TAMSAT-ALERT API (Version 2)

Guide to installing and running the API

Overview

This guide provides users with an overview of the API, instructions on how to set up and run the API and provides several test cases. We strongly recommend you can successfully complete the test cases (see Section 7) before running the API operationally. If you have any queries, please contact TAMSAT Operations Lead, Dr Ross Maidment (r.i.maidment@reading.ac.uk).

Section 1: Summary

The TAMSAT-ALERT API is software (Python code) designed to allow users to easily generate tailored agricultural drought information for their crop growing season using TAMSAT soil moisture estimates. The API is designed for use anywhere in Africa.

The API provides soil moisture conditions (expressed as the water requirement satisfaction index (WRSI)) from:

1) the start of the season up to a current date in the season ('**WRSI current**')
2) the start of the season out to the end of the season, with soil moisture forecasts used to extend WRSI out to the end of the season ('**WRSI forecast**')

The software is designed so that users do not need to edit the Python code and only need provide minimal inputs to allow the API to run, namely:

- Period of interest
- Region of interest
- The current date
- The climatological period from which soil moisture anomalies are derived
- Meteorological tercile forecast weights

With this information, the API will download the required data, compute the drought metrics, apply the forecast weighting and output the drought metrics in several formats, depending on the data type.

Section 2: Changes since the first version of the TAMSAT-ALERT API

Version 2.0 of the TAMSAT-ALERT API provides the following updates from the original version of the TAMSAT-ALERT API, released in 2022:

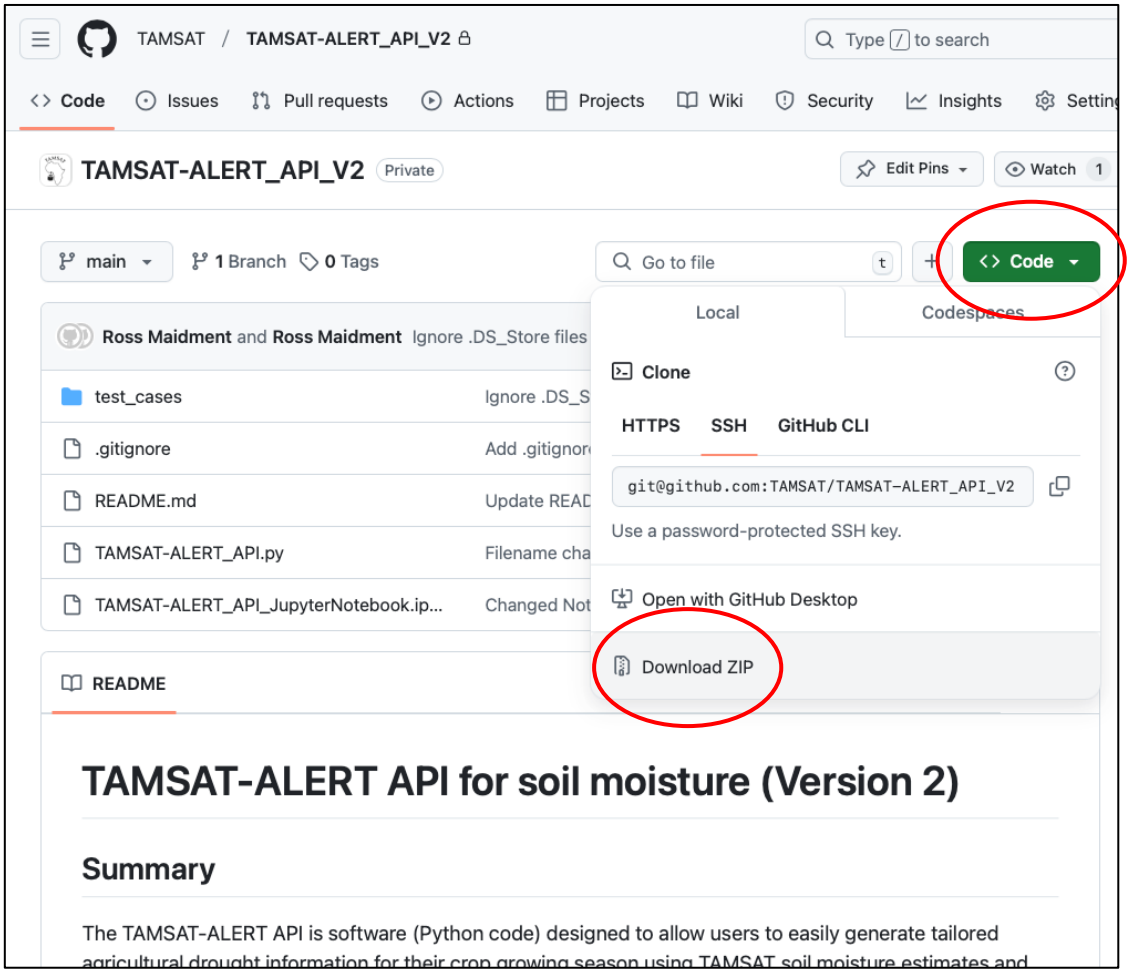
- Water requirement satisfaction index (WRSI) from season start to current date (the previous API only provided forecasts).
- Uses the most recent version of TAMSAT soil moisture dataset (https://www.tamsat.org.uk) developed within the UK Earth Observation Climate Information Service (EOCIS) project.
- Accounts for spatially variable start and end of growing seasons (these inputs need to be created in advance).
- Accounts for spatially variable tercile forecasts in which to weight the TAMSAT-ALERT soil moisture forecasts (this input needs to be created in advance).
- Provides users with the option to weight the TAMSAT-ALERT soil moisture forecasts using spatially variable 6-week ECMWF-S2S tercile precipitation forecasts.

The TAMSAT team will continue to further develop the tool to improve functionality and fix any bugs, but by providing the software it this way, it will be easy for users to keep up-to-date with the latest version whilst continuing to operationally produce soil moisture updates and forecasts.

Section 3: TAMSAT-ALERT API Prerequisites

To run the TAMSAT-ALERT API, you will need the following:

- Python installed on your laptop/PC. We recommend doing this through Anaconda, as described in Section 4.
- Around 6Gb of storage space – this is for Anaconda and the API driving data.
- Download the '**TAMSAT-ALERT_API_V2**' repository from the TAMSAT GitHub repository https://github.com/TAMSAT/TAMSAT-ALERT_API_V2. To do this, click on 'Code' (green button) and then select 'Download ZIP'. If you have already done this step to obtain this guide, move to Section 4.



- Unzip it and save the folder to a suitable location on your PC. You should make a note of this directory. If you are using an external hard drive to save the data, you should save the folder to the external hard drive. Do not move, edit or delete any of the files or folders.

Section 4: Anaconda and Python Installation

The API requires Python 3.9 or later (we cannot guarantee it will work on earlier versions of Python). While the API can be executed on the command line providing arguments, we have also provided a Jupyter notebook in which users can use to run the Python API. Both options are described in Section 5.

Why use Python?

Python is an interpreted, object-oriented, high-level programming language that can be used for many tasks (e.g. data processing, visualisation, web design amongst others). It's high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very easy for beginners to start learning how to run and write Python code. Python's easy to learn syntax emphasizes readability and reduces the cost of program maintenance. The Python interpreter and the extensive standard library are available open source for all major platforms. Python comes installed on Macs and with Linux, but you'll need to install it yourself if you're using Windows.

Options for Python Installation

There are several ways in which users can download Python (full details can be found here: https://realpython.com/installing-python/), some of which include:

- **Direct from the Python website** (https://www.python.org/downloads). This will check your operating system and provide a link for the most recent version of Python for your operating system. If you're using a Mac, Python will already be installed, but it might be out of date so it will need to be updated.
- **Through an Integrated Development Environment (IDE)** such as Spyder, PyCharm and JupyterLab. An IDE is a program that integrate several tools specifically designed for software development that includes an editor designed to handle code (e.g. syntax highlighting and auto-completion) and tools for debugging and testing, which can significantly speed up your work. More details can be found here: https://realpython.com/python-ides-code-editors-guide/.
- **Anaconda** – which is a distribution of the Python (and R) programming languages for scientific computing that aims to simplify package management and deployment. As such, it is software that can be used to install and manage Python.

Why Use Anaconda?

The advantage of using Anaconda is that it comes with IDEs such as Spyder and PyCharm, as well as most of the Python libraries users will need. It also comes with Jupyter Notebook which is an incredibly powerful tool for interactively developing, running and presenting Python code.

As Anaconda comes with (almost) everything a beginner need to run Python code, the rest of this section will focus on installing and operating Anaconda and running Python

code through Jupyter Notebooks. Key details on Anaconda and it's installation have been extracted from DataQuest's guide on using Jupyter Notebooks with Anaconda: https://www.dataquest.io/blog/jupyter-notebook-tutorial/.

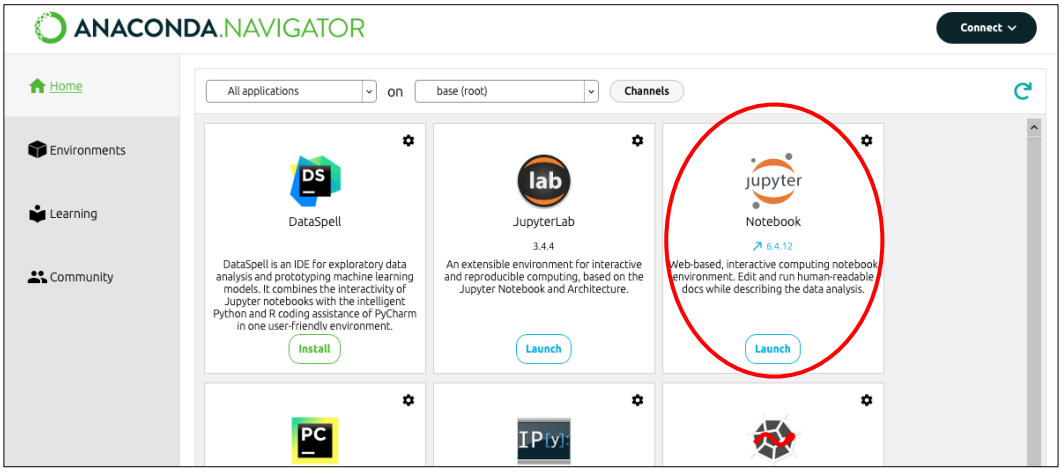
What is a Jupyter Notebook?

A Jupyter Notebook is an open-source web application that allows users to share documents with text, live Python code, images and more. Using Notebooks is now a major part of the data science workflow at companies and organisations across the globe.

Installing Anaconda

The easiest way to get started with Jupyter Notebooks is by installing Anaconda. Anaconda includes the Python language, libraries, editors (such as Jupyter Notebooks) and a package manager. To install Anaconda, Anaconda have provided a video on how to do this for Windows and Mac.

To begin using Anaconda, search for the Anaconda Navigator from the Start Menu in Windows or from the Applications folder if you're on a Mac. Once opened, you should see something like this:



If you see this screen, you have successfully installed Anaconda and are ready to start using Jupyter Notebooks. More details about the Anaconda Navigator can be found here.

Installing Python packages required by the TAMSAT-ALERT API

The TAMSAT-ALERT API uses several Python packages which add functionality to your Python environment. To ensure you have all the required packages installed:

[Windows]

1) Open 'Anaconda Prompt'. This can be found in the 'Anaconda' folder in your Start menu.

2) Type each of the following commands in turn into the Anaconda Prompt window. Some of these may already be installed or may need updating, and some will take a little while to run.

```
conda install wget
conda install xarray
conda install numpy
conda install matplotlib
conda install scipy
conda install pandas
conda install cartopy
```

[Mac]

1) Open the Anaconda Navigator.
2) Navigate to “Environments”.
3) Select your environment where the package needs to be installed (in most cases this will be ‘base (root)’.
4) Click on the green “play” button next to the environment name.
5) Click “Open Terminal” from your environment.
6) In the terminal that appears, type each of the following:

```
conda install wget
conda install xarray
conda install numpy
conda install matplotlib
conda install scipy
conda install pandas
conda install cartopy
```

Note: If the package is not available in any of the Anaconda repositories, type the following:

```
pip install <package-name>
```

Once you have successfully installed all the required packages, you are ready to try reproducing the TAMSAT-ALERT test cases. You will interact with the TAMSAT-ALERT API through Anaconda, at no point will you be required to directly edit the code.

Section 5: Running the TAMSAT-ALERT API

The API require users to specify the following six input arguments given in the table below, with details on how to run the API using (1) a Jupyter Notebook and (2) on the command line shown thereafter.

Argument	Description	Example

poi_start	Start date for period of interest (poi), usually the start of the growing season. Can be either:
	1. Fixed date in format YYYY-MM-DD.
	2. Filename (including path) of a netCDF file providing spatially varying start dates.
		For (1):
		2024-03-01
		(1st March 2024)
		
		For (2):
		F:/TAMSAT-ALERT_API.v2.0/season_start.nc

poi_end	Same as *poi_start*, but for the end of the growing season. Note, users may specify a combination of both fixed and spatially variable dates - e.g. a spatially variable date for *poi_start* and a fixed date for the *poi_end*.
	
	Bear in mind that TAMSAT-ALERT soil moisture forecasts run for 160 (just over 5 months) days ahead. This means that your period of interest cannot extend beyond 160 days from the current date (see below). If it does, you will get an error message and the API will not run.
		For (1):
		2024-05-31
		(31st May 2024)
		
		For (2):
		F:/TAMSAT-ALERT_API.v2.0/season_end.nc

current_date	The date in the season up to which soil moisture estimates are considered and soil moisture forecasts there afterwards. Usually, this date will be today's date. Can be either:
	1. Date in format YYYY-MM-DD. If the date provided is after the last available date of TAMSAT soil moisture estimates, *current_date* will be reset to this date.
	2. '**LATEST**' if wanting to use the most recent soil moisture estimates. This is recommended for operational use.
	
	Given TAMSAT-ALERT forecast files are not issued everyday (typically every 5 days), the API will find the most recent forecast file and run your forecast from this date.
	
	Your output files (see Section 6) will be labelled with either the current date for the 'WRSI
		For (1):
		2024-05-10
		(10th May 2024)
		
		For (2):
		'LATEST'

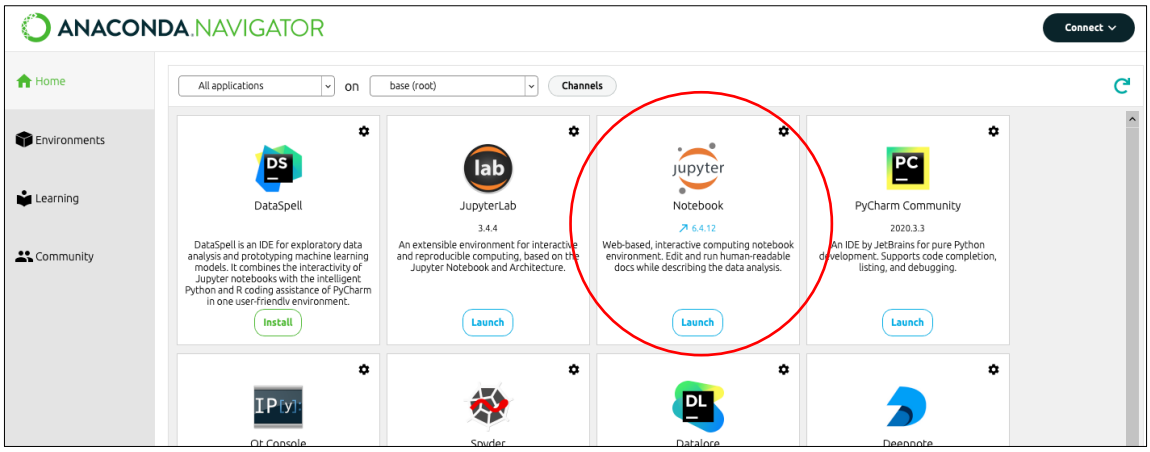
	current' outputs and the forecast date for the 'WRSI forecast' outputs.	
clim_years	The start and end year of the climatological period over which anomalies are computed. This could be, for example, the WMO reference period (1991-2020). Format must be two comma-separated years.	**1991,2020**
coords	Domain coordinates in the following order: N,S,W,E. Must be a comma-separated list of the four coordinates. If your longitudes are west of the Prime Meridian (0° longitude) or your latitudes are south of the Equator (0° latitude) you will need to include a minus sign (-) before the number.	**5.50,-2.5,38.0,45.5**

weights	The TAMSAT-ALERT tool uses a tercile rainfall forecast to weight the soil moisture ensemble forecast. You can obtain a tercile rainfall forecast from any source that is relevant to your region and period of interest. For example, if interested in Kenya's March-April-May rainy season you could use ICPAC's GHACOF rainfall forecasts or another relevant product.
	
	Weights should be proportional and should add up to 1.0. If the tercile rainfall forecast you are using instead adds up to 100, simply dived each value by 100.
	
	The weights give the probability of upper, middle and lower rainfall terciles for the period of interest. These probabilities which are used to weight the TAMSAT-ALERT ensemble forecast can be either:
	1. List of three numbers corresponding to the tercile probabilities in order of [upper, middle, lower.] This can be obtained, for instance, from the GHACOF seasonal rainfall forecast.
	2. '**ECMWF_S2S**' if using the European Centre for Medium Range Weather Forecasts (ECMWF) 6-week S2S tercile rainfall forecasts.
	
	If you cannot find a relevant tercile rainfall forecast, do not want to use the ECWMF S2S
		For (1):
		0.33,0.34,0.33
		(for equal weights)
		
		0.60,0.40,0.20
		(higher weight assigned to upper tercile if above-average rainfall is forecast)
		
		0.10,0.40,0.50
		(higher weight assigned to lower tercile if below-average rainfall is forecast)
		
		For (2):
		ECMWF_S2S

	forecasts or do not want to weight your soil moisture ensemble, then you should use an even weighting of 0.33, 0.34, 0.33 for the upper, middle, lower terciles respectively.	

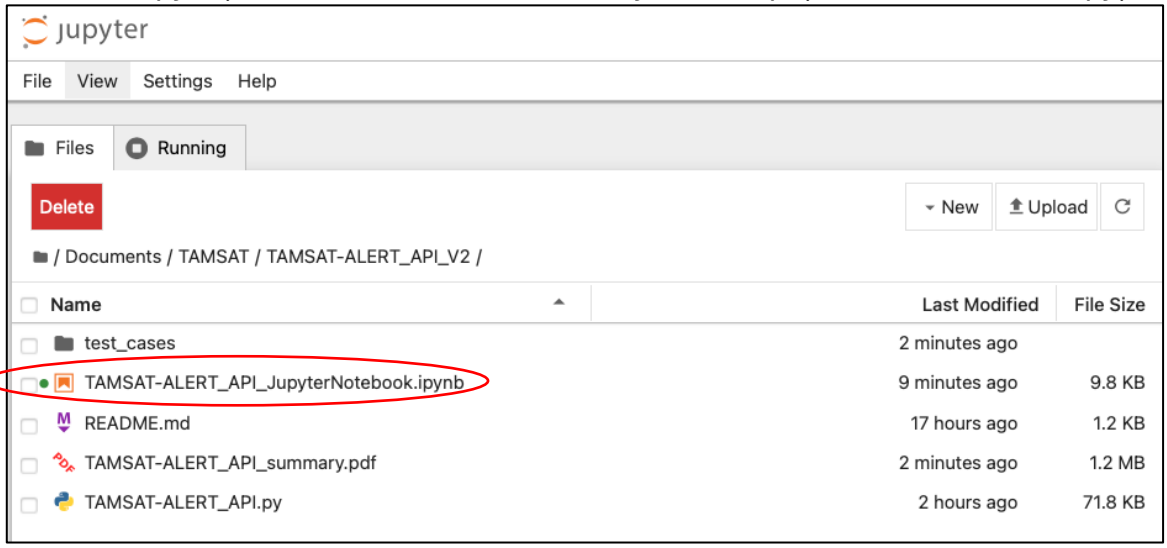
Option 1: Running the API using the TAMSAT-ALERT API Jupyter Notebook

1) Open the Anaconda Navigator and launch the Jupyter Notebook (each installation may have different options but should look similar):

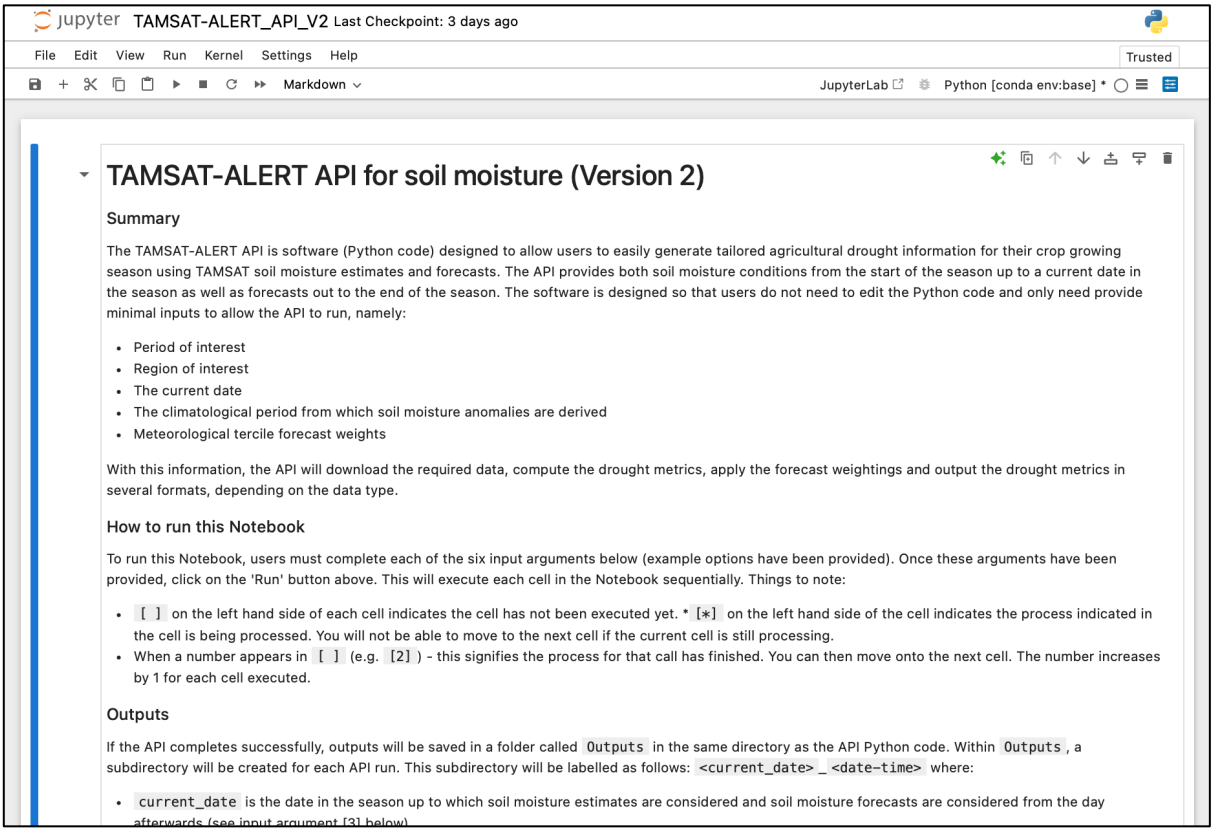


2) This will launch a web browser which will display the Jupyter Notebook interface where you can see all the files in your current directory. If your environment is 'base(root)', then this will show the directories at root level.

3) Now navigate to the folder where the TAMSAT-ALERT API folder is saved. Here, you will see the TAMSAT-ALERT API Jupyter Notebook (*TAMSAT-ALERT_API_Jupyter Notebook.ipynb*) and the TAMSAT-ALERT API Python script *(TAMSAT-ALERT_API.py)*.



4) Click on the Jupyter Notebook – this will open the Notebook as shown below:



5) To run the API, follow the steps given within the Notebook.

Option 2: Running the API using the command line

[Windows]

1) Open 'Anaconda Prompt'. This can be found in the 'Anaconda' folder in your Start menu.
2) Change directory to where you have put the TAMSAT-ALERT_API.v2 folder:

```
cd <directory_path>
```

3) Run the following command:

```
python TAMSAT-ALERT_API.v2.py -poi_start=<poi_start>
-poi_end=<poi_end> -current_date=<current_date>
-clim_year=<clim_years> -coords=<coords> -weights=<weights>
```

where <argument> should be replaced with your requirements.

[Mac]

1) Open a Terminal.
2) Change directory to where you have put the TAMSAT-ALERT_API.v2 folder:

```
cd <directory_path>
```

3) Run the following command:

```
python TAMSAT-ALERT_API.v2.py -poi_start=<poi_start>
-poi_end=<poi_end> -current_date=<current_date>
-clim_year=<clim_years> -coords=<coords> -weights=<weights>
```

where <argument> should be replaced with your requirements.

Section 6: TAMSAT-ALERT API outputs

If the API completes successfully, outputs will be saved in a folder called `outputs` in the same directory as the API Python code. Within `outputs`, a subdirectory will be created for each API run. This directory will be labelled as follows:

`<current_date>_<date-time>`

where:

- `<current_date>` is the date in the season up to which soil moisture estimates are considered and soil moisture forecasts are considered from the day afterwards.
- `<date-time>` is the date and time at which the API was executed.

Within this directory, the following are created:

- a file is created called 'API_input_arguments.csv' which provides a record of the arguments provided to the API
- two sub-directories; `data` and `plots`, which contains the API data and graphical outputs respectively.

Under `data`, the following files are created:

File	Description
wrsi-current_ <POI>_<current_date>.nc e.g. wrsi-current_MAM2024_2024-04-10.nc	*NetCDF file.* The mean soil moisture (WRSI) from the start of season until the current date in the season. Variables provided: • **wrsi_current** (WRSI for given period) • **wrsi_current_clim** (WRSI climatology for given period using reference period provided by '*clim_years*') • **wrsi_current_sd** (WRSI standard deviation for given period using reference period provided by '*clim_years*') • **wrsi_current_anom** (WRSI anomaly for given period, i.e. *wrsi_current - wrsi_current_clim*) • **wrsi_current_percent_anom** (WRSI % anomaly for given period, i.e. (*wrsi_current/ wrsi_current_clim)*100*)
wrsi-forecast_ <POI>_<forecast_date>.nc e.g. wrsi-current_MAM2024_2024-04-10.nc	*NetCDF file.* The mean soil moisture (WRSI) from the start of season out to the end of the season. Soil moisture forecasts are used to extend the soil moisture from the current date out to the end of the season. Variables provided:

	• **wrsi_forecast_ens_mean** (Forecast ensemble mean WRSI for the period of interest) • **wrsi_forecast_ens_sd** (Forecast ensemble standard deviation of WRSI for the period of interest) • **wrsi_clim** (WRSI climatology for period of interest using reference period provided by '*clim_years*') • **wrsi_sd** (WRSI standard deviation for period of interest using reference period provided by '*clim_years*') • **wrsi_forecast_anom** (WRSI anomaly for period of interest, i.e. *wrsi_forecast_ens_mean - wrsi_clim*) • **wrsi_forecast_percent_anom** (WRSI percentage anomaly for period of interest, i.e. (*wrsi_forecast_ens_mean/wrsi_clim)*100*)
ensemble-wrsi-forecast_<POI>_<forecast_date>.nc	*NetCDF file*. The forecasted soil moisture each day throughout the forecast period for each ensemble member.
prob_lower_tercile_<POI>_<forecast_date>.nc	*NetCDF file*. The probability that soil moisture in the forecast period with fall into the lowest tercile, for each grid cell in the region of interest.
terciles_<POI>_<forecast_date>.csv	CSV file. The probabilities of soil moisture (WRSI) in the forecast period falling into either the lower, mid or upper tercile. Values represent the average over the domain.

Under `plots`, the following files are created:

File	Description
wrsi-current_map_<POI>_<current_date>.png	*Image file*. The mean soil moisture (WRSI) from the start of season until the current date in the season. Maps provided show: • **WRSI climatology** for given period using reference period provided by '*clim_years*' • **WRSI current** for given period • **WRSI % anomaly** for given period, i.e. (*wrsi_current/ wrsi_current_clim)*100*)

wrsi-forecast_map_ <POI>_<forecast_date>.png	*Image file*. The mean soil moisture (WRSI) from the start of season out to the end of the season. Soil moisture forecasts are used to extend the soil moisture from the current date out to the end of the season. Maps provided show: • **WRSI climatology** for period of interest using reference period provided by '*clim_years*' • **WRSI forecast** derived from the forecast ensemble mean WRSI for the period of interest • **WRSI forecast % anomaly** derived from the forecast ensemble mean WRSI for the period of interest, i.e. (*wrsi_forecast_ens_mean/wrsi_clim)*100*)
probability_map_ <POI>_<forecast_date>.png	*Image file*. A map of the region of interest showing the probability that soil moisture in the forecast period will fall into the lowest tercile.
probability_distribution_ <POI>_<forecast_date>.png	*Image file*. The probability distribution of forecasted soil moisture (WRSI) compared to climatology. Values represent the average over the domain.
timeseries_ <POI>_<forecast_date>.png	*Image file*. A time series of current and ensemble soil moisture forecasts compared to the climatology. Values represent the average over the domain.

Where:

- <POI> is the period of interest, corresponding to the start and end of the season. If spatially variable start/end of season dates are provided, the API will use the earliest (for start of season) and latest (for end of season) dates within the domain. To construct the <POI> string, the first letter of the months within the season are used – e.g. 'MAM' is used to represent March, April and May.
- <current_date> is the date specified by the user and presents the date in the season up to which soil moisture estimates are considered (and soil moisture forecasts there afterwards).
- <forecast_date> is the date from which the TAMSAT-ALERT forecasts begin. They will typically be within a few days of <current_date>, depending on the availability of the TAMSAT-ALERT forecast files.

Section 7: Reproducing test cases

The following test cases are provided so that users can test they are running the TAMSAT-ALERT API correctly. For each test case, outputs have been provided within the `test_cases` folder on the Github repository for comparison.

Test case 1: Seasonal soil moisture estimates and forecasts issued on 10th April 2024 for Kenya's March-May (Long-rains) rainy season.

Run the test case using either the command line option or the provided Jupyter Notebook:

Command line:

1) Navigate to the folder where the TAMSAT-ALERT API Python code is saved.

2) Type the following code into the 'Anaconda Prompt' window, remembering to change directory to where the API Python script is located:

```
python TAMSAT-ALERT_API.v2.py -poi_start=2024-03-01 -poi_end=2024-05-31
-current_date=2024-04-10 -clim_year=1991,2020 -coords=6,-5,32,43
-weights=0.33,0.34,0.33
```

Jupyter Notebook:

1) Open the TAMSAT-ALERT API Jupyter Notebook as detailed in Section 5.

2) Populate the fields using the following information and then run the Notebook.

```
poi_start='2024-03-01'
poi_end='2024-05-30'
current_date=''2024-04-10
clim_year='1991,2020''
coords='6,-5,32,43'
weights='0.33,0.34,0.33'
```

This might take some time to run depending on your internet speed as it will have to download both historic and forecast data.

Once completed, you should find the following files in the `outputs` folder within the subfolder created for your API run:

Files under `data` folder
• wrsi-current_MAM2024_2024-04-10.nc • wrsi-forecast_MAM2024_2024-04-06.nc • ensemble-wrsi-forecast_MAM2024_2024-04-06.nc • prob_lower_tercile_MAM2024_2024-04-06.nc • terciles_MAM2024_2024-04-06.nc

Files under `plots` folder
• wrsi-current_map_MAM2024_2024-04-10.png • wrsi-forecast_map_MAM2024_2024-04-06.png • probability_map_MAM2024_2024-04-06.png • probability_distribution_MAM2024_2024-04-06.png • timeseries_MAM2024_2024-04-06_32.0-43.0--5.0-6.0.png

You can compare your plots against those stored in the `test_cases` folder. They should be the same. If they are, well done, you have successfully reproduced test case 1. You can now move onto test case 2. If not, check that your input code matches that above. If you get any errors, please copy your input code and the error message and send it to Ross Maidment (TAMSAT Operations Lead) at r.i.maidment@reading.ac.uk.

Test case 2: Seasonal soil moisture estimates and forecasts issued on 10th May 2024 for Kenya's March-May (Long-rains) rainy season.

- Type the required code into the 'Anaconda Prompt' window or modify the Jupyter Notebook – this time, you'll have to work out what the code is for yourself. Hint: you'll only need to change the current date.
- If using a Jupyter Notebook, you can edit the existing notebook or duplicate the existing notebook for test case 2.
- Once completed, compare your plots to the those in the `test_cases` folder.
- Once you have successfully reproduced the test cases, you are ready to develop your own bespoke forecasts.

Test case 3 (optional): Seasonal soil moisture estimates and forecasts issued on 20th April 2024 for Kenya's March-May (Long-rains) rainy season using weights provided by the ECMWF S2S terciles forecasts.

- Type the required code into the 'Anaconda Prompt' window or modify the Jupyter Notebook – again, you'll have to work out what the code is for yourself. Hint: you'll need to change the arguments for the current date and the weights.
- Once completed, compare your plots to the those in the `test_cases` folder.

Test case 4 (optional) - TBC: Seasonal soil moisture estimates and forecasts issued on 10th May 2024 for Kenya's March-May (Long-rains) rainy season using a spatially variable season start date.

- Type the required code into the 'Anaconda Prompt' window or modify the Jupyter Notebook – again, you'll have to work out what the code is for yourself. Hint: you can use the same inputs for test case 2, you'll need to change the arguments for the poi_start from '2024-05-10' to the file path of where the spatially variable season start file is (this file is provided within the test case 4 folder on Github).
- Once completed, compare your plots to the those in the `test_cases` folder.