sen1mosaic Documentation

Release 1

Samuel Bowers

Dec 02, 2020

Contents

This is a set of tools to aid in the production of large-scale seasonal mosaic products from Sentinel-1 data.

Building large-scale mosaics of Sentinel-1 data for land cover mapping is more difficult than with more commonly-used optical data (e.g. Landsat), with existing tools still under-development and hard to use. The goal of this repository is to streamline this processing chain with a set of straightforward command line tools.

This repository contains three command-line based scripts to perform the following tasks:

- Downloading Sentinel-1 data from the Copernicus Open Access Hub for a particular latitude/longitude window, specifying date ranges and ascending/descending orbits. This is based on the Sentinelsat utility.
- Executing SNAP graph processing tool to calibrate, filter and perform terrain-correction on Sentinel-1 GRD images.
- Mosaicking pre-processed Sentinel-1 files into larger GeoTIFF files that are suitable for image classification.

CHAPTER 1

More information

For more information, refer to the ‘Satellite Monitoring for Forest Management’ webpage..

CHAPTER 2

How do I get set up?

These tools are written in Python for use in Linux. You will need to have first successfully installed the following:

- Sentinelhub: A library for searching and downloading Sentinel products.
- SNAP: Pre-processing tools for Sentinel-1 data.

The modules used in these scripts are all available in Anaconda Python.

CHAPTER 3

Who do I talk to?

Written and maintained by Samuel Bowers (sam.bowers@ed.ac.uk).

CHAPTER 4

Contents:

4.1 Setup instructions

4.1.1 Requirements

This toolset is written for use in Linux.

You will need access to a PC or server with at least:

- Python 3
- 8 GB of RAM to run SNAP.
- 8+ GB of RAM to combine images into a mosaic tile (depending on resolution/extent).

4.1.2 Installing Anaconda Python

These tools are written in Python. We recommend the Anaconda distribution of Python, which contains all the modules necessary to run these scripts.

To install Anaconda Python, open a terminal window, change directory to the location you'd like to install Anaconda Python, and run the following commands:

```
wget https://repo.anaconda.com/archive/Anaconda2-5.1.0-Linux-x86_64.sh
chmod +x Anaconda2-5.1.0-Linux-x86_64.sh
./Anaconda2-5.1.0-Linux-x86_64.sh
```

If this has functioned, on executing `python` in a terminal window, you should ssee the following:

```
Python 2.7.14 |Anaconda, Inc.| (default, Dec  7 2017, 17:05:42)
[GCC 7.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

4.1.3 Setting up your Anaconda environment

Note: The Anaconda environment required for sen1mosaic and sen2mosaic is identical. If you already have a sen2mosaic environment set up, it can be used in place of a new environment.

To ensure you are working with the appropriate version of Python as well as the correct modules, we recommend that you create an Anaconda virtual environment set up for running `sen1mosaic`. This is done by running the following commands in your terminal or the Anaconda prompt (recommended procedure):

```
conda create -n sen1mosaic -c conda-forge python=3.7 scipy pandas psutil scikit-image␣
↪gdal pyshp opencv
```

Activate the `sen1mosaic` environment whenever opening a new terminal window by running this command:

```
conda activate sen1mosaic
```

4.1.4 Installing SNAP

SNAP is an ESA toolset used for (amongst other things) pre-processing data from Sentinel-1. SNAP for Linux can be downloaded from http://step.esa.int/main/download/.

To install SNAP, open a terminal window, change directory to the location you'd like to download SNAP, and run the following commands:

```
wget http://step.esa.int/downloads/6.0/installers/esa-snap_sentinel_unix_6_0.sh
bash esa-snap_sentinel_unix_6_0.sh
```

. . . and follow the instructions. The default installation instructions should work fine with sen1mosaic.

Sen1mosaic uses the 'graph processing tool' to operate SNAP commands from the command line. To give access system-wide to the graph processing tool, you'll need to add an alias to your `.bashrc` file as follows:

```
echo 'alias gpt=~/snap/bin/gpt' >> ~/.bashrc
```

It's a good idea to increase the memory allocation to SNAP. This is controlled by the text file `~/snap/bin/gpt.vmoptions`. This can be done with following line:

```
echo '-Xmx8G' >> ~/snap/bin/gpt.vmoptions
```

Some SNAP operations are currently having trouble with the latest Sentinel-1 data (after March 2018). This can be fixed by installing updated through the SNAP GUI (`Help >> Check for Updates`), or with the following line in the terminal:

```
snap --nosplash --nogui --modules --update-all
```

For further details and up-to-date installation instructions, see the SNAP website.

4.1.5 Installing sentinelsat

Sentinelsat is the toolset used to access data from the Sentinel-1 archive at the Copernicus Open Access Data Hub.

Up-to-date installation instructions can be found here.

At the time of writing, the installation process is as follows:

```
pip install sentinelsat
```

4.1.6 Installing sen1mosaic

sen1mosaic can be downloaded to a machine from its repository . To do this, open a terminal window and input:

```
git clone https://github.org/smfm-project/sen1mosaic.git
```

To install sen1mosaic, navigate to the sen1mosaic directory and run the following within your sen1mosaic environment

```
python setup.py install
```

To avoid having to reference the full path of the Python scripts in sen1mosaic, it's a good idea add the following line to your `.bashrc` file:

```
echo "alias s1m='_s1m() { python ~/sen1mosaic/cli/\"\$1\".py \$(shift; echo \"\$@\") ;
↪}; _s1m'" >> ~/.bashrc
```

4.1.7 Installing sen2mosaic

sen1mosaic makes use of some of the functons of sen2mosaic. To install sen2mosaic:

```
git clone https://github.com/smfm-project/sen2mosaic.git
```

To install sen2mosaic, navigate to the sen2mosaic directory and run the following within your sen1mosaic environment

```
python setup.py install
```

4.1.8 Is there a Dockerfile?

Coming soon!

4.1.9 Where do I get help?

For help installing SNAP, it's best to refer to the ESA STEP forum. For assistance in setting up and using sen1mosaic, email sam.bowers@ed.ac.uk.

4.2 Command line tools

The most straightforward way of using sen1mosaic it to call its various stages from the Linux command line. Here the functionality of each of the three commands is explained. In the next section we show how it can be used by example.

Note: Remember, each command line script has a help flag, which can point you in the right direction when in doubt.

4.2.1 Downloading Sentinel-1 data

Data from Sentinel-1 are available from the Copernicus Open Access Data Hub, which has a graphical interface to download scenes from selected areas. Whilst useful for smaller areas, generating mosaics at national scales requires a volume of data which makes this extremely labour intensive.

The alternative is to download data using the API Hub. This system allows users to search for files using conditions on the command line, and automatically download files. To interface with the API hub, we use an open source utility called Sentinelsat. This operates both as a command line tool, and as a Python API, which we use here. You will need to sign up for an account at Scihub.

`download.py` is a program to interface with Sentinelsat to download Sentinel-1 files, specifying a particular latitude/longitude ranges, dates and orbital directions.

Help for `download.py` can be viewed by typing `s1m download --help`:

```
usage: download.py [-h] -u USER -p PASS -a LONMIN LATMIN LONMAX LATMAX -s
                   YYYYMMDD -e YYYYMMDD [-o PATH] [-d DIR]

Download Sentinel-1 IW GRD data from the Copernicus Open Access Hub,
specifying a latitude/longitude range, date ranges and ascending/descending
orbits. Files that are already present in the destination directory won't be
re-downloaded.

Required arguments:
-u USER, --user USER  Scihub username
-p PASS, --password PASS
                      Scihub password
-a LONMIN LATMIN LONMAX LATMAX, --search_area LONMIN LATMIN LONMAX LATMAX
                      Extent of search area, in format <lonmin latmin lonmax
                      latmax>.
-s YYYYMMDD, --start YYYYMMDD
                      Start date for search in format YYYYMMDD.
-e YYYYMMDD, --end YYYYMMDD
                      End date for search in format YYYYMMDD.

Optional arguments:
-o PATH, --output_dir PATH
                      Specify an output directory. Defaults to the current
                      working directory.
-d DIR, --direction DIR
                      Specify <ASCENDING> or <DESCENDING> if only a single
                      orbital direction should be downloaded. Defaults to
                      downloading both.
```

For example, to download all data for the August-September 2017 for the longitudes of 34 to 35 degrees and latitudes of -19 to -18 degrees (around Gorongosa National Park in Mozambique), specifying an output location, use the following command:

```
s1m download -u user.name -p supersecret -a 34 -19 35 -18 -s 20170801 -e 20170930 -o /
↪path/to/S1_data/
```

Note: sen1mosaic is only compatible with Sentinel-1 data in Ground Range Detected (GRD) Interferometic Wide swath (IW) mode. If you already have access to Sentinel-1 GRD IW data, you can skip straight to the next section. This may be the case if you're using a cloud platform where Sentinel-1 data archives are stored at the same location as servers.

4.2.2 Pre-processing Sentinel-1 data

Once you have Sentinel-1 (GRD IW) data, the next step is to calibrate, filter, and perform geometric correction on the data.

`preprocess.py` takes a list of Sentinel-1 .zip files as input, and inititates a series of SNAP processing chains.

Help for `preprocess.py` can be viewed by typing `s1m preprocess --help`:

```
usage: preprocess.py [-h] [-o PATH] [-n STR] [-t PATH] [-ms N] [-m N] [-f]
                     [-u UNITS] [-s] [-no] [-g PATH] [-st START] [-en END]
                     [-v] [-p N]
                     [S1_FILES [S1_FILES ...]]

Pre-process Sentinel-1 IW GRD data from the Copernicus Open Access Hub to
radiometric/terrain corrected images.

Positional arguments:
S1_FILES              Input files. Specify a valid S1 input file (.zip),
                        multiple files through wildcards, or a directory.
                        Defaults to processing all S1 files in current working
                        directory.

Optional arguments:
-o PATH, --output_dir PATH
                        Output directory for processed files. Defaults to
                        current working directory.
-n STR, --output_name STR
                        String to be included in output filenames for
                        identification. Defaults to 'processed'.
-t PATH, --temp_dir PATH
                        Output directory for intermediate files. Defaults to
                        current working directory.
-ms N, --max_scenes N
                        Maximum number of scenes from an overpass to
                        reconstitute and process together. Higher values
                        result in fewer output files with fewer artefacts at
                        scene boundaries, but require more RAM. Defaults to 3
                        scenes.
-m N, --multilook N   Multilooking reduces image noise by degrading output
                        resolution from ~10 x 10 m by a factor. Defaults to 2
                        (~20 x 20 m output).
-f, --speckle_filter  Apply a speckle filter (Refined Lee) to output images.
-u UNITS, --output_units UNITS
                        Output units, set to either decibels (default) or
                        natural.
-s, --short           Perform a more rapid processing chain, ommitting some
                        nonessential preprocessing steps.
-no, --noorbit        Skip downloading of a precise orbit file.
-g PATH, --gpt PATH   Path to graph processing tool. Defaults to
                        ~/snap/bin/gpt.
-st START, --start START
                        Start date for tiles to include in format YYYYMMDD.
                        Defaults to processing all dates.
-en END, --end END    End date for tiles to include in format YYYYMMDD.
                        Defaults to processing all dates.
-v, --verbose         Print script progress.
-p N, --processes N   Specify a maximum number of tiles to process in
```

(continues on next page)

(continued from previous page)

```
                        parallel. Note: more processes will require more
                        resources. Defaults to 1.
```

For example, to run `preprocess.py` on a set of Sentinel-1 GRD IW .zip files in a directory (specifying an output and a temporary files directory), use the following command:

```
s1m preprocess -o /path/to/S1_data/ -t /scratch/ /path/to/S1_data/
```

4.2.3 Processing to GeoTiff tiles

The final step is to process Sentinel-1 data into a user-specified tiling grid. This script takes Sentinel-1 .dim files or a directory containing .dim files as input, selects the tiles that fall within the specified spatial extent, and mosaics available data into single-band GeoTiff files for easy use in classification systems.

`mosaic.py` takes input .dim files and generates an output image with a specifed extent (xmin, ymin, xmax, ymax) and projection EPSG code as input. The script outputs a mean, minimum, maximum, and standard deviation of Sentinel-1 backscatter for each available polarisation.

Help for `mosaic.py` can be viewed by typing `s1m mosaic --help`:

```
usage: mosaic.py [-h] [-te XMIN YMIN XMAX YMAX] [-e EPSG] [-res RES]
                 [-st START] [-en END] [-o PATH] [-n NAME] [-p POL] [-v]
                 [S1_FILES [S1_FILES ...]]

Collate preprocessed Sentinel-1 data into mosaicked tiles. This script mosaics
Sentinel-1 data into a customisable grid square, based on specified UTM
coordinate bounds. Files are output as GeoTiffs of mean, min, max, and
standard deviation of each available backscatter.

required arguments:
-te XMIN YMIN XMAX YMAX, --target_extent XMIN YMIN XMAX YMAX
                        Extent of output image tile, in format <xmin, ymin,
                        xmax, ymax>.
-e EPSG, --epsg EPSG  EPSG code for output image tile CRS. This must be UTM.
                        Find the EPSG code of your output CRS as
                        https://www.epsg-registry.org/.
-res RES, --resolution RES
                        Output resolution in metres. If not specified,
                        defaults to 20m.

optional arguments:
S1_FILES              Input files from preprocess.py. Specify a valid S1
                        input file (.dim), multiple files through wildcards,
                        or a directory. Defaults to processing all S1 files in
                        current working directory.
-st START, --start START
                        Start date for tiles to include in format YYYYMMDD.
                        Defaults to processing all dates.
-en END, --end END    End date for tiles to include in format YYYYMMDD.
                        Defaults to processing all dates.
-o PATH, --output_dir PATH
                        Output directory. If nothing specified, downloads will
                        output to the present working directory, given a
                        standard filename.
-n NAME, --output_name NAME
```

(continues on next page)

(continued from previous page)

```
                        Optionally specify a string to precede output
                        filename.
-p POL, --pol POL     Specify a single polarisation ('VV' or 'VH') or
                        'both'. Defaults to processing both.
-v, --verbose         Print script progress.
```

For example, to run `mosaic.py` in the directory `/path/to/S1_data/` which contains pre-processed Sentinel-1 files to create a 200 x 200 km output tile in the UTM36S projection at 20 m resolution, input:

```
s1m mosaic -te 600000 7900000 800000 8100000 -e 32736 -r 20 /path/to/S1_data
```

To do the same operation, but specifying an output directory and a name to prepend to outputs from this tile, input:

```
s1m mosaic -te 600000 7900000 800000 8100000 -e 32736 -r 20 -o /path/to/output/ -n␣
↪tile_1 /path/to/S1_data/
```

4.3 Worked example on the command line

Here we'll show you by example how the sen1mosaic processing chain works in practice. We will focus on an example from Zambezia Province of Mozambique, with the aim of creating a composite GeoTiff mosaic product for the province. The extent of Zambezia Province extends roughly from **35 to 40** degrees longitude and **-19 to -15** degrees latitude. We will generate a mosaic for the dry season (**August**) of **2016**, in anticipation of multiple seasonally-specific mosaics improving classification accuracy.

4.3.1 Preparation

First ensure that you've followed setup successfully.

Open a terminal, and use `cd` to navigate to the location you'd like to store data.

```
cd /home/user/DATA
mkdir worked_example
cd worked_example
```

Use mkdir to make a directory for the region to be downloaded, and navigate to that directory.

```
mkdir zambezia_data
cd zambezia_data
```

4.3.2 Downloading data

The first step is to download Sentinel-1 GRD IW data from the Copernicus Open Access Data Hub.

For this we use the `download.py` tool. We will need to specify a Scihub username and password (sign up for an account at Scihub), the extent to download (in degrees lat/lon), and a start and end date in the format YYYYMMDD.

We will limit the download to only the ascending passes, which limits data volume and results in a cleaner mosaic where the geometric distortions from terrain are consistent. At our Mozambique study site Sentinel-1 returns data from both the ascendinfg and descending pass, but note there are regions where data are only available from a single overpass (see: https://sentinel.esa.int/web/sentinel/missions/sentinel-1/observation-scenario)

These options can be encoded as follows:

```
s1m download -u user.name -p supersecret -a 35 -19 40 -15 -s 20170501 -e 20170630 -d␣
↪ASCENDING
```

As we didn't specify the option `-o` (`--output`), data will output to the current working directory. The data format will be `.zip`, which can be read directly by sen1mosiac without extraction.

Wait for all files to finish downloading before proceeding to the next step. By the time the processing is complete, your `zambezia_data/` directory should contain a list of Sentinel1 .zip files (show files in the currenty working directory with the command `ls`).

```
S1B_IW_GRDH_1SDV_20170803T160647_20170803T160716_006777_00BED2_4455.zip
S1B_IW_GRDH_1SDV_20170803T160716_20170803T160741_006777_00BED2_9B1E.zip
S1B_IW_GRDH_1SDV_20170803T160741_20170803T160806_006777_00BED2_6BCE.zip
...
S1B_IW_GRDH_1SDV_20170827T160807_20170827T160832_007127_00C900_F95D.zip
S1B_IW_GRDH_1SDV_20170829T155108_20170829T155137_007156_00C9CB_7F9D.zip
S1B_IW_GRDH_1SDV_20170829T155137_20170829T155202_007156_00C9CB_49E9.zip
```

4.3.3 Preprocessing Sentinel-1 GRD data

The next step is to perform the preprocessing steps that convert raw Sentinel-1 data to usable terrain corrected images. We do this with the graph processing tool (`gpt`) bundled with SNAP.

To perform atmospheric correction and cloud masking we call the tool `preprocess.py`. We need to specify Sentinel-1 input files, a directory containing Sentinel-1 .zip files, or a single Sentinel-1 .zip file. We will use default options, except for adding in a speckle filter to reduce image noise.

To process all Sentinel-1 input files, we can submit the following line:

```
s1m preprocess -f -v
```

This command will loop through each Sentinel-1 input file, stitch together images from the same satellite overpass and process them sequentially. You might alternatively want to specify multiple processes to run similaneously (with the `-p` flag), although bear in mind that this will require access to a large quantity of memory.

Here we didn't specify the options `-o` (`--output_dir`) which can be used to output data to a location other than the directory containing input files, or the `-r` (`--remove`) option, which would delete Sentinel-1 .zip files once data is finished processing.

Wait for all files to be processed to level 2A before proceeding. If you run `ls` again, your `zambezia_data/` directory should now contain a new set of files:

```
S1_L2_20170803_160647_160831_006777_00BED2.data
S1_L2_20170803_160647_160831_006777_00BED2.dim
S1_L2_20170805_155107_155201_006806_00BF9E.data
S1_L2_20170805_155107_155201_006806_00BF9E.dim
...
S1_L2_20170827_160648_160832_007127_00C900.data
S1_L2_20170827_160648_160832_007127_00C900.dim
S1_L2_20170829_155108_155202_007156_00C9CB.data
S1_L2_20170829_155108_155202_007156_00C9CB.dim
```

4.3.4 Generating a mosaic for classification

After you have preprocessed all the Sentinel-1 .zip files, the final step is to mosaic these into a larger tiling system in preparation for image classification. The tool `mosaic.py` will generate summary statistics (mean, min, max, and

standard deviation) for each input polarisation in the widely-used GeoTiff format.

Here we will generate a single output tile coverng the entirity of Zambezia province at 50 m resolution with the limits **710,500 to 1,250,000** m Eastings and **7,890,000 - 8,330,000** m Northings (**UTM 36S**). We'll generate mosaic for both VV and VH polarisations, and output data a name ('worked example') to identfy this tile.

To perform this step, we can run the following script:

```
s1m mosaic -te 710500 7890000 1250000 8340000 -e 32736 -r 50 -v -n worked_example
```

Here we didn't specify an input directory (the script defaults to processing all compatable files in the current working directory) or the `-o` (`--output_dir`) option, meaning that results will be output to the current working directory. Once processing is complte, you can use `ls` to view the newly created output files:

```
worked_example_max_VH_R50m.tif
worked_example_max_VV_R50m.tif
worked_example_mean_VH_R50m.tif
worked_example_mean_VV_R50m.tif
worked_example_mean_VVVH_R50m.tif
worked_example_min_VH_R50m.tif
worked_example_min_VV_R50m.tif
worked_example_stdev_VH_R50m.tif
worked_example_stdev_VV_R50m.tif
worked_example_VVmean_VHmean_VVVH.vrt
worked_example_VVmin_VHmin_VVstdev.vrt
```

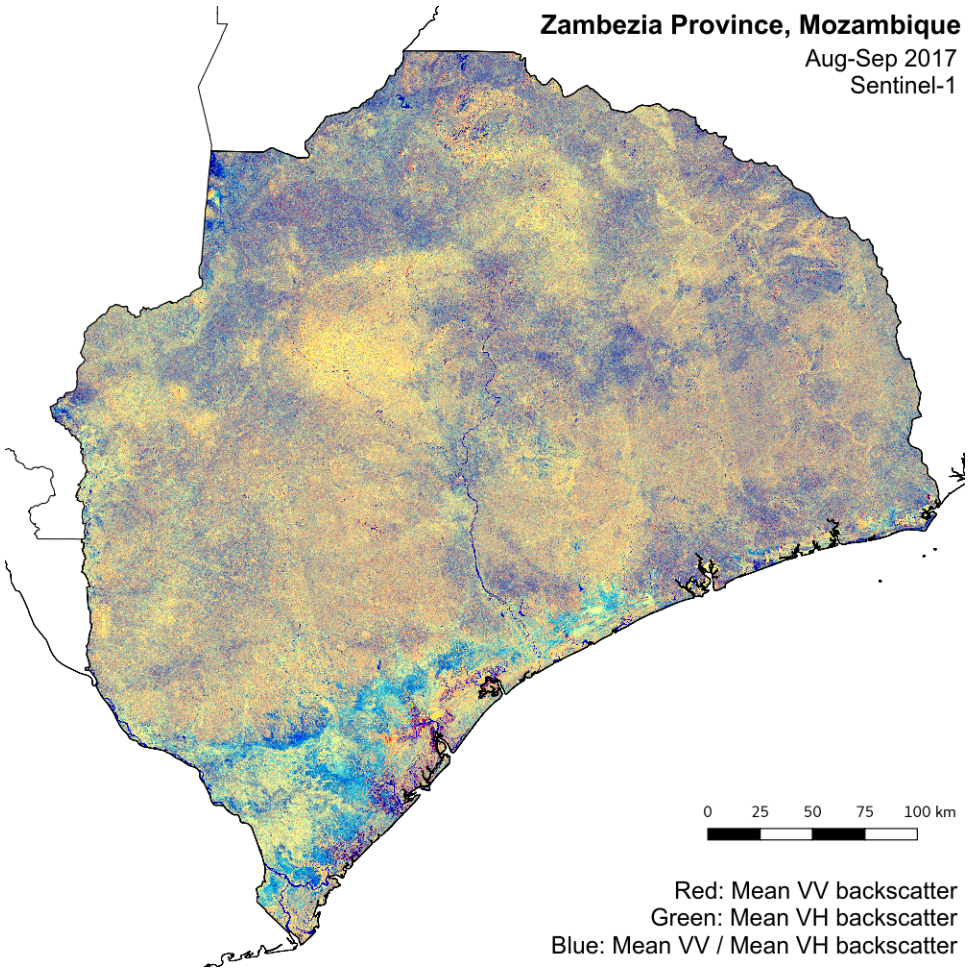
4.3.5 Viewing data

In addition to a GeoTiff files with summary statistics for each Sentinel-1 polarisation (VV, VH), `mosaic.py` outputs two 3-band GDAL virtual dataset files (`.vrt`). These are labelled `_VVmean_VHmean_VVVH_R*.vrt` and `_VVmin_VHmin_VVstdev_R*.vrt`, and can be opened in QGIS to show two different false colour composites of Sentinel-1 data. The first shows:

- Red: mean VV backscatter
- Green: mean VH backscatter
- Blue: mean VV dividided by mean VH backscatter

and the second shows:

- Red: minimum VV backscatter
- Green: minimum VH backscatter
- Blue: standard deviation of VV backscatter



4.3.6 See also

This example required a lot of manual typing. We can achieve further automation through Python. To see an example of how to run achieve the same results in Python, see the scripts in the sectiomn worked_example_python.

4.4 Using sen1mosaic in Python

This is harder than the command line, but you may be interested in importing the sen1mosaic functions into Python in order to customise the processing chain.

To make sen1mosaic accesible in Python, edit your `.bashrc` file (located at `~/.bashrc`) to contain the line:

```
export PYTHONPATH=$PYTHONPATH:/path/to/sen1mosaic/
```

You should now be able to import each of the four modules in Python as follows:

```
import sen1mosaic.download
import sen1mosaic.preprocess
import sen1mosaic.mosaic
```

Help for each function can be accessed interactively from Python. For example:

```
>>> help(sen1mosaic.download.connectToAPI)
        Help on function connectToAPI in module sen1mosaic.download:

        connectToAPI(username, password)
        Connect to the SciHub API with sentinelsat.

        Args:
            username: Scihub username. Sign up at https://scihub.copernicus.eu/.
            password: Scihub password.
```

On this page each of the functions from the download, preprocess, and mosaic modules are documented. Note that the `main()` function in each is what is driven by the command line tools, so in addition to it's component parts you can call the entire processing chain from Python.

4.4.1 Download module

sen1mosaic.download.**connectToAPI**(*username*, *password*)
Connect to the SciHub API with sentinelsat. Sign up at https://scihub.copernicus.eu/.

Parameters

- **username** – Scihub username.
- **password** – Scihub password.

sen1mosaic.download.**download**(*products_df*, *output_dir = os.getcwd()*)
Downloads all images from a dataframe produced by sentinelsat.

Parameters

- **products_df** – Pandas dataframe from search() function.
- **output_dir** – Optionally specify an output directory. Defaults to the present working directory.

sen1mosaic.download.**removeDuplicates**(*products_df*, *data_dir = os.getcwd()*)
Remove images from search results that have already been downloaded

Parameters

- **products_df** – Pandas dataframe from search() function.
- **data_dir** – Directory containing Sentinel-1 data. Defaults to current working directory.

Returns A dataframe with duplicate files removed.

sen1mosaic.download.**search**(*search_area*, *start = '20140403'*, *end = datetime.datetime.today().strftime('%Y%m%d')*, *direction= '*'*)
Searches for Sentinel-1 GRD IW images that meet conditions of date range and extent.

Parameters

- **search_area** – A list in the format [minlon, minlat, maxlon, maxlat]
- **start** – Start date for search in format YYYYMMDD. Start date may not precede 20140403, the launch date of Sentinel1-. Defaults to 20140403.
- **end** – End date for search in format YYYYMMDD. Defaults to today's date.

Returns A pandas dataframe with details of scenes matching conditions.

4.4.2 Preprocessing module

4.4.3 Mosaicking module

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index