# 1 Predicting Snow Water Equivalent in regions in Western United States

**Estimating snow water equivalent (SWE) at a high spatiotemporal resolution over the Western U.S. using near real-time data sources**

**Author**: Hanis Zulmuthi

May 2022

Open in Colab
(https://colab.research.google.com/github/hanis-z/Snow-water-equivalent/blob/main/src/MODIS-DEM-Preprocessing_colab(01.1).ipynb)

Source: Reddit.com (https://www.reddit.com/r/EarthPorn/comments/a6ewla/snow_and_flowing_water_is_one_of_the_most_magical/?utm_source=ifttt)

## 1.1 Overview

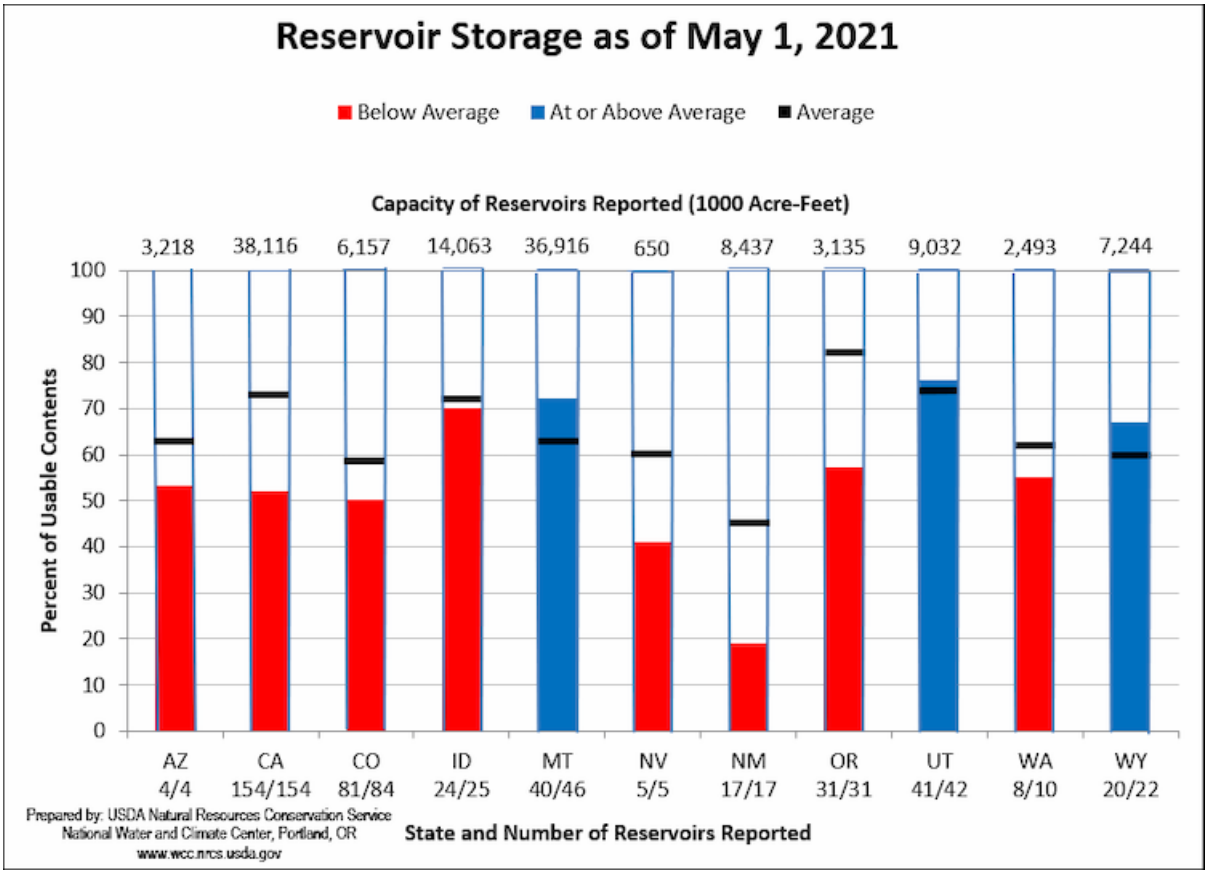
This project budded from a competition titled Snowcast Showdown (https://www.drivendata.org/competitions/90/competition-reclamation-snow-water-eval/page/431/) on Driven Data (https://www.drivendata.org/). The goal of the project is to develop a predictive model to estimate the distribution of Snow Water Equivalent (SWE) at a high spatiotemporal resolution over the Western U.S. This predictive model will assist NOAA in their National Integrated Drought Information System (NIDIS) (https://www.drought.gov/), an initiative to monitor snow drought in the wester United States.

## 1.2 Introduction

Snow Water Equivalent (SWE) is a common snowpack measurement used by hydrologists and water managers to gage amount of liquid water contained within snowpack. It is equal to the amount of water contained within the snowpack when it melts. It can be thought of as the depth of water that would theoretically result if you melted the entire snowpack instantaneously [1] (https://www.nrcs.usda.gov/wps/portal/nrcs/detail/nv/snow/?cid=nrcseprd1746821#:~:text=Snow%20Water%20Equivalent%20(SWE)%20is,the%20snowpack%20when%20it%20melts.).

Water in a snow pack is determined by depth, density, type of snow, changes in the pack, previous freeze/thaw cycles, recent rainfall events, etc. Available water is the amount of water that would be released if the snow pack melted. SWE is an important measure of availability of water resources, since it relates to the runoff of rivers and variations in groundwater levels, so knowing how much water is available in the snow pack is valuable for those managing reservoirs and flood forecasting [2] (https://www.campbellsci.ca/snow-water-equivalent-measurement)[3] (http://www.eumetrain.org/data/3/358/navmenu.php?tab=7&page=1.0.0#:~:text=Climatology%20of%20snow%20cover%20and%20snow%20water%20equivalent,-Table%20of%20Contents&text=SWE%20is%20an%20important%20measure,the%20age%20of%20snow%20cover.).

Reports (https://www.drought.gov/drought-status-updates/water-year-2021-snow-drought-conditions-summary-and-impacts-west) by NOAA (through their National Integrated Drought Information System (NIDIS) (https://www.drought.gov/) program) on the intensifying snow drought over western U.S raises the alarm on the importance of predicting SWE as accurately possible, especially for remote, high elevation areas where manual ground measure measurements are not feasible. It was reported that the loww snowpack, rapid and early snow melts and poor runoffs had resulted in a significant drop in water supply in the summer of 2021 (fig 1).




Source: NIDIS, Drought.gov (https://www.drought.gov/drought-status-updates/water-year-2021-snow-drought-conditions-summary-and-impacts-west)

## 1.3 Data Understanding

**Historical Ground Measures data:** Ground measures help provide regularly collected, highly accurate point estimates of SWE at designated stations. Ground measures data range from 2013-2019 and 2020-2021 was provided in ground_measures_train_features.csv (./data/ground_measures_train_features.csv) and ground_measures_test_features.csv (./data/ground_measures_test_features.csv). The ground measures data are from Snow Telemetry (SNOTEL) (https://www.nrcs.usda.gov/wps/portal/wcc/home/) and California Data Exchange Center (CDEC) (https://cdec.water.ca.gov/). The dataset used from these sources is available in this repo here (./data/).

***SNOTEL (https://www.nrcs.usda.gov/wps/portal/wcc/home/):*** The Snow Telemetry (SNOTEL) program consists of automated and semi-automated data collection sites across the Western U.S.

***CDEC (https://cdec.water.ca.gov/):*** The California Data Exchange Center (CDEC) facilitates the collection, storage, and exchange of hydrologic and climate information to support real-time flood management and water supply needs in California. CDEC operates data collection sites similar to SNOTEL within California.

Ground-based sites from SNOTEL and CDEC are used both as an input data source and in ground truth labels for our predictive model. ***Note that, sites that we are predicting SWE for, are entirely distinct from those in the features data.***

**MODIS Satellite Imagery (https://microsoft.github.io/AIforEarthDataSets/data/modis.html):** The MODIS satellite images consist of MODIS/Terra and MODIS/Aqua Snow Cover Daily L3 Global 500m SIN Grid. Terra's orbit around the Earth is timed so that it passes from north to south across the equator in the morning, while Aqua passes south to north over the equator in the afternoon. Snow-covered land typically has very high reflectance in visible bands and very low reflectance in shortwave infrared bands. The Normalized Difference Snow Index (NDSI) reveals the magnitude of this difference. The snow cover algorithm calculates NDSI for all land and inland water pixels in daylight using MODIS band 4 (visible green) and band 6 (shortwave near-infrared).

The satellite imageries from MODIS were not used for modelling due to contraints in computing power and memory. We did however, pull down the satellite images from their Azure blob () and saved it as numpy arrays of pixels. This process was done in this notebook (./src/MODIS-DEM-Preprocessing_colab.ipynb) that was executed in Google Colab (https://colab.research.google.com/?utm_source=scs-index).

## 1.4 Import packages

In [1]:
```
# #Run this cell to check if you have conda
!conda --version
```

```
conda 4.12.0
```

In [ ]:
```
# #Run this cell and the following 2 cells to check packages
# import sys
# sys.path
```

In [ ]:
```
# !ls /usr/local/lib/python3.7/dist-packages
```

In [2]:
```
!ls /usr/local/lib/python3.7/site-packages
```

```
affine
affine-2.3.1.dist-info
aiohttp
aiohttp-3.8.1.dist-info
aiosignal
aiosignal-1.2.0.dist-info
asynctest
asynctest-0.13.0.dist-info
async_timeout
async_timeout-4.0.2.dist-info
attr
attrs
attrs-21.4.0.dist-info
azure
azure_core-1.24.0.dist-info
azure_storage_blob-12.12.0.dist-info
brotli
brotlipy-0.7.0-py3.7.egg-info
cachetools
cachetools-5.1.0.dist-info
certifi
certifi-2022.5.18-py3.7.egg-info
cffi
cffi-1.14.5.dist-info
_cffi_backend.cpython-37m-x86_64-linux-gnu.so
CHANGELOG.md
chardet
chardet-4.0.0.dist-info
charset_normalizer
charset_normalizer-2.0.12.dist-info
click
click-7.1.2.dist-info
click_plugins
click_plugins-1.1.1.dist-info
cligj
cligj-0.7.2.dist-info
conda
conda-4.12.0-py3.7.egg-info
conda_env
conda_package_handling
conda_package_handling-1.7.2.dist-info
cryptography
cryptography-3.4.5.dist-info
dateutil
decorator-5.1.1.dist-info
decorator.py
_distutils_hack
distutils-precedence.pth
dotenv
easy_install.py
frozenlist
frozenlist-1.3.0.dist-info
fsspec
fsspec-2022.3.0.dist-info
gcsfs
gcsfs-2022.3.0.dist-info
geojson
geojson-2.5.0.dist-info
google
google_api_core-2.8.0.dist-info
google_api_core-2.8.0-py3.10-nspkg.pth
googleapis_common_protos-1.56.1.dist-info
googleapis_common_protos-1.56.1-py3.10-nspkg.pth
google_auth-2.6.6.dist-info
google_auth-2.6.6-py3.10-nspkg.pth
google_auth_oauthlib
google_auth_oauthlib-0.5.1.dist-info
google_cloud_core-2.3.0.dist-info
google_cloud_core-2.3.0-py3.10-nspkg.pth
```

```
google_cloud_storage-2.3.0.dist-info
google_cloud_storage-2.3.0-py3.10-nspkg.pth
google_crc32c
google_crc32c-1.3.0.dist-info
google_crc32c.libs
google_resumable_media-2.3.3.dist-info
google_resumable_media-2.3.3-py3.10-nspkg.pth
idna
idna-2.10.dist-info
importlib_metadata
importlib_metadata-4.11.3.dist-info
isodate
isodate-0.6.1.dist-info
LICENSE
mamba
mamba-0.8.0.dist-info
msrest
msrest-0.6.21.dist-info
multidict
multidict-6.0.2.dist-info
numpy
numpy-1.21.1.dist-info
oauthlib
oauthlib-3.2.0.dist-info
OpenSSL
packaging
packaging-21.3.dist-info
pandas
pandas-1.3.1-py3.7.egg-info
pip
pip-21.0.1-py3.9.egg-info
pkg_resources
planetary_computer
planetary_computer-0.4.6.dist-info
protobuf-3.20.1.dist-info
protobuf-3.20.1-py3.7-nspkg.pth
pyasn1
pyasn1-0.4.8.dist-info
pyasn1_modules
pyasn1_modules-0.2.8.dist-info
__pycache__
pycosat-0.6.3.dist-info
pycosat.cpython-37m-x86_64-linux-gnu.so
pycparser
pycparser-2.20.dist-info
pydantic
pydantic-1.9.1.dist-info
pyOpenSSL-20.0.1.dist-info
pyparsing
pyparsing-3.0.9.dist-info
pyproj
pyproj-3.1.0.dist-info
PySocks-1.7.1.dist-info
pystac
pystac-1.4.0.dist-info
pystac_client
pystac_client-0.3.3.dist-info
python_dateutil-2.8.2.dist-info
python_dotenv-0.20.0.dist-info
pytz
pytz-2022.1.dist-info
rasterio
rasterio-1.2.4.dist-info
README.md
README.txt
requests
requests-2.25.1.dist-info
requests_oauthlib
requests_oauthlib-1.3.1.dist-info
rioxarray
rioxarray-0.9.1.dist-info
rsa
```

```
rsa-4.8.dist-info
ruamel_yaml
ruamel_yaml_conda-0.15.80.dist-info
scipy
scipy-1.7.0.dist-info
setuptools
setuptools-49.6.0.post20210108-py3.7.egg-info
six-1.15.0.dist-info
six.py
snuggs
snuggs-1.4.7.dist-info
sockshandler.py
socks.py
test_pycosat.py
tests
tqdm
tqdm-4.59.0.dist-info
typing_extensions-4.2.0.dist-info
typing_extensions.py
urllib3
urllib3-1.26.3.dist-info
wget-3.2.dist-info
wget.py
wheel
wheel-0.36.2-py3.6.egg-info
xarray
xarray-0.20.2.dist-info
xontrib
yarl
yarl-1.7.2.dist-info
zipp-3.8.0.dist-info
zipp.py
```

In [5]:

```
#Run this cell for conda colab
!pip install -q condacolab
import condacolab
condacolab.install()
!conda --version
```

```
⏬ Downloading https://github.com/jaimergp/miniforge/releases/latest/download/Mambaforge-colab-Linux-x86_64.sh... (https://github.com/jaimergp/miniforge/releases/latest/download/Mambaforge-colab-Linux-x86_64.sh...)
📦 Installing...
📌 Adjusting configuration...
🩹 Patching environment...
⏲ Done in 0:00:19
🔁 Restarting kernel...
conda 4.9.2
```

In [4]:

```
#Run this cell to install necessary packages
!conda install --channel conda-forge rioxarray --yes
!pip install azure.storage.blob
!pip install azure-core
!pip install wget
!pip install geojson
!pip install gcsfs
!pip install pystac_client
!pip install planetary_computer
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata
source.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /usr/local

  added / updated specs:
    - rioxarray


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    affine-2.3.1               |     pyhd8ed1ab_0          17 KB  conda-forge
    attrs-21.4.0               |     pyhd8ed1ab_0          49 KB  conda-forge
    boost-cpp-1.74.0           |       hc6e9bd1_3        16.3 MB  conda-forge
    ca-certificates-2022.5.18  |       ha878542_0         144 KB  conda-forge
    cairo-1.16.0               |    h6cf1ce9_1008         1.5 MB  conda-forge
    certifi-2022.5.18          |   py37h89c1867_0         150 KB  conda-forge
    cfitsio-3.470              |       hb418390_7         1.3 MB  conda-forge
    click-7.1.2                |     pyh9f0ad1d_0          64 KB  conda-forge
    click-plugins-1.1.1        |             py_0           9 KB  conda-forge
    cligj-0.7.2                |     pyhd8ed1ab_1          10 KB  conda-forge
    conda-4.12.0               |   py37h89c1867_0         1.0 MB  conda-forge
    cudatoolkit-11.1.1         |       h6406543_8        1.20 GB  conda-forge
    curl-7.75.0                |       h979ede3_0         147 KB  conda-forge
    expat-2.4.1                |       h9c3ff4c_0         182 KB  conda-forge
    fontconfig-2.13.1          |    hba837de_1005         357 KB  conda-forge
    freetype-2.10.4            |       h0708190_1         890 KB  conda-forge
    freexl-1.0.6               |       h7f98852_0          48 KB  conda-forge
    geos-3.9.1                 |       h9c3ff4c_2         1.1 MB  conda-forge
    geotiff-1.6.0              |       hcf90da6_5         296 KB  conda-forge
    gettext-0.19.8.1           |    h0b5b191_1005         3.6 MB  conda-forge
    giflib-5.2.1               |       h36c2ea0_2          77 KB  conda-forge
    hdf4-4.2.15                |       h10796ff_3         950 KB  conda-forge
    hdf5-1.10.6                |nompi_h6a2412b_1114         3.1 MB  conda-forge
    importlib-metadata-4.11.3  |   py37h89c1867_1          33 KB  conda-forge
    importlib_metadata-4.11.3  |       hd8ed1ab_1           4 KB  conda-forge
    jpeg-9d                    |       h36c2ea0_0         264 KB  conda-forge
    json-c-0.15                |       h98cffda_0         274 KB  conda-forge
    kealib-1.4.14              |       hcc255d8_2         186 KB  conda-forge
    libblas-3.9.0              |11_linux64_openblas          12 KB  conda-forge
    libcblas-3.9.0             |11_linux64_openblas          11 KB  conda-forge
    libdap4-3.20.6             |       hd7c4107_2        11.3 MB  conda-forge
    libgdal-3.2.2              |       h804b7da_0        13.2 MB  conda-forge
    libgfortran-ng-12.1.0      |      h69a702a_16          23 KB  conda-forge
    libgfortran5-12.1.0        |      hdcd56e2_16         1.8 MB  conda-forge
    libglib-2.68.3             |       h3e27bee_0         3.1 MB  conda-forge
    libkml-1.3.0               |    h238a007_1014         591 KB  conda-forge
    liblapack-3.9.0            |11_linux64_openblas          11 KB  conda-forge
    libnetcdf-4.7.4            |nompi_h56d31a8_107         1.3 MB  conda-forge
    libopenblas-0.3.17         |pthreads_h8fe5266_1         9.2 MB  conda-forge
    libpng-1.6.37              |       h21135ba_2         306 KB  conda-forge
    libpq-13.2                 |       hfd2b0eb_2         2.7 MB  conda-forge
    librttopo-1.1.0            |       h1185371_6         235 KB  conda-forge
```

```
  libspatialite-5.0.1        |       h20cb978_4         4.4 MB  conda-forge
  libtiff-4.2.0              |       hbd63e13_2         639 KB  conda-forge
  libuuid-2.32.1             |    h7f98852_1000          28 KB  conda-forge
  libwebp-base-1.2.0         |       h7f98852_2         815 KB  conda-forge
  libxcb-1.13                |    h7f98852_1003         395 KB  conda-forge
  numpy-1.21.1               |   py37h038b26d_0         6.1 MB  conda-forge
  openjpeg-2.4.0             |       hb52868f_1         444 KB  conda-forge
  openssl-1.1.1k             |       h7f98852_0         2.1 MB  conda-forge
  packaging-21.3             |     pyhd8ed1ab_0          36 KB  conda-forge
  pandas-1.3.1               |   py37h219a48f_0        12.7 MB  conda-forge
  pcre-8.45                  |       h9c3ff4c_0         253 KB  conda-forge
  pixman-0.40.0              |       h36c2ea0_0         627 KB  conda-forge
  poppler-21.03.0            |       h93df280_0        15.9 MB  conda-forge
  poppler-data-0.4.11        |       hd8ed1ab_0         3.6 MB  conda-forge
  postgresql-13.2            |       h6303168_2         5.3 MB  conda-forge
  proj-8.0.0                 |       h277dcde_0         3.1 MB  conda-forge
  pthread-stubs-0.4          |    h36c2ea0_1001           5 KB  conda-forge
  pyparsing-3.0.9            |     pyhd8ed1ab_0          79 KB  conda-forge
  pyproj-3.1.0               |   py37h8627986_0         513 KB  conda-forge
  python-dateutil-2.8.2      |     pyhd8ed1ab_0         240 KB  conda-forge
  python_abi-3.7             |          2_cp37m           4 KB  conda-forge
  pytz-2022.1                |     pyhd8ed1ab_0         242 KB  conda-forge
  rasterio-1.2.4             |   py37h1339def_1         8.3 MB  conda-forge
  rioxarray-0.9.1            |     pyhd8ed1ab_0          44 KB  conda-forge
  scipy-1.7.0                |   py37h29e03ee_1        21.7 MB  conda-forge
  snuggs-1.4.7               |             py_0           8 KB  conda-forge
  tiledb-2.2.9               |       h91fcb0e_0         4.0 MB  conda-forge
  typing_extensions-4.2.0    |     pyha770c72_1          27 KB  conda-forge
  tzcode-2021a               |       h7f98852_2          68 KB  conda-forge
  tzdata-2022a               |       h191b570_0         121 KB  conda-forge
  xarray-0.20.2              |     pyhd8ed1ab_0         628 KB  conda-forge
  xerces-c-3.2.3             |       h9d8b166_2         1.8 MB  conda-forge
  xorg-kbproto-1.0.7         |    h7f98852_1002          27 KB  conda-forge
  xorg-libice-1.0.10         |       h7f98852_0          58 KB  conda-forge
  xorg-libsm-1.2.3           |    hd9c2040_1000          26 KB  conda-forge
  xorg-libx11-1.7.2          |       h7f98852_0         941 KB  conda-forge
  xorg-libxau-1.0.9          |       h7f98852_0          13 KB  conda-forge
  xorg-libxdmcp-1.1.3        |       h7f98852_0          19 KB  conda-forge
  xorg-libxext-1.3.4         |       h7f98852_1          54 KB  conda-forge
  xorg-libxrender-0.9.10     |    h7f98852_1003          32 KB  conda-forge
  xorg-renderproto-0.11.1    |    h7f98852_1002           9 KB  conda-forge
  xorg-xextproto-7.3.0       |    h7f98852_1002          28 KB  conda-forge
  xorg-xproto-7.0.31         |    h7f98852_1007          73 KB  conda-forge
  zipp-3.8.0                 |     pyhd8ed1ab_0          12 KB  conda-forge
  ------------------------------------------------------------
                                           Total:        1.37 GB

The following NEW packages will be INSTALLED:

  affine             conda-forge/noarch::affine-2.3.1-pyhd8ed1ab_0
  attrs              conda-forge/noarch::attrs-21.4.0-pyhd8ed1ab_0
  boost-cpp          conda-forge/linux-64::boost-cpp-1.74.0-hc6e9bd1_3
  cairo              conda-forge/linux-64::cairo-1.16.0-h6cf1ce9_1008
  cfitsio            conda-forge/linux-64::cfitsio-3.470-hb418390_7
  click              conda-forge/noarch::click-7.1.2-pyh9f0ad1d_0
  click-plugins      conda-forge/noarch::click-plugins-1.1.1-py_0
  cligj              conda-forge/noarch::cligj-0.7.2-pyhd8ed1ab_1
  cudatoolkit        conda-forge/linux-64::cudatoolkit-11.1.1-h6406543_8
  curl               conda-forge/linux-64::curl-7.75.0-h979ede3_0
  expat              conda-forge/linux-64::expat-2.4.1-h9c3ff4c_0
  fontconfig         conda-forge/linux-64::fontconfig-2.13.1-hba837de_1005
  freetype           conda-forge/linux-64::freetype-2.10.4-h0708190_1
  freexl             conda-forge/linux-64::freexl-1.0.6-h7f98852_0
  geos               conda-forge/linux-64::geos-3.9.1-h9c3ff4c_2
  geotiff            conda-forge/linux-64::geotiff-1.6.0-hcf90da6_5
  gettext            conda-forge/linux-64::gettext-0.19.8.1-h0b5b191_1005
  giflib             conda-forge/linux-64::giflib-5.2.1-h36c2ea0_2
  hdf4               conda-forge/linux-64::hdf4-4.2.15-h10796ff_3
  hdf5               conda-forge/linux-64::hdf5-1.10.6-nompi_h6a2412b_1114
  importlib-metadata conda-forge/linux-64::importlib-metadata-4.11.3-py37h89c1867_1
  importlib_metadata conda-forge/noarch::importlib_metadata-4.11.3-hd8ed1ab_1
  jpeg               conda-forge/linux-64::jpeg-9d-h36c2ea0_0
```

```
  json-c             conda-forge/linux-64::json-c-0.15-h98cffda_0
  kealib             conda-forge/linux-64::kealib-1.4.14-hcc255d8_2
  libblas            conda-forge/linux-64::libblas-3.9.0-11_linux64_openblas
  libcblas           conda-forge/linux-64::libcblas-3.9.0-11_linux64_openblas
  libdap4            conda-forge/linux-64::libdap4-3.20.6-hd7c4107_2
  libgdal            conda-forge/linux-64::libgdal-3.2.2-h804b7da_0
  libgfortran-ng     conda-forge/linux-64::libgfortran-ng-12.1.0-h69a702a_16
  libgfortran5       conda-forge/linux-64::libgfortran5-12.1.0-hdcd56e2_16
  libglib            conda-forge/linux-64::libglib-2.68.3-h3e27bee_0
  libkml             conda-forge/linux-64::libkml-1.3.0-h238a007_1014
  liblapack          conda-forge/linux-64::liblapack-3.9.0-11_linux64_openblas
  libnetcdf          conda-forge/linux-64::libnetcdf-4.7.4-nompi_h56d31a8_107
  libopenblas        conda-forge/linux-64::libopenblas-0.3.17-pthreads_h8fe5266_1
  libpng             conda-forge/linux-64::libpng-1.6.37-h21135ba_2
  libpq              conda-forge/linux-64::libpq-13.2-hfd2b0eb_2
  librttopo          conda-forge/linux-64::librttopo-1.1.0-h1185371_6
  libspatialite      conda-forge/linux-64::libspatialite-5.0.1-h20cb978_4
  libtiff            conda-forge/linux-64::libtiff-4.2.0-hbd63e13_2
  libuuid            conda-forge/linux-64::libuuid-2.32.1-h7f98852_1000
  libwebp-base       conda-forge/linux-64::libwebp-base-1.2.0-h7f98852_2
  libxcb             conda-forge/linux-64::libxcb-1.13-h7f98852_1003
  numpy              conda-forge/linux-64::numpy-1.21.1-py37h038b26d_0
  openjpeg           conda-forge/linux-64::openjpeg-2.4.0-hb52868f_1
  packaging          conda-forge/noarch::packaging-21.3-pyhd8ed1ab_0
  pandas             conda-forge/linux-64::pandas-1.3.1-py37h219a48f_0
  pcre               conda-forge/linux-64::pcre-8.45-h9c3ff4c_0
  pixman             conda-forge/linux-64::pixman-0.40.0-h36c2ea0_0
  poppler            conda-forge/linux-64::poppler-21.03.0-h93df280_0
  poppler-data       conda-forge/noarch::poppler-data-0.4.11-hd8ed1ab_0
  postgresql         conda-forge/linux-64::postgresql-13.2-h6303168_2
  proj               conda-forge/linux-64::proj-8.0.0-h277dcde_0
  pthread-stubs      conda-forge/linux-64::pthread-stubs-0.4-h36c2ea0_1001
  pyparsing          conda-forge/noarch::pyparsing-3.0.9-pyhd8ed1ab_0
  pyproj             conda-forge/linux-64::pyproj-3.1.0-py37h8627986_0
  python-dateutil    conda-forge/noarch::python-dateutil-2.8.2-pyhd8ed1ab_0
  pytz               conda-forge/noarch::pytz-2022.1-pyhd8ed1ab_0
  rasterio           conda-forge/linux-64::rasterio-1.2.4-py37h1339def_1
  rioxarray          conda-forge/noarch::rioxarray-0.9.1-pyhd8ed1ab_0
  scipy              conda-forge/linux-64::scipy-1.7.0-py37h29e03ee_1
  snuggs             conda-forge/noarch::snuggs-1.4.7-py_0
  tiledb             conda-forge/linux-64::tiledb-2.2.9-h91fcb0e_0
  typing_extensions  conda-forge/noarch::typing_extensions-4.2.0-pyha770c72_1
  tzcode             conda-forge/linux-64::tzcode-2021a-h7f98852_2
  tzdata             conda-forge/noarch::tzdata-2022a-h191b570_0
  xarray             conda-forge/noarch::xarray-0.20.2-pyhd8ed1ab_0
  xerces-c           conda-forge/linux-64::xerces-c-3.2.3-h9d8b166_2
  xorg-kbproto       conda-forge/linux-64::xorg-kbproto-1.0.7-h7f98852_1002
  xorg-libice        conda-forge/linux-64::xorg-libice-1.0.10-h7f98852_0
  xorg-libsm         conda-forge/linux-64::xorg-libsm-1.2.3-hd9c2040_1000
  xorg-libx11        conda-forge/linux-64::xorg-libx11-1.7.2-h7f98852_0
  xorg-libxau        conda-forge/linux-64::xorg-libxau-1.0.9-h7f98852_0
  xorg-libxdmcp      conda-forge/linux-64::xorg-libxdmcp-1.1.3-h7f98852_0
  xorg-libxext       conda-forge/linux-64::xorg-libxext-1.3.4-h7f98852_1
  xorg-libxrender    conda-forge/linux-64::xorg-libxrender-0.9.10-h7f98852_1003
  xorg-renderproto   conda-forge/linux-64::xorg-renderproto-0.11.1-h7f98852_1002
  xorg-xextproto     conda-forge/linux-64::xorg-xextproto-7.3.0-h7f98852_1002
  xorg-xproto        conda-forge/linux-64::xorg-xproto-7.0.31-h7f98852_1007
  zipp               conda-forge/noarch::zipp-3.8.0-pyhd8ed1ab_0

The following packages will be UPDATED:

  ca-certificates                     2020.12.5-ha878542_0 --> 2022.5.18-ha878542_0
  certifi                         2020.12.5-py37h89c1867_1 --> 2022.5.18-py37h89c1867_0
  conda                               4.9.2-py37h89c1867_0 --> 4.12.0-py37h89c1867_0
  openssl                                1.1.1j-h7f98852_0 --> 1.1.1k-h7f98852_0
  python_abi                                   3.7-1_cp37m --> 3.7-2_cp37m



Downloading and Extracting Packages
xorg-libice-1.0.10   | 58 KB     | : 100% 1.0/1 [00:00<00:00,  8.95it/s]
xorg-libx11-1.7.2    | 941 KB    | : 100% 1.0/1 [00:00<00:00,  4.98it/s]
```

```
conda-4.12.0          | 1.0 MB    | : 100% 1.0/1 [00:00<00:00,  4.13it/s]
pyparsing-3.0.9       | 79 KB     | : 100% 1.0/1 [00:00<00:00, 22.45it/s]
freexl-1.0.6          | 48 KB     | : 100% 1.0/1 [00:00<00:00, 23.84it/s]
xorg-libxrender-0.9.  | 32 KB     | : 100% 1.0/1 [00:00<00:00, 25.61it/s]
numpy-1.21.1          | 6.1 MB    | : 100% 1.0/1 [00:01<00:00,  1.13s/it]
geotiff-1.6.0         | 296 KB    | : 100% 1.0/1 [00:00<00:00,  9.51it/s]
fontconfig-2.13.1     | 357 KB    | : 100% 1.0/1 [00:00<00:00, 11.32it/s]
libgfortran5-12.1.0   | 1.8 MB    | : 100% 1.0/1 [00:00<00:00,  3.31it/s]
affine-2.3.1          | 17 KB     | : 100% 1.0/1 [00:00<00:00, 27.04it/s]
xorg-kbproto-1.0.7    | 27 KB     | : 100% 1.0/1 [00:00<00:00, 22.61it/s]
giflib-5.2.1          | 77 KB     | : 100% 1.0/1 [00:00<00:00, 23.67it/s]
libcblas-3.9.0        | 11 KB     | : 100% 1.0/1 [00:00<00:00, 30.61it/s]
pthread-stubs-0.4     | 5 KB      | : 100% 1.0/1 [00:00<00:00, 23.31it/s]
expat-2.4.1           | 182 KB    | : 100% 1.0/1 [00:00<00:00, 16.56it/s]
libpq-13.2            | 2.7 MB    | : 100% 1.0/1 [00:00<00:00,  1.98it/s]
libblas-3.9.0         | 12 KB     | : 100% 1.0/1 [00:00<00:00, 21.90it/s]
proj-8.0.0            | 3.1 MB    | : 100% 1.0/1 [00:00<00:00,  1.72it/s]
libnetcdf-4.7.4       | 1.3 MB    | : 100% 1.0/1 [00:00<00:00,  4.22it/s]
cairo-1.16.0          | 1.5 MB    | : 100% 1.0/1 [00:00<00:00,  3.44it/s]
openssl-1.1.1k        | 2.1 MB    | : 100% 1.0/1 [00:00<00:00,  3.20it/s]
postgresql-13.2       | 5.3 MB    | : 100% 1.0/1 [00:01<00:00,  1.25s/it]
libspatialite-5.0.1   | 4.4 MB    | : 100% 1.0/1 [00:00<00:00,  1.34it/s]
tiledb-2.2.9          | 4.0 MB    | : 100% 1.0/1 [00:00<00:00,  1.39it/s]
xorg-libxext-1.3.4    | 54 KB     | : 100% 1.0/1 [00:00<00:00, 24.25it/s]
libglib-2.68.3        | 3.1 MB    | : 100% 1.0/1 [00:00<00:00,  1.75it/s]
cfitsio-3.470         | 1.3 MB    | : 100% 1.0/1 [00:00<00:00,  4.36it/s]
xorg-libxdmcp-1.1.3   | 19 KB     | : 100% 1.0/1 [00:00<00:00, 26.84it/s]
curl-7.75.0           | 147 KB    | : 100% 1.0/1 [00:00<00:00, 20.21it/s]
xorg-libsm-1.2.3      | 26 KB     | : 100% 1.0/1 [00:00<00:00, 19.42it/s]
importlib-metadata-4  | 33 KB     | : 100% 1.0/1 [00:00<00:00, 18.26it/s]
python-dateutil-2.8.  | 240 KB    | : 100% 1.0/1 [00:00<00:00, 17.13it/s]
libgdal-3.2.2         | 13.2 MB   | : 100% 1.0/1 [00:03<00:00,  3.09s/it]
cligj-0.7.2           | 10 KB     | : 100% 1.0/1 [00:00<00:00, 21.67it/s]
geos-3.9.1            | 1.1 MB    | : 100% 1.0/1 [00:00<00:00,  4.28it/s]
tzdata-2022a          | 121 KB    | : 100% 1.0/1 [00:00<00:00, 10.98it/s]
scipy-1.7.0           | 21.7 MB   | : 100% 1.0/1 [00:03<00:00,  3.16s/it]
packaging-21.3        | 36 KB     | : 100% 1.0/1 [00:00<00:00, 24.70it/s]
xorg-xextproto-7.3.0  | 28 KB     | : 100% 1.0/1 [00:00<00:00, 24.14it/s]
poppler-data-0.4.11   | 3.6 MB    | : 100% 1.0/1 [00:00<00:00,  1.80it/s]
click-7.1.2           | 64 KB     | : 100% 1.0/1 [00:00<00:00, 23.62it/s]
certifi-2022.5.18     | 150 KB    | : 100% 1.0/1 [00:00<00:00, 19.98it/s]
ca-certificates-2022  | 144 KB    | : 100% 1.0/1 [00:00<00:00, 17.39it/s]
libuuid-2.32.1        | 28 KB     | : 100% 1.0/1 [00:00<00:00, 27.27it/s]
libgfortran-ng-12.1.  | 23 KB     | : 100% 1.0/1 [00:00<00:00, 21.49it/s]
jpeg-9d               | 264 KB    | : 100% 1.0/1 [00:00<00:00, 10.78it/s]
rioxarray-0.9.1       | 44 KB     | : 100% 1.0/1 [00:00<00:00, 22.30it/s]
importlib_metadata-4  | 4 KB      | : 100% 1.0/1 [00:00<00:00, 28.89it/s]
rasterio-1.2.4        | 8.3 MB    | : 100% 1.0/1 [00:01<00:00,  1.28s/it]
libkml-1.3.0          | 591 KB    | : 100% 1.0/1 [00:00<00:00,  7.25it/s]
xorg-renderproto-0.1  | 9 KB      | : 100% 1.0/1 [00:00<00:00, 30.90it/s]
gettext-0.19.8.1      | 3.6 MB    | : 100% 1.0/1 [00:00<00:00,  1.39it/s]
libopenblas-0.3.17    | 9.2 MB    | : 100% 1.0/1 [00:01<00:00,  1.50s/it]
tzcode-2021a          | 68 KB     | : 100% 1.0/1 [00:00<00:00, 11.87it/s]
poppler-21.03.0       | 15.9 MB   | : 100% 1.0/1 [00:02<00:00,  2.40s/it]
librttopo-1.1.0       | 235 KB    | : 100% 1.0/1 [00:00<00:00, 16.26it/s]
xerces-c-3.2.3        | 1.8 MB    | : 100% 1.0/1 [00:00<00:00,  1.69it/s]
pandas-1.3.1          | 12.7 MB   | : 100% 1.0/1 [00:02<00:00,  2.40s/it]
xorg-xproto-7.0.31    | 73 KB     | : 100% 1.0/1 [00:00<00:00, 22.84it/s]
libxcb-1.13           | 395 KB    | : 100% 1.0/1 [00:00<00:00,  7.98it/s]
kealib-1.4.14         | 186 KB    | : 100% 1.0/1 [00:00<00:00, 17.97it/s]
freetype-2.10.4       | 890 KB    | : 100% 1.0/1 [00:00<00:00,  5.36it/s]
click-plugins-1.1.1   | 9 KB      | : 100% 1.0/1 [00:00<00:00, 19.81it/s]
libpng-1.6.37         | 306 KB    | : 100% 1.0/1 [00:00<00:00, 11.06it/s]
xorg-libxau-1.0.9     | 13 KB     | : 100% 1.0/1 [00:00<00:00, 30.18it/s]
hdf5-1.10.6           | 3.1 MB    | : 100% 1.0/1 [00:00<00:00,  2.27it/s]
python_abi-3.7        | 4 KB      | : 100% 1.0/1 [00:00<00:00, 29.77it/s]
hdf4-4.2.15           | 950 KB    | : 100% 1.0/1 [00:00<00:00,  5.51it/s]
openjpeg-2.4.0        | 444 KB    | : 100% 1.0/1 [00:00<00:00,  7.79it/s]
attrs-21.4.0          | 49 KB     | : 100% 1.0/1 [00:00<00:00, 23.65it/s]
pyproj-3.1.0          | 513 KB    | : 100% 1.0/1 [00:00<00:00,  3.20it/s]
cudatoolkit-11.1.1    | 1.20 GB   | : 100% 1.0/1 [02:27<00:00, 147.72s/it]
boost-cpp-1.74.0      | 16.3 MB   | : 100% 1.0/1 [00:04<00:00,  4.82s/it]
```

```
pytz-2022.1          | 242 KB    | : 100% 1.0/1 [00:00<00:00,  9.17it/s]
libtiff-4.2.0        | 639 KB    | : 100% 1.0/1 [00:00<00:00,  7.09it/s]
liblapack-3.9.0      | 11 KB     | : 100% 1.0/1 [00:00<00:00, 27.64it/s]
json-c-0.15          | 274 KB    | : 100% 1.0/1 [00:00<00:00, 12.65it/s]
zipp-3.8.0           | 12 KB     | : 100% 1.0/1 [00:00<00:00, 25.47it/s]
libdap4-3.20.6       | 11.3 MB   | : 100% 1.0/1 [00:01<00:00,  1.64s/it]
libwebp-base-1.2.0   | 815 KB    | : 100% 1.0/1 [00:00<00:00,  4.34it/s]
pixman-0.40.0        | 627 KB    | : 100% 1.0/1 [00:00<00:00,  8.31it/s]
typing_extensions-4. | 27 KB     | : 100% 1.0/1 [00:00<00:00, 23.19it/s]
pcre-8.45            | 253 KB    | : 100% 1.0/1 [00:00<00:00, 12.63it/s]
snuggs-1.4.7         | 8 KB      | : 100% 1.0/1 [00:00<00:00, 24.27it/s]
xarray-0.20.2        | 628 KB    | : 100% 1.0/1 [00:00<00:00,  5.67it/s]
Preparing transaction: done
Verifying transaction: done
Executing transaction: | By downloading and using the CUDA Toolkit conda packages, you accept the te
rms and conditions of the CUDA End User License Agreement (EULA): https://docs.nvidia.com/cuda/eula/
index.html (https://docs.nvidia.com/cuda/eula/index.html)

done
Collecting azure.storage.blob
  Downloading azure_storage_blob-12.12.0-py3-none-any.whl (366 kB)
     |████████████████████████████████| 366 kB 5.1 MB/s
Collecting msrest>=0.6.21
  Downloading msrest-0.6.21-py2.py3-none-any.whl (85 kB)
     |████████████████████████████████| 85 kB 4.4 MB/s
Collecting azure-core<2.0.0,>=1.23.1
  Downloading azure_core-1.24.0-py3-none-any.whl (178 kB)
     |████████████████████████████████| 178 kB 73.7 MB/s
Requirement already satisfied: cryptography>=2.1.4 in /usr/local/lib/python3.7/site-packages (from a
zure.storage.blob) (3.4.5)
Requirement already satisfied: six>=1.11.0 in /usr/local/lib/python3.7/site-packages (from azure-cor
e<2.0.0,>=1.23.1->azure.storage.blob) (1.15.0)
Requirement already satisfied: requests>=2.18.4 in /usr/local/lib/python3.7/site-packages (from azur
e-core<2.0.0,>=1.23.1->azure.storage.blob) (2.25.1)
Requirement already satisfied: typing-extensions>=4.0.1 in /usr/local/lib/python3.7/site-packages (f
rom azure-core<2.0.0,>=1.23.1->azure.storage.blob) (4.2.0)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.7/site-packages (from cryptograp
hy>=2.1.4->azure.storage.blob) (1.14.5)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/site-packages (from cffi>=1.12-
>cryptography>=2.1.4->azure.storage.blob) (2.20)
Collecting requests-oauthlib>=0.5.0
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Collecting isodate>=0.6.0
  Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
     |████████████████████████████████| 41 kB 707 kB/s
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from ms
rest>=0.6.21->azure.storage.blob) (2022.5.18)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from req
uests>=2.18.4->azure-core<2.0.0,>=1.23.1->azure.storage.blob) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/site-packages (from requests
>=2.18.4->azure-core<2.0.0,>=1.23.1->azure.storage.blob) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (from
requests>=2.18.4->azure-core<2.0.0,>=1.23.1->azure.storage.blob) (1.26.3)
Collecting oauthlib>=3.0.0
  Downloading oauthlib-3.2.0-py3-none-any.whl (151 kB)
     |████████████████████████████████| 151 kB 88.8 MB/s
Installing collected packages: oauthlib, requests-oauthlib, isodate, msrest, azure-core, azure.stora
ge.blob
Successfully installed azure-core-1.24.0 azure.storage.blob isodate-0.6.1 msrest-0.6.21 oauthlib-3.
2.0 requests-oauthlib-1.3.1
Requirement already satisfied: azure-core in /usr/local/lib/python3.7/site-packages (1.24.0)
Requirement already satisfied: typing-extensions>=4.0.1 in /usr/local/lib/python3.7/site-packages (f
rom azure-core) (4.2.0)
Requirement already satisfied: requests>=2.18.4 in /usr/local/lib/python3.7/site-packages (from azur
e-core) (2.25.1)
Requirement already satisfied: six>=1.11.0 in /usr/local/lib/python3.7/site-packages (from azure-cor
e) (1.15.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from req
uests>=2.18.4->azure-core) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/site-packages (from requests
>=2.18.4->azure-core) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from re
```

```
quests>=2.18.4->azure-core) (2022.5.18)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (from
requests>=2.18.4->azure-core) (1.26.3)
Collecting wget
  Downloading wget-3.2.zip (10 kB)
Building wheels for collected packages: wget
  Building wheel for wget (setup.py) ... done
  Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9680 sha256=d02e8578eca24d61538b47
7b582818182d828ef42c9b2fea8f3eb7de1481a574
  Stored in directory: /root/.cache/pip/wheels/a1/b6/7c/0e63e34eb06634181c63adacca38b79ff8f35c37e3c1
3e3c02
Successfully built wget
Installing collected packages: wget
Successfully installed wget-3.2
Collecting geojson
  Downloading geojson-2.5.0-py2.py3-none-any.whl (14 kB)
Installing collected packages: geojson
Successfully installed geojson-2.5.0
Collecting gcsfs
  Downloading gcsfs-2022.3.0-py2.py3-none-any.whl (25 kB)
Collecting fsspec==2022.3.0
  Downloading fsspec-2022.3.0-py3-none-any.whl (136 kB)
     |████████████████████████████████| 136 kB 7.6 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/site-packages (from gcsfs) (2.2
5.1)
Collecting aiohttp<4
  Downloading aiohttp-3.8.1-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.
manylinux2010_x86_64.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 42.5 MB/s
Collecting google-cloud-storage
  Downloading google_cloud_storage-2.3.0-py2.py3-none-any.whl (107 kB)
     |████████████████████████████████| 107 kB 94.4 MB/s
Collecting decorator>4.1.2
  Downloading decorator-5.1.1-py3-none-any.whl (9.1 kB)
Collecting google-auth-oauthlib
  Downloading google_auth_oauthlib-0.5.1-py2.py3-none-any.whl (19 kB)
Collecting google-auth>=1.2
  Downloading google_auth-2.6.6-py2.py3-none-any.whl (156 kB)
     |████████████████████████████████| 156 kB 94.7 MB/s
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.7.2-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.man
ylinux2010_x86_64.whl (271 kB)
     |████████████████████████████████| 271 kB 62.0 MB/s
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_
64.manylinux2014_x86_64.whl (144 kB)
     |████████████████████████████████| 144 kB 75.5 MB/s
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/site-packages (f
rom aiohttp<4->gcsfs) (4.2.0)
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (94 kB)
     |████████████████████████████████| 94 kB 3.8 MB/s
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting charset-normalizer<3.0,>=2.0
  Downloading charset_normalizer-2.0.12-py3-none-any.whl (39 kB)
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.2.0-py3-none-any.whl (8.2 kB)
Collecting asynctest==0.13.0
  Downloading asynctest-0.13.0-py3-none-any.whl (26 kB)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/site-packages (from aiohttp
<4->gcsfs) (21.4.0)
Collecting cachetools<6.0,>=2.0.0
  Downloading cachetools-5.1.0-py3-none-any.whl (9.2 kB)
Collecting rsa<5,>=3.1.4
  Downloading rsa-4.8-py3-none-any.whl (39 kB)
Collecting pyasn1-modules>=0.2.1
  Downloading pyasn1_modules-0.2.8-py2.py3-none-any.whl (155 kB)
     |████████████████████████████████| 155 kB 99.2 MB/s
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/site-packages (from google-aut
h>=1.2->gcsfs) (1.15.0)
Collecting pyasn1<0.5.0,>=0.4.6
```

```
  Downloading pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
     |████████████████████████████████| 77 kB 7.1 MB/s
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.7/site-packages (from yarl<2.0,>=
1.0->aiohttp<4->gcsfs) (2.10)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/site-packages (f
rom google-auth-oauthlib->gcsfs) (1.3.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/site-packages (from reque
sts-oauthlib>=0.7.0->google-auth-oauthlib->gcsfs) (3.2.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from req
uests->gcsfs) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (from
requests->gcsfs) (1.26.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from re
quests->gcsfs) (2022.5.18)
Collecting protobuf
  Downloading protobuf-3.20.1-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.0 MB)
     |████████████████████████████████| 1.0 MB 80.2 MB/s
Collecting google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5
  Downloading google_api_core-2.8.0-py3-none-any.whl (114 kB)
     |████████████████████████████████| 114 kB 98.5 MB/s
Collecting google-cloud-core<3.0dev,>=2.3.0
  Downloading google_cloud_core-2.3.0-py2.py3-none-any.whl (29 kB)
Collecting google-resumable-media>=2.3.2
  Downloading google_resumable_media-2.3.3-py2.py3-none-any.whl (76 kB)
     |████████████████████████████████| 76 kB 6.7 MB/s
Collecting googleapis-common-protos<2.0dev,>=1.52.0
  Downloading googleapis_common_protos-1.56.1-py2.py3-none-any.whl (211 kB)
     |████████████████████████████████| 211 kB 93.8 MB/s
Collecting google-crc32c<2.0dev,>=1.0
  Downloading google_crc32c-1.3.0-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (38 kB)
Installing collected packages: pyasn1, rsa, pyasn1-modules, protobuf, cachetools, googleapis-common-
protos, google-auth, multidict, google-crc32c, google-api-core, frozenlist, yarl, google-resumable-m
edia, google-cloud-core, charset-normalizer, asynctest, async-timeout, aiosignal, google-cloud-stora
ge, google-auth-oauthlib, fsspec, decorator, aiohttp, gcsfs
Successfully installed aiohttp-3.8.1 aiosignal-1.2.0 async-timeout-4.0.2 asynctest-0.13.0 cachetools
-5.1.0 charset-normalizer-2.0.12 decorator-5.1.1 frozenlist-1.3.0 fsspec-2022.3.0 gcsfs-2022.3.0 goo
gle-api-core-2.8.0 google-auth-2.6.6 google-auth-oauthlib-0.5.1 google-cloud-core-2.3.0 google-cloud
-storage-2.3.0 google-crc32c-1.3.0 google-resumable-media-2.3.3 googleapis-common-protos-1.56.1 mult
idict-6.0.2 protobuf-3.20.1 pyasn1-0.4.8 pyasn1-modules-0.2.8 rsa-4.8 yarl-1.7.2
```

```
Collecting pystac_client
  Downloading pystac_client-0.3.3-py3-none-any.whl (19 kB)
Collecting pystac>=1.4.0
  Downloading pystac-1.4.0-py3-none-any.whl (137 kB)
     |████████████████████████████████| 137 kB 5.0 MB/s eta 0:00:01
Requirement already satisfied: requests>=2.25 in /usr/local/lib/python3.7/site-packages (from pys
tac_client) (2.25.1)
Requirement already satisfied: python-dateutil>=2.7.0 in /usr/local/lib/python3.7/site-packages
(from pystac>=1.4.0->pystac_client) (2.8.2)
Requirement already satisfied: typing-extensions>=3.7 in /usr/local/lib/python3.7/site-packages
(from pystac>=1.4.0->pystac_client) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-da
teutil>=2.7.0->pystac>=1.4.0->pystac_client) (1.15.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from
requests>=2.25->pystac_client) (4.0.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (f
rom requests>=2.25->pystac_client) (1.26.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/site-packages (from reque
sts>=2.25->pystac_client) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from
requests>=2.25->pystac_client) (2022.5.18)
Installing collected packages: pystac, pystac-client
Successfully installed pystac-1.4.0 pystac-client-0.3.3
Collecting planetary_computer
  Downloading planetary_computer-0.4.6-py3-none-any.whl (14 kB)
Requirement already satisfied: pystac-client>=0.2.0 in /usr/local/lib/python3.7/site-packages (fr
om planetary_computer) (0.3.3)
Requirement already satisfied: pystac>=1.0.0 in /usr/local/lib/python3.7/site-packages (from plan
etary_computer) (1.4.0)
Requirement already satisfied: pytz>=2020.5 in /usr/local/lib/python3.7/site-packages (from plane
```

```
tary_computer) (2022.1)
Collecting pydantic[dotenv]>=1.7.3
  Downloading pydantic-1.9.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.1 MB)
     |████████████████████████████████| 11.1 MB 5.0 MB/s
Requirement already satisfied: requests>=2.25.1 in /usr/local/lib/python3.7/site-packages (from p
lanetary_computer) (2.25.1)
Requirement already satisfied: click>=7.1 in /usr/local/lib/python3.7/site-packages (from planeta
ry_computer) (7.1.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/site-packag
es (from pydantic[dotenv]>=1.7.3->planetary_computer) (4.2.0)
Collecting python-dotenv>=0.10.4
  Downloading python_dotenv-0.20.0-py3-none-any.whl (17 kB)
Requirement already satisfied: python-dateutil>=2.7.0 in /usr/local/lib/python3.7/site-packages
(from pystac>=1.0.0->planetary_computer) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-da
teutil>=2.7.0->pystac>=1.0.0->planetary_computer) (1.15.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/site-packages (from reque
sts>=2.25.1->planetary_computer) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from
requests>=2.25.1->planetary_computer) (2022.5.18)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.7/site-packages (f
rom requests>=2.25.1->planetary_computer) (1.26.3)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.7/site-packages (from
requests>=2.25.1->planetary_computer) (4.0.0)
Installing collected packages: python-dotenv, pydantic, planetary-computer
Successfully installed planetary-computer-0.4.6 pydantic-1.9.1 python-dotenv-0.20.0
```

In [3]:

```
import pandas as pd
from pyproj import Proj
from pystac_client import Client
import planetary_computer
import rasterio
import geojson as gsn
# from osgeo import gdal
# from osgeo import gdalconst

import tempfile
import wget
import math
import random
import numpy as np
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm

import os
import pickle
from collections import defaultdict
from datetime import datetime, timedelta

import xarray as xr
import rioxarray as rxr
from azure.storage.blob import ContainerClient

import gcsfs

modis_account_name = 'modissa'
modis_container_name = 'modis-006'
modis_account_url = 'https://' + modis_account_name + '.blob.core.windows.net/'
modis_blob_root = modis_account_url + modis_container_name + '/'

# This file is provided by NASA; it indicates the lat/lon extents of each
# NOTE: this was from tutorial, not actually helpful because unprojected?

modis_tile_extents_url = modis_blob_root + 'sn_bound_10deg.txt'

temp_dir = os.path.join(tempfile.gettempdir(),'modis_snow')
os.makedirs(temp_dir,exist_ok=True)
fn = os.path.join(temp_dir,modis_tile_extents_url.split('/')[-1])
# wget.download(modis_tile_extents_url, fn)


modis_container_client = ContainerClient(account_url=modis_account_url,
                                         container_name=modis_container_name,
                                                  credential=None)
```

executed in 28ms, finished 18:01:07 2022-05-16

Let's clone the repository to access the data in the repo for analysis.

In [2]:

```
! git clone https://github.com/hanis-z/Snow-water-equivalent.git
```

```
fatal: destination path 'Snow-water-equivalent' already exists and is not an empty directory.
```

Below, I'm establishing my google cloud storage to save my satellite images to.

In [4]:
```
from google.colab import auth
auth.authenticate_user()

# https://cloud.google.com/resource-manager/docs/creating-managing-projects
project_id = 'snow-water-equivalent'
# !gcloud config set project {project_id}
project_name = 'Snow-Water-Equivalent'
bucket_name = 'modis-swe'

fs = gcsfs.GCSFileSystem(projectstring=project_id)
fs.ls(bucket_name)
```

Out[4]:
```
['modis-swe/Copernicus_DSM_COG_10_N35_00_E138_00_DEM.tif',
 'modis-swe/ModisSnowImagesA.npy',
 'modis-swe/ModisSnowImagesT.npy',
 'modis-swe/ModisSnowImages_testA.npy',
 'modis-swe/ModisSnowImages_testT.npy',
 'modis-swe/cell_snow_idsA.pkl',
 'modis-swe/cell_snow_idsT.pkl',
 'modis-swe/cell_snow_ids_testA.pkl',
 'modis-swe/cell_snow_ids_testT.pkl']
```

## 1.5 MODIS data

### 1.5.1 Helper Functions

These helper functions are from a notebook example provided by DrivenData (https://www.drivendata.org/competitions/86/competition-reclamation-snow-water-dev/page/417/) here (https://nbviewer.org/github/microsoft/AIforEarthDataSets/blob/main/data/modis.ipynb). I also adopted the code (https://github.com/M-Harrington/SnowComp) to load MODIS data from DrivenData user @themrharrington (https://community.drivendata.org/u/themrharrington).

In [8]:

```
def lat_lon_to_modis_tile(lat,lon):
    '''converts lat lon to modis tiles but reconstructing grid and its projection'''

    CELLS = 2400
    VERTICAL_TILES = 18
    HORIZONTAL_TILES = 36
    EARTH_RADIUS = 6371007.181
    EARTH_WIDTH = 2 * math.pi * EARTH_RADIUS

    TILE_WIDTH = EARTH_WIDTH / HORIZONTAL_TILES
    TILE_HEIGHT = TILE_WIDTH
    CELL_SIZE = TILE_WIDTH / CELLS

    MODIS_GRID = Proj(f'+proj=sinu +R={EARTH_RADIUS} +nadgrids=@null +wktext')

    x, y = MODIS_GRID(lon, lat)
    h = (EARTH_WIDTH * .5 + x) / TILE_WIDTH
    v = -(EARTH_WIDTH * .25 + y - (VERTICAL_TILES - 0) * TILE_HEIGHT) / TILE_HEIGHT

    return int(h), int(v)


def list_blobs_in_folder(container_name,folder_name):
    """
    List all blobs in a virtual folder in an Azure blob container
    """

    files = []
    generator = modis_container_client.list_blobs(name_starts_with=folder_name)
    for blob in generator:
        files.append(blob.name)
    return files


def list_hdf_blobs_in_folder(container_name,folder_name):
    """"
    List .hdf files in a folder
    """

    files = list_blobs_in_folder(container_name,folder_name)
    files = [fn for fn in files if fn.endswith('.hdf')]
    return files

# daynum = '2014236'
def daynum_gen(date_time):
    '''converts date time objects to filename'''
    doy = date_time.timetuple().tm_yday
    year = date_time.year
    return str(year) + '{:03d}'.format(doy)
```

executed in 23ms, finished 17:39:05 2022-05-16

In [9]:

```
def images_downloader(tiles, centroids, out_dataset, prod_name, verbose = False):
    """"""
    cell_ids = []
    i = 0
    for date_tile in tiles.keys():
        print("\n",i)

        date = date_tile[0]
        daynum = daynum_gen(date)
        daynum_og = daynum #to save later
        tile_num = (date_tile[1],date_tile[2])


        folder = prod_name + '/' + '{:0>2d}/{:0>2d}'.format(date_tile[1],date_tile[2]) + '/' + dayn

        # Find all HDF files from this tile on this day
        filenames = list_hdf_blobs_in_folder(modis_container_name,folder)
        print('Found {} matching file(s):'.format(len(filenames)))
        for fn in filenames:
            print(fn)
        file_root = filenames.copy()

        if len(file_root) > 1: #images may come in multiples
            print("multiple files found: ", len(file_root))
            blob_name1 = filenames[0]
            blob_name2 = filenames[1]

            # Download to a temporary file
            url1 = modis_blob_root + blob_name1
            url2 = modis_blob_root + blob_name2

            filename = os.path.join(temp_dir,blob_name1.replace('/','_'))
            if not os.path.isfile(filename):
                wget.download(url1,filename)

            filename = os.path.join(temp_dir,blob_name2.replace('/','_'))
            if not os.path.isfile(filename):
                wget.download(url2,filename)
            rds1 = rxr.open_rasterio(filename)
            rds2 = rxr.open_rasterio(filename)

            #find highest quality image
            rds1_quality = ((rds1.NDSI_Snow_Cover_Basic_QA.values >0) | (rds1.NDSI_Snow_Cover_Basic
            rds2_quality = ((rds2.NDSI_Snow_Cover_Basic_QA.values >0) | (rds2.NDSI_Snow_Cover_Basic

            rds = rds1 if rds1_quality >= rds2_quality else rds2

        else:
            # Work with the first returned URL
            file_found = False
            breaker = 1
            while not file_found and breaker <= 5:
                try:
                    blob_name = filenames[0]
                    file_found = True
                except IndexError:
                    print("No file found: tile {} date {}".format(tile_num,daynum))
                    date -= timedelta(days=1)
                    daynum = daynum_gen(date)

                    breaker +=1
                    print("trying:", daynum)
            if breaker == 5:
                raise ValueError("Image", tile_num, daynum, "not found")


            # Download to a temporary file
            url = modis_blob_root + blob_name
            filename = os.path.join(temp_dir,blob_name.replace('/','_'))
            if not os.path.isfile(filename):
                wget.download(url,filename)
```

```
            rds = rxr.open_rasterio(filename)

        #####reproject#####
        image = rds.rio.reproject(dst_crs="EPSG:4326")
        for var in image.data_vars:
            image[var]=image[var].astype(image[var].dtype,keep_attrs = False)


        #####create blocks around centroids#####
        cells = tiles[date_tile]
        for cell in cells:
            center = centroids[cell]


            x_idx = np.nanargmin(np.abs(image.x.values - center[0]))
            y_idx = np.nanargmin(np.abs(image.y.values - center[1]))

            #subset 21x21 square
            xmin, xmin_actual, xmax = max(x_idx -10, 0) , x_idx -10, x_idx + 11
            ymin, ymin_actual, ymax = max(y_idx -10, 0) , y_idx -10, y_idx + 11

            sub_image = image[dict(x= slice(xmin,xmax), y= slice(ymin,ymax))]

            try: # in case we're against boundary
                sub_image = sub_image.squeeze().to_array().to_numpy()
                out_dataset[i] = sub_image
            except ValueError as e:
                #flip and reflip before saving because coding's hard
                sub_image = np.swapaxes(sub_image, 1,2)

                image_shape = tuple(image.dims[d] for d in ['x', 'y'])
                simage_shape = sub_image.shape
                if verbose:
                    print(e)
                    print("Out of bounds error, padding with 0 for day/grid:", daynum_og, cell)

                    print("input shape: ", image_shape, "output shape", simage_shape)
                    print("max/min", xmax, ymax, xmin, ymin)

                #pad with necessary columns
                if xmin_actual < 0:
                    fill = np.zeros((out_dataset.shape[1],
                                     0-xmin_actual, simage_shape[1]))
                    sub_image = np.concatenate((fill, sub_image), axis= 1)
                    simage_shape = sub_image.shape
                    if verbose:
                        print("off left")
                        print("updated simage_shape", simage_shape)

                elif xmax > image_shape[0]:
                    fill = np.zeros((out_dataset.shape[1],
                                     xmax- image_shape[0], simage_shape[1]))
                    sub_image = np.concatenate((sub_image, fill), axis=1)
                    simage_shape = sub_image.shape
                    print("off right")
                    print("updated simage_shape", simage_shape)

                if ymin_actual < 0 :
                    fill = np.zeros((out_dataset.shape[1],
                                    21, 0-ymin_actual ))
                    sub_image = np.concatenate((fill, sub_image), axis=2)
                    simage_shape = sub_image.shape
                    if verbose:
                        print("off up")
                        print("updated simage_shape", simage_shape)

                elif ymax > image_shape[1]:
                    fill = np.zeros((out_dataset.shape[1],
                                     21, ymax - image_shape[1] ))
                    sub_image = np.concatenate((sub_image,fill), axis=2)
                    simage_shape = sub_image.shape
```

```
                    if verbose:
                        print("off down")
                        print("updated simage_shape", simage_shape)

                sub_image = np.swapaxes(sub_image, 1,2)
                out_dataset[i] = sub_image



            cell_ids.append((cell, daynum_og))

            i+=1



    return cell_ids, out_dataset
```

executed in 52ms, finished 17:59:44 2022-05-16

### 1.5.2 Ground Measures Data

In [4]:

```
gm_md = pd.read_csv("Snow-water-equivalent/data/ground_measures_metadata.csv")
gm_md
```

executed in 94ms, finished 17:39:08 2022-05-16

Out[4]:

| | station_id | name | elevation_m | latitude | longitude | state |
|---|---|---|---|---|---|---|
| **0** | CDEC:ADM | Adin Mountain | 1889.760000 | 41.237000 | -120.792000 | California |
| **1** | CDEC:AGP | Agnew Pass | 2880.360000 | 37.726631 | -119.141731 | California |
| **2** | CDEC:ALP | Alpha (Smud) | 2316.480000 | 38.804192 | -120.215652 | California |
| **3** | CDEC:BCB | Blackcap Basin | 3139.440000 | 37.066685 | -118.773010 | California |
| **4** | CDEC:BCH | Beach Meadows | 2331.720000 | 36.126095 | -118.293457 | California |
| **...** | ... | ... | ... | ... | ... | ... |
| **759** | SNOTEL:994_WA_SNTL | Epa Quinault Open | 91.440002 | 46.483330 | -123.966667 | Washington |
| **760** | SNOTEL:995_WA_SNTL | Epa Quinault Can | 91.440002 | 46.483330 | -123.966667 | Washington |
| **761** | SNOTEL:996_WA_SNTL | NWA Heather Mdws | 1267.968018 | 48.849998 | -121.666672 | Washington |
| **762** | SNOTEL:998_WA_SNTL | Easy Pass | 1606.296021 | 48.859329 | -121.438950 | Washington |
| **763** | SNOTEL:999_WA_SNTL | Marten Ridge | 1072.895996 | 48.762920 | -121.698227 | Washington |

764 rows × 6 columns

### 1.5.3 Cell grids

This [dataset (%22../data/grid_cells.geojson%22)](%22../data/grid_cells.geojson%22) provides us spatial information about the grid cells in the submission format.

In [5]:

```
path = "Snow-water-equivalent/data/grid_cells.geojson"
with open(path) as f:
    gj = gsn.load(f)
print(len(gj['features']))
# gj['features']
```

executed in 1.58s, finished 17:40:13 2022-05-16

```
18130
```

In [6]:

```
gj.keys()
```

executed in 12ms, finished 17:40:14 2022-05-16

Out[6]:

```
dict_keys(['type', 'crs', 'features'])
```

In [7]:
```
gj['features'][1]
```
executed in 14ms, finished 17:40:15 2022-05-16

Out[7]:
```
{"geometry": {"coordinates": [[[-107.076787, 37.780424], [-107.076787, 37.787523], [-107.08577, 37.7
87523], [-107.08577, 37.780424], [-107.076787, 37.780424]]], "type": "Polygon"}, "properties": {"cel
l_id": "000617d8-8c14-43e2-b708-7e3a69fe3cc3", "region": "central rockies"}, "type": "Feature"}
```

Let's transform our geojson data into a dataframe so its easier to look at and work with

In [8]:
```
grid_cell_df = pd.DataFrame.from_dict(gj['features'])
grid_cell_df['cell_id'] = [x['cell_id'] for x in grid_cell_df['properties']]
grid_cell_df['coordinates'] = [x['coordinates'][0][0:] for x in grid_cell_df['geometry']]
grid_cell_df['region'] = [x['region'] for x in grid_cell_df['properties']]
grid_cell_df = grid_cell_df.drop(['type','geometry','properties'],axis=1)
grid_cell_df
```
executed in 410ms, finished 17:40:17 2022-05-16

Out[8]:

| | cell_id | coordinates | region |
|---|---|---|---|
| **0** | 0003f387-71c4-48f6-b2b0-d853bd4f0aba | [[-118.718953, 37.074192], [-118.718953, 37.08... | sierras |
| **1** | 000617d8-8c14-43e2-b708-7e3a69fe3cc3 | [[-107.076787, 37.780424], [-107.076787, 37.78... | central rockies |
| **2** | 000863e7-21e6-477d-b799-f5675c348627 | [[-119.401673, 37.024005], [-119.401673, 37.03... | other |
| **3** | 000ba8d9-d6d5-48da-84a2-1fa54951fae1 | [[-119.320824, 37.431707], [-119.320824, 37.43... | sierras |
| **4** | 00146204-d4e9-4cd8-8f86-d1ef133c5b6d | [[-118.521324, 36.657353], [-118.521324, 36.66... | sierras |
| **...** | ... | ... | ... |
| **18125** | ffdfb5a4-91a0-41a9-a4d5-501b04ef6326 | [[-118.620138, 37.117184], [-118.620138, 37.12... | sierras |
| **18126** | ffe43514-2c92-43b6-bd84-d183806aca65 | [[-123.49799, 47.901318], [-123.49799, 47.9073... | other |
| **18127** | ffeabc13-7c6f-4b63-b043-19c8f15e0345 | [[-119.644218, 37.879756], [-119.644218, 37.88... | sierras |
| **18128** | fff95195-ccc9-40b7-b302-a0d8570c86bc | [[-123.372226, 47.732416], [-123.372226, 47.73... | other |
| **18129** | fffb4d40-5947-4922-9f05-5d8b5a243d84 | [[-123.794435, 47.520516], [-123.794435, 47.52... | other |

18130 rows × 3 columns

Estimate centroids for lat_lon calculations by taking mean of points (not actual centroid because of projection and great circle distance?)

In [9]:
```
#test for one row
list(np.mean(grid_cell_df['coordinates'][1],axis=0))
```
executed in 25ms, finished 17:40:19 2022-05-16

Out[9]:
```
[-107.0803802, 37.7832636]
```

In [10]:
```
#cellid : centroid
centroids = { x['cell_id']:list(np.mean(x['coordinates'],axis=0)) for index, x in grid_cell_df.iter
centroids
```

executed in 3.62s, finished 17:40:25 2022-05-16

Out[10]:
```
{'0003f387-71c4-48f6-b2b0-d853bd4f0aba': [-118.72254619999998,
  37.077058799999996],
 '000617d8-8c14-43e2-b708-7e3a69fe3cc3': [-107.0803802, 37.7832636],
 '000863e7-21e6-477d-b799-f5675c348627': [-119.4052662, 37.0268738],
 '000ba8d9-d6d5-48da-84a2-1fa54951fae1': [-119.32441759999999, 37.4345602],
 '00146204-d4e9-4cd8-8f86-d1ef133c5b6d': [-118.5249172, 36.660235799999995],
 '0017d1c4-64cb-426d-9158-3f6521d2dd22': [-119.4322152, 37.2417204],
 '0020c632-3d5c-4509-b4ee-6b63a89bf2ff': [-118.90220920000002, 36.8545572],
 '00211c19-7ea8-4f21-a2de-1d6216186a96': [-106.9456332, 38.742496800000005],
 '0021411f-e7b5-48d7-9d36-abecbc255821': [-123.3129372, 47.9879678],
 '00226e82-e747-4f03-9c5d-3eef8ebe515e': [-120.0520532, 38.003026199999994],
 '0027a004-df14-4d66-a3e4-e987336b8814': [-106.80190259999999, 38.8824902],
 '002ccd85-65b3-4903-8725-4590d1f5611e': [-123.96870759999999, 47.6078018],
 '002cec08-f455-4e8c-8682-4d69861f4120': [-119.4232322, 37.8329446],
 '0036f966-3430-45f3-b6a2-803e678a1c2b': [-106.9546162, 39.0291872],
 '003bc010-4187-4fba-9c3c-29ca50b15a78': [-118.61474819999998,
  36.580927200000005],
 '003ccf34-6c35-4546-b8c5-b926fbe5ffbb': [-108.0775102, 38.987304800000004],
 '003cec54-9e23-4c5b-8577-fc968ba1e9d2': [-105.9485032, 39.5367682],
```

In [13]:
```
len(centroids)
```

executed in 15ms, finished 17:50:56 2022-05-16

Out[13]: 18130

## 1.5.4 Training data

### 1.5.4.1 Ground measures of Training data

Below is the ground measures from for our training data from SNOTEL and CDEC sites.

In [14]:
```
gm_train_feat = pd.read_csv("Snow-water-equivalent/data/ground_measures_train_features.csv")
gm_train_feat.rename(columns={'Unnamed: 0': "station_id"},inplace=True)
gm_train_feat
```

executed in 85ms, finished 17:41:24 2022-05-16

Out[14]:

| | station_id | 2013-01-01 | 2013-01-08 | 2013-01-15 | 2013-01-22 | 2013-01-29 | 2013-02-05 | 2013-02-12 | 2013-02-19 | 2013-02-26 | ... | 2019-05-28 | 2019-06-04 | 2019-06-11 | 2019-06-18 | 2 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | CDEC:ADM | 5.90 | 5.90 | 6.50 | 6.50 | 7.40 | 7.60 | 7.40 | 8.00 | 8.00 | ... | NaN | NaN | NaN | NaN | |
| **1** | CDEC:AGP | 17.52 | 17.54 | 17.85 | 17.39 | 18.03 | 17.70 | 17.65 | 16.66 | 17.21 | ... | NaN | NaN | NaN | NaN | |
| **2** | CDEC:ALP | 12.75 | 13.32 | 14.26 | 14.02 | 13.39 | 13.25 | 14.30 | 13.95 | 15.73 | ... | 29.52 | 20.81 | 8.71 | 0.30 | |
| **3** | CDEC:BCB | 4.30 | 4.42 | 4.62 | 4.53 | 4.67 | 4.90 | 4.90 | 5.06 | 5.11 | ... | NaN | NaN | NaN | NaN | |
| **4** | CDEC:BCH | 2.88 | 3.00 | 3.48 | 3.84 | 3.96 | 4.44 | 5.40 | 5.16 | 3.60 | ... | 0.84 | 0.60 | 0.36 | 0.36 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **695** | SNOTEL:989_ID_SNTL | 9.00 | 10.20 | 10.90 | 11.10 | 12.80 | 14.10 | 14.40 | 14.60 | 17.60 | ... | 0.00 | 0.00 | 0.00 | 0.00 | |
| **696** | SNOTEL:990_WA_SNTL | 27.50 | 29.10 | 31.50 | 31.90 | 33.40 | 33.90 | 35.40 | 36.50 | 38.80 | ... | 6.00 | 0.10 | 0.00 | 0.00 | |
| **697** | SNOTEL:992_UT_SNTL | 4.10 | 4.10 | 4.40 | 4.50 | 4.80 | 5.10 | 5.20 | 5.30 | 6.10 | ... | 0.00 | 0.00 | 0.00 | 0.00 | |
| **698** | SNOTEL:998_WA_SNTL | 48.40 | 55.50 | 61.50 | 62.20 | 67.50 | 70.10 | 72.90 | 77.00 | 83.30 | ... | 53.60 | 36.10 | 31.30 | 8.50 | |
| **699** | SNOTEL:999_WA_SNTL | 33.10 | 37.50 | 40.80 | 42.50 | 47.50 | 51.00 | 53.90 | 56.40 | 64.50 | ... | 1.30 | 0.00 | 0.00 | 0.00 | |

700 rows × 214 columns

### 1.5.4.2 Train Labels

In [15]:
```
y_train = pd.read_csv("Snow-water-equivalent/data/train_labels.csv")
y_train
```
executed in 219ms, finished 17:41:27 2022-05-16

Out[15]:

| | cell_id | 2013-01-01 | 2013-01-08 | 2013-01-15 | 2013-01-22 | 2013-01-29 | 2013-02-05 | 2013-02-12 | 2013-02-19 | 2013-02-26 | ... | 2019-06-13 | 2019-06-14 | 2019-06-18 | 2019-06-24 | 2019-06-25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0003f387-71c4-48f6-b2b0-d853bd4f0aba | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **1** | 000617d8-8c14-43e2-b708-7e3a69fe3cc3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |
| **2** | 000ba8d9-d6d5-48da-84a2-1fa54951fae1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 0.0 | NaN | NaN | NaN |
| **3** | 0017d1c4-64cb-426d-9158-3f6521d2dd22 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |

In [16]:
```
y_train_melt = y_train.melt(id_vars=["cell_id"]).dropna()
y_train_melt.rename(columns={"variable":'date'},inplace=True)
y_train_grouped = y_train_melt.groupby('cell_id')

#grab date/cell_id combos
#cell_id : [dates]
dates = {key:(list(group['date'])) for key, group in y_train_grouped}
```
executed in 2.59s, finished 17:41:41 2022-05-16

In [17]:
```
len(y_train_melt)
```
executed in 13ms, finished 17:41:43 2022-05-16

Out[17]: 91490

Even after dropping nulls, we still have 91490 training labels to work use as ground truth. Now let's compile overlapping tiles.

In [ ]:

```
# create dictionary tiles (date, lat, lon) : [cell_ids]
counter = 0
tiles_training = defaultdict(list)
for cell, date_list in dates.items():
    for date in date_list:

        modis_tile = lat_lon_to_modis_tile(centroids[cell][1], centroids[cell][0])
        tiles_training[(datetime.fromisoformat(date),) + modis_tile].append(cell)
        counter += 1
print("total squares:", counter)
print("Now saving tiles dictionary as .json...")

# #Save this counter and tiles dictionary as json
# to_save = {'Counter':counter,'tiles':tiles_training}
# output_path = f"{bucket_name}/tiles_training.json"
# with fs.open(output_path, 'wb') as handle:
#   json.dump(handle,to_save)

# print("Saving completed")
```

executed in 1m 41.2s, finished 17:47:53 2022-05-16

```
total squares: 91490
Now saving tiles dictionary as .json...

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-39-fbb090d18d57> in <module>()
     15 output_path = f"{bucket_name}/tiles_training.json"
     16 with fs.open(output_path, 'wb') as handle:
---> 17   json.dump(handle,to_save)
     18
     19 print("Saving completed")

NameError: name 'json' is not defined
```

#### 1.5.4.3 Load Modis Terra Training images (morning)

In [ ]:

```
## Testing cell
# import gcsfs

# project_name = 'Snow-Water-Equivalent'
# bucket_name = 'modis-swe'

# fs = gcsfs.GCSFileSystem(projectstring=project_id)
# fs.ls(bucket_name)


# test_np = np.random.rand(3,4,2)
# output_path = f"{bucket_name}/test.npy"
# with fs.open(output_path, 'wb') as handle:
#   np.save(handle,test_np)


# path_ids = "{}/test.pkl".format(bucket_name)
# with fs.open(path_ids, 'wb') as handle:
#     pickle.dump(dates, handle)
```

In [ ]:

```
product = 'MOD10A1' # Terra -morning
# product = 'MCD43A4'

#initialize empty array
dataset_t = np.empty((counter, 7, 21, 21)) #(image, band, row, column)

# download dataset
cell_ids, dataset_t = images_downloader(tiles_training, centroids, dataset_t, product)

#####save output#####
output_path = f"{bucket_name}/ModisSnowImagesT.npy"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_t)

path_ids = f"{bucket_name}/cell_snow_idsT.pkl"
with fs.open(path_ids, 'wb') as handle:
  pickle.dump(cell_ids, handle)
```

executed in 1h 21m 8s, finished 19:22:22 2022-05-16

```
 0
Found 1 matching file(s):
MOD10A1/08/05/2018116/MOD10A1.A2018116.h08v05.006.2018118031402.hdf

 1468
Found 1 matching file(s):
MOD10A1/08/05/2019075/MOD10A1.A2019075.h08v05.006.2019077031831.hdf

 2937
Found 1 matching file(s):
MOD10A1/08/05/2019108/MOD10A1.A2019108.h08v05.006.2019110033147.hdf

 4406
Found 1 matching file(s):
MOD10A1/08/05/2019118/MOD10A1.A2019118.h08v05.006.2019120033847.hdf

 5875
Found 1 matching file(s):
```

In [ ]:

```
cell_ids
```

Out[84]:

```
[('0003f387-71c4-48f6-b2b0-d853bd4f0aba', '2018116'),
 ('0020c632-3d5c-4509-b4ee-6b63a89bf2ff', '2018116'),
 ('00559e5b-310a-4514-8123-2a3074828d74', '2018116'),
 ('00748c91-2e34-44ea-ab63-51c10bf77497', '2018116'),
 ('0095cd70-ad71-485a-a670-64fc5604db98', '2018116'),
 ('00b78996-1639-40d2-a943-2126ec4ef2e1', '2018116'),
 ('00c0b91c-9aa7-4d9b-a44f-abb91acd49d2', '2018116'),
 ('0158d4f1-037e-421a-aeed-58db7a086512', '2018116'),
 ('01724a51-925e-4d33-879f-837f436383b4', '2018116'),
 ('01867838-b8c1-45b2-bf01-a262d18c95ec', '2018116'),
 ('01909e72-97ba-4c53-85d6-15d4410a6b57', '2018116'),
 ('01a0f55a-2c5e-4539-8c74-7d315031b51f', '2018116'),
 ('01a191f1-2d34-4886-a72f-53e3c24f7356', '2018116'),
 ('01b0d56b-f1d2-436d-a0ae-bc605eede64a', '2018116'),
 ('01ccdf49-83ed-470f-8d82-3c89973bbb8c', '2018116'),
 ('01ee5268-f209-420e-a888-59ac08261065', '2018116'),
 ('01f9c9c0-d9a6-416b-96fe-2c43d250517b', '2018116'),
 ('020d17b2-4720-4c0b-b876-20aa707bf4c2', '2018116'),
 ('0266d61d-71e0-4ebc-895d-a42a872a90b9', '2018116'),
```

In [ ]:

```
len(cell_ids)
```

executed in 13ms, finished 11:32:54 2022-05-17

Out[43]: 91490

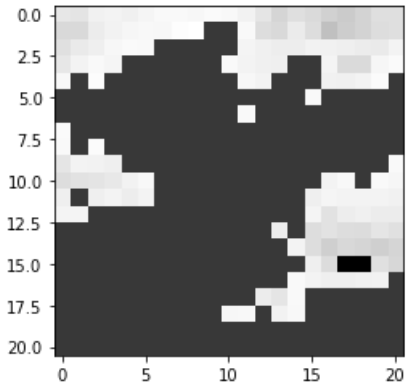
In [ ]:
```
dataset_t.shape
```

Out[85]:
```
(91490, 7, 21, 21)
```

In [ ]:
```
plt.imshow(dataset_t[1000,4,:,:],cmap='Greys')
```

executed in 259ms, finished 11:28:09 2022-05-17

Out[44]:
```
<matplotlib.image.AxesImage at 0x7f818abf6ad0>
```




In [ ]:
```
dataset_t[1000,4,:,:].mean()
```

executed in 17ms, finished 11:25:04 2022-05-17

Out[45]:
```
93.70521541950113
```

In [ ]:
```
dataset_t[1000,4,0,:]
```

executed in 17ms, finished 11:28:39 2022-05-17

Out[46]:
```
array([36., 38., 28., 26., 29., 23., 19., 18., 21., 17., 20., 27., 37.,
       49., 42., 45., 51., 55., 50., 42., 42.])
```

#### 1.5.4.4 Load MODIS Aqua training images (afternoon)

In [ ]:

```
product = 'MYD10A1' # Aqua -afternoon

#initialize empty array
dataset_a = np.empty((counter, 7, 21, 21)) #(image, band, row, column)

# download dataset
cell_ids, dataset_a = images_downloader(tiles_training, centroids, dataset_a, product)


#####save output#####
output_path = f"{bucket_name}/ModisSnowImagesA.npy"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_a)

path_ids = f"{bucket_name}/cell_snow_idsA.pkl"
with fs.open(path_ids, 'wb') as handle:
  pickle.dump(cell_ids, handle)
```

executed in 1h 19m 50s, finished 21:34:34 2022-05-16

```
 0
Found 1 matching file(s):
MYD10A1/08/05/2018116/MYD10A1.A2018116.h08v05.006.2018118031436.hdf

 1468
Found 1 matching file(s):
MYD10A1/08/05/2019075/MYD10A1.A2019075.h08v05.006.2019079193507.hdf

 2937
Found 1 matching file(s):
MYD10A1/08/05/2019108/MYD10A1.A2019108.h08v05.006.2019110032220.hdf

 4406
Found 1 matching file(s):
MYD10A1/08/05/2019118/MYD10A1.A2019118.h08v05.006.2019120031417.hdf

 5875
Found 1 matching file(s):
```

In [ ]:

```
#####save output#####
output_path = f"{bucket_name}/ModisSnowImagesA.npy"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_a)

path_ids = f"{bucket_name}/cell_snow_idsA.pkl"
with fs.open(path_ids, 'wb') as handle:
  pickle.dump(cell_ids, handle)
```

## 1.5.5 Testing Data

### 1.5.5.1 Ground measures of testing data

The dataset below is the ground truth data for our testing data. This will be reserved to test our model performance on unseen data.

In [18]:
```
gm_test_feat = pd.read_csv("Snow-water-equivalent/data/ground_measures_test_features.csv")
gm_test_feat.rename(columns={'Unnamed: 0': "station_id"},inplace=True)
gm_test_feat
```

executed in 86ms, finished 21:54:00 2022-05-16

Out[18]:

| | station_id | 2020-01-07 | 2020-01-14 | 2020-01-21 | 2020-01-28 | 2020-02-04 | 2020-02-11 | 2020-02-18 | 2020-02-25 | 2020-03-03 | ... | 2021-04-27 | 2021-05-04 | 2021-05-11 | 2021-05-18 | 2 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | CDEC:ADM | 4.50 | 5.50 | 7.30 | 8.30 | 8.10 | 8.20 | 9.30 | 8.50 | 7.90 | ... | 0.30 | NaN | NaN | NaN | |
| **1** | CDEC:AGP | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | |
| **2** | CDEC:ALP | 12.72 | 13.78 | 17.12 | 18.07 | 18.17 | 18.38 | 17.71 | 16.05 | 14.62 | ... | 3.34 | 0.31 | 0.02 | 0.00 | |
| **3** | CDEC:BCB | 12.20 | 12.20 | 13.30 | 13.35 | 12.85 | 12.72 | 12.72 | 12.80 | 13.16 | ... | 15.44 | 11.94 | 5.91 | 1.10 | |
| **4** | CDEC:BCH | 6.60 | 5.76 | 5.16 | 7.68 | 4.68 | 1.32 | 0.84 | 0.84 | 0.24 | ... | 0.12 | 0.24 | 0.24 | 0.24 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **695** | SNOTEL:989_ID_SNTL | 6.80 | 12.50 | 13.10 | 14.40 | 16.30 | 19.10 | 19.80 | 19.80 | 19.70 | ... | 6.10 | 0.00 | 0.00 | 0.00 | |
| **696** | SNOTEL:990_WA_SNTL | 13.80 | 17.00 | 20.30 | 24.90 | 26.70 | 29.40 | 29.80 | 31.10 | 32.60 | ... | 41.20 | 38.10 | 35.90 | 32.10 | 2 |
| **697** | SNOTEL:992_UT_SNTL | 4.40 | 5.00 | 5.80 | 6.20 | 6.30 | 6.80 | 7.20 | 7.40 | 7.80 | ... | 0.00 | 0.00 | 0.00 | 0.00 | |
| **698** | SNOTEL:998_WA_SNTL | 37.90 | 47.00 | 51.80 | 61.90 | 69.00 | 73.40 | 77.30 | 81.40 | 83.60 | ... | 96.00 | 95.10 | 95.50 | 94.20 | 9 |
| **699** | SNOTEL:999_WA_SNTL | 17.70 | 23.60 | 27.90 | 32.00 | 33.70 | 39.90 | 44.20 | 48.10 | 52.60 | ... | 68.10 | 64.30 | 64.00 | 57.50 | 5 |

700 rows × 58 columns

#### 1.5.5.2 Testing labels

In [19]:
```
y_test = pd.read_csv("Snow-water-equivalent/data/labels_2020_2021.csv")
y_test
```

executed in 131ms, finished 21:54:07 2022-05-16

Out[19]:

| | cell_id | 2020-01-07 | 2020-01-14 | 2020-01-21 | 2020-01-28 | 2020-02-04 | 2020-02-11 | 2020-02-18 | 2020-02-25 | 2020-03-03 | ... | 2021-04-27 | 2021-05-04 | 2021-05-11 | 2021-05-18 | 2021-05-25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 000863e7-21e6-477d-b799-f5675c348627 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 0.0 | NaN | NaN | NaN |
| **1** | 000ba8d9-d6d5-48da-84a2-1fa54951fae1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 0.0 | NaN | NaN | NaN |
| **2** | 00146204-d4e9-4cd8-8f86-d1ef133c5b6d | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | 0.5 | NaN | NaN | NaN |
| **3** | 00211c19-7ea8-4f21-a2de-1d6216186a96 | NaN | NaN | NaN | NaN | NaN | 4.6 | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN |

In [20]:
```
y_test_melt = y_test.melt(id_vars=["cell_id"]).dropna()
y_test_melt.rename(columns={"variable":'date'},inplace=True)
len(y_test_melt)
```

Out[20]: 45241

In [21]:

```
y_test_melt = y_test.melt(id_vars=["cell_id"]).dropna()
y_test_melt.rename(columns={"variable":'date'},inplace=True)
y_test_grouped = y_test_melt.groupby('cell_id')

#grab date/cell_id combos
#cell_id : [dates]
dates_sub = {key:(list(group['date'])) for key, group in y_test_grouped}
dates_sub
```

executed in 1.76s, finished 22:05:42 2022-05-16

Out[21]:

```
{'000863e7-21e6-477d-b799-f5675c348627': ['2020-05-26',
  '2020-06-09',
  '2021-02-23',
  '2021-03-30',
  '2021-05-04'],
 '000ba8d9-d6d5-48da-84a2-1fa54951fae1': ['2020-04-14',
  '2020-05-05',
  '2020-05-26',
  '2020-06-09',
  '2021-02-23',
  '2021-03-30',
  '2021-05-04'],
 '00146204-d4e9-4cd8-8f86-d1ef133c5b6d': ['2020-04-14',
  '2020-05-05',
  '2020-05-26',
  '2021-03-30',
  '2021-05-04'],
 '00211c19-7ea8-4f21-a2de-1d6216186a96': ['2020-02-11'],
 '00226e82-e747-4f03-9c5d-3eef8ebe515e': ['2021-02-23', '2021-04-27'],
```

In [ ]:

```
# create dictionary tiles_test (date, lat, lon) : [cell_ids]
counter_test = 0
tiles_test = defaultdict(list)
for cell, date_list in dates_sub.items():
    for date in date_list:

        modis_tile = lat_lon_to_modis_tile(centroids[cell][1], centroids[cell][0])
        tiles_test[(datetime.fromisoformat(date),) + modis_tile].append(cell)
        counter_test += 1
print("total squares:", counter_test)

# print("Now saving tiles dictionary as .json...")

# #Save this counter and tiles dictionary as json
# to_save = {'Counter':counter_test,'tiles':tiles_test}
# output_path = f"{bucket_name}/tiles_test.json"
# with fs.open(output_path, 'wb') as handle:
#   json.dump(handle,to_save)

# print("Saving completed")
```

executed in 5m 8s, finished 22:11:05 2022-05-16

```
total squares: 45241
```

#### 1.5.5.3 Load MODIS Terra Test images (Morning)

In [45]:

```
product = 'MOD10A1'

#initialize empty array
dataset_test_t = np.empty((counter_test, 7, 21, 21)) #(image, band, row, column)

# download dataset
cell_ids_sub, dataset_test_t = images_downloader(tiles_test, centroids, dataset_test_t, product)

#####save output#####
output_path = f"{bucket_name}/ModisSnowImages_testT.npy"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_test_t)

path_ids = f"{bucket_name}/cell_snow_ids_testT.pkl"
with fs.open(path_ids, 'wb') as handle:
  pickle.dump(cell_ids_sub, handle)
```

executed in 40ms, finished 22:53:44 2022-05-16

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-45-ea2a487e068f> in <module>()
      2
      3 #initialize empty array
----> 4 dataset_test_t = np.empty((counter_test, 7, 21, 21)) #(image, band, row, column)
      5
      6 # download dataset

NameError: name 'counter_test' is not defined
```

#### 1.5.5.4 Load MODIS Aqua Test images (Afternoon)

In [ ]:

```
product = 'MYD10A1'

#initialize empty array
dataset_test_a = np.empty((counter_test, 7, 21, 21)) #(image, band, row, column)

# download dataset
cell_ids_sub, dataset_test_a = images_downloader(tiles_test, centroids, dataset_test_a, product)

#####save output#####
output_path = f"{bucket_name}/ModisSnowImages_testA.npy"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_test_a)

path_ids = f"{bucket_name}/cell_snow_ids_testA.pkl"
with fs.open(path_ids, 'wb') as handle:
  pickle.dump(cell_ids_sub, handle)
```

executed in 40ms, finished 22:53:49 2022-05-16

```
 0
Found 1 matching file(s):
MYD10A1/08/05/2020147/MYD10A1.A2020147.h08v05.006.2020149033856.hdf

 3222
Found 1 matching file(s):
MYD10A1/08/05/2020161/MYD10A1.A2020161.h08v05.006.2020163035410.hdf

 5007
Found 1 matching file(s):
MYD10A1/08/05/2021054/MYD10A1.A2021054.h08v05.006.2021056034727.hdf

 8040
Found 1 matching file(s):
MYD10A1/08/05/2021089/MYD10A1.A2021089.h08v05.006.2021091042441.hdf

 11593
Found 1 matching file(s):
```

# 2 Corpenicus DEM

### 2.0.1 Training data

In [23]:

```
centroids['1b15d267-cd94-4c73-a938-cbe89c079dbb']
```

Out[23]: [-106.8108862, 38.889482799999996]

In [16]:

```
def download_DEM(coords):
  client = Client.open(
  "https://planetarycomputer.microsoft.com/api/stac/v1",
  ignore_conformance=True,
  )
  #Perform a STAC API query against the finer resolution cop-dem-glo-30.
  coords = coords
  print(f'Searching DEM at {coords}')
  search = client.search(
      collections=["cop-dem-glo-30"],
      intersects={"type": "Point", "coordinates": coords},
  )
  items = list(search.get_items())
  print(f"Returned {len(items)} items")
  signed_asset = planetary_computer.sign(items[0].assets["data"])
  # url = items[0].assets["data"].href
  # blob_name = str(items[0].assets["data"].href).split('/')[5]

  print('Read & return DEM image as 3D array')

  # # Method 1
  # # Download to a temporary file
  # filename = os.path.join(temp_dir,blob_name.replace('/','_'))
  # if not os.path.isfile(filename):
  #     wget.download(url,filename)

  # # Open raster file

  # with rasterio.open(filename, 'r') as ds:
  #   arr = ds.read(1)  # read all raster values

  # Method 2
  # Open raster file
  with rasterio.Env(GDAL_DISABLE_READDIR_ON_OPEN=True, CPL_VSIL_CURL_ALLOWED_EXTENSIONS="tif"):
    with rasterio.open(signed_asset.href, 'r') as ds:
      arr = ds.read(1)  # read all raster values
  arr_mean = arr.mean()
  arr_var = arr.var()

  ## Method 3
  # with rasterio.open(signed_asset.href, 'r') as ds:
  #   arr = ds.read()  # read all raster values

  #Return array of our DEM image
  return arr_mean,arr_var
```

In [17]:

```
mean,var = download_DEM(centroids['011cfd7c-58ad-4d0d-b035-6437788b033a'])
print(mean)
print(var)
```

```
Searching DEM at [-119.2076366, 36.775448999999995]
Returned 1 items
Read & return DEM image as 3D array
212.5251
95698.1
```

In [ ]:

```
# dem_np = np.empty((len(centroids), 1, 3600, 3600)) #(image, band, row, column)

dem_dict={}

i=0
for cell_id, coords in centroids.items():
  print(f'Grabbing DEM image no. {i} for cell grid {cell_id}')
  print('--'*10)
  mean,var = download_DEM(coords)
  dem_dict[cell_id] = [mean,var]
  i+=1
```

```
Grabbing DEM image no. 0 for cell grid 0003f387-71c4-48f6-b2b0-d853bd4f0aba
--------------------
Searching DEM at [-118.72254619999998, 37.077058799999996]
Returned 1 items
Read & return DEM image as 3D array
Grabbing DEM image no. 1 for cell grid 000617d8-8c14-43e2-b708-7e3a69fe3cc3
--------------------
Searching DEM at [-107.0803802, 37.7832636]
Returned 1 items
Read & return DEM image as 3D array
Grabbing DEM image no. 2 for cell grid 000863e7-21e6-477d-b799-f5675c348627
--------------------
Searching DEM at [-119.4052662, 37.0268738]
Returned 1 items
Read & return DEM image as 3D array
Grabbing DEM image no. 3 for cell grid 000ba8d9-d6d5-48da-84a2-1fa54951fae1
--------------------
Searching DEM at [-119.32441759999999, 37.4345602]
Returned 1 items
```

In [66]:

```
centroids
```

Out[66]:

```
{'0003f387-71c4-48f6-b2b0-d853bd4f0aba': [-118.72254619999998,
  37.077058799999996],
 '000617d8-8c14-43e2-b708-7e3a69fe3cc3': [-107.0803802, 37.7832636],
 '000863e7-21e6-477d-b799-f5675c348627': [-119.4052662, 37.0268738],
 '000ba8d9-d6d5-48da-84a2-1fa54951fae1': [-119.32441759999999, 37.4345602],
 '00146204-d4e9-4cd8-8f86-d1ef133c5b6d': [-118.5249172, 36.660235799999995],
 '0017d1c4-64cb-426d-9158-3f6521d2dd22': [-119.4322152, 37.2417204],
 '0020c632-3d5c-4509-b4ee-6b63a89bf2ff': [-118.90220920000002, 36.8545572],
 '00211c19-7ea8-4f21-a2de-1d6216186a96': [-106.9456332, 38.742496800000005],
 '0021411f-e7b5-48d7-9d36-abecbc255821': [-123.3129372, 47.9879678],
 '00226e82-e747-4f03-9c5d-3eef8ebe515e': [-120.0520532, 38.003026199999994],
 '0027a004-df14-4d66-a3e4-e987336b8814': [-106.80190259999999, 38.8824902],
 '002ccd85-65b3-4903-8725-4590d1f5611e': [-123.96870759999999, 47.6078018],
 '002cec08-f455-4e8c-8682-4d69861f4120': [-119.4232322, 37.8329446],
 '0036f966-3430-45f3-b6a2-803e678a1c2b': [-106.9546162, 39.0291872],
 '003bc010-4187-4fba-9c3c-29ca50b15a78': [-118.61474819999998,
  36.580927200000005],
 '003ccf34-6c35-4546-b8c5-b926fbe5ffbb': [-108.0775102, 38.987304800000004],
 '003cec54-9e23-4c5b-8577-fc968ba1e9d2': [-105.9485032, 39.5367682],
```

In [2]:

```
dem_np
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-ab733d14ee27> in <module>()
----> 1 dem_np

NameError: name 'dem_np' is not defined
```

In [44]:

```
len(centroids)
```

Out[44]: 18130

In [33]:

```
import csv
account_name = 'elevationeuwest'
container_name = 'copernicus-dem'
account_url = 'https://' + account_name + '.blob.core.windows.net'
blob_root = account_url + '/' + container_name + '/'

cdem_content_extension = '.tif'

# List of blobs and their lat/lon ranges, as left/bottom/right/top
#
# One for the 30m DEM...
cdem_30_extents_url = account_url + '/' + container_name + '/' + 'index/copernicus-dem-30-bounds.cs

# ...and one for the 90m DEM.
cdem_90_extents_url = account_url + '/' + container_name + '/' + 'index/copernicus-dem-90-bounds.cs

# Load the lat/lon table for the 30m DEM
temp_dir = os.path.join(tempfile.gettempdir(),'copernicus-dem')
os.makedirs(temp_dir,exist_ok=True)
fn = os.path.join(temp_dir,cdem_30_extents_url.split('/')[-1])
if not os.path.isfile(fn):
  wget.download(cdem_30_extents_url, fn, bar=None)

# Load this file into a table, where each row is (name,left,bottom,right,top)
with open(fn, newline='') as f:
    reader = csv.reader(f)
    cdem_30_tile_extents = list(reader)

# Remove the header and convert str to float
cdem_30_tile_extents = cdem_30_tile_extents[1:]
for i in range(0,len(cdem_30_tile_extents)):
    for j in range(1,5):
        cdem_30_tile_extents[i][j] = float(cdem_30_tile_extents[i][j])
```

In [39]:

```
def lat_lon_to_copernicus_dem_tile(lat,lon,extents):
    """
    Get the tile path corresponding to the given lat/lon from the table in "extents"
    """

    i_tile = None
    for i in range(0,len(extents)):
        found_matching_tile = \
        lat >= extents[i][2] \
        and lon <= extents[i][3] \
        and lat <= extents[i][4] \
        and lon >= extents[i][1]
        if found_matching_tile:
            i_tile = i
            break

    assert i_tile is not None, 'Could not find tile for coordinate {},{}'.format(lat,lon)
    blob_name = extents[i_tile][0]
    return blob_name
```

In [68]:
```
# Interesting places for looking at DEM data
everest = [27.9881,86.9250]
seattle = [47.6062,-122.3321]
grand_canyon = [36.101690, -112.107676]
mount_fuji = [35.3606, 138.7274]
mont_blanc = [45.832778, 6.865000]
invalid = [-15.1,41]

tile_of_interest = everest

tile_name = lat_lon_to_copernicus_dem_tile(centroids['1b15d267-cd94-4c73-a938-cbe89c079dbb'][0],cen
url = blob_root + tile_name
print('Plotting tile at:\n{}'.format(url))
rds = rasterio.open(url)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-68-e7c8a9352d54> in <module>()
      9 tile_of_interest = everest
     10
---> 11 tile_name = lat_lon_to_copernicus_dem_tile(centroids['1b15d267-cd94-4c73-a938-cbe89c079dbb']
[0],centroids['1b15d267-cd94-4c73-a938-cbe89c079dbb'][1],cdem_30_tile_extents)
     12 url = blob_root + tile_name
     13 print('Plotting tile at:\n{}'.format(url))

<ipython-input-39-1b3fc831f1cf> in lat_lon_to_copernicus_dem_tile(lat, lon, extents)
     11             break
     12
---> 13     assert i_tile is not None, 'Could not find tile for coordinate {},{}'.format(lat,lon)
     14     blob_name = extents[i_tile][0]
     15     return blob_name

AssertionError: Could not find tile for coordinate -106.8108862,38.889482799999996
```

In [53]:
```
temp_dir
```

Out[53]:
```
'/tmp/copernicus-dem'
```

In [63]:
```
url
```

Out[63]:
```
'https://elevationeuwest.blob.core.windows.net/copernicus-dem/COP30_hh/Copernicus_DSM_COG_10_N27_00_
E086_00_DEM.tif'
```

In [64]:
```
blob_name='Copernicus_DSM_COG_10_N27_00_E086_00_DEM.tif'
```

In [58]:
```
filename
```

Out[58]:
```
'/tmp/copernicus-dem/Copernicus_DSM_COG_10_N35_00_E138_00_DEM.tif'
```

In [65]:
```
# Download to a temporary file
filename = os.path.join(temp_dir,blob_name.replace('/','_'))
if not os.path.isfile(filename):
    wget.download(url,filename)

rds = rasterio.open(filename)
```

In [61]:
```
# Download to a temporary file
filename = os.path.join(temp_dir,blob_name.replace('/','_'))
if not os.path.isfile(filename):
    wget.download(url,filename)

rds = rxr.open_rasterio(filename)
```

In [59]:
```
rds
```

Out[59]:
```
<xarray.DataArray (band: 1, y: 3600, x: 3600)>
[12960000 values with dtype=float32]
Coordinates:
  * band         (band) int64 1
  * x            (x) float64 138.0 138.0 138.0 138.0 ... 139.0 139.0 139.0 139.0
  * y            (y) float64 36.0 36.0 36.0 36.0 36.0 ... 35.0 35.0 35.0 35.0
    spatial_ref  int64 0
Attributes:
    scale_factor:  1.0
    add_offset:    0.0
```

In [ ]:
```
output_path = f"{bucket_name}/{blob_name}"
with fs.open(output_path, 'wb') as handle:
  np.save(handle,dataset_test_a)
```

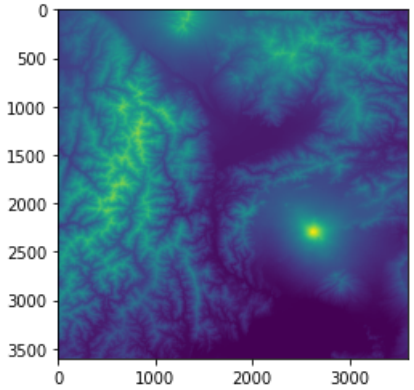
In [42]:
```
src = rasterio.open(url)
d = src.read(1)
d
```

Out[42]:
```
array([[968.17426, 968.1576 , 964.515  , ..., 485.64798, 479.80908,
        466.68808],
       [984.7898 , 977.51447, 969.6498 , ..., 486.8953 , 485.5122 ,
        472.66592],
       [982.55286, 969.0126 , 957.158  , ..., 473.1398 , 470.76645,
        464.55292],
       ...,
       [522.9197 , 519.202  , 517.8268 , ..., 235.85004, 236.10023,
        236.97673],
       [518.0397 , 504.67368, 496.105  , ..., 226.95879, 222.91801,
        220.38797],
       [513.45306, 500.36423, 488.86935, ..., 216.82407, 220.10077,
        224.50966]], dtype=float32)
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

In [43]:
```
plt.imshow(d)
```

Out[43]:
```
<matplotlib.image.AxesImage at 0x7f7087a40a10>
```



In [28]:
```
client = Client.open(
    "https://planetarycomputer.microsoft.com/api/stac/v1",
    ignore_conformance=True,
)
```

In [29]:
```
cell1 = centroids['0ccb87f5-e4fd-407d-8ef6-1daa355abeba']
search = client.search(
    collections=["cop-dem-glo-30"],
    intersects={"type": "Point", "coordinates": cell1},
)
items = list(search.get_items())
print(f"Returned {len(items)} items")
```

```
Returned 1 items
```

In [78]:
```
string = str(items[0].assets["data"].href)
string = string.split('/')[5]
string
```

Out[78]:
```
'Copernicus_DSM_COG_10_N37_00_W108_00_DEM.tif'
```

In [77]:

Out[77]:
```
'Copernicus_DSM_COG_10_N37_00_W108_00_DEM.tif'
```

In [31]:
```
signed_asset = planetary_computer.sign(items[0].assets["data"])
signed_asset
```

Out[31]:
```
<Asset href=https://elevationeuwest.blob.core.windows.net/copernicus-dem/COP30_hh/Copernicus_DSM_COG
_10_N37_00_W108_00_DEM.tif?st=2022-05-18T17%3A28%3A56Z&se=2022-05-19T18%3A13%3A56Z&sp=rl&sv=2020-06-
12&sr=c&skoid=c85c15d6-d1ae-42d4-af60-e2ca0f81359b&sktid=72f988bf-86f1-41af-91ab-2d7cd011db47&skt=20
22-05-18T07%3A20%3A52Z&ske=2022-05-25T07%3A20%3A52Z&sks=b&skv=2020-06-12&sig=q3XqdEldlOtAFjAY%2BsQrt
iAR756oRuueipLnXBpKVN8%3D>
```

In [39]:
```
signed_asset = planetary_computer.sign(items[0].assets["data"])

with rasterio.open(signed_asset.href, 'r') as ds:
    arr2 = ds.read()  # read all raster values

print(arr.shape)  # this is a 3D numpy array, with dimensions [band, row, col]
```

```
(1, 3600, 3600)
```

In [41]:
```
numpy_experiment = np.empty((2, 1, 3600, 3600)) #(image, band, row, column)
numpy_experiment[0] = arr2
numpy_experiment[0].shape
```

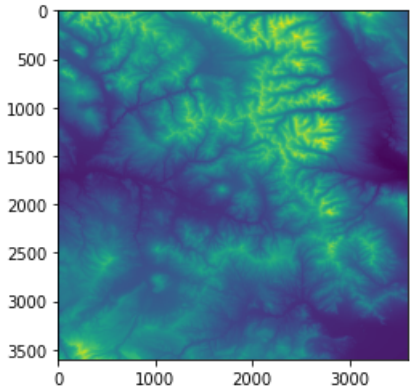
Out[41]:
```
(1, 3600, 3600)
```

In [43]:
```
numpy_experiment[0].mean()
```

Out[43]:
```
2856.887177975021
```

In [27]:
```
plt.imshow(arr[0,:,:])
```

Out[27]: <matplotlib.image.AxesImage at 0x7f7089814ed0>




In [ ]:
```
arr[0,:,:].mean()
```

Out[88]: 2364.2617

### 2.0.2 Test data

In [ ]:
```

```