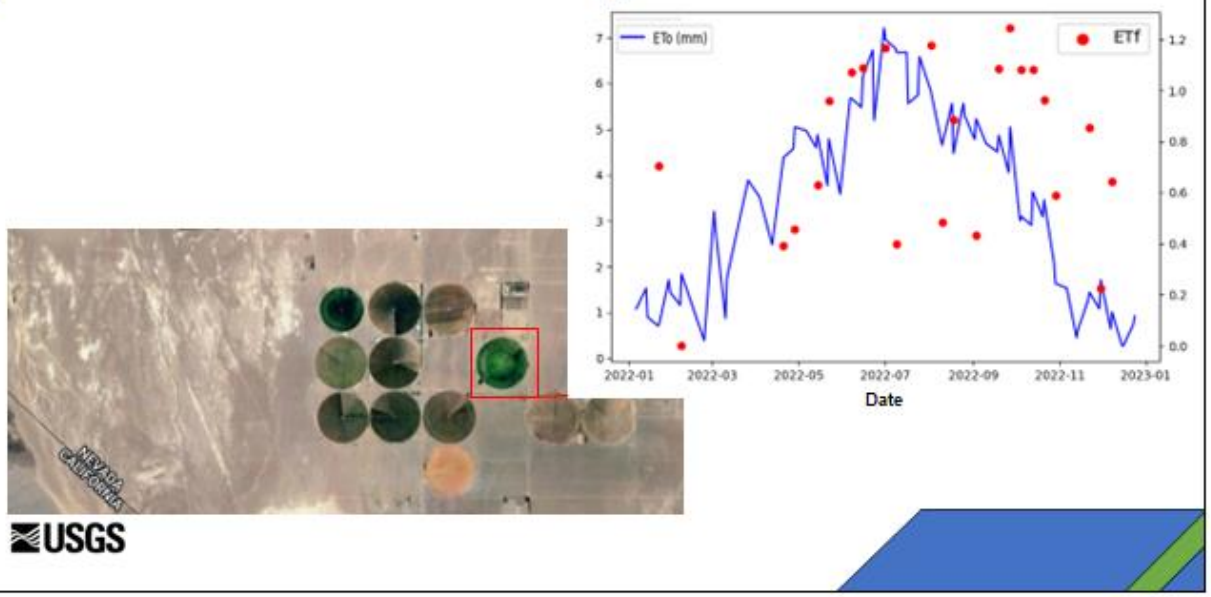


Fig. 4. ET,*F* from METRIC for: (a) dry bean crop in southern Idaho in 2000 (solid symbols) compared to a K_c curve [published by the USBR AgriMet (2000) (open circles) (adapted from Tasumi et al. 2005a,b]; (b) sugar beet crop where a cubic spline has been used to interpolate between satellite image dates

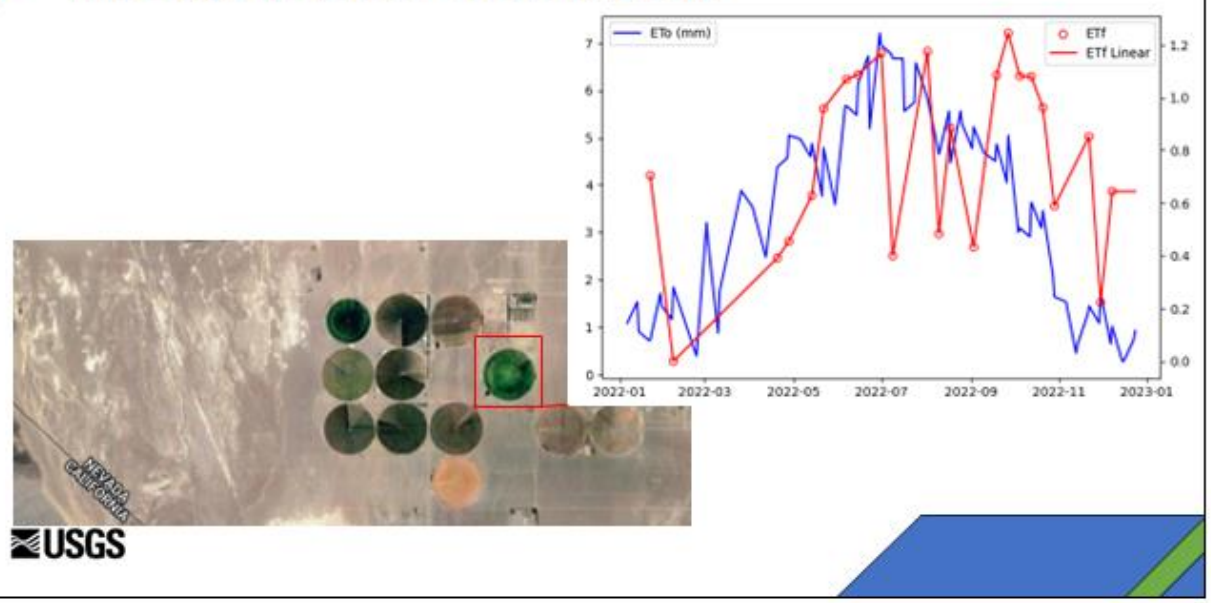
48

Illustration of Interpolation



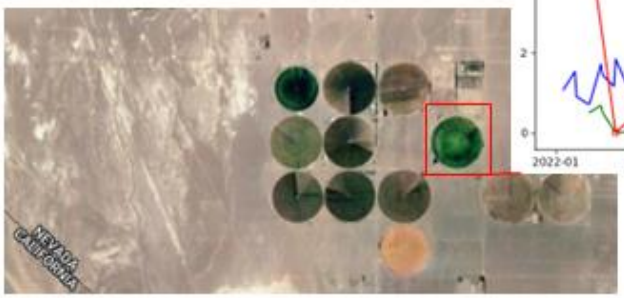
49

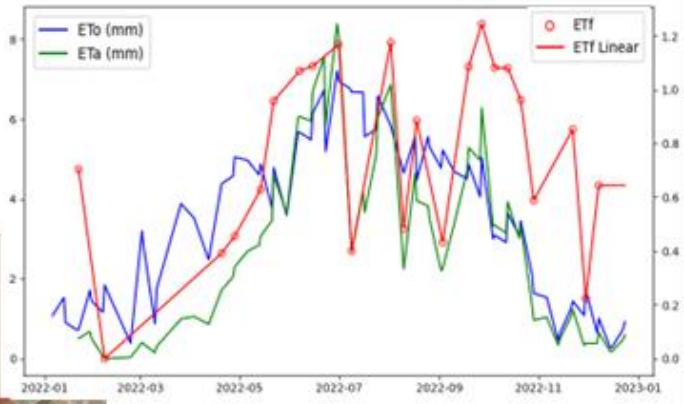
Interpolation continued



50

Interpolation continued

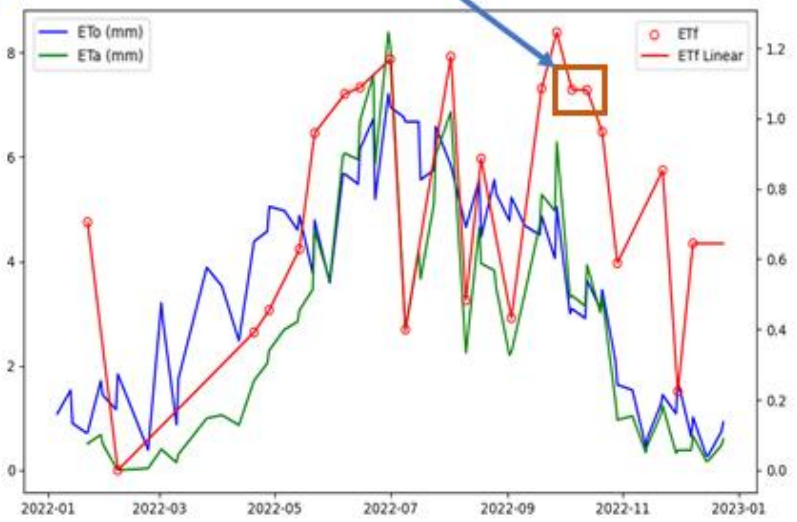




USGS

51

Two identical ETfs in a row... A telltale sign of gap-filling



Now we have daily ETa, which can be aggregated into:

- Monthly ETa
- Yearly ETa
- Seasonal ETa
- Anything you want (except for hourly!)

USGS

52

Installation

I) Using Anaconda to install needed code base and Python modules

Anaconda is a distribution software of Python programming languages for scientific computing, that helps developers with code package management and deployment. You will need **Anaconda and Anaconda Prompt** installed on your computer. Visit https://www.anaconda.com/download/success for the appropriate download for your operating system.

Once the install of Anaconda is completed, **open Anaconda Prompt**. The application should look like this:

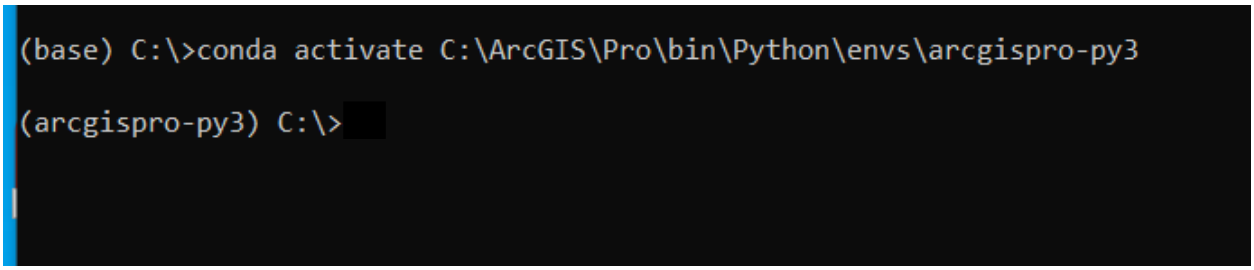
The **(base)** prompt is the current environment that is active within Anaconda. One can create different environments with different Python packages for different projects. Here, we are having 2 options for environments:

a) ArcPy code environment

The code base using the Python ArcPy module provided by Esri requires the environment called **arcgispro-py3** to be activated. This environment doesn't need to be created but is provided with Esri's ArcPro software installation. To activate the environment enter

conda activate PATH_TO_ENV\arcgispro-py3

Here is an example:



Make sure you can open and are signed in to ArcPro, if you aren't the license may not activate properly and it gives an error using the **ArcPy** module.

b) Open Source code environment

If there is no environment alreadythat you can utilize to process the Python code, we need to create one from scratch. For the Open Source code base we are creating an environment named **opensource_et_env** and are installing Python packagerasterio to do the rastercalculation instead of the ArcPy module. To do that we first list all the environments that are possibly already available from previous projects.

conda env list

```
(base) C:\Users\       \scenes\bulk-downloader-master>conda env list
# conda environments:
#
base                  *  C:\Users\skagone\AppData\Local\miniforge3
bulk_dl_espa             C:\Users\skagone\AppData\Local\miniforge3\envs\bulk_dl_espa


(base) C:\Users\       \scenes\bulk-downloader-master>conda create --name opensource_et_env
```

In this example we have the default environment called **base** and a **bulk_dl_espa** environment. To run the code base, we will create a new environment named **opensource_et_env** with

conda create --name opensource_et_env

```
(base) C:\Users\       \scenes\bulk-downloader-master>conda create --name opensource_et_env
Retrieving notices: ...working... done
Channels:
 - conda-forge
Platform: win-64
Collecting package metadata (repodata.json): /
```

This will create an empty environment where all the needed Python packages can be installed. To use and install packages to this environment we will activate it first with

conda activate opensource_et_env

```
(base) C:\Users\skagone\scenes\bulk-downloader-master>conda activate opensource_et_env

(opensource_et_env) C:\Users\skagone\scenes\bulk-downloader-master>
```

And then install package: rasterio.

conda install rasterio

```
(opensource_et_env) C:\Users\skagone\scenes\bulk-downloader-master>conda install rasterio
Channels:
 - conda-forge
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
```

With that we can move to the next step: to pip install the code base from the GITLab repository.

II) Pip Install from GITLab repository

a) ArcPy code

The ArcPy code base can be downloaded or cloned from eros_hydro / ssebop_espa_arcgis · GitLab (usgs.gov) or is provided as a zip file (ssebop_espa_arcgis-main.zip) in the Workshop folder at Index of /project/SSEBop/WaterSciCon2024 (usgs.gov).

b) Open source code

The ArcPy code base can be downloaded or cloned from eros_hydro / ssebop_espa_opensource · GitLab (usgs.gov) or is provided as a zip file (ssebop_espa_opensource-main.zip) in the Workshop folder at Index of /project/SSEBop/WaterSciCon2024 (usgs.gov).

Next, use the cd command in the anaconda prompt window to navigate to the folder where you saved the downloaded code from GIT or the workshop folder. In this example the code is saved in a directory called **PycharmProjects.**

```
(arcgispro-py3) C:\>D:

(arcgispro-py3) D:\>cd D:\Users\[redacted]\PycharmProjects\ssebop_espa_arcgis

(arcgispro-py3) D:\Users\[redacted]\PycharmProjects\ssebop_espa_arcgis>
```

Once in the directory, Pip install the code to this folder (.) with no dependencies (–no-deps) because those are already provided in the activated environment.

Pip install . --no-deps

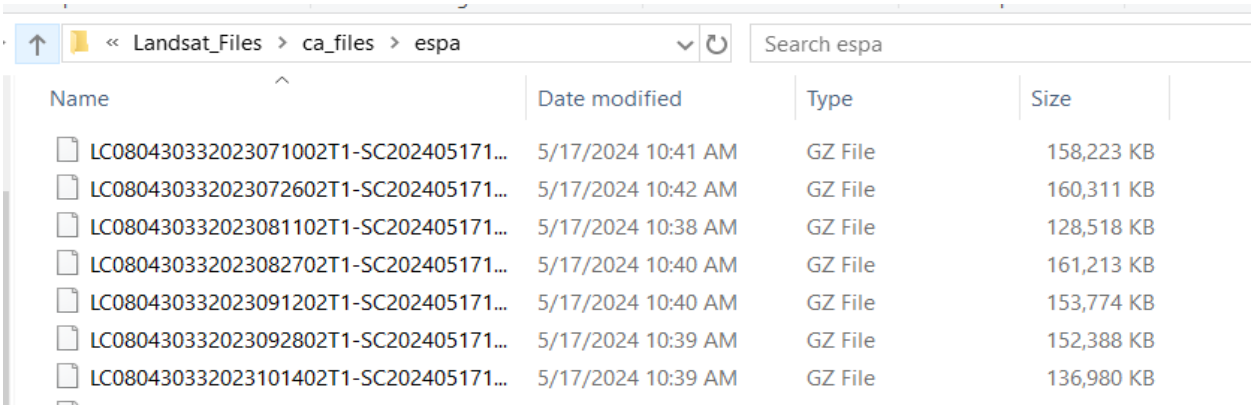
```
(arcgispro-py3) D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis>pip install . --no-deps
Processing d:\users\gparrish\pycharmprojects\ssebop_espa_arcgis
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: ssebop-espa
  Building wheel for ssebop-espa (setup.py) ... done
  Created wheel for ssebop-espa: filename=ssebop_espa-0.0.4-py3-none-any.whl size=11936 sha256=c2
9a9b46bdee29d0f534be96558b920ca09f3bf3
  Stored in directory: D:\Users\gparrish\AppData\Local\Temp\1\pip-ephem-wheel-cache-1ophkbbw\whee
d64cd540715a411c99ae4f8e3c0aea109d1aa3
Successfully built ssebop-espa
Installing collected packages: ssebop-espa
  Attempting uninstall: ssebop-espa
    Found existing installation: ssebop-espa 0.0.3
    Uninstalling ssebop-espa-0.0.3:
      Successfully uninstalled ssebop-espa-0.0.3
Successfully installed ssebop-espa-0.0.4

(arcgispro-py3) D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis>
```

Once the code base was successfully installed, we can move on to processing the Python code to create different ET products, such as monthly or annual time series.

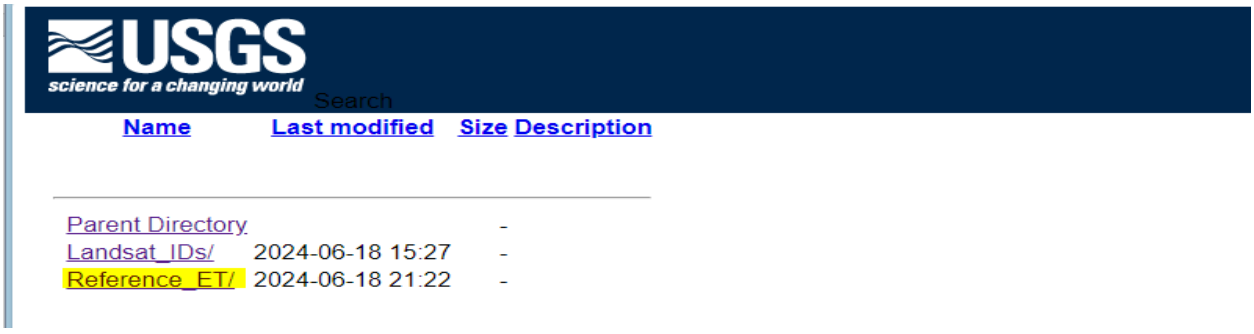
Processing the Python code

The code base consists of 4 scripts, each representing 1 step of the process. There is also a GUI (Graphical User Interface) available to simplify the processing of the data (gui.py). The order in which they are to be run (1-4) is indicated on each tab in the GUI menu.

Next, locate the folder where you stored the tar.gz files from the ESPA website. In our case, this location (or directory structure) is: **//Landsat/ca_files/espa.** There are all in all 63 files for 2 Landsat scenes (p43r33, p43r34), but we are demonstrating the scripts and the functionality on a smaller subset of files for the workshop.

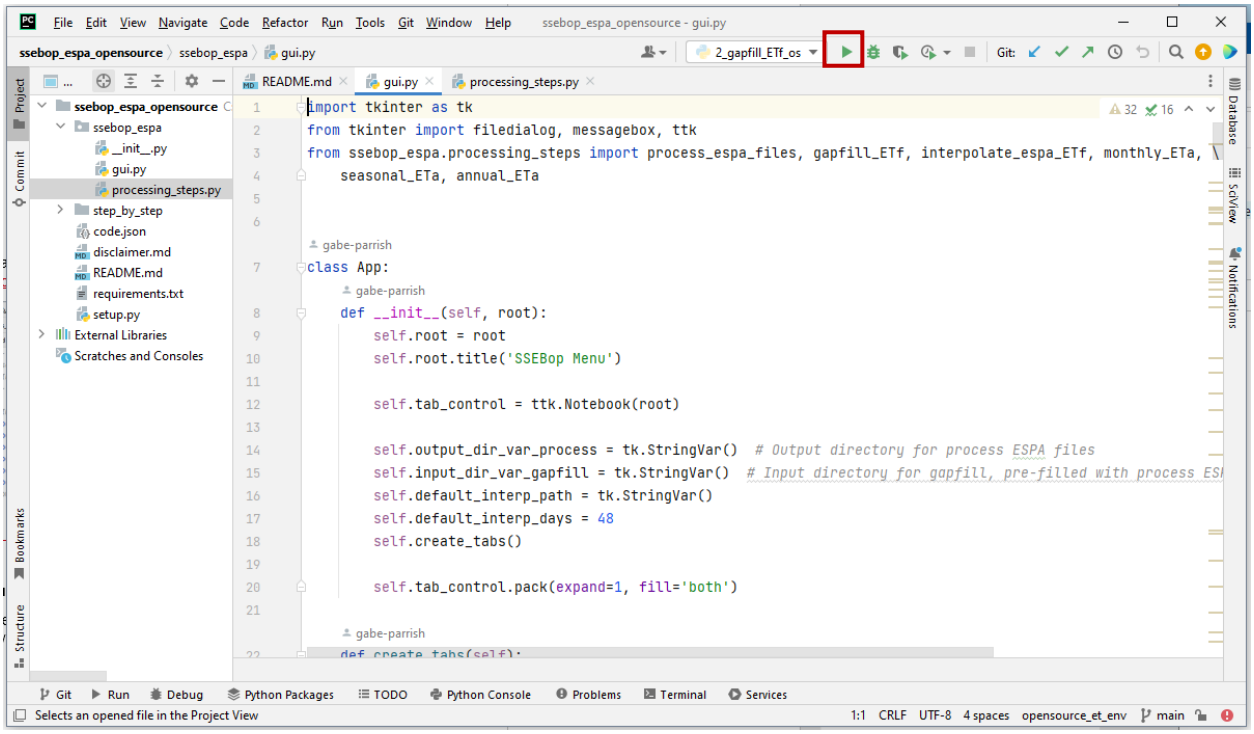


In addition to tar.gz files, we need a reference ET dataset to create actual ET (ETa) data in millimeter (mm). Therefore, download the reference ET dataset provided in the Workshop materials under https://edcftp.cr.usgs.gov/project/SSEBop/WaterSciCon2024/Reference_ET/ for this demonstration. Other available datasets can be used for your specific project needs.

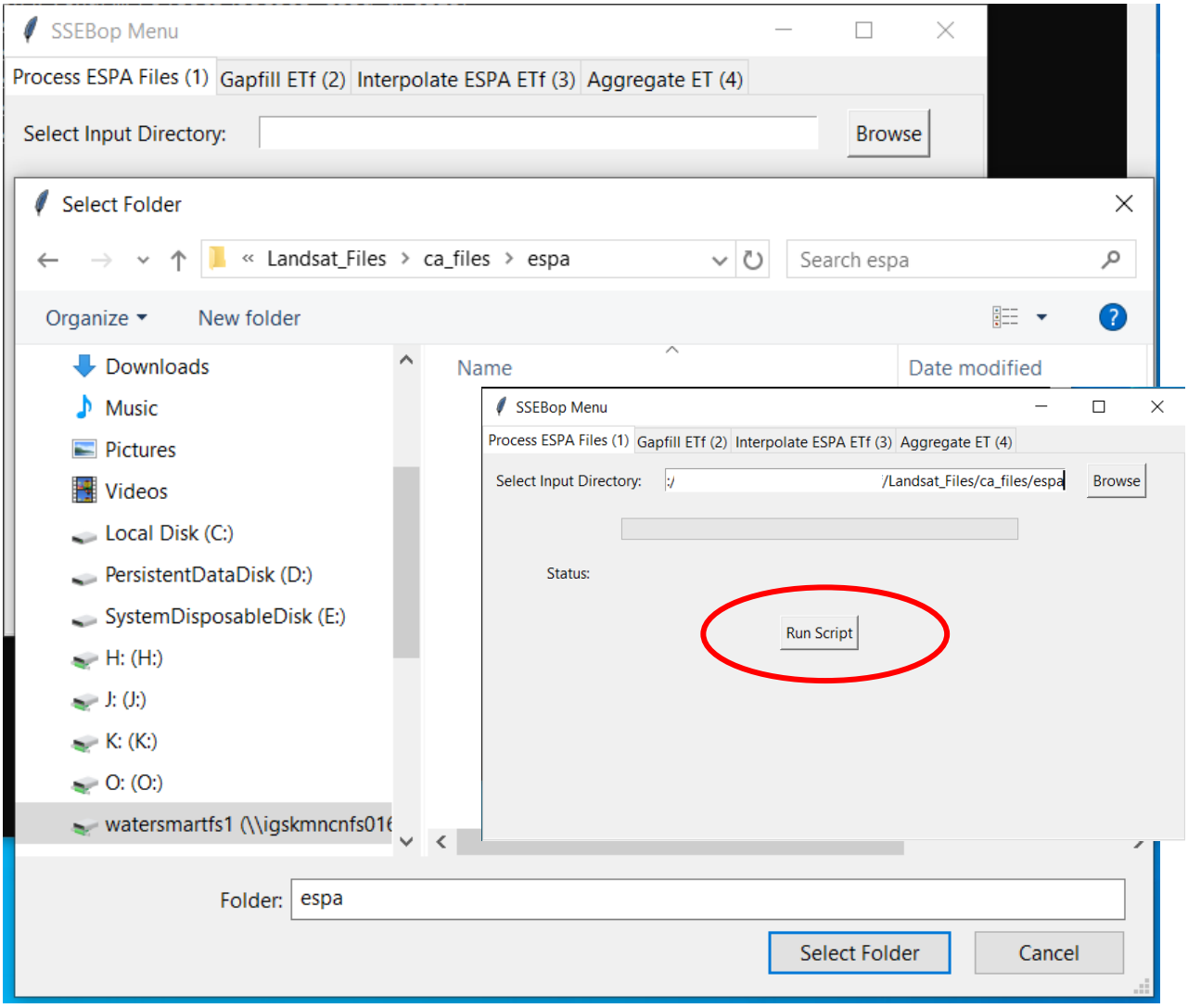


1) Process ESPA Files

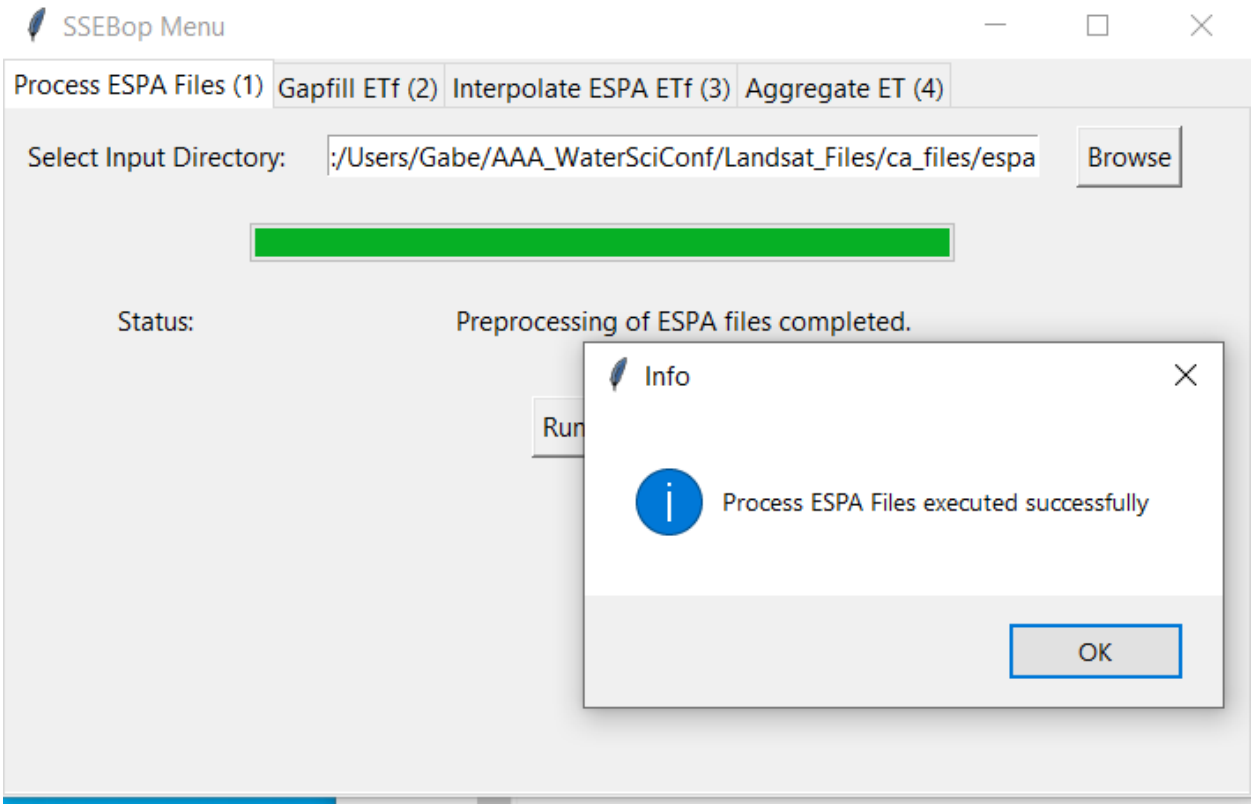
To start processing the data open an User Interface, such as PyCharm, VS Code, etc. and open the downloaded code (ssebop_espa_opensource-main.zip). Here, we are using PyCharm.



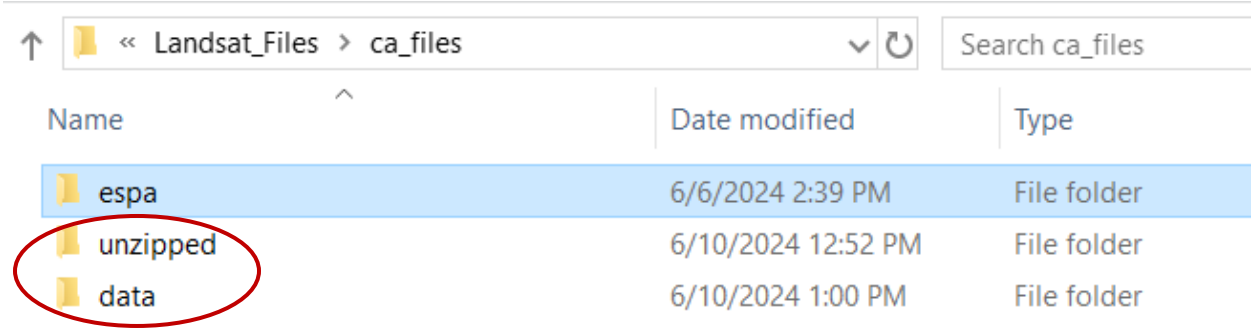
Open the script **gui.py** in the folder **ssbeop_espa** and **Run** the script by clicking the **Run** button (green triangle). Here the GUI interface should pop up. Click on the first tab named **"Process ESPA Files (1)"** and push the Browse button to navigate to the location where the tar.gz files downloaded from ESPA are located.



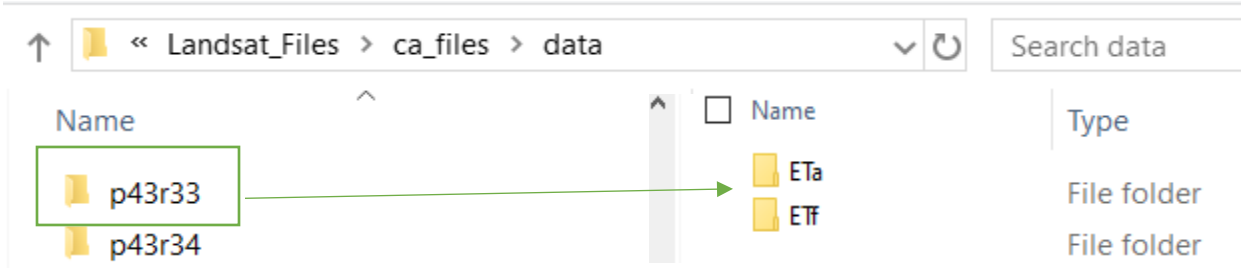
Run the script by clicking the **Run Script** button. Be prepared to ignore (Not Responding) message. The script takes about 5-10 mins depending on the number of files processed. When it all completes the following message is displayed:



Along with 2 new folders in your directory:



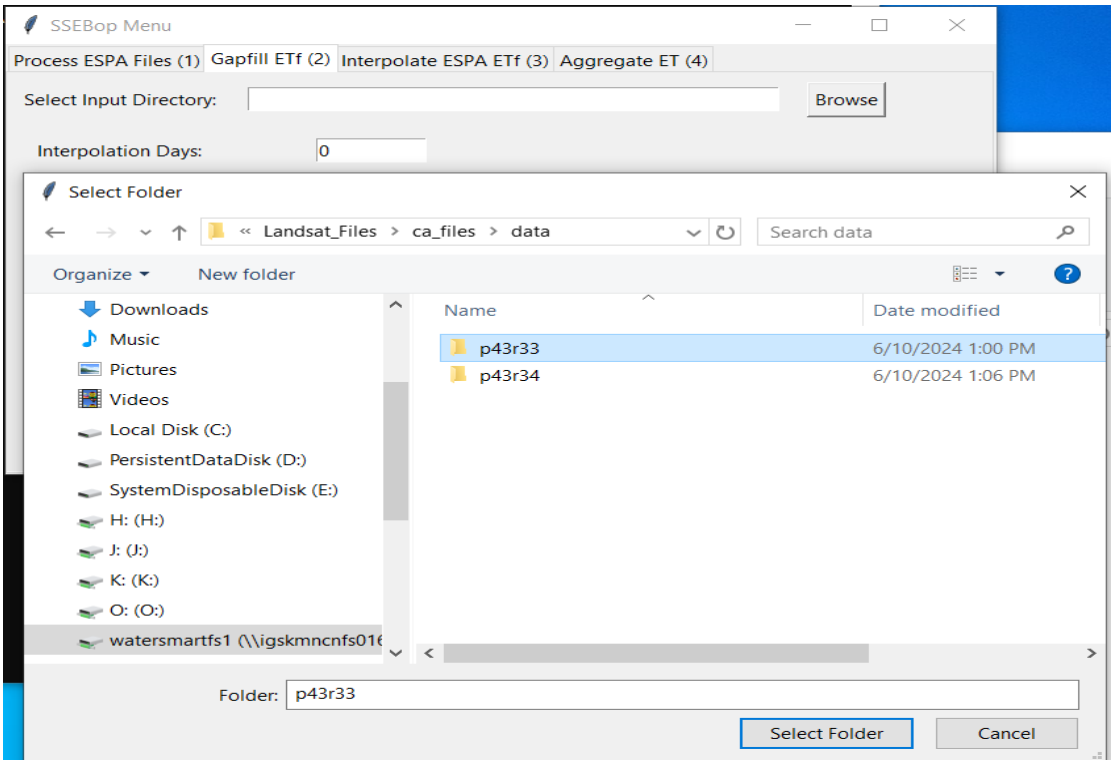
The overpass ETf and ETa data will be placed in the folder **//Landsat/ca_files/data/pXXrXX.** The ETf data is scaled by 10000 and renamed to filename **etf_YYYYMMDD.tif**. The ETa data is scaled by 1000 and renamed to filename **eta_YYYYMMDD.tif**.



2) Gapfill the ETf rasters

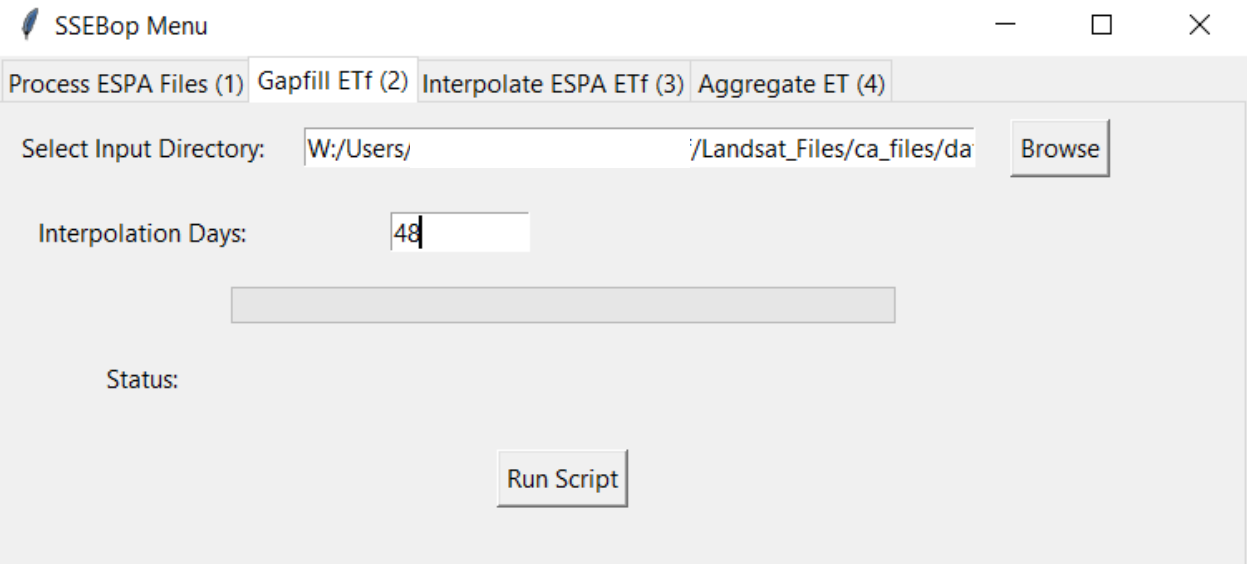
Once you have reviewed the data created, go to the second tab, "**Gapfill ETf (2)**" and begin to fill out the fields needed to run the script.

First, select one of the path/row folders (ex. p43r34):

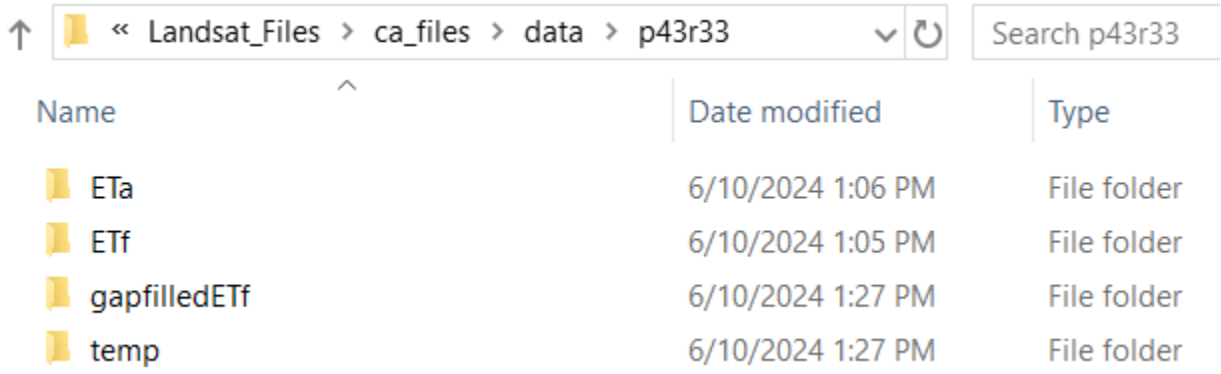


This step looks forwards and backwards to fill gaps in images using the linear interpolation approach. The "**Interpolation Days**" argument controls how far forwards and backwards the script will look to find rasters to fill gaps left by clouds, NoData, etc.

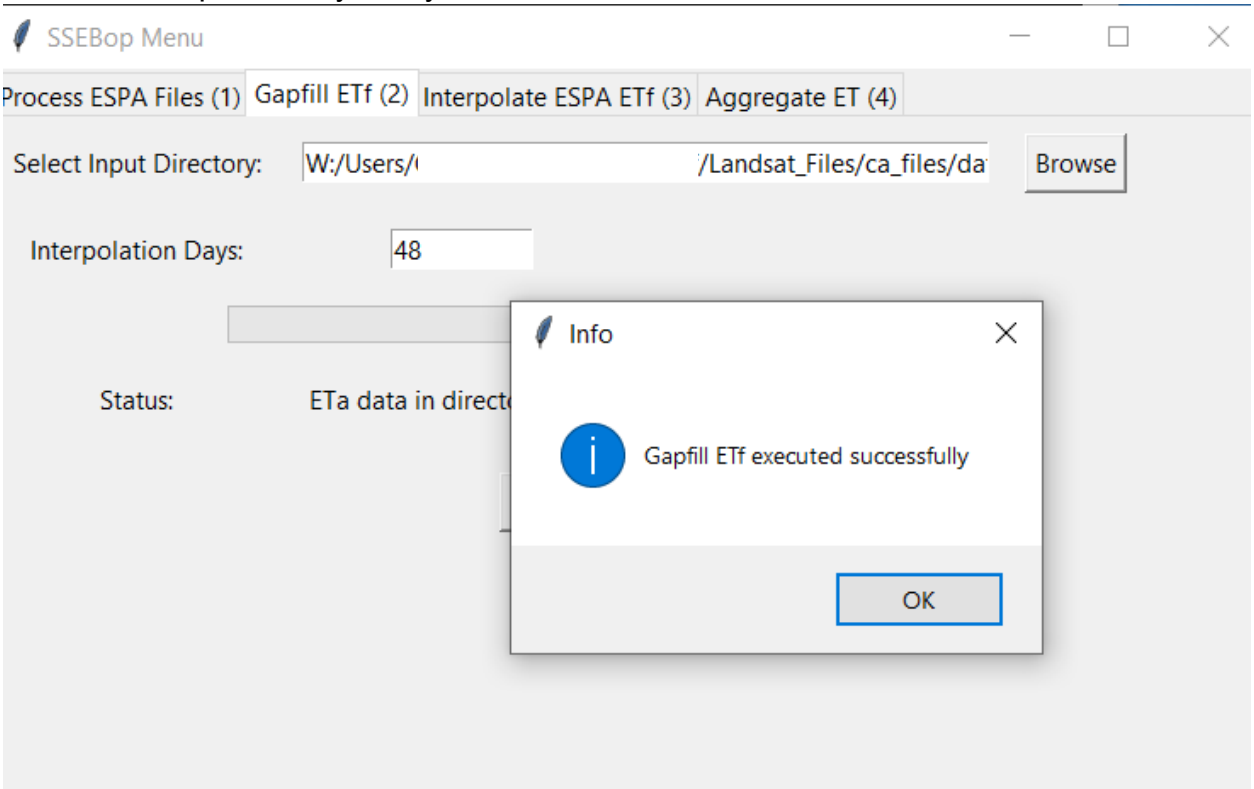
Set Interpolation Days to 48 (but you may increase the number or decrease the number as you need). This is the number of days backwards and forwards the gap filling script will look for valid pixels. It is a compromise between a too-short interval that would leave too many gaps and a too long interval that would interpolate ETf linearly between two very distant dates, adding to uncertainty and leading to errors of commission, rather than errors of omission.



At this point the script can be run. A "**temp**" file and a "**gapfilledETf**" folder are created within the path/row directory.

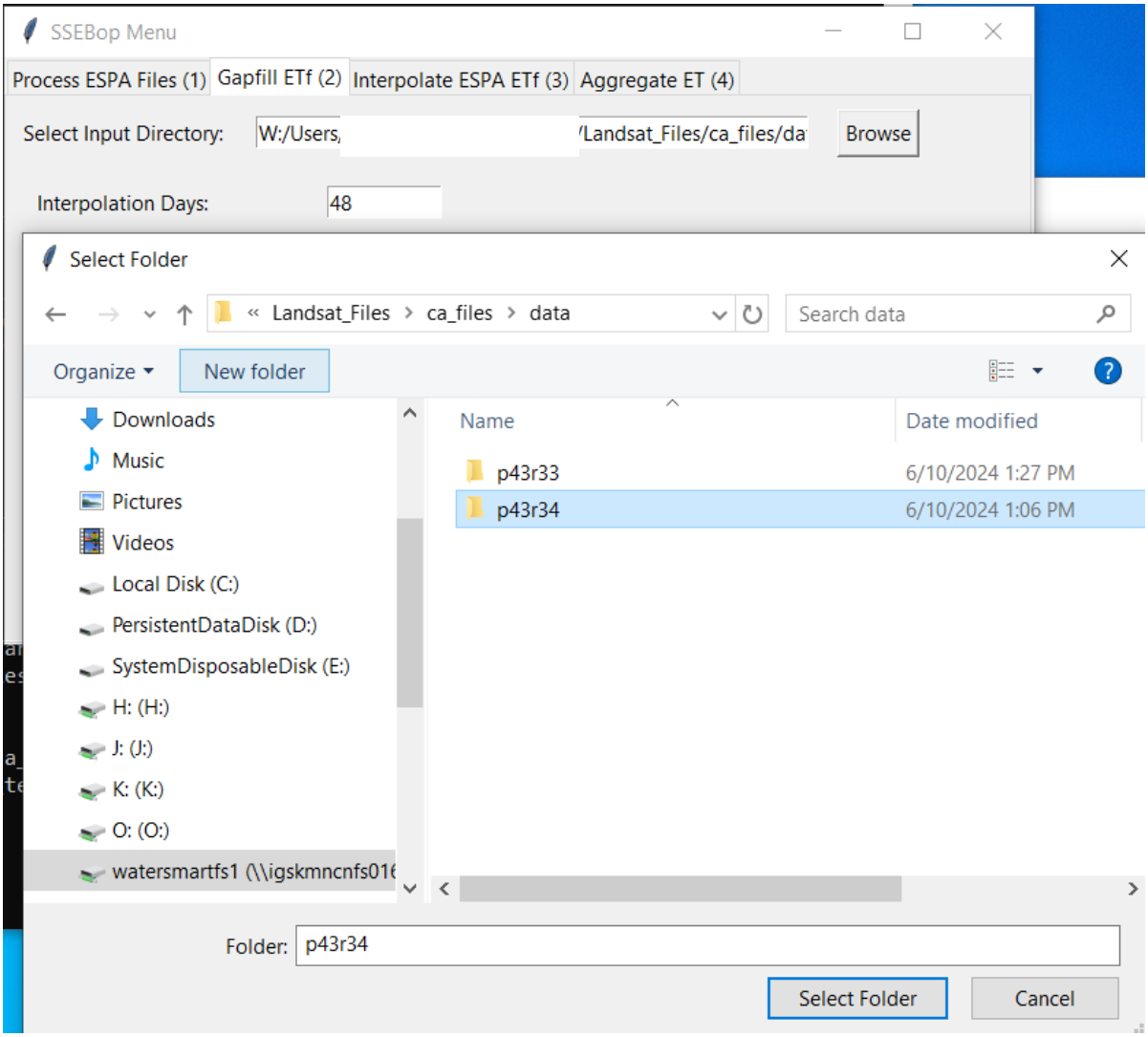


When the script has fully run, you will see the confirmation window.



Click "**OK**"

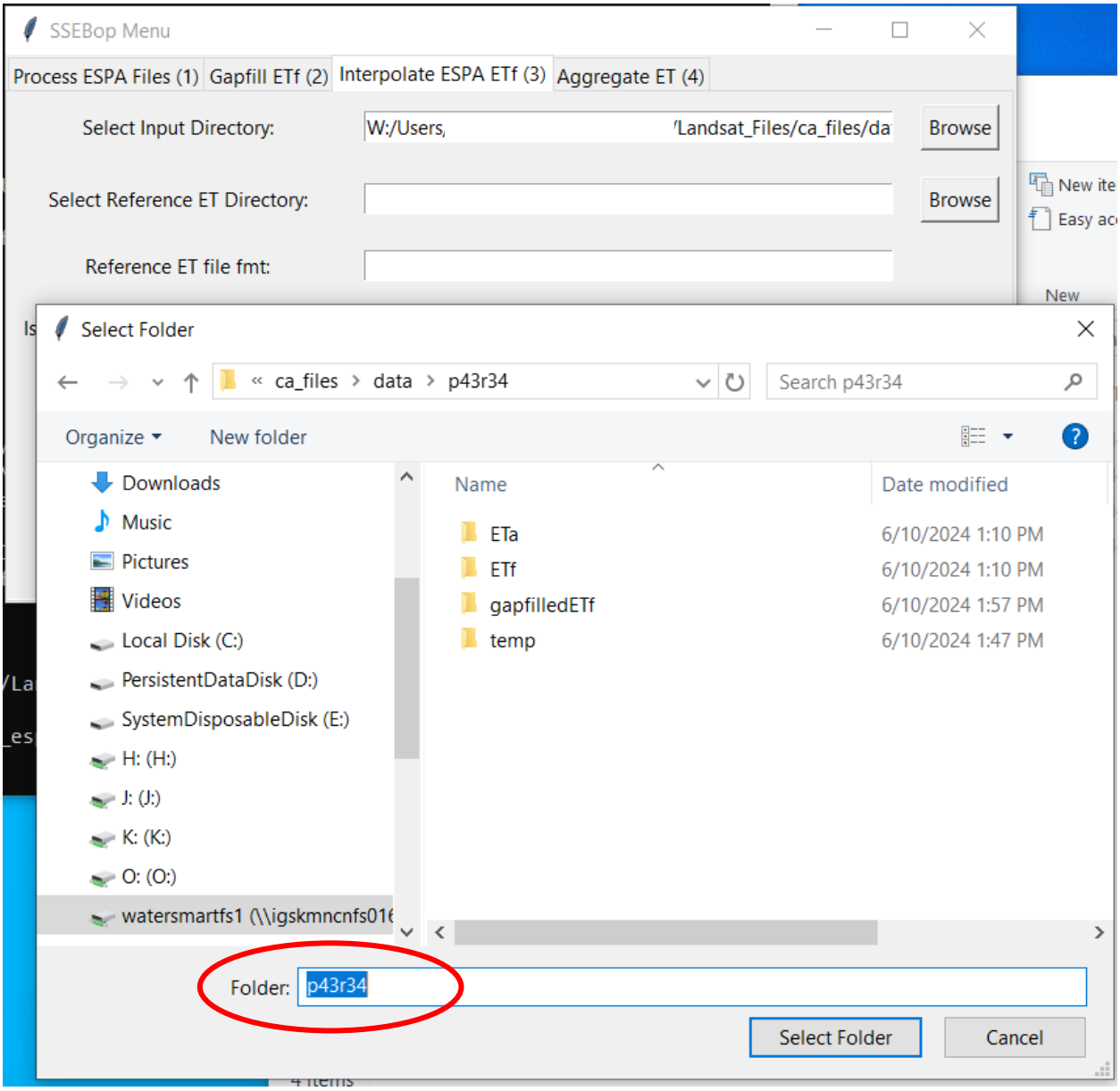
Make sure to repeat the gap filling steps above for each path/row folder you have in your directory. Simply change the Input Directory and leave the Interpolation Days the same and rerun.



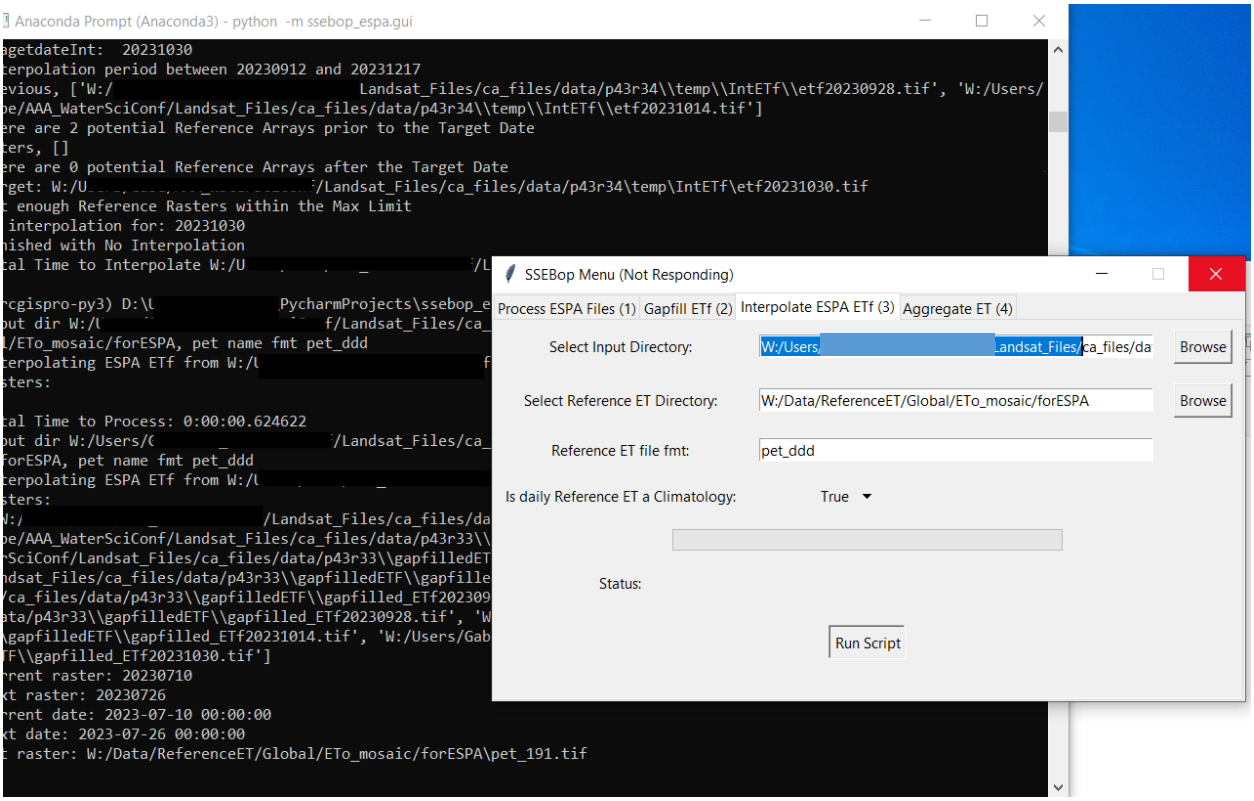
3) Interpolate Daily ETf

This process takes the gap filled ETfs and interpolates between them to approximate daily ETfs between satellite overpasses. The resulting ETf rasters will be multiplied by the corresponding daily reference ET raster to produce an actual ET (ETa) raster in millimeter (mm) for that day.

To run the interpolation script "**Interpolate ESPA ETf (3)**", select one of the path/row folders as the input directory.

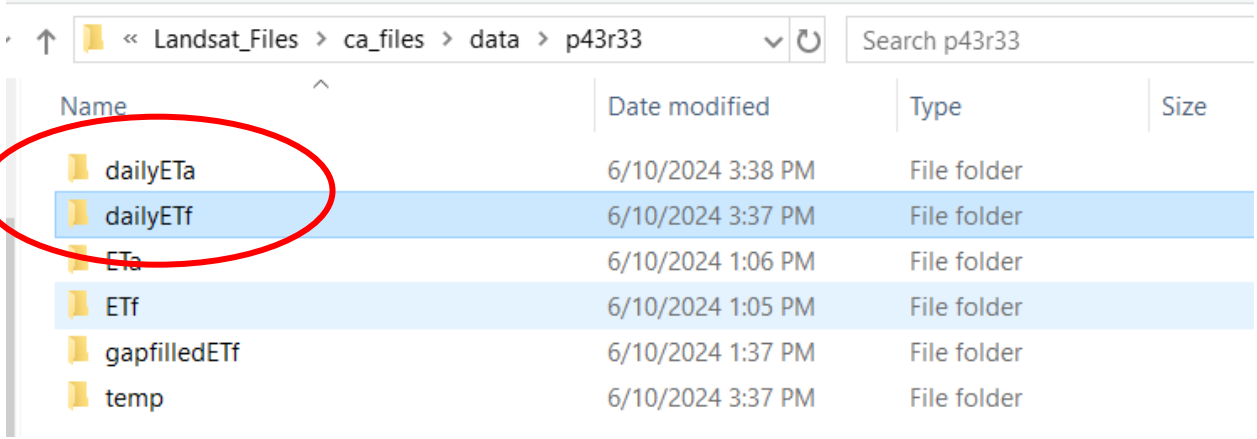


Now you will need to specify a directory containing a reference ET dataset. And, you need to specify the file format and whether or not the reference ET is a daily climatology dataset (average over 20 or 30 years). At this time, the code only supports the use of daily climatology data with a naming convention of “pet_ddd” (ddd = day of the year) as in pet_001.tif or pet_365.tif. The dataset used by ESPA is provided in the Workshop folder (https://edcftp.cr.usgs.gov/project/SSEBop/WaterSciCon2024/) and you should have it saved on your computer.



Once you have populated all the fields, you may run the script until the process is complete. Repeat the same for the additional path/row folders you may have.

Notice that new folders have appeared in the path/row directory named "**dailyETa**" and "**dailyETf**":

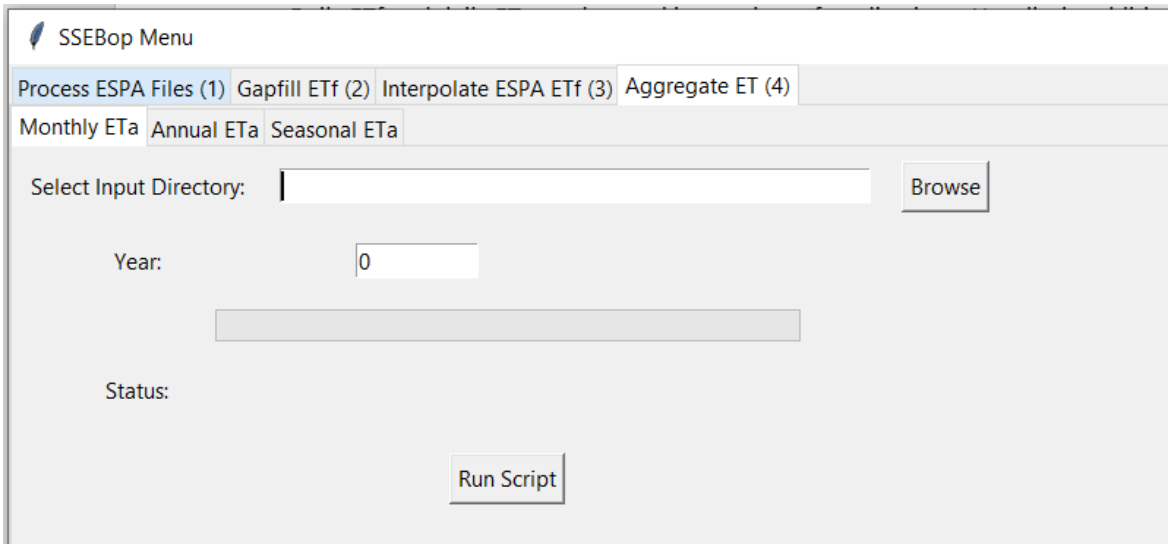


Daily ETf and daily ETa can be used in a variety of applications. Usually, in addition to studying the daily dynamics of ETa or ETf, people are interested in longer term cumulative fluxes of ET. The most common of these are monthly, annual, and seasonal cumulative ET values.

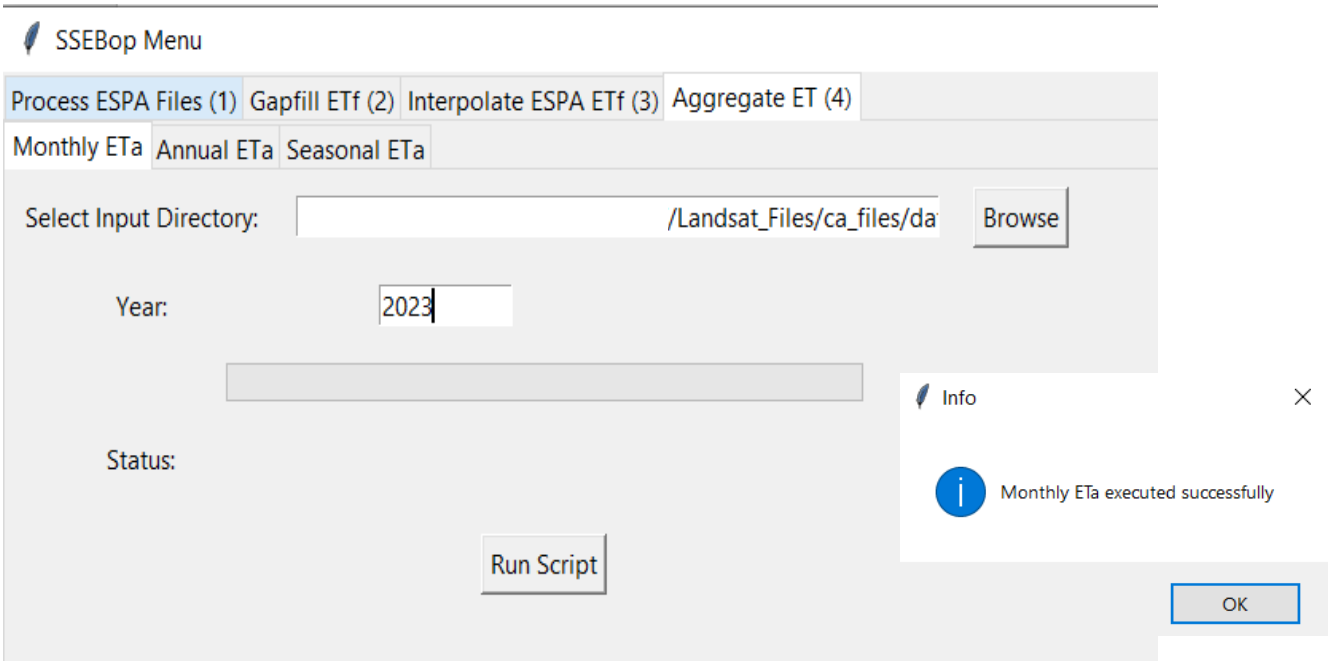
4) Aggregate Daily ETa to monthly, annual, or seasonal summaries

For each of the aggregation operations, the input directory will be the **dailyETa** folder in the path/row directory. Click on the tab "**Aggregate ET (4)**" and choose the aggregation period of your choice. Here we will demonstrate all 3 options. The monthly, annual, and seasonal data are in millimeter per time period.

a) Monthly Aggregation

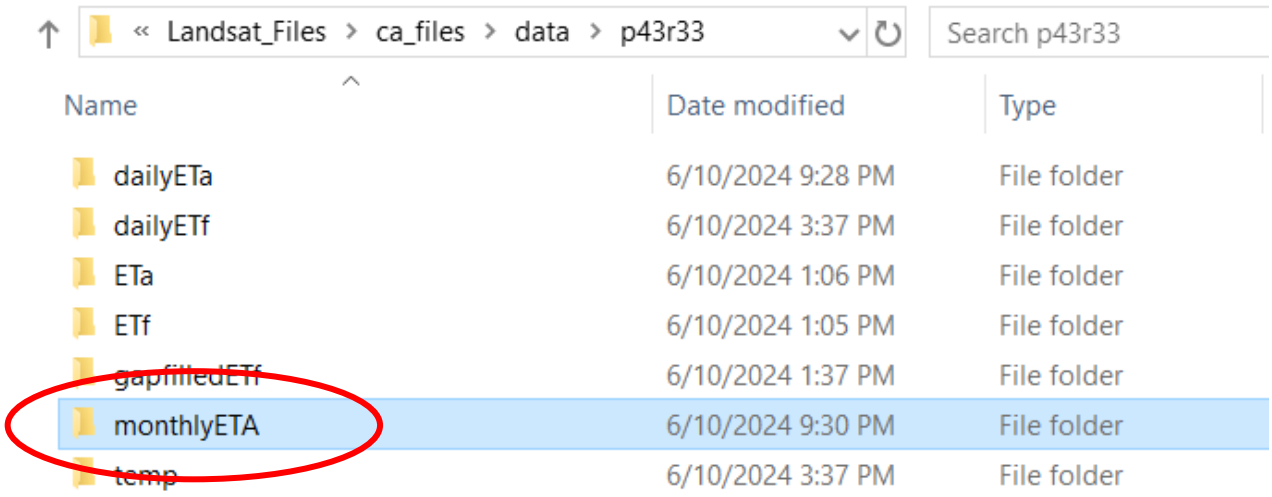


For every complete month of daily ETa rasters within the specified year, the application will accumulate the rasters on a monthly basis in mm and output them in the path/row folder.



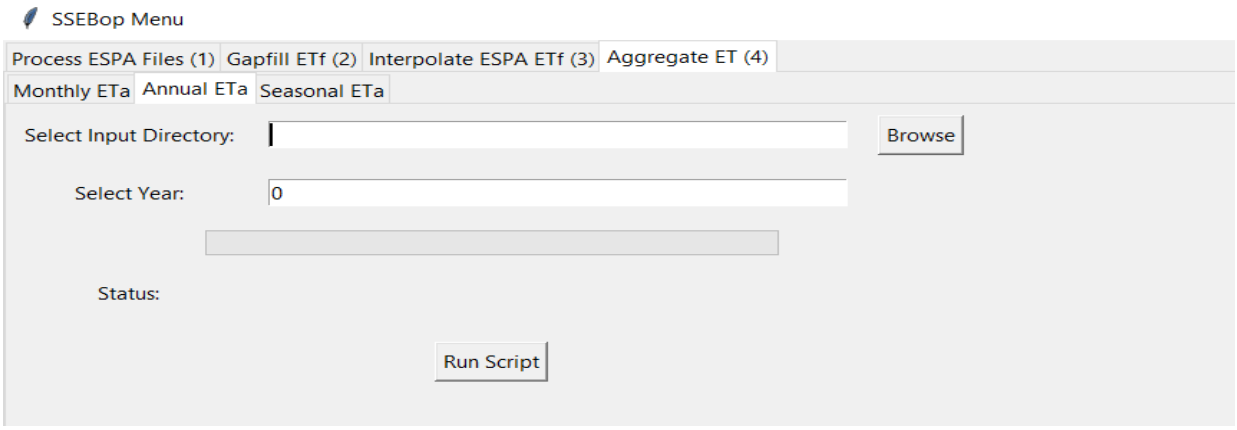
When the script is finished you will see the familiar pop-up window, Click **OK**.

A new directory will appear in the path/row folder named "**monthlyETa**" containing your rasters.

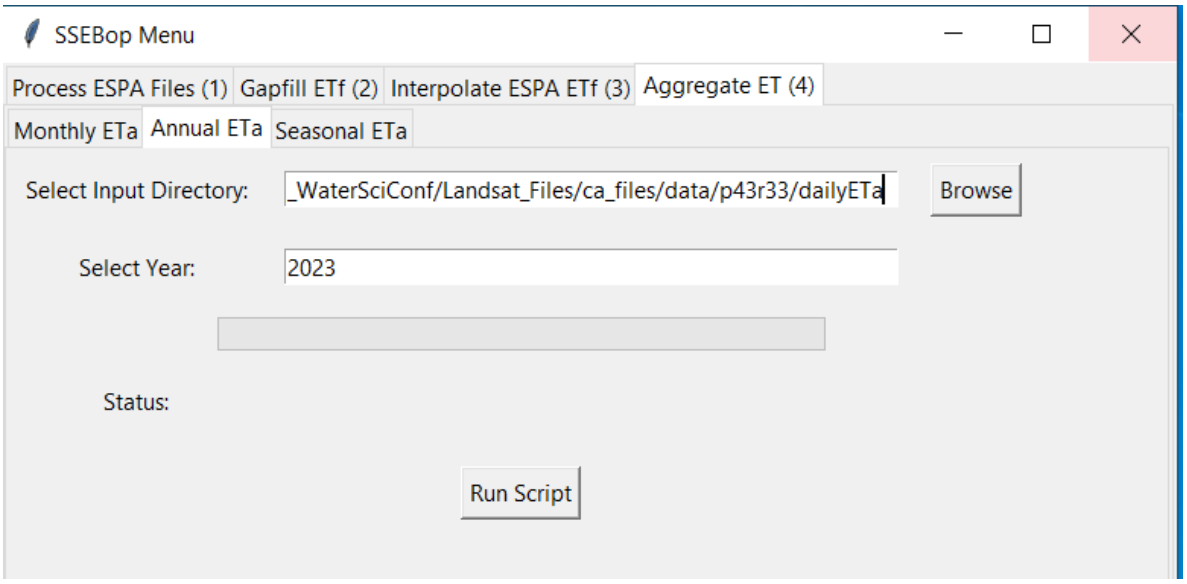


b) Annual Aggregation

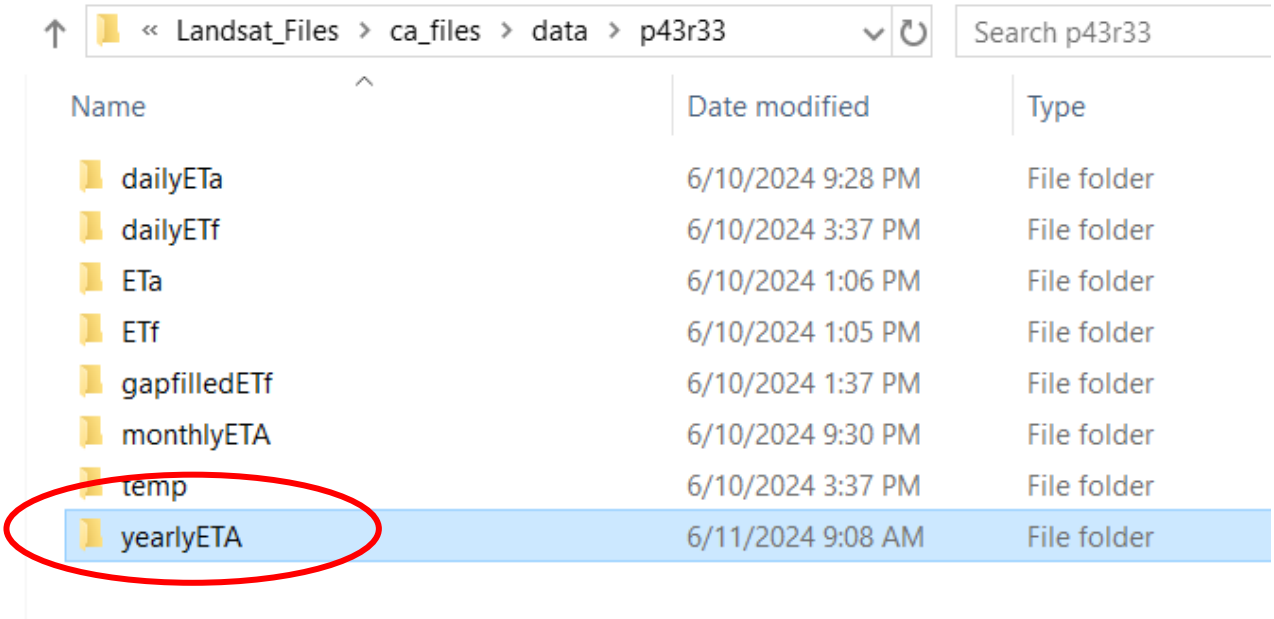
For every complete year of daily ETa rasters available within the input folder (dailyETa), within the specified year, the application will accumulate the rasters on an annual basis in mm and output it in the path/row folder.



When you complete the fields, the GUI should look like this:



When the script finishes, you will see the completion dialog, along with an "**yearlyETa**" folder in the path/row directory.

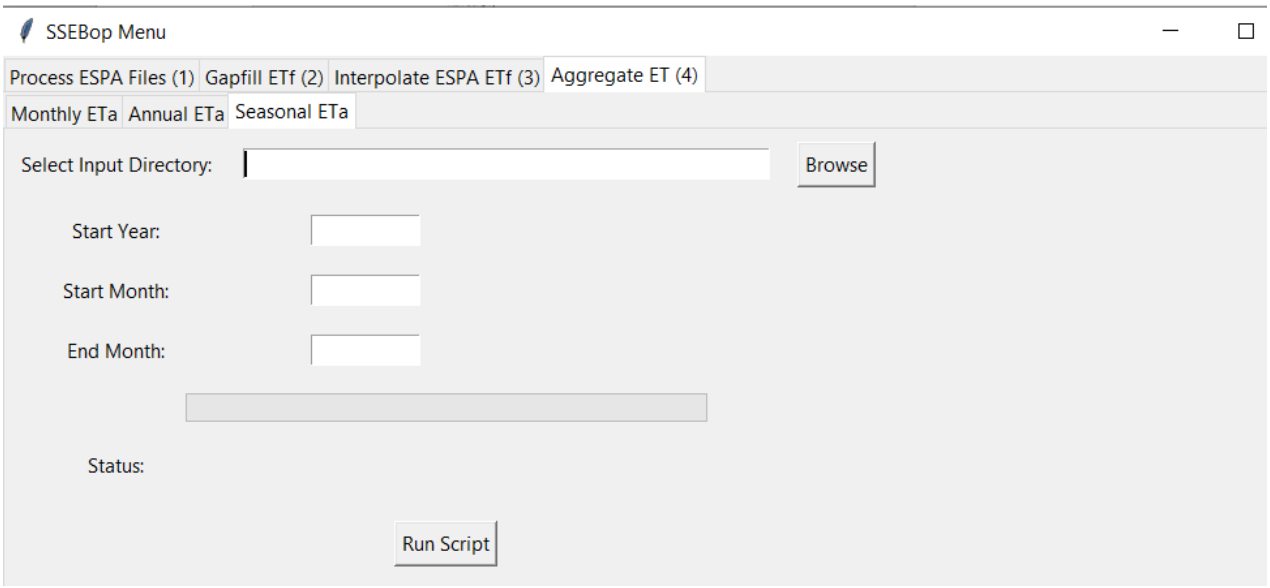


If you run the script for a year without enough daily rasters within dailyETa, you will not see a completion dialog and will see the following exception:

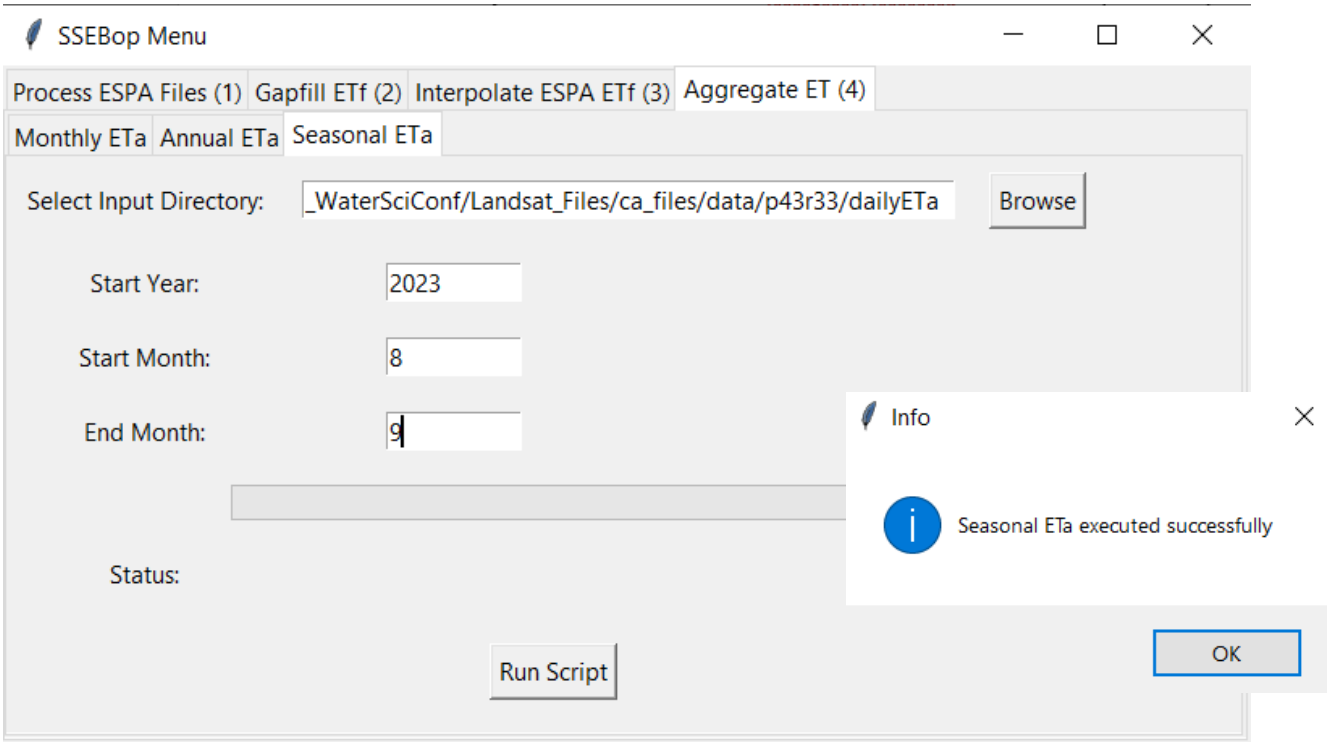
```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\gui.py", line 219, in <lambda>
    command=lambda: self.run_script_annual(input_dir_var.get(), year.get(), progress_var,
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\gui.py", line 348, in run_script_annual
    annual_ETa(input_dir, year, progress=update_progress, status_msg=update_status, root=self.root)
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\processing_steps.py", line 589, in annual_ETa
    raise Exception(f'Incomplete set of daily rasters for year {year}, you must process '
Exception: Incomplete set of daily rasters for year 2023, you must process more rasters prior to aggregating.
```

c) Seasonal Aggregation

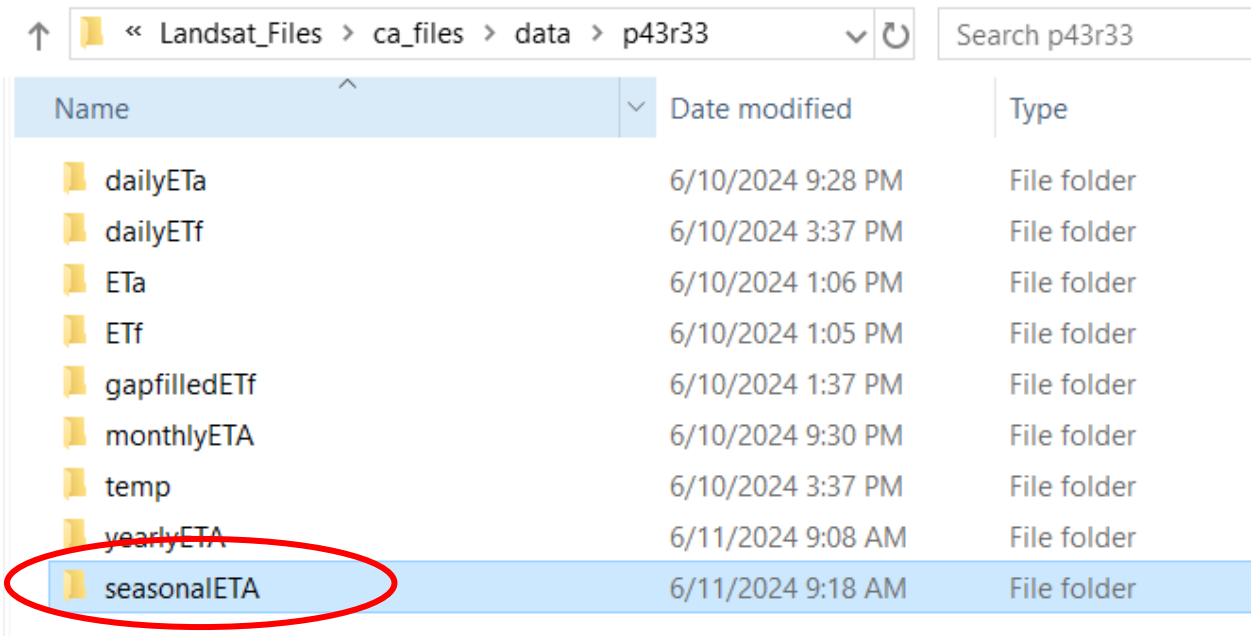
For a complete set of months of daily ETa rasters within the specified year, the application will accumulate the rasters for a user-defined season and output it in mm in the path/row folder. The seasons are not predefined, but chosen by you and your needs. If the start month is a larger number than the end month, the script will assume that the season being accumulated will go into the following calendar year (i.e. the growing season in the southern hemisphere). Seasonal aggregation of periods greater than 1 year are not supported at this time.



Like the other scripts, you select the dailyETa directory as your input location. A start year, start month and end month also must be specified. The example below would be for a season from August to September of 2023.



Once complete you will see the seasonal completion dialog and you will see a new directory for the seasonal results.



Moreover, the seasonal results are marked in the filename with the specific choices made as to the month and year start/end of the season:

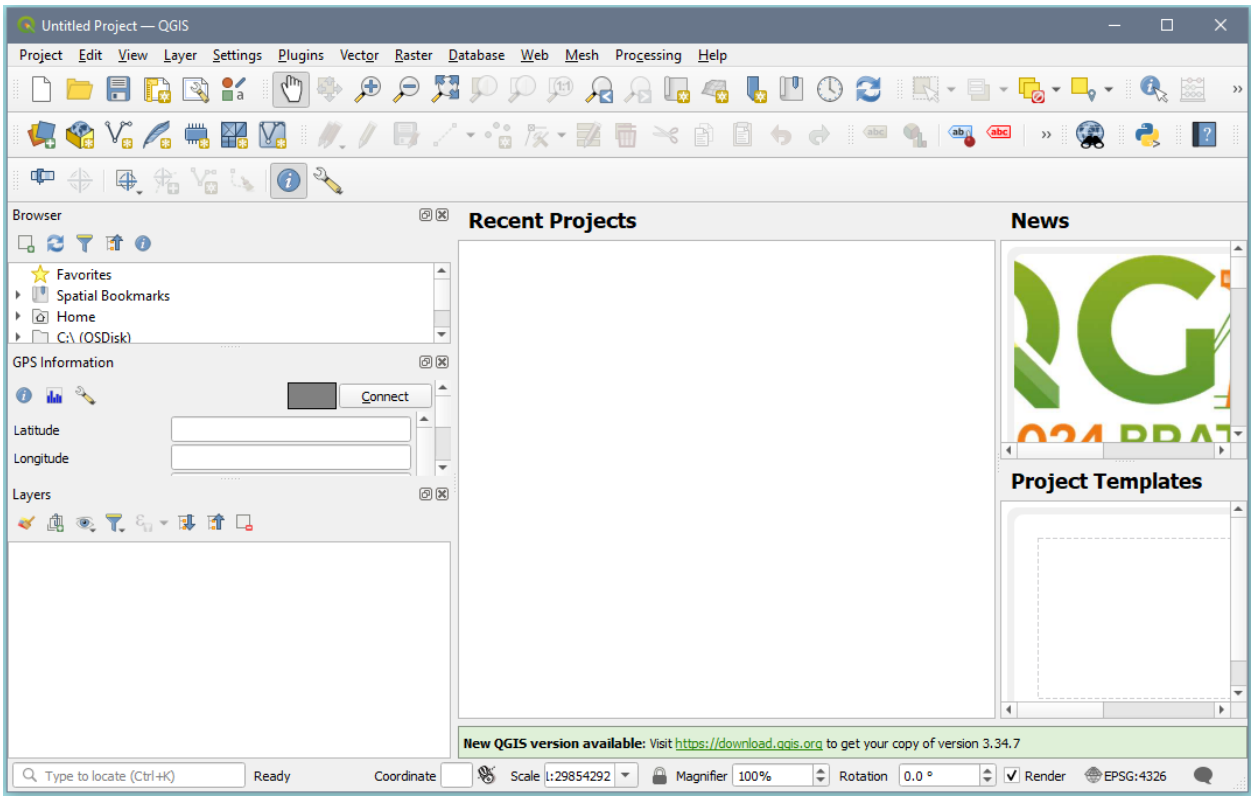
If a user attempts to do a seasonal aggregation when the months are not complete, the terminal prompt will issue an Exception warning in order to prevent an underestimation based on missing data.

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\gui.py", line 252, in <lambda>
    command=lambda: self.run_script_seasonal(input_dir_var.get(), start_month_var.get(),
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\gui.py", line 365, in run_script_seasonal
    seasonal_ETa(input_dir, int(start_month), int(end_month), year, progress=update_progress, status_msg=update_status, r
oot=self.root)
  File "D:\Users\gparrish\PycharmProjects\ssebop_espa_arcgis\ssebop_espa\processing_steps.py", line 626, in seasonal_ETa
    raise Exception(f'Month {m} of year {y} does not have a complete set of daily ETa rasters, '
Exception: Month 7 of year 2023 does not have a complete set of daily ETa rasters, therefore the season cannot be aggrega
ted. Try a different date range or adjust your interpolation.
```

Exploring the aggregated raster data in QGIS

Now that the data was created for monthly, annual, or seasonal time periods, lets inspect the data using a GIS mapping application. We are using QGIS for this demonstration.

First, open the QGIS application.



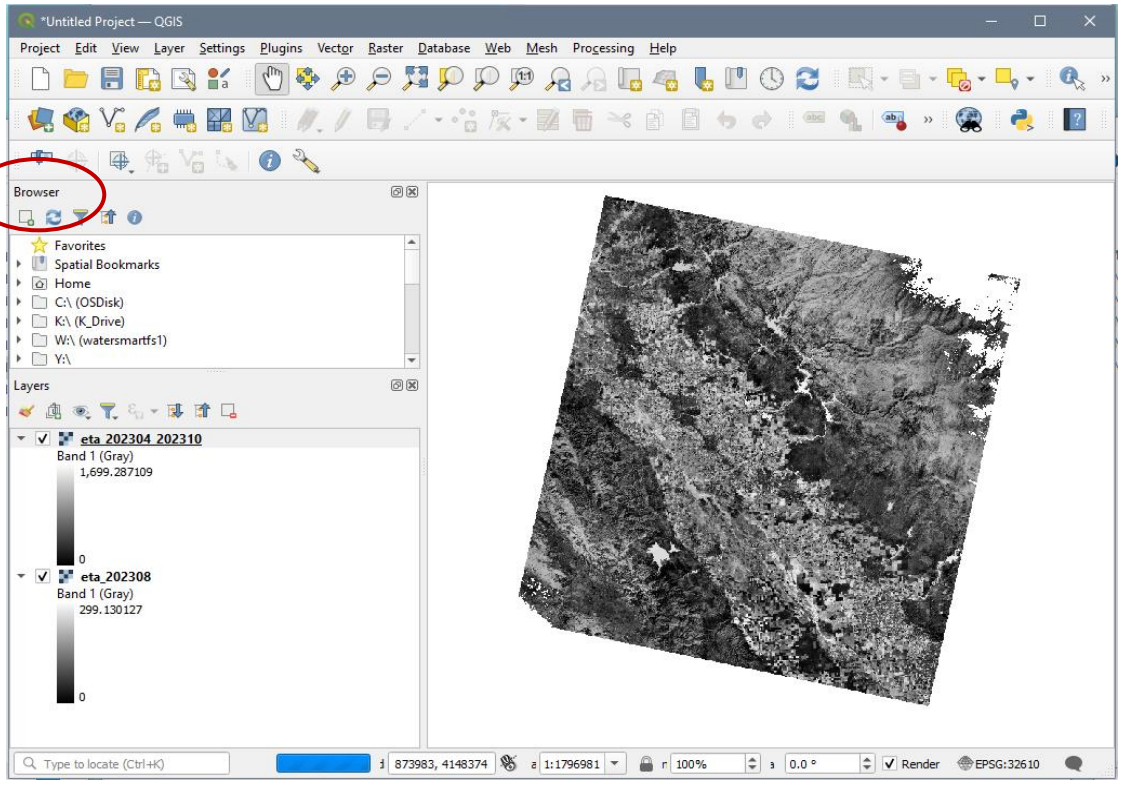
Add a monthly raster and seasonal raster of your choosing. We are using rasters:

a) eta_202308.tif (August 2023) for path/row p43r34

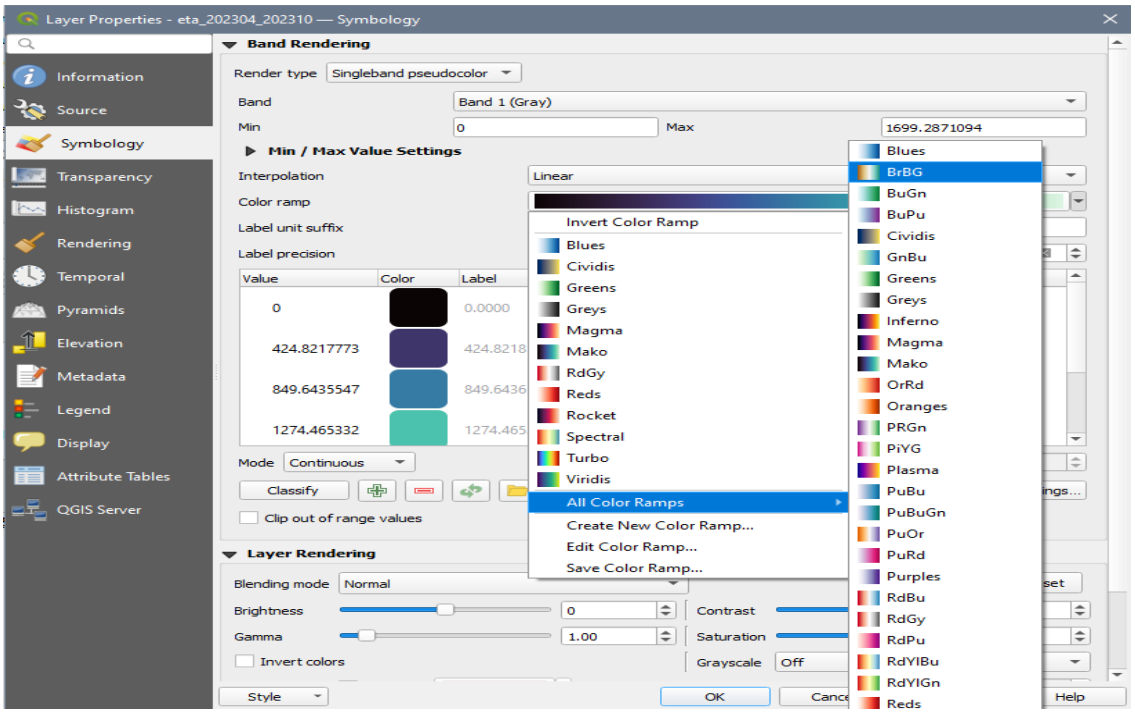
b) eta_202308_202309.tif for path/row p43r34

for this demonstration.

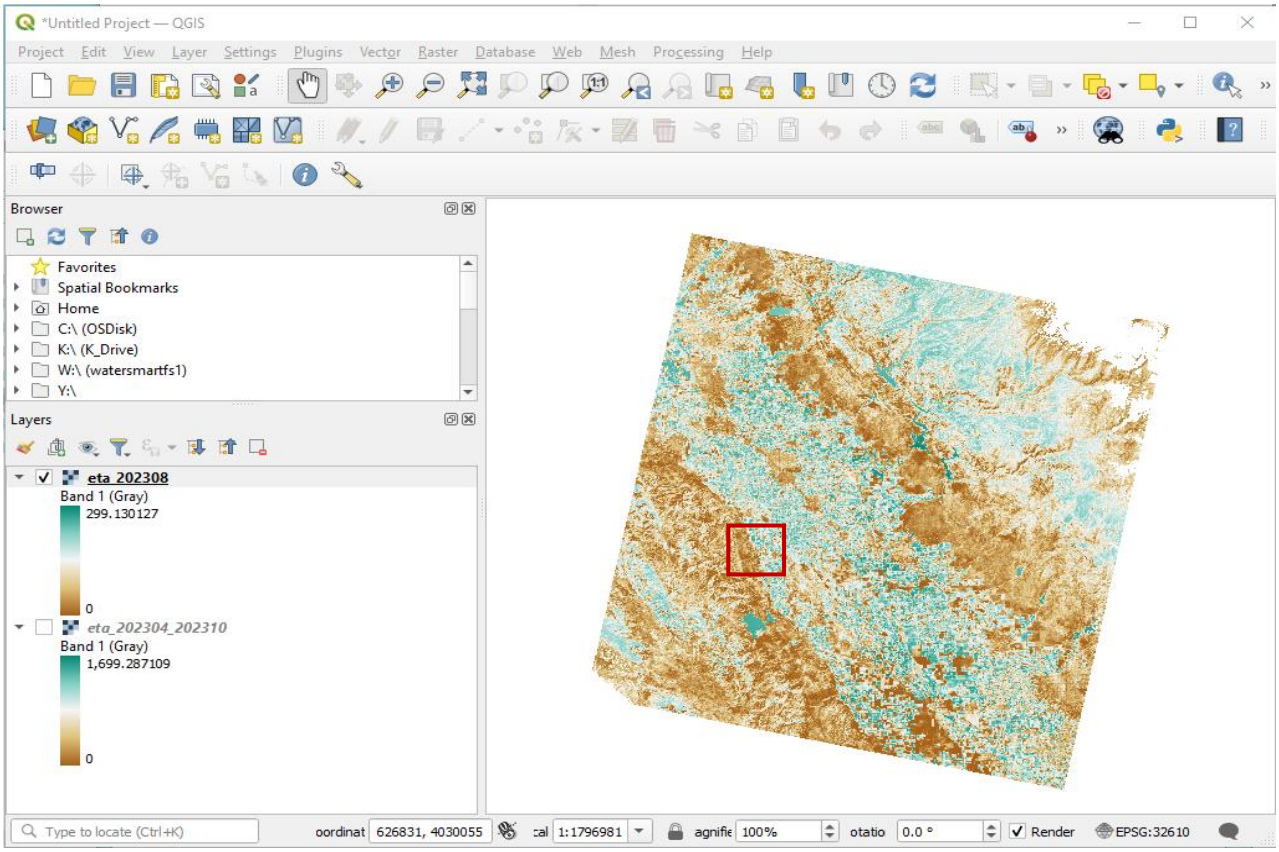
To add data in QGIS, drag and drop the raster file from the folder directory into the mapping window in QGIS or use the Brower Window (top left panel).



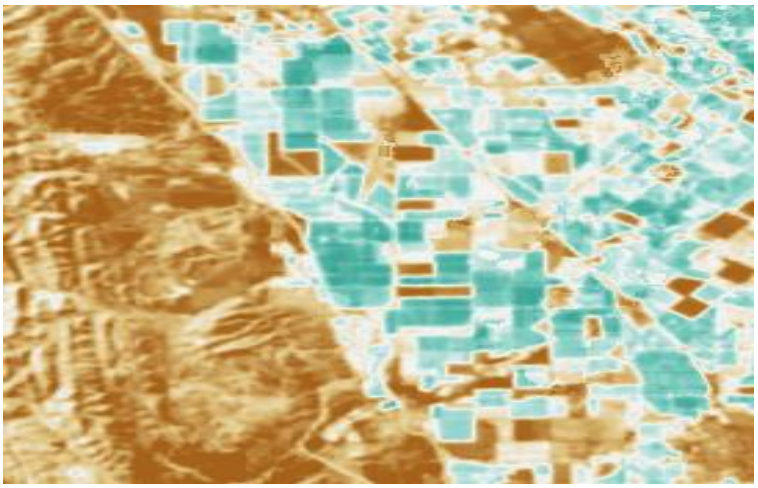
For easier analyses we are changing the color ramp from black-white to brown-green-blue. Right click on the bold file name of the raster and select Layer Properties → Symbology. Here we need to change the **Render type** to **Singleband pseudocolor** and select a **color ramp** best representing the data. Here we chose **BrBG**.



The raster looks like this:

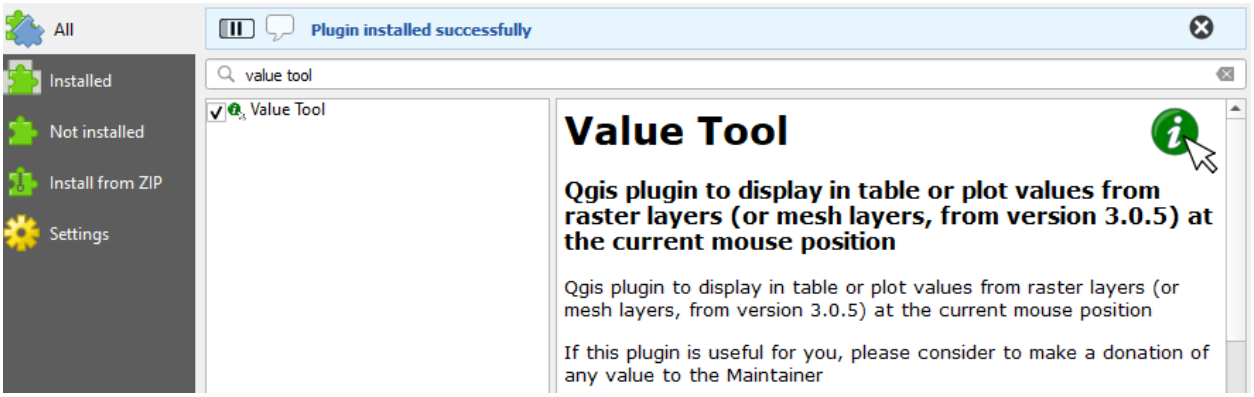


The green (lighter) areas represent high ET values and the brown (darker) areas are low ET values. The area going from the bottom right of the image to the top left on the left side are irrigated fields in the Central Valley. A closer look reveals the individual fields.

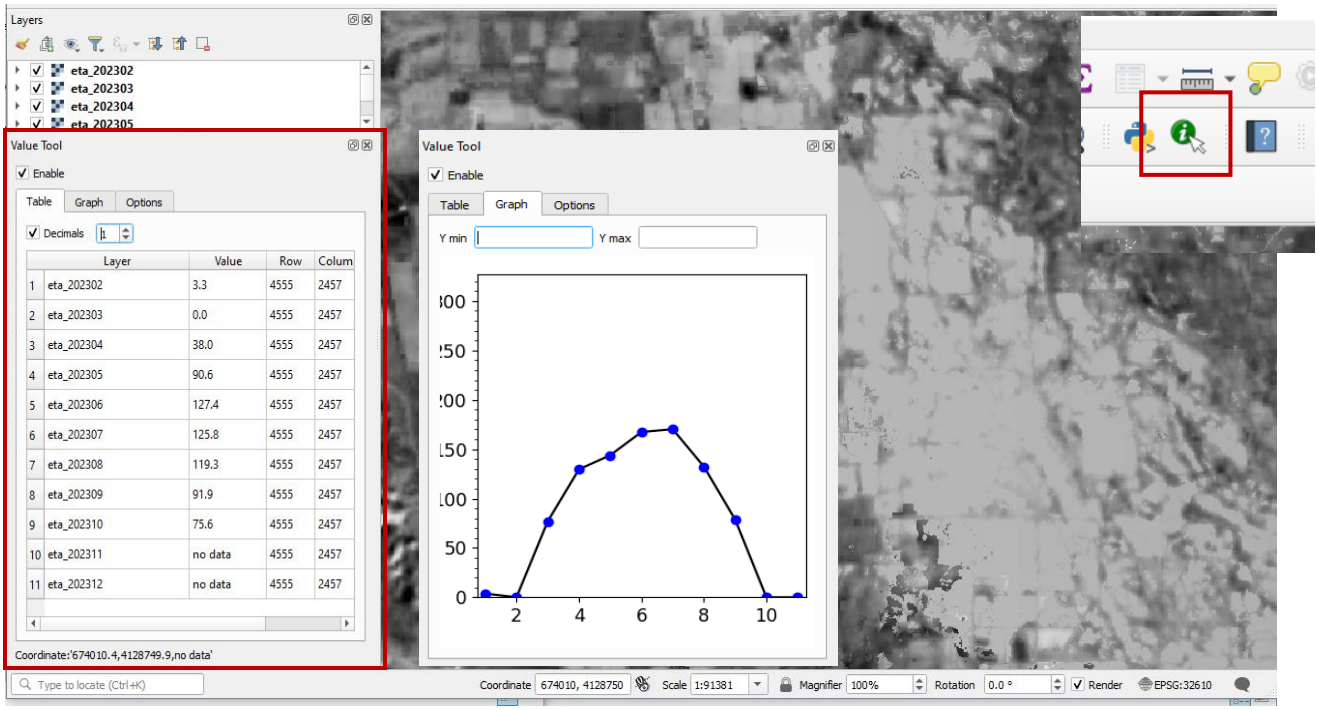


One tool we wanted to point out in QGIS is the Value Tool. The tool allows for viewing the value of a pixel for multiple rasters at the same time in table or graph format.

The Value tool is a plugin to QGIS. Navigate on the top bar to **Plugins**, **Manage and Install Plugins…**., search for **Value Tool**, click **Install** button.



The Plugin will appear on the 2. Row of tool bar towards the right side of the window. Click the icon and the tool panel will show on the left side, usually under the Layers panel.



With the Tool activated, you can pan around the mapping window and investigate the values for each pixel of interest. The Table view will provide the pixel value, here ETa in mm per month, while the Graph view provides a line graph for a better understanding.

Additional functionality and tools in QGIS will be demonstrated as requested in the workshop.