0.1 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).

0.2 Note:

Please ensure that you are running this notebook with the correct conda environment (../geo_env.yml) provided in the project's repo. Run the following lines in your command prompt to activate the environment using the .yml file provided. Ensure that you are in the correct directory in your command prompt before executing the lines below.

```
conda env create -f geo_env.yml
conda activate geo_env
```

Before we begin, let's first import our packages to run this notebook.

0.3 Import packages

In [1]:

```
import pandas as pd
import geojson as gsn
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#import spatial analysis packages
import geopandas as gpd
import contextily as cx
from shapely.geometry import Point
import geoplot.crs as gcrs
import geoplot as gplt
import pyproj
from shapely.geometry import Polygon
import haversine as hs
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots

import pickle
from datetime import date
```

executed in 8.16s, finished 08:04:17 2022-06-01

1 Data Preprocessing

1.1 Ground Measures Data

1.1.1 Load Ground Measure data

Ground measure data provides us with the measure snow water equivalent (SWE) at SNOTEC and CDEC sites across western U.S.

In [2]:
```
gm_md = pd.read_csv("../data/ground_measures_metadata.csv")
gm_md = gpd.GeoDataFrame(gm_md, geometry=gpd.points_from_xy(gm_md.longitude, gm_md.latitude))
gm_md
```

executed in 106ms, finished 08:04:18 2022-06-01

Out[2]:

	station_id	name	elevation_m	latitude	longitude	state	geometry
0	CDEC:ADM	Adin Mountain	1889.760000	41.237000	-120.792000	California	POINT (-120.79200 41.23700)
1	CDEC:AGP	Agnew Pass	2880.360000	37.726631	-119.141731	California	POINT (-119.14173 37.72663)
2	CDEC:ALP	Alpha (Smud)	2316.480000	38.804192	-120.215652	California	POINT (-120.21565 38.80419)
3	CDEC:BCB	Blackcap Basin	3139.440000	37.066685	-118.773010	California	POINT (-118.77301 37.06668)
4	CDEC:BCH	Beach Meadows	2331.720000	36.126095	-118.293457	California	POINT (-118.29346 36.12609)
...	...	...	...	...	...	...	...
759	SNOTEL:994_WA_SNTL	Epa Quinault Open	91.440002	46.483330	-123.966667	Washington	POINT (-123.96667 46.48333)
760	SNOTEL:995_WA_SNTL	Epa Quinault Can	91.440002	46.483330	-123.966667	Washington	POINT (-123.96667 46.48333)
761	SNOTEL:996_WA_SNTL	NWA Heather Mdws	1267.968018	48.849998	-121.666672	Washington	POINT (-121.66667 48.85000)
762	SNOTEL:998_WA_SNTL	Easy Pass	1606.296021	48.859329	-121.438950	Washington	POINT (-121.43895 48.85933)
763	SNOTEL:999_WA_SNTL	Marten Ridge	1072.895996	48.762920	-121.698227	Washington	POINT (-121.69823 48.76292)

764 rows × 7 columns

In [3]:
```
gm_md.info()
```

executed in 27ms, finished 08:04:18 2022-06-01

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 764 entries, 0 to 763
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   station_id   764 non-null    object  
 1   name         764 non-null    object  
 2   elevation_m  764 non-null    float64 
 3   latitude     764 non-null    float64 
 4   longitude    764 non-null    float64 
 5   state        764 non-null    object  
 6   geometry     764 non-null    geometry
dtypes: float64(3), geometry(1), object(3)
memory usage: 41.9+ KB
```

In [4]:
```
gm_md= gm_md.set_crs('epsg:4326')
```

executed in 25ms, finished 08:04:18 2022-06-01

1.2 Cell grids

1.2.1 Load cell grid data

This dataset (%22../data/grid_cells.geojson%22) provides us spatial information about the grid cells that we're modelling for.

In [5]:
```
path = "../data/grid_cells.geojson"
with open(path) as f:
    gj = gsn.load(f)
print(len(gj['features']))
# gj['features']
```

executed in 1.22s, finished 08:04:19 2022-06-01

```
18130
```

In [6]:
```
gj.keys()
```

executed in 15ms, finished 08:04:19 2022-06-01

Out[6]:
```
dict_keys(['type', 'crs', 'features'])
```

In [7]:
```
gj['features'][1]
```

executed in 29ms, finished 08:04:19 2022-06-01

Out[7]:
```
{"geometry": {"coordinates": [[[-107.076787, 37.780424], [-107.076787, 37.787523], [-107.08577, 37.787523], [-107.08577, 37.780424], [-107.076787, 37.780424]]], "type": "Polygon"}, "properties": {"cell_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 166ms, finished 08:04:19 2022-06-01

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

In [9]:
```
#Make new columns for latitude and longitude of the center of the grid cell

grid_cell_df['centr_lat'] = [list(np.mean(x['coordinates'],axis=0))[0] for index, x in grid_cell_df
grid_cell_df['centr_lon'] = [list(np.mean(x['coordinates'],axis=0))[1] for index, x in grid_cell_df
grid_cell_df
```

executed in 3.39s, finished 08:04:23 2022-06-01

Out[9]:

	cell_id	coordinates	region	centr_lat	centr_lon
0	0003f387-71c4-48f6-b2b0-d853bd4f0aba	[[-118.718953, 37.074192], [-118.718953, 37.08...	sierras	-118.722546	37.077059
1	000617d8-8c14-43e2-b708-7e3a69fe3cc3	[[-107.076787, 37.780424], [-107.076787, 37.78...	central rockies	-107.080380	37.783264
2	000863e7-21e6-477d-b799-f5675c348627	[[-119.401673, 37.024005], [-119.401673, 37.03...	other	-119.405266	37.026874
3	000ba8d9-d6d5-48da-84a2-1fa54951fae1	[[-119.320824, 37.431707], [-119.320824, 37.43...	sierras	-119.324418	37.434560
4	00146204-d4e9-4cd8-8f86-d1ef133c5b6d	[[-118.521324, 36.657353], [-118.521324, 36.66...	sierras	-118.524917	36.660236
...	...	...	...	...	...
18125	ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	[[-118.620138, 37.117184], [-118.620138, 37.12...	sierras	-118.623732	37.120049
18126	ffe43514-2c92-43b6-bd84-	[[-123.49799, 47.901318], [-123.49799,	other	-123.501584	47.903727

In [10]:
```
#Check data types in this dataframe
grid_cell_df.info()
```

executed in 27ms, finished 08:04:23 2022-06-01

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18130 entries, 0 to 18129
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   cell_id      18130 non-null  object 
 1   coordinates  18130 non-null  object 
 2   region       18130 non-null  object 
 3   centr_lat    18130 non-null  float64
 4   centr_lon    18130 non-null  float64
dtypes: float64(2), object(3)
memory usage: 708.3+ KB
```

1.3 Training data

1.3.1 Ground measures

Below is the SWE values from ground stations for our training dataset.

In [11]:
```
gm_train_feat = pd.read_csv("../data/ground_measures_train_features.csv")
gm_train_feat.rename(columns={'Unnamed: 0': "station_id"},inplace=True)
gm_train_feat
```

executed in 91ms, finished 08:04:23 2022-06-01

Out[11]:

	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	20
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

In [12]:
```
# stations_train = list(gm_train_feat['station_id'].values)
# len(stations_train)
```

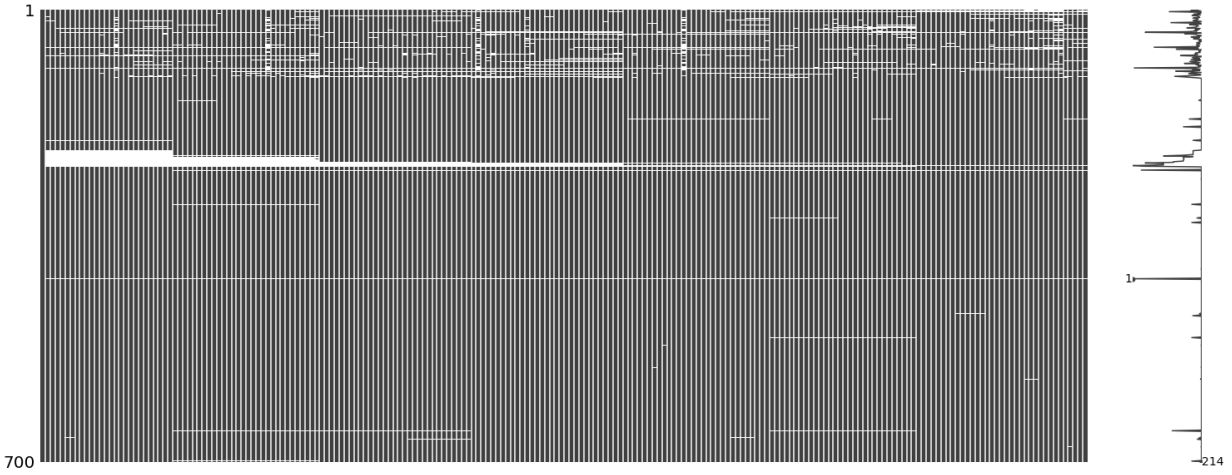
executed in 13ms, finished 08:04:23 2022-06-01

Let's find out if these stations carry any nulls.

In [13]:
```
import missingno as msno

msno.matrix(gm_train_feat)
```

executed in 709ms, finished 08:04:23 2022-06-01

Out[13]: <AxesSubplot:>



Based on the above heatmap of NaNs, there are a number of stations that have completely null values, and a number of stations that are mostly null values. Let's find out which stations have the most nulls.

In [15]:
```
# Transpose and set station id as columns
gm_t=gm_train_feat.T
gm_t.columns = gm_t.iloc[0]
gm_t = gm_t[1:]
gm_t
```

executed in 59ms, finished 08:04:23 2022-06-01

Out[15]:

station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL	CDE
2013-01-01	5.9	17.52	12.75	4.3	2.88	6.6	7.2	19.32	16.56	
2013-01-08	5.9	17.54	13.32	4.42	3.0	7.92	7.28	19.32	17.04	
2013-01-15	6.5	17.85	14.26	4.62	3.48	8.76	7.79	20.16	18.72	
2013-01-22	6.5	17.39	14.02	4.53	3.84	9.0	7.8	20.28	19.2	
2013-01-29	7.4	18.03	13.39	4.67	3.96	10.44	8.35	20.52	19.08	
...	...	...	...	...	...	...	...	...	...	
2019-12-03	0.7	0.0	5.69	NaN	2.88	3.0	NaN	NaN	6.72	
2019-12-10	1.2	0.6	8.04	NaN	4.56	3.36	NaN	NaN	5.88	
2019-12-17	3.4	0.2	10.74	NaN	4.68	4.56	NaN	NaN	6.12	
2019-12-24	3.7	NaN	12.67	NaN	5.04	5.04	NaN	NaN	10.56	
2019-12-31	3.4	NaN	12.57	NaN	6.0	5.64	NaN	NaN	11.64	

213 rows × 700 columns

In [16]:
```
#Look at percentage of null values
(gm_t.isna().sum().sort_values(ascending=False)[:50])*100/213
```

executed in 44ms, finished 08:04:23 2022-06-01

Out[16]:
```
station_id
SNOTEL:549_NV_SNTL     100.000000
SNOTEL:1286_MT_SNTL    100.000000
CDEC:SSM                98.122066
SNOTEL:305_CO_SNTL      87.793427
SNOTEL:1287_MT_SNTL     83.568075
CDEC:FRW                82.159624
SNOTEL:1272_NV_SNTL     82.159624
CDEC:LLP                69.014085
SNOTEL:1277_CA_SNTL     55.399061
SNOTEL:1280_UT_SNTL     55.399061
SNOTEL:1252_CO_SNTL     55.399061
SNOTEL:1278_UT_SNTL     55.399061
CDEC:BCB                46.948357
CDEC:CRL                44.600939
SNOTEL:878_WY_SNTL      42.723005
SNOTEL:1269_UT_SNTL     40.845070
SNOTEL:1271_AZ_SNTL     40.845070
CDEC:WHW                38.967136
CDEC:TNY                37.558685
CDEC:MDW                36.619718
CDEC:PSN                30.516432
SNOTEL:1256_WA_SNTL     26.291080
SNOTEL:1263_WA_SNTL     26.291080
SNOTEL:1251_CO_SNTL     26.291080
SNOTEL:1257_WA_SNTL     26.291080
SNOTEL:1261_UT_SNTL     26.291080
SNOTEL:1262_NV_SNTL     26.291080
SNOTEL:1148_UT_SNTL     26.291080
SNOTEL:1258_CA_SNTL     25.821596
SNOTEL:1259_WA_SNTL     25.352113
CDEC:SHM                24.882629
CDEC:GIN                24.413146
CDEC:WTM                23.943662
CDEC:DPO                21.126761
CDEC:FRN                20.657277
CDEC:VLC                18.309859
SNOTEL:1133_WY_SNTL     17.840376
CDEC:CWD                17.370892
CDEC:SCT                16.901408
CDEC:WC3                16.901408
CDEC:UTY                15.023474
CDEC:AGP                15.023474
CDEC:TUM                15.023474
CDEC:RRM                15.023474
CDEC:GRV                15.023474
SNOTEL:379_WY_SNTL      14.084507
SNOTEL:677_ID_SNTL      14.084507
SNOTEL:998_WA_SNTL      14.084507
SNOTEL:418_WA_SNTL      14.084507
CDEC:BLA                13.615023
dtype: float64
```

Quite a number of these stations have more than 50% of their data as nulls. Let's drop stations that have more than 60% nulls and impute the other stations will nulls using KNN imputer.

1.3.1.1 Dropping nulls in ground measure stations

In [17]:
```
# Create mask for stations more than 60% nulls
mask = ((gm_t.isna().sum().sort_values(ascending=False))*100/213)<60

#Filter and get the list
filter_list = list((gm_t.isna().sum().sort_values(ascending=False))[mask].index)
filter_list

#Drop the stations in the original dataframe
gm_train_feat = gm_train_feat[(gm_train_feat['station_id'].isin(filter_list))]
gm_train_feat
```

executed in 123ms, finished 08:04:24 2022-06-01

Out[17]:

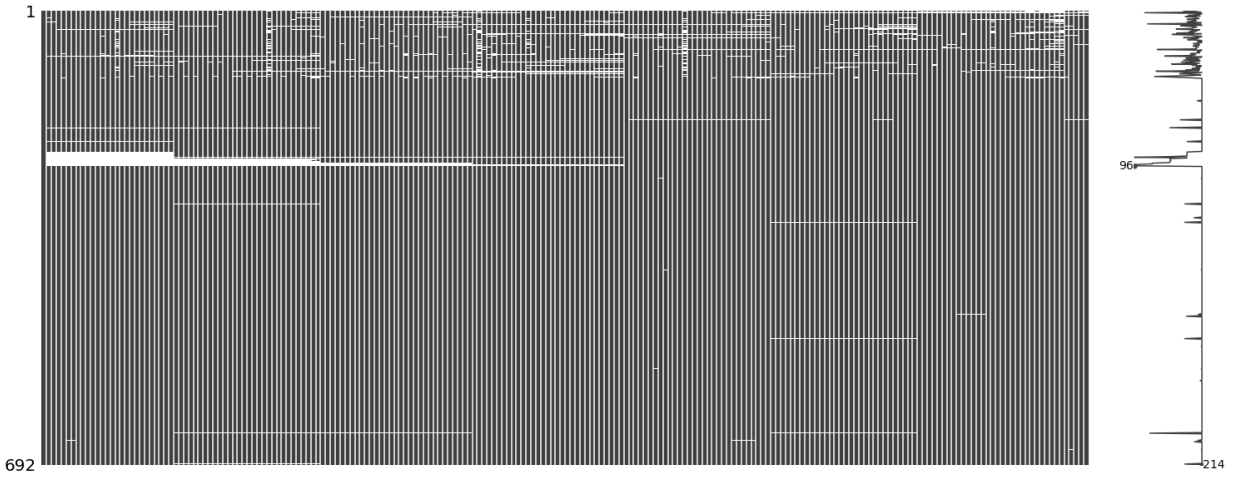
	station_id	2013-01-01	2013-01-08	2013-01-15	2013-01-22	2013-01-29	2013-02-05	2013-02-12	2013-02-19	2013-02-26	...	2019-05-28	2019-06-04	2019-06-11	2019-06-18	20
0	CDEC:ADM	5.90	5.90	6.50	6.50	7.40	7.60	7.40	8.00	8.00	...	NaN	NaN	NaN	NaN	
1	CDEC:AGP	17.52	17.54	17.85	17.39	18.03	17.70	17.65	16.66	17.21	...	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	

692 rows × 214 columns

In [18]:
```
import missingno as msno

msno.matrix(gm_train_feat)
```

executed in 630ms, finished 08:04:24 2022-06-01

Out[18]: <AxesSubplot:>



1.3.1.2 Interpolate missing values using Spline interpolation

We'll treat each station as a time series to do interpolation. This will ensure we take into account any seasonality or temporal pattern in our data.

In [19]:
```
# Transpose and set station id as columns
gm_t=gm_train_feat.T
gm_t.columns = gm_t.iloc[0]
gm_t = gm_t[1:]
gm_t
```

executed in 59ms, finished 08:04:24 2022-06-01

Out[19]:

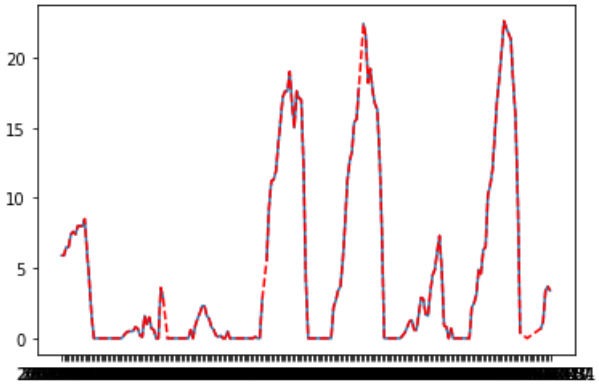
station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL	CDE
2013-01-01	5.9	17.52	12.75	4.3	2.88	6.6	7.2	19.32	16.56	
2013-01-08	5.9	17.54	13.32	4.42	3.0	7.92	7.28	19.32	17.04	
2013-01-15	6.5	17.85	14.26	4.62	3.48	8.76	7.79	20.16	18.72	
2013-01-22	6.5	17.39	14.02	4.53	3.84	9.0	7.8	20.28	19.2	
2013-01-29	7.4	18.03	13.39	4.67	3.96	10.44	8.35	20.52	19.08	
...	...	...	...	...	...	...	...	...	...	
2019-12-03	0.7	0.0	5.69	NaN	2.88	3.0	NaN	NaN	6.72	
2019-12-10	1.2	0.6	8.04	NaN	4.56	3.36	NaN	NaN	5.88	
2019-12-17	3.4	0.2	10.74	NaN	4.68	4.56	NaN	NaN	6.12	
2019-12-24	3.7	NaN	12.67	NaN	5.04	5.04	NaN	NaN	10.56	
2019-12-31	3.4	NaN	12.57	NaN	6.0	5.64	NaN	NaN	11.64	

213 rows × 692 columns

In [20]:
```
#Set date to datetime and as index
# gm_t.index = pd.to_datetime(gm_t.index)

#Test interpolation code on 1 station
plt.plot(gm_t ['CDEC:ADM'])
interpolated = gm_t['CDEC:ADM'].astype(float).interpolate(option='spline')

plt.plot(interpolated, color='r', linestyle='--')
```

executed in 2.43s, finished 08:04:27 2022-06-01

Out[20]:
```
[<matplotlib.lines.Line2D at 0x162d7cb6500>]
```



In [21]:
```
#Get list of our columns
stations_list = list(gm_t.columns)


#Apply interpolation to every station
for station in stations_list:
    gm_t[station] = gm_t[station].astype(float).interpolate(option='spline')

#Sniff test
gm_t
```

executed in 1.41s, finished 08:04:28 2022-06-01

Out[21]:

station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL
2013-01-01	5.9	17.52	12.75	4.30	2.88	6.60	7.20	19.32	16.56
2013-01-08	5.9	17.54	13.32	4.42	3.00	7.92	7.28	19.32	17.04
2013-01-15	6.5	17.85	14.26	4.62	3.48	8.76	7.79	20.16	18.72
2013-01-22	6.5	17.39	14.02	4.53	3.84	9.00	7.80	20.28	19.20
2013-01-29	7.4	18.03	13.39	4.67	3.96	10.44	8.35	20.52	19.08
...	...	...	...	...	...	...	...	...	...
2019-12-03	0.7	0.00	5.69	81.22	2.88	3.00	7.33	21.24	6.72

In [22]:
```
#Look at percentage of null values
(gm_t.isna().sum().sort_values(ascending=False)[:50])*100/213
```

executed in 90ms, finished 08:04:28 2022-06-01

Out[22]:
```
station_id
SNOTEL:1280_UT_SNTL    55.399061
SNOTEL:1278_UT_SNTL    55.399061
SNOTEL:1277_CA_SNTL    55.399061
SNOTEL:1252_CO_SNTL    55.399061
SNOTEL:1271_AZ_SNTL    40.845070
SNOTEL:1269_UT_SNTL    40.845070
CDEC:PSN               26.291080
SNOTEL:1263_WA_SNTL    26.291080
SNOTEL:1262_NV_SNTL    26.291080
SNOTEL:1261_UT_SNTL    26.291080
SNOTEL:1257_WA_SNTL    26.291080
SNOTEL:1251_CO_SNTL    26.291080
SNOTEL:1256_WA_SNTL    26.291080
SNOTEL:1148_UT_SNTL    26.291080
SNOTEL:1258_CA_SNTL    25.821596
SNOTEL:1259_WA_SNTL    25.352113
SNOTEL:1243_NV_SNTL    12.206573
SNOTEL:1244_NV_SNTL    12.206573
SNOTEL:1247_UT_SNTL    12.206573
SNOTEL:1248_UT_SNTL    12.206573
SNOTEL:1249_UT_SNTL    12.206573
SNOTEL:1254_NM_SNTL    12.206573
SNOTEL:1242_NV_SNTL    12.206573
SNOTEL:1236_UT_SNTL    12.206573
SNOTEL:1187_CO_SNTL    12.206573
CDEC:CDP                0.938967
CDEC:BLC                0.469484
SNOTEL:616_WY_SNTL      0.000000
SNOTEL:615_NV_SNTL      0.000000
SNOTEL:617_AZ_SNTL      0.000000
SNOTEL:651_OR_SNTL      0.000000
SNOTEL:614_OR_SNTL      0.000000
SNOTEL:619_OR_SNTL      0.000000
SNOTEL:613_MT_SNTL      0.000000
SNOTEL:612_UT_SNTL      0.000000
SNOTEL:610_ID_SNTL      0.000000
SNOTEL:621_UT_SNTL      0.000000
SNOTEL:650_ID_SNTL      0.000000
SNOTEL:622_CO_SNTL      0.000000
SNOTEL:649_MT_SNTL      0.000000
SNOTEL:623_ID_SNTL      0.000000
SNOTEL:625_WY_SNTL      0.000000
SNOTEL:626_UT_SNTL      0.000000
SNOTEL:627_ID_SNTL      0.000000
SNOTEL:628_UT_SNTL      0.000000
SNOTEL:629_CO_SNTL      0.000000
SNOTEL:633_CA_SNTL      0.000000
SNOTEL:637_ID_SNTL      0.000000
SNOTEL:640_AZ_SNTL      0.000000
SNOTEL:642_WA_SNTL      0.000000
dtype: float64
```

In [23]:
```
#Check for patterns in the missing values

dump_list = list((gm_t.isna().sum().sort_values(ascending=False)[:10]).index)

gm_t[dump_list].head(100)
```

executed in 105ms, finished 08:04:28 2022-06-01

Out[23]:

station_id	SNOTEL:1280_UT_SNTL	SNOTEL:1278_UT_SNTL	SNOTEL:1277_CA_SNTL	SNOTEL:1252_CO_SNTL	SNOTEL:1271_
2013-01-01	NaN	NaN	NaN	NaN	
2013-01-08	NaN	NaN	NaN	NaN	
2013-01-15	NaN	NaN	NaN	NaN	
2013-01-22	NaN	NaN	NaN	NaN	
2013-01-29	NaN	NaN	NaN	NaN	
...	...	...	...	...	
2016-01-26	NaN	NaN	NaN	NaN	
2016-02-02	NaN	NaN	NaN	NaN	
2016-02-09	NaN	NaN	NaN	NaN	
2016-02-16	NaN	NaN	NaN	NaN	
2016-02-23	NaN	NaN	NaN	NaN	

100 rows × 10 columns

So all the remaining stations that have more than 50% of missing values seem to have missing values between 2013 until 2016. While stations with 40% of missing values after interpolation are missing values from 2013 until the end of 2015.

It's impossible to interpolate 4 straight years of missing data. So, let's rid of the stations that have more than 50% of missing values from those 4 missing years.

In [24]:
```
gm_train_feat
```
executed in 44ms, finished 08:04:28 2022-06-01

Out[24]:

	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	20
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	

692 rows × 214 columns

In [25]:

```
#Transpose back to our original dataframe shape.
gm_train_feat=gm_t.T
gm_train_feat.reset_index(inplace=True)

#Drop more stations
dump_list = list((gm_t.isna().sum().sort_values(ascending=False)[:4]).index)

#Add on stations that doesn't have data the whole of 2019.
dump_list.extend(['CDEC:BCB','CDEC:BGP','CDEC:BIM'])

#Filter and get the list
mask = ~(gm_train_feat['station_id'].isin(dump_list))

#Drop the stations in the original dataframe
gm_train_feat = gm_train_feat[mask]
gm_train_feat
```

executed in 61ms, finished 08:04:28 2022-06-01

Out[25]:

	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-
0	CDEC:ADM	5.90	5.90	6.50	6.50	7.40	7.60	7.40	8.00	8.00	...	0.116667	0.233333	0.3500
1	CDEC:AGP	17.52	17.54	17.85	17.39	18.03	17.70	17.65	16.66	17.21	...	0.251613	0.201290	0.1509
2	CDEC:ALP	12.75	13.32	14.26	14.02	13.39	13.25	14.30	13.95	15.73	...	29.520000	20.810000	8.7100
4	CDEC:BCH	2.88	3.00	3.48	3.84	3.96	4.44	5.40	5.16	3.60	...	0.840000	0.600000	0.3600
5	CDEC:BFL	6.60	7.92	8.76	9.00	10.44	10.80	11.16	11.28	12.00	...	0.270000	0.150000	0.2700
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
687	SNOTEL:989_ID_SNTL	9.00	10.20	10.90	11.10	12.80	14.10	14.40	14.60	17.60	...	0.000000	0.000000	0.0000
688	SNOTEL:990_WA_SNTL	27.50	29.10	31.50	31.90	33.40	33.90	35.40	36.50	38.80	...	6.000000	0.100000	0.0000
689	SNOTEL:992_UT_SNTL	4.10	4.10	4.40	4.50	4.80	5.10	5.20	5.30	6.10	...	0.000000	0.000000	0.0000
690	SNOTEL:998_WA_SNTL	48.40	55.50	61.50	62.20	67.50	70.10	72.90	77.00	83.30	...	53.600000	36.100000	31.3000
691	SNOTEL:999_WA_SNTL	33.10	37.50	40.80	42.50	47.50	51.00	53.90	56.40	64.50	...	1.300000	0.000000	0.0000

685 rows × 214 columns

Let's update our ground measures station list with the updated stations

In [26]:
```
stations_list = list(gm_train_feat.station_id.unique())
gm_md = gm_md[gm_md['station_id'].isin(stations_list)]
gm_md
```

executed in 42ms, finished 08:04:28 2022-06-01

Out[26]:

	station_id	name	elevation_m	latitude	longitude	state	geometry
0	CDEC:ADM	Adin Mountain	1889.760000	41.237000	-120.792000	California	POINT (-120.79200 41.23700)
1	CDEC:AGP	Agnew Pass	2880.360000	37.726631	-119.141731	California	POINT (-119.14173 37.72663)
2	CDEC:ALP	Alpha (Smud)	2316.480000	38.804192	-120.215652	California	POINT (-120.21565 38.80419)
4	CDEC:BCH	Beach Meadows	2331.720000	36.126095	-118.293457	California	POINT (-118.29346 36.12609)
5	CDEC:BFL	Big Flat	1554.480000	41.077599	-122.942230	California	POINT (-122.94223 41.07760)
...	...	...	...	...	...	...	...
755	SNOTEL:989_ID_SNTL	Moscow Mountain	1432.560059	46.805000	-116.853500	Idaho	POINT (-116.85350 46.80500)
756	SNOTEL:990_WA_SNTL	Beaver Pass	1106.423950	48.879299	-121.255501	Washington	POINT (-121.25550 48.87930)
758	SNOTEL:992_UT_SNTL	Bear River RS	2675.229492	40.885201	-110.827698	Utah	POINT (-110.82770 40.88520)
762	SNOTEL:998_WA_SNTL	Easy Pass	1606.296021	48.859329	-121.438950	Washington	POINT (-121.43895 48.85933)
763	SNOTEL:999_WA_SNTL	Marten Ridge	1072.895996	48.762920	-121.698227	Washington	POINT (-121.69823 48.76292)

685 rows × 7 columns

1.3.2 Training labels

Below is our ground truth data for each cell grid for each date.

In [218]:
```
train_labels = pd.read_csv("../data/train_labels.csv")
train_labels = train_labels.melt(id_vars=["cell_id"]).dropna()
train_labels.rename({'variable':'dates'},axis=1,inplace=True)
train_labels
```

executed in 1.25s, finished 16:33:22 2022-05-31

Out[218]:

	cell_id	dates	value
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0
...	...	...	...
2904295	fd4492f2-8aa9-4279-bdc0-73991786943f	2019-12-31	1.3
2904321	fde3221a-9ce3-45a9-857f-bd196b07aa05	2019-12-31	5.6
2904323	fdeb8912-f9d1-445d-aadb-e943534f67fe	2019-12-31	8.8
2904336	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2019-12-31	2.9
2904371	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2019-12-31	4.7

91490 rows × 3 columns

1.3.2.1 Mapping ground measure to training set cell grids

We want to use the SWE of 20 of the nearest ground stations as model features for our cell grids. We will have to which ground stations are nearest to our cell grid first. For this, we will be using [cdist (https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html)](https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cdist.html) package from scipy.

In [ ]:
```
## This code block gives the wrong distance

# from scipy.spatial.distance import cdist

# mat = dt(gm_md[['latitude','longitude']],grid_cell_df[['centr_lat','centr_lon']],metric='euclidea
# mat
# nn_train_df = pd.DataFrame(mat, index=gm_md['station_id'], columns=grid_cell_df['cell_id'])
# nn_df
```
executed in 0ms, finished 22:15:33 2022-05-23

In [101]:
```
# Function that Calculates haversine distance
def distance_from(loc1,loc2):
    dist=hs.haversine(loc1,loc2)
    return round(dist,2)
```
executed in 10ms, finished 18:32:22 2022-05-30

In [ ]:
```
# #Test code block for one station

# station='SNOTEL:988_ID_SNTL'
# station_mask = gm_md['station_id']==station
# station_coord = tuple(gm_md[station_mask][['latitude','longitude']].iloc[0])
# nn_df[station] = [distance_from(station_coord, coord) for coord in zip(grid_cell_df['centr_lat'],
```
executed in 0ms, finished 22:15:33 2022-05-23

In [102]:
```
nn_df = pd.DataFrame()
for station in stations_list:
    station_mask = gm_md['station_id']==station
    station_coord = tuple(gm_md[station_mask][['latitude','longitude']].iloc[0])
    nn_df[station] = [distance_from(station_coord,coord) for coord in zip(grid_cell_df['centr_lat']
nn_df
```
executed in 1m 5.79s, finished 18:33:29 2022-05-30

```
C:\Users\hanis\AppData\Local\Temp\ipykernel_11148\1461231269.py:5: PerformanceWarning: DataFrame
is highly fragmented.  This is usually the result of calling `frame.insert` many times, which has
poor performance.  Consider joining all columns at once using pd.concat(axis=1) instead. To get a
de-fragmented frame, use `newframe = frame.copy()`
  nn_df[station] = [distance_from(station_coord,coord) for coord in zip(grid_cell_df['centr_la
t'],grid_cell_df['centr_lon'])]
C:\Users\hanis\AppData\Local\Temp\ipykernel_11148\1461231269.py:5: PerformanceWarning: DataFrame
is highly fragmented.  This is usually the result of calling `frame.insert` many times, which has
poor performance.  Consider joining all columns at once using pd.concat(axis=1) instead. To get a
de-fragmented frame, use `newframe = frame.copy()`
  nn_df[station] = [distance_from(station_coord,coord) for coord in zip(grid_cell_df['centr_la
t'],grid_cell_df['centr_lon'])]
C:\Users\hanis\AppData\Local\Temp\ipykernel_11148\1461231269.py:5: PerformanceWarning: DataFrame
is highly fragmented.  This is usually the result of calling `frame.insert` many times, which has
poor performance.  Consider joining all columns at once using pd.concat(axis=1) instead. To get a
de-fragmented frame, use `newframe = frame.copy()`
  nn_df[station] = [distance_from(station_coord,coord) for coord in zip(grid_cell_df['centr_la
t'],grid_cell_df['centr_lon'])]
```

In [103]:
```
#Get cell grids list
cells_list = list(grid_cell_df['cell_id'].values)

#Transpose and change column names to cell ids
nn_df = nn_df.T
nn_df.columns = cells_list
nn_df
```

executed in 140ms, finished 18:33:30 2022-05-30

Out[103]:

	0003f387-71c4-48f6-b2b0-d853bd4f0aba	000617d8-8c14-43e2-b708-7e3a69fe3cc3	000863e7-21e6-477d-b799-f5675c348627	000ba8d9-d6d5-48da-84a2-1fa54951fae1	00146204-d4e9-4cd8-8f86-d1ef133c5b6d	0017d1c4-64cb-426d-9158-3f6521d2dd22	0020c632-3d5c-4509-b4ee-6b63a89bf2ff
CDEC:ALP	11316.95	12537.87	11245.54	11247.33	11344.73	11239.10	11301.63
CDEC:BCB	11152.07	12361.60	11081.36	11082.53	11180.21	11074.65	11137.21
CDEC:BCH	11058.25	12263.24	10987.82	10988.75	11086.53	10981.01	11043.57
CDEC:BFL	11523.96	12762.74	11451.47	11454.32	11551.08	11445.49	11507.88
CDEC:BLC	11360.93	12585.38	11289.31	11291.30	11388.59	11282.96	11345.47
...	...	...	...	...	...	...	...
SNOTEL:989_ID_SNTL	12243.88	13455.88	12172.82	12174.29	12271.85	12166.24	12228.79
SNOTEL:990_WA_SNTL	12401.12	13639.72	12328.54	12331.49	12428.18	12322.61	12384.97
SNOTEL:992_UT_SNTL	11724.32	12882.68	11656.43	11655.41	11753.58	11648.83	11711.15
SNOTEL:998_WA_SNTL	12396.39	13635.85	12323.76	12326.76	12423.41	12317.85	12380.21
SNOTEL:999_WA_SNTL	12382.30	13622.87	12309.60	12312.67	12409.27	12303.72	12366.06

673 rows × 18130 columns

Get 20 nearest neighbor and put it in our cell grid dataframe.

In [32]:
```
#Sniff test

nn_df['000863e7-21e6-477d-b799-f5675c348627'].sort_values(ascending=True)[:20]
```

executed in 43ms, finished 22:30:03 2022-05-23

Out[32]:
```
SNOTEL:755_NM_SNTL     10871.55
SNOTEL:1048_NM_SNTL    10887.82
SNOTEL:877_AZ_SNTL     10896.26
SNOTEL:757_NM_SNTL     10904.58
SNOTEL:902_AZ_SNTL     10925.68
SNOTEL:486_NM_SNTL     10937.44
SNOTEL:416_AZ_SNTL     10939.33
SNOTEL:1271_AZ_SNTL    10941.75
SNOTEL:617_AZ_SNTL     10944.10
SNOTEL:1127_AZ_SNTL    10949.27
SNOTEL:308_AZ_SNTL     10953.64
SNOTEL:519_AZ_SNTL     10953.82
SNOTEL:1140_AZ_SNTL    10954.08
CDEC:PSC               10969.46
CDEC:QUA               10982.31
SNOTEL:1034_NM_SNTL    10986.70
CDEC:BCH               10987.82
CDEC:CSV               10996.13
SNOTEL:640_AZ_SNTL     11003.00
CDEC:WTM               11006.81
Name: 000863e7-21e6-477d-b799-f5675c348627, dtype: float64
```

In [104]:
```
#Helper function

def nn_20(cell_id,df):
    """
    Get 20 nearest neighbor from dataframe and return a list of the stations names.
    """
    station_list = list(df[cell_id].sort_values(ascending=True)[:20].index)
    return station_list
```

executed in 16ms, finished 18:33:32 2022-05-30

In [106]:
```
#Put the list of nearest neighbors in our cell grid dataframe.

grid_cell_df['nearest_neighb'] = [nn_20(x,nn_df) for x in grid_cell_df['cell_id']]
grid_cell_df.set_index('cell_id')
```

executed in 4.64s, finished 18:34:49 2022-05-30

Out[106]:

cell_id	coordinates	region	centr_lat	centr_lon	geometry	nearest_neighb
0003f387-71c4-48f6-b2b0-d853bd4f0aba	[[-118.718953, 37.074192], [-118.718953, 37.08...	sierras	-118.722546	37.077059	POLYGON ((-118.71895 37.07419, -118.71895 37.0...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
000617d8-8c14-43e2-b708-7e3a69fe3cc3	[[-107.076787, 37.780424], [-107.076787, 37.78...	central rockies	-107.080380	37.783264	POLYGON ((-107.07679 37.78042, -107.07679 37.7...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
000863e7-21e6-477d-b799-f5675c348627	[[-119.401673, 37.024005], [-119.401673, 37.03...	other	-119.405266	37.026874	POLYGON ((-119.40167 37.02401, -119.40167 37.0...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
000ba8d9-d6d5-48da-84a2-1fa54951fae1	[[-119.320824, 37.431707], [-119.320824, 37.43...	sierras	-119.324418	37.434560	POLYGON ((-119.32082 37.43171, -119.32082 37.4...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
00146204-d4e9-4cd8-8f86-d1ef133c5b6d	[[-118.521324, 36.657353], [-118.521324, 36.66...	sierras	-118.524917	36.660236	POLYGON ((-118.52132 36.65735, -118.52132 36.6...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
...	...	...	...	...	...	...
ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	[[-118.620138, 37.117184], [-118.620138, 37.12...	sierras	-118.623732	37.120049	POLYGON ((-118.62014 37.11718, -118.62014 37.1...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
ffe43514-2c92-43b6-bd84-d183806aca65	[[-123.49799, 47.901318], [-123.49799, 47.9073...	other	-123.501584	47.903727	POLYGON ((-123.49799 47.90132, -123.49799 47.9...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
ffeabc13-7c6f-4b63-b043-19c8f15e0345	[[-119.644218, 37.879756], [-119.644218, 37.88...	sierras	-119.647811	37.882592	POLYGON ((-119.64422 37.87976, -119.64422 37.8...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
fff95195-ccc9-40b7-b302-a0d8570c86bc	[[-123.372226, 47.732416], [-123.372226, 47.73...	other	-123.375819	47.734833	POLYGON ((-123.37223 47.73242, -123.37223 47.7...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
fffb4d40-5947-4922-9f05-5d8b5a243d84	[[-123.794435, 47.520516], [-123.794435, 47.52...	other	-123.798028	47.522942	POLYGON ((-123.79443 47.52052, -123.79443 47.5...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...

18130 rows × 6 columns

In [35]:
```
#Sniff test
grid_cell_df['nearest_neighb'].explode().unique()
```
executed in 58ms, finished 22:30:23 2022-05-23

Out[35]:
```
array(['SNOTEL:755_NM_SNTL', 'SNOTEL:1048_NM_SNTL', 'SNOTEL:877_AZ_SNTL',
       'SNOTEL:757_NM_SNTL', 'SNOTEL:902_AZ_SNTL', 'SNOTEL:486_NM_SNTL',
       'SNOTEL:416_AZ_SNTL', 'SNOTEL:1271_AZ_SNTL', 'SNOTEL:617_AZ_SNTL',
       'SNOTEL:1127_AZ_SNTL', 'SNOTEL:519_AZ_SNTL', 'SNOTEL:308_AZ_SNTL',
       'SNOTEL:1140_AZ_SNTL', 'CDEC:PSC', 'SNOTEL:1034_NM_SNTL',
       'CDEC:QUA', 'CDEC:BCH', 'CDEC:CSV', 'SNOTEL:640_AZ_SNTL',
       'SNOTEL:488_AZ_SNTL', 'SNOTEL:861_AZ_SNTL', 'SNOTEL:1139_AZ_SNTL',
       'SNOTEL:1121_AZ_SNTL', 'CDEC:WTM', 'SNOTEL:927_AZ_SNTL'],
      dtype=object)
```

1.3.2.2 Add nearest neighboring stations to training set dataframe

Add neighbors values to our dataframe.

In [54]:
```
#Function to help us pull nearest neighbor list

def get_neighb_list(df_idx,label_df, grid_df):
    """
    Get the neareset neighbor list for the row
    """
    cell_id = label_df['cell_id'].iloc[df_idx]
    nn_list = grid_df[grid_df['cell_id'] == cell_id]['nearest_neighb'].values[0]

    return nn_list

def get_gm_value(df_idx, station_idx, gm_df,label_df,nn_list):
    value = gm_df[gm_df['station_id'] == nn_list[station_idx]][label_df['dates'].iloc[df_idx]].valu
    return value
```
executed in 10ms, finished 22:45:28 2022-05-23

In [63]:
```
#Test function
nn_list = get_neighb_list(0,train_labels,grid_cell_df)
get_gm_value(0,0,gm_train_feat,train_labels,nn_list)
```
executed in 19ms, finished 07:58:49 2022-05-24

Out[63]:
```
2.0
```

In [66]:

```
for k in range (0,20):
    print (f"Working on neighbor {k+1}")
    neighbor=[]
    for i in range(0,len(train_labels)):
        nn_list = get_neighb_list(i,train_labels,grid_cell_df)
        try:
            value = get_gm_value(i,k,gm_train_feat,train_labels,nn_list)
        except:
            value = np.NaN
        neighbor.append(value)
#         print (neighbor)
    col_name = f'neighbor_{k+1}'
    train_labels[col_name] = neighbor
```

executed in 1h 18m 38s, finished 09:19:26 2022-05-24

```
Working on neighbor 1
Working on neighbor 2
Working on neighbor 3
Working on neighbor 4
Working on neighbor 5
Working on neighbor 6
Working on neighbor 7
Working on neighbor 8
Working on neighbor 9
Working on neighbor 10
Working on neighbor 11
Working on neighbor 12
Working on neighbor 13
Working on neighbor 14
Working on neighbor 15
Working on neighbor 16
Working on neighbor 17
Working on neighbor 18
Working on neighbor 19
Working on neighbor 20
```

In [67]:
```
train_labels
```
executed in 118ms, finished 09:20:14 2022-05-24

Out[67]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3	2.0	1.6	3.2	6.4	2.7	2.0	2.8
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0	2.0	1.6	3.2	6.4	2.7	2.0	2.8
...	...	...	...	...	...	...	...	...	...	...
2904295	fd4492f2-8aa9-4279-bdc0-73991786943f	2019-12-31	1.3	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904321	fde3221a-9ce3-45a9-857f-bd196b07aa05	2019-12-31	5.6	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904323	fdeb8912-f9d1-445d-aadb-e943534f67fe	2019-12-31	8.8	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904336	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2019-12-31	2.9	1.3	2.7	4.0	6.7	2.1	1.5	2.8
2904371	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2019-12-31	4.7	1.3	2.7	6.7	4.0	2.1	1.5	2.8

91490 rows × 23 columns

Let's check for null is our new dataframe.

In [69]:
```
train_labels.isna().sum()
```

executed in 54ms, finished 09:20:24 2022-05-24

Out[69]:
```
cell_id            0
dates              0
value              0
neighbor_1     41032
neighbor_2     41032
neighbor_3     41032
neighbor_4     41032
neighbor_5     41032
neighbor_6     41032
neighbor_7     41032
neighbor_8     49066
neighbor_9     47731
neighbor_10    44338
neighbor_11    41641
neighbor_12    41032
neighbor_13    41032
neighbor_14    41032
neighbor_15    41032
neighbor_16    41032
neighbor_17    41032
neighbor_18    41032
neighbor_19    41032
neighbor_20    41032
dtype: int64
```

That's a lot of nulls of the same amount across the neighbors. Most of these nulls probably come from our own code where we set values to null if date exist in grid cell dataframe and not in ground stations dataframe ! Some of the neighbors have more nulls than others too. Let's analyze these and decide what we should do next.

In [71]:
```
train_labels[train_labels['neighbor_1'].isna()]
```

executed in 67ms, finished 09:21:49 2022-05-24

Out[71]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
152344	00ef834b-a1de-4305-b7ef-0800961c7fa2	2013-04-03	28.6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
152357	01438219-bc42-47bb-8a4f-d50d4c8bcfed	2013-04-03	31.6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
152424	02fa9f4d-a454-4e73-8aea-4d19c4ee3e37	2013-04-03	28.7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
152446	03968d16-5231-4140-b602-61356a1bd393	2013-04-03	15.2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
152474	04149267-2eeb-4910-8360-f76b2df2ad8d	2013-04-03	23.6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
2838988	fc374249-6186-4986-9c9e-3a3ed64c5aa2	2019-06-24	5.4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2839006	fcb3f308-86b0-48d5-b053-43844d3ec5b4	2019-06-24	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2839105	ff0b561c-3a91-44e7-8054-52f90bcabc59	2019-06-24	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2839109	ff1540b9-01f2-410b-90aa-678f3c4d8ea2	2019-06-24	12.4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2839151	ffca2dc2-ed02-4006-82a3-fa093863fa68	2019-06-24	7.9	NaN	NaN	NaN	NaN	NaN	NaN	NaN

41032 rows × 23 columns

In [119]:
```
train_labels[train_labels['neighbor_1'].isna()]['dates'].unique() == train_labels[train_labels['nei
```

executed in 33ms, finished 12:10:27 2022-05-20

Out[119]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True])
```

Since 50% have these dates that contain null, we'll be dropping them. That's a lot of data to drop but we'll still have 50000 observations to train our model on. It will be ok..

In [86]:
```
#Get dates with missing values for all neighbors
null_dates = list(train_labels[train_labels['neighbor_1'].isna()]['dates'].unique())
null_dates
```

executed in 24ms, finished 12:19:35 2022-05-24

Out[86]:
```
['2013-04-03',
 '2013-04-29',
 '2013-05-03',
 '2013-05-25',
 '2013-06-01',
 '2013-06-08',
 '2016-02-08',
 '2016-03-26',
 '2016-04-01',
 '2016-04-03',
 '2016-04-04',
 '2016-04-07',
 '2016-04-16',
 '2016-05-09',
 '2016-05-27',
 '2016-06-26',
 '2017-01-28',
 '2017-01-29',
 '2018-03-04',
 '2018-03-30',
 '2018-03-31',
 '2018-04-22',
 '2018-04-23',
 '2018-04-25',
 '2018-04-26',
 '2018-05-24',
 '2018-05-28',
 '2018-06-01',
 '2018-06-02',
 '2019-03-09',
 '2019-03-15',
 '2019-03-16',
 '2019-03-17',
 '2019-03-24',
 '2019-03-25',
 '2019-03-29',
 '2019-04-07',
 '2019-04-08',
 '2019-04-17',
 '2019-04-18',
 '2019-04-19',
 '2019-04-21',
 '2019-04-27',
 '2019-04-28',
 '2019-05-01',
 '2019-05-02',
 '2019-05-03',
 '2019-06-05',
 '2019-06-08',
 '2019-06-09',
 '2019-06-10',
 '2019-06-13',
 '2019-06-14',
 '2019-06-24']
```

In [92]:
```
#Create mask to filter rows with dates.
mask = train_labels['dates'].isin(null_dates)

#filter out the rows with the dates
train_labels_cleaned = train_labels[~mask]

#Sniff test
train_labels_cleaned.isna().sum()*100/len(train_labels_cleaned)
```

executed in 39ms, finished 12:26:02 2022-05-24

Out[92]:
```
cell_id         0.000000
dates           0.000000
value           0.000000
neighbor_1      0.000000
neighbor_2      0.000000
neighbor_3      0.000000
neighbor_4      0.000000
neighbor_5      0.000000
neighbor_6      0.000000
neighbor_7      0.000000
neighbor_8     15.922153
neighbor_9     13.276388
neighbor_10     6.551984
neighbor_11     1.206944
neighbor_12     0.000000
neighbor_13     0.000000
neighbor_14     0.000000
neighbor_15     0.000000
neighbor_16     0.000000
neighbor_17     0.000000
neighbor_18     0.000000
neighbor_19     0.000000
neighbor_20     0.000000
dtype: float64
```

In [91]:
```
train_labels_cleaned
```

executed in 64ms, finished 12:23:24 2022-05-24

Out[91]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3	2.0	1.6	3.2	6.4	2.7	2.0	2.8
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0	2.0	1.6	3.2	6.4	2.7	2.0	2.8
...	...	...	...	...	...	...	...	...	...	...
2904295	fd4492f2-8aa9-4279-bdc0-73991786943f	2019-12-31	1.3	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904321	fde3221a-9ce3-45a9-857f-bd196b07aa05	2019-12-31	5.6	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904323	fdeb8912-f9d1-445d-aadb-e943534f67fe	2019-12-31	8.8	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904336	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2019-12-31	2.9	1.3	2.7	4.0	6.7	2.1	1.5	2.8
2904371	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2019-12-31	4.7	1.3	2.7	6.7	4.0	2.1	1.5	2.8

50458 rows × 23 columns

Now that we've dropped dates that are complete nulls, we still see that eighboring ground stations 8, 9, 10 still have nulls at specific dates. This is not a ton (between 1 - 15% of our dataset. We can impute this in our modeling workflow.

In [99]:
```
train_labels_cleaned[train_labels_cleaned['neighbor_8'].isna()]
```

executed in 52ms, finished 12:31:22 2022-05-24

Out[99]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8
294	06442780-9a68-424b-9fbe-0184e7d14d2e	2013-01-01	3.6	2.0	1.6	3.2	6.4	2.7	2.0	2.8
325	06ec1554-a0ea-41cf-85f8-43f4b48c7599	2013-01-01	13.4	2.0	1.6	3.2	6.4	2.7	2.0	2.8
...	...	...	...	...	...	...	...	...	...	...
1011032	f1f17cee-3fca-44bc-8974-b72d5f995a4a	2015-06-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1011216	f6106ef5-e6fd-423b-9034-59bce18319da	2015-06-30	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1011242	f6a782f0-4b17-4989-815c-2398b77b09dd	2015-06-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1011407	fa92fd14-b8db-48b2-be54-f57db0a05bda	2015-06-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1011564	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2015-06-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

8034 rows × 23 columns

In [98]:
```
#Dates of missing values in neighbor 8
train_labels_cleaned[train_labels_cleaned['neighbor_8'].isna()]['dates'].unique()

#Another method of checking for the same thing.
# np.setdiff1d(list(train_labels[train_labels['neighbor_8'].isna()]['dates'].unique()), null_dates)
```

executed in 39ms, finished 12:30:30 2022-05-24

Out[98]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True])
```

In [240]:
```
train_labels_cleaned
```

executed in 69ms, finished 13:48:06 2022-05-24

Out[240]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3	2.0	1.6	3.2	6.4	2.7	2.0	2.8
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0	2.0	1.6	3.2	6.4	2.7	2.0	2.8
...	...	...	...	...	...	...	...	...	...	...
2904295	fd4492f2-8aa9-4279-bdc0-73991786943f	2019-12-31	1.3	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904321	fde3221a-9ce3-45a9-857f-bd196b07aa05	2019-12-31	5.6	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904323	fdeb8912-f9d1-445d-aadb-e943534f67fe	2019-12-31	8.8	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904336	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2019-12-31	2.9	1.3	2.7	4.0	6.7	2.1	1.5	2.8
2904371	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2019-12-31	4.7	1.3	2.7	6.7	4.0	2.1	1.5	2.8

50458 rows × 23 columns

1.3.2.3 Geospatial info

The geospatial information that we will be adding to our training dataframe will be:

- **Region**: This feature has 3 categories (Sierras, central rockis, other)
- **Latitude** & **Longitude**: The latitude of the centroid of the cell grid. We will fuzz this out to 1 decimal places and lose the accuracy of our coordinates for precision. This will reduce cardinality in this feature. For perspective, 1 decimal place can practically distinguish two counties. Here (https://gis.stackexchange.com/questions/8650/measuring-accuracy-of-latitude-and-longitude#:~:text=The%20first%20decimal%20place%20is,one%20village%20from%20the%20next.) and (here (https://www.forensicdjs.com/blog/gps-coordinates-many-decimal-places-need/) is more information on decimals and resolution.

In [249]:
```
grid_cell_df
```

executed in 86ms, finished 16:56:48 2022-05-24

Out[249]:

	cell_id	coordinates	region	centr_lat	centr_lon	geometry	nearest_neighb
0	0003f387-71c4-48f6-b2b0-d853bd4f0aba	[[-118.718953, 37.074192], [-118.718953, 37.08...	sierras	-118.722546	37.077059	POLYGON ((-118.71895 37.07419, -118.71895 37.0...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
1	000617d8-8c14-43e2-b708-7e3a69fe3cc3	[[-107.076787, 37.780424], [-107.076787, 37.78...	central rockies	-107.080380	37.783264	POLYGON ((-107.07679 37.78042, -107.07679 37.7...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
2	000863e7-21e6-477d-b799-f5675c348627	[[-119.401673, 37.024005], [-119.401673, 37.03...	other	-119.405266	37.026874	POLYGON ((-119.40167 37.02401, -119.40167 37.0...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
3	000ba8d9-d6d5-48da-84a2-1fa54951fae1	[[-119.320824, 37.431707], [-119.320824, 37.43...	sierras	-119.324418	37.434560	POLYGON ((-119.32082 37.43171, -119.32082 37.4...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
4	00146204-d4e9-4cd8-8f86-d1ef133c5b6d	[[-118.521324, 36.657353], [-118.521324, 36.66...	sierras	-118.524917	36.660236	POLYGON ((-118.52132 36.65735, -118.52132 36.6...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
...	...	...	...	...	...	...	...
18125	ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	[[-118.620138, 37.117184], [-118.620138, 37.12...	sierras	-118.623732	37.120049	POLYGON ((-118.62014 37.11718, -118.62014 37.1...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
18126	ffe43514-2c92-43b6-bd84-d183806aca65	[[-123.49799, 47.901318], [-123.49799, 47.9073...	other	-123.501584	47.903727	POLYGON ((-123.49799 47.90132, -123.49799 47.9...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
18127	ffeabc13-7c6f-4b63-b043-19c8f15e0345	[[-119.644218, 37.879756], [-119.644218, 37.88...	sierras	-119.647811	37.882592	POLYGON ((-119.64422 37.87976, -119.64422 37.8...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
18128	fff95195-ccc9-40b7-b302-a0d8570c86bc	[[-123.372226, 47.732416], [-123.372226, 47.73...	other	-123.375819	47.734833	POLYGON ((-123.37223 47.73242, -123.37223 47.7...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...
18129	fffb4d40-5947-4922-9f05-5d8b5a243d84	[[-123.794435, 47.520516], [-123.794435, 47.52...	other	-123.798028	47.522942	POLYGON ((-123.79443 47.52052, -123.79443 47.5...	[SNOTEL:755_NM_SNTL, SNOTEL:1048_NM_SNTL, SNOT...

18130 rows × 7 columns

In [259]:
```
round(grid_cell_df[grid_cell_df['cell_id'] == '0003f387-71c4-48f6-b2b0-d853bd4f0aba']['centr_lat'].
```

executed in 20ms, finished 17:29:45 2022-05-24

Out[259]:
```
-118.7
```

In [260]:
```
train_labels_cleaned['latitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr_l
train_labels_cleaned['longitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr_
train_labels_cleaned['region'] = [grid_cell_df[grid_cell_df['cell_id'] == cell_id]['region'].values
train_labels_cleaned
```

executed in 6m 49s, finished 17:37:04 2022-05-24

```
C:\Users\hanis\AppData\Local\Temp\ipykernel_9044\903773265.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  train_labels_cleaned['latitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr_
lat'].values[0],1) for cell_id in train_labels_cleaned['cell_id']]
C:\Users\hanis\AppData\Local\Temp\ipykernel_9044\903773265.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  train_labels_cleaned['longitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr
_lon'].values[0],1) for cell_id in train_labels_cleaned['cell_id']]
C:\Users\hanis\AppData\Local\Temp\ipykernel_9044\903773265.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  train_labels_cleaned['region'] = [grid_cell_df[grid_cell_df['cell_id'] == cell_id]['region'].value
s[0] for cell_id in train_labels_cleaned['cell_id']]
```

Out[260]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3	2.0	1.6	3.2	6.4	2.7	2.0	2.8
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0	2.0	1.6	3.2	6.4	2.7	2.0	2.8
...	...	...	...	...	...	...	...	...	...	...
2904295	fd4492f2-8aa9-4279-bdc0-73991786943f	2019-12-31	1.3	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904321	fde3221a-9ce3-45a9-857f-bd196b07aa05	2019-12-31	5.6	1.3	2.7	6.7	4.0	2.1	1.5	2.8
2904323	fdeb8912-f9d1-445d-aadb-e943534f67fe	2019-12-31	8.8	1.3	2.7	6.7	4.0	2.1	1.5	2.8

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
2904336	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2019-12-31	2.9	1.3	2.7	4.0	6.7	2.1	1.5	2.8
2904371	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2019-12-31	4.7	1.3	2.7	6.7	4.0	2.1	1.5	2.8

50458 rows × 26 columns

In [261]:
```
#Saving dataframe to file for now, just in case.
train_labels_cleaned.to_csv('../data/train_labels_final.csv')
```
executed in 797ms, finished 18:40:33 2022-05-24

1.4 Testing Data

1.4.1 Ground measures

Below is the SWE values from ground stations for our testing dataset.

In [72]:
```
gm_test_feat = pd.read_csv("../data/ground_measures_test_features.csv")
gm_test_feat.rename(columns={'Unnamed: 0': "station_id"},inplace=True)
gm_test_feat
```
executed in 155ms, finished 18:29:06 2022-05-30

Out[72]:

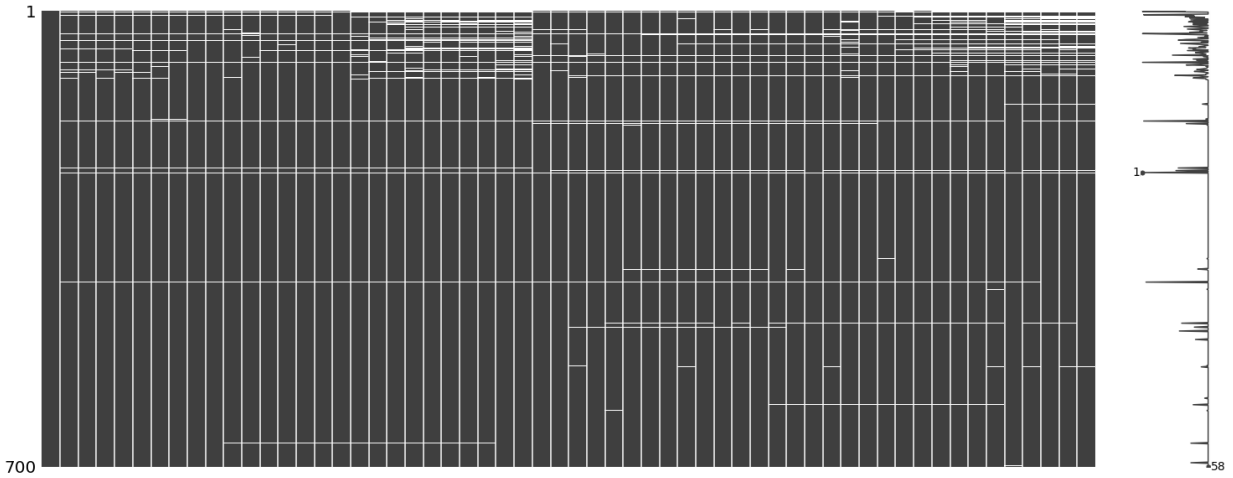
	station_id	2020-01-07	2020-01-14	2020-01-21	2020-01-28	2020-02-04	2020-02-11	2020-02-18	2020-02-25	2020-03-03	...	2021-04-27	2021-05-04	2021-05-11	2021-05-18
0	CDEC:ADM	4.50	5.50	7.30	8.30	8.10	8.20	9.30	8.50	7.90	...	0.30	NaN	NaN	NaN
1	CDEC:AGP	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
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
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

We're starting off with 700 ground measure stations and 58 dates, starting in January 2020 to June 2021, for our test set. Let's look at nulls in our ground measure stations.

In [73]:
```
import missingno as msno

msno.matrix(gm_test_feat)
```

executed in 637ms, finished 18:29:09 2022-05-30

Out[73]: <AxesSubplot:>



The nulls in our ground measures testing set isn't as bad as our training set.

1.4.1.1 Dropping nulls in ground measure testing data

In [74]:
```
# Transpose and set station id as columns
gm_test_t=gm_test_feat.T
gm_test_t.columns = gm_test_t.iloc[0]
gm_test_t = gm_test_t[1:]
gm_test_t
```

executed in 67ms, finished 18:29:11 2022-05-30

Out[74]:

station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL
2020-01-07	4.5	NaN	12.72	12.2	6.6	5.76	NaN	NaN	12.36
2020-01-14	5.5	NaN	13.78	12.2	5.76	7.08	NaN	NaN	16.68
2020-01-21	7.3	NaN	17.12	13.3	5.16	9.72	NaN	NaN	20.88
2020-01-28	8.3	NaN	18.07	13.35	7.68	10.32	NaN	NaN	21.84
2020-02-04	8.1	NaN	18.17	12.85	4.68	10.2	NaN	NaN	21.0
2020-02-11	8.2	NaN	18.38	12.72	1.32	10.8	NaN	NaN	21.48
2020-02-18	9.3	NaN	17.71	12.72	0.84	10.08	NaN	NaN	21.84
2020-02-25	8.5	NaN	16.05	12.8	0.84	10.2	NaN	NaN	21.84
2020-03-03	7.9	NaN	14.62	13.16	0.24	10.56	NaN	NaN	20.64
2020-03-10	7.3	NaN	12.79	12.82	0.24	10.8	NaN	NaN	20.16
2020-03-17	6.2	NaN	16.3	14.33	0.84	11.4	NaN	NaN	24.96
2020-03-24	6.0	NaN	17.52	14.88	1.2	11.76	NaN	NaN	27.96
2020-03-31	7.0	NaN	19.35	16.0	0.6	12.48	NaN	NaN	30.0
2020-04-07	7.9	NaN	22.31	19.29	0.84	13.08	NaN	NaN	33.36
2020-04-14	4.5	NaN	21.04	19.14	0.84	10.08	NaN	NaN	31.8
2020-04-21	0.0	NaN	18.58	20.1	0.72	5.28	NaN	20.2	29.04
2020-04-28	NaN	NaN	12.68	18.84	0.36	0.72	NaN	2.54	22.56
2020-05-05	NaN	NaN	2.83	14.36	0.24	0.72	NaN	0.32	12.0
2020-05-12	NaN	NaN	0.14	9.32	0.12	0.72	NaN	0.53	0.24
2020-05-19	NaN	NaN	1.2	6.69	0.48	0.72	NaN	0.87	0.36
2020-05-26	NaN	NaN	0.23	3.13	0.36	0.72	NaN	0.64	0.0
2020-06-02	NaN	NaN	0.2	1.21	0.36	0.84	NaN	0.63	0.0
2020-06-09	NaN	NaN	0.0	1.22	0.12	0.72	NaN	0.65	0.0
2020-06-16	NaN	NaN	0.0	1.32	0.24	0.72	NaN	0.63	0.0
2020-06-23	NaN	NaN	0.01	1.41	0.36	0.84	NaN	0.64	0.0
2020-06-30	NaN	NaN	0.0	1.39	0.12	0.72	0.28	0.7	0.0

station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL
2020-12-01	2.2	NaN	4.53	6.96	0.12	0.12	NaN	0.64	0.48
2020-12-08	2.1	NaN	4.35	7.06	0.24	0.72	NaN	0.68	0.0
2020-12-15	2.7	NaN	5.83	8.11	0.24	1.2	NaN	2.1	1.8
2020-12-22	3.5	NaN	7.66	8.62	2.4	2.64	NaN	2.66	1.8
2020-12-29	4.5	NaN	8.98	9.65	0.84	4.44	NaN	3.33	3.6
2021-01-05	6.0	NaN	9.65	9.36	1.32	6.36	NaN	3.6	5.16
2021-01-12	6.5	NaN	10.2	9.13	1.44	7.68	NaN	3.71	4.92
2021-01-19	6.2	NaN	8.43	9.33	0.48	7.8	NaN	3.57	5.04
2021-01-26	6.7	NaN	9.93	9.78	0.84	8.28	NaN	4.63	3.72
2021-02-02	7.5	NaN	17.68	17.56	2.52	12.48	NaN	7.49	10.2
2021-02-09	8.5	NaN	17.33	17.47	2.4	12.96	NaN	7.71	10.56
2021-02-16	10.4	NaN	19.56	19.12	2.88	15.36	NaN	8.54	11.64
2021-02-23	12.2	NaN	20.31	19.11	1.56	17.04	NaN	8.24	11.76
2021-03-02	12.3	NaN	19.9	19.01	0.36	16.92	NaN	8.2	10.68
2021-03-09	13.1	NaN	20.07	18.96	0.12	18.48	NaN	7.79	10.44
2021-03-16	14.1	NaN	22.11	20.61	0.96	20.64	NaN	8.34	10.44
2021-03-23	13.9	NaN	23.81	21.95	0.72	22.2	NaN	8.31	9.24
2021-03-30	11.7	NaN	23.23	20.14	0.24	22.68	NaN	9.38	6.12
2021-04-06	8.0	NaN	18.13	19.08	0.24	21.12	NaN	6.48	2.04
2021-04-13	2.4	NaN	12.4	17.94	0.12	18.84	NaN	0.0	NaN
2021-04-20	NaN	NaN	8.01	16.82	0.24	15.48	NaN	0.0	NaN
2021-04-27	0.3	NaN	3.34	15.44	0.12	12.36	NaN	0.06	NaN
2021-05-04	NaN	NaN	0.31	11.94	0.24	5.04	NaN	0.72	NaN
2021-05-11	NaN	NaN	0.02	5.91	0.24	0.12	NaN	0.7	NaN
2021-05-18	NaN	NaN	0.0	1.1	0.24	0.12	NaN	0.71	NaN
2021-05-25	NaN	NaN	0.13	0.86	0.12	0.12	NaN	0.7	NaN
2021-06-01	NaN	NaN	0.01	1.04	0.24	0.0	NaN	0.74	NaN
2021-06-08	NaN	NaN	0.0	1.12	0.12	0.0	NaN	0.74	NaN
2021-06-15	NaN	NaN	0.0	1.04	0.24	0.12	NaN	0.75	NaN

station_id	CDEC:ADM	CDEC:AGP	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BGP	CDEC:BIM	CDEC:BKL
2021-06-22	NaN	NaN	0.0	1.17	0.24	0.0	NaN	0.8	NaN
2021-06-29	NaN	NaN	0.0	1.25	0.36	0.12	NaN	0.72	NaN

57 rows × 700 columns

In [75]:
```
#Look at percentage of null values
(gm_test_t.isna().sum().sort_values(ascending=False)[:50])*100/57
```

executed in 39ms, finished 18:29:14 2022-05-30

Out[75]:
```
station_id
SNOTEL:305_CO_SNTL     100.000000
CDEC:AGP               100.000000
CDEC:SCN               100.000000
CDEC:FRW               100.000000
SNOTEL:1133_WY_SNTL     98.245614
CDEC:BGP                98.245614
SNOTEL:549_NV_SNTL      94.736842
CDEC:PDS                54.385965
CDEC:VLC                50.877193
SNOTEL:302_OR_SNTL      49.122807
SNOTEL:1286_MT_SNTL     45.614035
CDEC:HIG                45.614035
CDEC:GEM                43.859649
SNOTEL:655_OR_SNTL      43.859649
CDEC:HRS                42.105263
SNOTEL:642_WA_SNTL      40.350877
CDEC:CWF                36.842105
CDEC:ADM                35.087719
CDEC:BLA                35.087719
CDEC:HGM                33.333333
CDEC:SHM                33.333333
SNOTEL:1138_NM_SNTL     33.333333
CDEC:DPO                31.578947
CDEC:MDW                31.578947
CDEC:LBD                29.824561
CDEC:CDP                29.824561
CDEC:FLL                28.070175
CDEC:BLD                28.070175
CDEC:BIM                26.315789
SNOTEL:984_WA_SNTL      26.315789
CDEC:BMW                26.315789
SNOTEL:911_WA_SNTL      26.315789
CDEC:RCC                22.807018
CDEC:CSL                22.807018
SNOTEL:817_WA_SNTL      22.807018
CDEC:WHW                22.807018
CDEC:LVM                22.807018
CDEC:CHP                22.807018
SNOTEL:648_WA_SNTL      21.052632
CDEC:BKL                21.052632
CDEC:HVN                21.052632
CDEC:BSK                21.052632
CDEC:TK2                21.052632
CDEC:MNT                19.298246
SNOTEL:672_WA_SNTL      19.298246
CDEC:CXS                19.298246
CDEC:LLP                19.298246
CDEC:SSM                17.543860
CDEC:RP2                17.543860
CDEC:GRM                15.789474
dtype: float64
```

Some of these stations have more than 50% of missing values.

In [76]:

```
# Create mask for stations more than 60% nulls
mask = ((gm_test_t.isna().sum().sort_values(ascending=False))*100/57)>40

#Filter and get the list
filter_list = list((gm_test_t.isna().sum().sort_values(ascending=False))[mask].index)
# filter_list.extend(['CDEC:AGP','CDEC:ADM','CDEC:BGP','CDEC:BIM','CDEC:BKL','CDEC:BLA','CDEC:HIG',
#                     'CDEC:LBD','CDEC:CHP','CDEC:GEM','CDEC:HGM','CDEC:BLD','CDEC:BMW','CDEC:FRW',
filter_list.extend(['CDEC:BIM','CDEC:RCC', 'CDEC:BKL','CDEC:LBD','CDEC:CHP','CDEC:BLA','CDEC:MNT','
                    'CDEC:SHM'])


#Drop the stations in the original dataframe
gm_test_feat = gm_test_feat[~(gm_test_feat['station_id'].isin(filter_list))]
gm_test_feat.head(50)
```

executed in 110ms, finished 18:29:16 2022-05-30

Out[76]:

	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	2021-05-25	2021-06-01
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	0.13	0.01
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	0.86	1.04
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	0.12	0.24
5	CDEC:BFL	5.76	7.08	9.72	10.32	10.20	10.80	10.08	10.20	10.56	...	12.36	5.04	0.12	0.12	0.12	0.00
10	CDEC:BLC	3.24	5.40	7.44	6.84	6.36	6.12	5.04	3.36	1.44	...	0.45	0.42	0.44	0.54	0.36	0.40
12	CDEC:BLK	12.50	13.50	14.60	15.20	15.30	16.00	16.10	15.90	15.70	...	13.40	9.40	2.30	0.40	0.40	NaN
13	CDEC:BLS	12.72	13.32	15.72	16.56	16.56	15.96	16.80	16.80	16.32	...	13.20	8.64	0.60	0.24	0.24	0.24
14	CDEC:BMW	9.00	9.20	10.20	11.20	11.20	10.70	10.30	10.40	10.00	...	2.50	NaN	NaN	NaN	0.00	NaN
15	CDEC:BNK	8.88	11.16	15.24	16.44	17.04	18.96	19.20	19.56	18.48	...	8.28	0.00	0.00	0.00	0.00	0.00
16	CDEC:BSK	12.40	12.60	13.60	14.20	14.40	14.00	14.40	14.90	13.70	...	4.00	0.00	0.00	0.00	0.00	NaN
17	CDEC:CDP	6.10	9.30	9.90	12.10	12.30	12.40	12.50	12.50	12.60	...	3.20	0.00	NaN	NaN	0.10	NaN
18	CDEC:CHM	10.08	10.32	11.76	12.24	12.48	12.84	12.24	12.48	11.64	...	7.44	1.68	0.00	0.00	0.00	0.00
20	CDEC:CRL	12.68	12.08	12.81	13.14	13.08	12.98	13.17	13.20	13.98	...	7.48	0.18	0.24	0.42	0.26	0.27
21	CDEC:CSL	12.00	14.10	16.30	17.50	17.30	16.80	16.40	15.80	14.00	...	5.40	0.20	0.10	0.10	0.20	NaN
22	CDEC:CSV	7.53	7.60	8.30	8.31	8.34	8.25	8.52	8.68	8.11	...	0.19	0.22	0.27	0.21	0.20	0.32
23	CDEC:CWD	7.49	7.22	7.66	7.86	7.69	7.49	6.98	8.29	10.12	...	1.78	0.00	0.00	0.00	0.00	0.00
24	CDEC:CWF	1.10	2.00	2.80	2.10	0.80	1.00	0.70	0.20	0.00	...	0.00	NaN	NaN	0.00	0.00	0.00
25	CDEC:CXS	14.10	14.40	15.50	16.00	16.20	16.20	16.00	16.40	15.90	...	8.80	3.40	NaN	NaN	NaN	NaN
26	CDEC:DAN	8.96	9.35	10.34	9.86	10.57	11.36	11.60	10.73	10.50	...	10.10	6.05	1.06	1.02	0.99	0.92
27	CDEC:DDM	5.59	5.59	6.30	6.84	6.73	6.84	6.41	6.35	6.57	...	13.12	10.12	2.86	0.24	0.18	0.13
28	CDEC:DPO	8.32	4.85	5.75	8.00	6.14	3.84	1.76	1.03	0.00	...	NaN	NaN	NaN	NaN	NaN	1.79
29	CDEC:DSS	11.70	16.20	18.50	22.80	23.30	23.60	25.20	25.60	25.20	...	16.40	12.40	7.20	0.00	0.00	NaN
30	CDEC:EBB	14.60	14.70	15.80	16.90	17.00	17.00	17.50	17.90	17.70	...	15.50	8.20	0.10	0.10	0.10	0.00
31	CDEC:EP5	15.60	16.50	17.40	17.90	18.30	19.50	19.60	19.30	18.30	...	8.70	0.50	0.00	0.00	NaN	NaN
32	CDEC:FDC	15.70	16.40	17.80	18.30	18.20	18.70	19.00	19.20	18.00	...	14.70	11.20	5.20	0.00	0.10	0.00
33	CDEC:FLL	2.00	2.30	3.30	3.30	3.10	3.00	2.60	0.50	0.30	...	0.00	0.00	0.00	0.00	0.00	NaN
34	CDEC:FRN	13.08	14.88	18.24	19.56	20.04	20.04	20.04	19.20	16.44	...	11.40	1.08	0.00	0.00	0.00	0.12
37	CDEC:GIN	7.80	8.76	9.84	10.56	10.44	7.86	10.26	9.48	7.08	...	6.06	0.06	0.00	0.00	0.00	0.00
38	CDEC:GKS	2.28	3.84	6.60	6.60	6.60	6.84	7.08	6.36	5.40	...	1.32	0.36	0.36	0.36	0.36	0.36
39	CDEC:GNF	4.96	4.95	5.09	5.13	5.06	4.59	1.94	0.85	0.74	...	0.76	0.74	0.72	0.71	0.69	0.70
40	CDEC:GNL	15.48	15.60	17.04	18.24	18.48	18.72	18.60	18.72	18.84	...	20.28	16.20	6.24	0.60	0.48	0.48
41	CDEC:GRM	9.72	9.36	9.60	9.84	9.72	9.24	9.24	6.72	5.28	...	1.31	0.73	0.70	0.61	0.68	0.70
42	CDEC:GRV	8.04	7.56	8.16	8.52	7.80	8.04	7.20	6.00	4.68	...	0.96	0.60	0.60	0.48	0.60	0.72

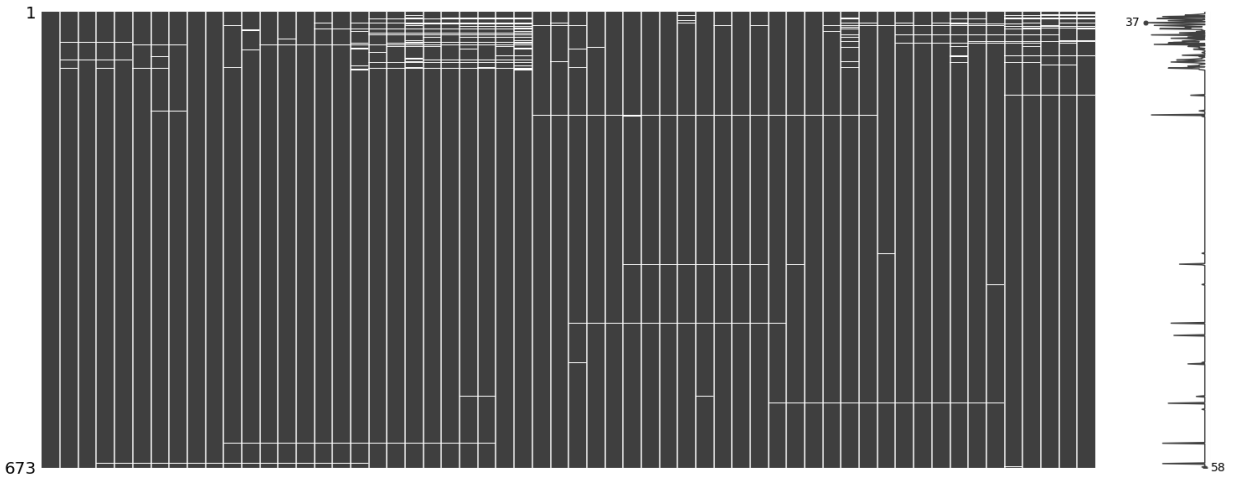
	station_id	2020-01-07	2020-01-14	2020-01-21	2020-01-28	2020-02-04	2020-02-11	2020-02-18	2020-02-25	2020-03-03	...	2021-04-27	2021-05-04	2021-05-11	2021-05-18	2021-05-25	2021-06-01
43	CDEC:GRZ	10.20	10.80	12.60	13.32	13.44	14.16	14.16	13.92	13.56	...	1.20	0.00	0.00	0.00	0.00	0.00
44	CDEC:HGM	7.90	7.90	9.10	9.80	10.00	10.20	9.80	8.60	7.70	...	NaN	NaN	NaN	NaN	NaN	NaN
46	CDEC:HMB	15.48	17.64	20.52	22.20	23.88	23.76	24.60	23.76	22.92	...	13.44	3.36	0.00	0.00	0.00	0.00
47	CDEC:HNT	7.80	7.56	8.52	9.12	8.88	8.76	8.52	8.88	9.48	...	0.24	0.24	0.24	0.12	0.12	0.12
48	CDEC:HOR	9.60	9.70	10.80	11.40	11.20	11.40	11.30	11.30	10.60	...	4.50	0.50	0.00	0.00	0.10	0.00
49	CDEC:HRK	8.19	10.43	11.77	12.33	13.23	13.43	12.76	12.25	10.80	...	1.48	0.04	0.07	0.00	0.07	0.00
51	CDEC:HVN	7.40	7.60	8.90	9.50	9.80	10.00	9.80	9.90	10.50	...	2.60	NaN	NaN	NaN	0.10	NaN
52	CDEC:HYS	10.80	11.88	14.04	16.08	15.84	15.72	16.92	16.56	16.44	...	16.32	12.00	4.44	0.24	0.12	0.00
53	CDEC:IDC	6.80	7.20	9.20	9.50	9.60	11.00	11.40	12.20	10.40	...	0.00	0.00	0.00	0.00	0.20	0.00
54	CDEC:IDP	13.10	13.80	15.40	16.60	16.50	16.80	16.80	16.90	17.40	...	23.20	20.80	17.70	14.20	13.70	7.70
55	CDEC:INN	5.00	4.30	5.60	5.10	4.90	4.30	6.80	6.40	4.80	...	0.20	0.10	0.20	NaN	NaN	NaN
56	CDEC:KTL	9.12	10.80	13.08	14.28	14.40	15.00	14.76	14.64	14.64	...	2.40	0.00	NaN	0.00	0.00	0.00
58	CDEC:LLP	NaN	NaN	NaN	NaN	36.26	36.55	37.48	37.67	38.83	...	46.48	44.35	36.99	24.54	17.84	1.44
59	CDEC:LVM	5.10	4.80	5.70	6.10	6.40	4.30	5.60	3.80	1.70	...	NaN	NaN	NaN	NaN	NaN	NaN
60	CDEC:LVT	18.50	19.00	22.00	23.00	22.80	22.90	23.00	23.10	23.80	...	35.70	30.70	25.50	19.40	17.20	9.40
61	CDEC:MDW	16.51	18.74	21.36	23.26	NaN	NaN	NaN	22.23	21.69	...	25.01	20.66	11.43	0.94	0.13	0.52
62	CDEC:MED	9.12	10.08	11.64	13.56	14.16	14.64	15.12	14.88	15.24	...	7.80	1.92	0.48	0.60	1.08	0.60

50 rows × 58 columns

In [77]:
```
import missingno as msno

msno.matrix(gm_test_feat)
```

executed in 334ms, finished 18:29:20 2022-05-30

Out[77]: <AxesSubplot:>



1.4.1.2 Interpolate missing values with Spline interpolation

In [78]:
```
# Transpose and set station id as columns
gm_test_t=gm_test_feat.T
gm_test_t.columns = gm_test_t.iloc[0]
gm_test_t = gm_test_t[1:]
gm_test_t
```

executed in 59ms, finished 18:29:23 2022-05-30

Out[78]:

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2020-01-07	12.72	12.2	6.6	5.76	3.24	12.5	12.72	9.0	8.88
2020-01-14	13.78	12.2	5.76	7.08	5.4	13.5	13.32	9.2	11.16
2020-01-21	17.12	13.3	5.16	9.72	7.44	14.6	15.72	10.2	15.24
2020-01-28	18.07	13.35	7.68	10.32	6.84	15.2	16.56	11.2	16.44
2020-02-04	18.17	12.85	4.68	10.2	6.36	15.3	16.56	11.2	17.04
2020-02-11	18.38	12.72	1.32	10.8	6.12	16.0	15.96	10.7	18.96
2020-02-18	17.71	12.72	0.84	10.08	5.04	16.1	16.8	10.3	19.2
2020-02-25	16.05	12.8	0.84	10.2	3.36	15.9	16.8	10.4	19.56
2020-03-03	14.62	13.16	0.24	10.56	1.44	15.7	16.32	10.0	18.48
2020-03-10	12.79	12.82	0.24	10.8	1.92	15.6	15.84	12.0	17.16
2020-03-17	16.3	14.33	0.84	11.4	9.36	20.2	22.8	14.9	17.16
2020-03-24	17.52	14.88	1.2	11.76	9.72	21.3	24.36	16.2	18.24
2020-03-31	19.35	16.0	0.6	12.48	10.92	23.1	27.0	17.3	19.2
2020-04-07	22.31	19.29	0.84	13.08	15.24	26.6	30.84	17.9	26.04
2020-04-14	21.04	19.14	0.84	10.08	11.76	27.4	30.12	17.0	24.72
2020-04-21	18.58	20.1	0.72	5.28	7.2	25.6	28.68	15.0	20.28
2020-04-28	12.68	18.84	0.36	0.72	1.56	21.2	24.0	9.8	6.36
2020-05-05	2.83	14.36	0.24	0.72	1.44	16.9	18.6	2.4	0.0
2020-05-12	0.14	9.32	0.12	0.72	1.8	10.3	7.56	NaN	0.0
2020-05-19	1.2	6.69	0.48	0.72	1.92	NaN	1.44	NaN	0.12
2020-05-26	0.23	3.13	0.36	0.72	1.68	2.1	0.36	NaN	0.0
2020-06-02	0.2	1.21	0.36	0.84	1.68	0.3	0.12	NaN	0.12
2020-06-09	0.0	1.22	0.12	0.72	1.32	0.2	0.12	0.0	0.0
2020-06-16	0.0	1.32	0.24	0.72	0.0	0.1	0.12	NaN	0.0
2020-06-23	0.01	1.41	0.36	0.84	0.24	0.1	0.24	NaN	0.12
2020-06-30	0.0	1.39	0.12	0.72	0.24	0.1	0.12	NaN	0.0

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2020-12-01	4.53	6.96	0.12	0.12	11.8	3.3	1.8	2.5	0.0
2020-12-08	4.35	7.06	0.24	0.72	11.81	3.3	1.8	2.6	0.0
2020-12-15	5.83	8.11	0.24	1.2	0.63	5.1	3.24	3.6	0.6
2020-12-22	7.66	8.62	2.4	2.64	1.47	5.8	4.08	4.2	1.32
2020-12-29	8.98	9.65	0.84	4.44	2.69	7.0	5.4	4.9	2.76
2021-01-05	9.65	9.36	1.32	6.36	2.52	7.8	6.48	6.0	5.4
2021-01-12	10.2	9.13	1.44	7.68	2.45	8.0	6.72	5.9	6.6
2021-01-19	8.43	9.33	0.48	7.8	0.96	7.5	5.4	6.1	6.48
2021-01-26	9.93	9.78	0.84	8.28	2.62	NaN	6.12	6.2	6.48
2021-02-02	17.68	17.56	2.52	12.48	9.51	14.4	15.24	10.0	13.68
2021-02-09	17.33	17.47	2.4	12.96	10.06	14.5	15.84	10.5	15.36
2021-02-16	19.56	19.12	2.88	15.36	12.11	16.5	17.52	12.0	17.16
2021-02-23	20.31	19.11	1.56	17.04	12.52	17.1	17.64	12.5	18.48
2021-03-02	19.9	19.01	0.36	16.92	11.91	17.5	17.28	12.3	18.84
2021-03-09	20.07	18.96	0.12	18.48	10.72	17.5	17.64	13.0	19.44
2021-03-16	22.11	20.61	0.96	20.64	13.46	19.6	19.68	13.7	21.6
2021-03-23	23.81	21.95	0.72	22.2	12.93	21.1	20.52	14.6	23.28
2021-03-30	23.23	20.14	0.24	22.68	9.5	21.7	20.52	14.1	22.8
2021-04-06	18.13	19.08	0.24	21.12	3.98	19.9	18.96	11.0	20.52
2021-04-13	12.4	17.94	0.12	18.84	0.31	17.1	17.16	8.4	17.4
2021-04-20	8.01	16.82	0.24	15.48	0.33	15.2	15.0	7.1	12.84
2021-04-27	3.34	15.44	0.12	12.36	0.45	13.4	13.2	2.5	8.28
2021-05-04	0.31	11.94	0.24	5.04	0.42	9.4	8.64	NaN	0.0
2021-05-11	0.02	5.91	0.24	0.12	0.44	2.3	0.6	NaN	0.0
2021-05-18	0.0	1.1	0.24	0.12	0.54	0.4	0.24	NaN	0.0
2021-05-25	0.13	0.86	0.12	0.12	0.36	0.4	0.24	0.0	0.0
2021-06-01	0.01	1.04	0.24	0.0	0.4	NaN	0.24	NaN	0.0
2021-06-08	0.0	1.12	0.12	0.0	0.34	NaN	0.0	NaN	0.0
2021-06-15	0.0	1.04	0.24	0.12	NaN	NaN	0.0	NaN	0.0

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2021-06-22	0.0	1.17	0.24	0.0	NaN	NaN	0.12	NaN	0.0
2021-06-29	0.0	1.25	0.36	0.12	NaN	NaN	0.24	NaN	0.0

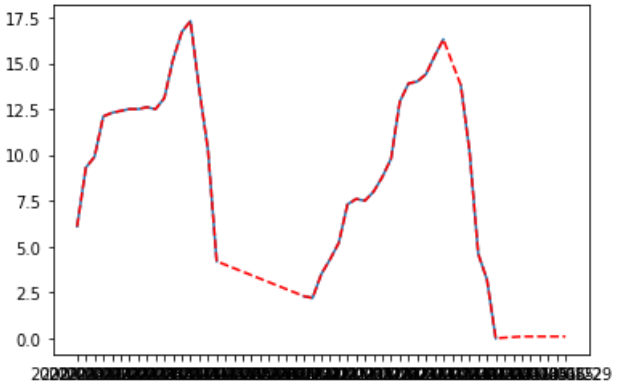
57 rows × 673 columns

In [79]:

```
#Set date to datetime and as index
# gm_t.index = pd.to_datetime(gm_t.index)

#Test interpolation code on 1 station
plt.plot(gm_test_t ['CDEC:CDP'])
interpolated = gm_test_t['CDEC:CDP'].astype(float).interpolate(option='spline')

plt.plot(interpolated, color='r', linestyle='--')
```

executed in 814ms, finished 18:29:25 2022-05-30

Out[79]: [<matplotlib.lines.Line2D at 0x15000611060>]



In [80]:
```
#Get list of our columns
stations_list = list(gm_test_t.columns)


#Apply interpolation to every station
for station in stations_list:
    gm_test_t[station] = gm_test_t[station].astype(float).interpolate(option='spline', limit_direct

#Sniff test
gm_test_t
```

executed in 658ms, finished 18:29:27 2022-05-30

Out[80]:

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2020-01-07	12.72	12.20	6.60	5.76	3.24	12.50	12.72	9.000	8.88
2020-01-14	13.78	12.20	5.76	7.08	5.40	13.50	13.32	9.200	11.16
2020-01-21	17.12	13.30	5.16	9.72	7.44	14.60	15.72	10.200	15.24
2020-01-28	18.07	13.35	7.68	10.32	6.84	15.20	16.56	11.200	16.44
2020-02-04	18.17	12.85	4.68	10.20	6.36	15.30	16.56	11.200	17.04
2020-02-11	18.38	12.72	1.32	10.80	6.12	16.00	15.96	10.700	18.96
2020-02-18	17.71	12.72	0.84	10.08	5.04	16.10	16.80	10.300	19.20
2020-02-25	16.05	12.80	0.84	10.20	3.36	15.90	16.80	10.400	19.56
2020-03-03	14.62	13.16	0.24	10.56	1.44	15.70	16.32	10.000	18.48
2020-03-10	12.79	12.82	0.24	10.80	1.92	15.60	15.84	12.000	17.16
2020-03-17	16.30	14.33	0.84	11.40	9.36	20.20	22.80	14.900	17.16
2020-03-24	17.52	14.88	1.20	11.76	9.72	21.30	24.36	16.200	18.24
2020-03-31	19.35	16.00	0.60	12.48	10.92	23.10	27.00	17.300	19.20
2020-04-07	22.31	19.29	0.84	13.08	15.24	26.60	30.84	17.900	26.04
2020-04-14	21.04	19.14	0.84	10.08	11.76	27.40	30.12	17.000	24.72
2020-04-21	18.58	20.10	0.72	5.28	7.20	25.60	28.68	15.000	20.28
2020-04-28	12.68	18.84	0.36	0.72	1.56	21.20	24.00	9.800	6.36
2020-05-05	2.83	14.36	0.24	0.72	1.44	16.90	18.60	2.400	0.00
2020-05-12	0.14	9.32	0.12	0.72	1.80	10.30	7.56	1.920	0.00
2020-05-19	1.20	6.69	0.48	0.72	1.92	6.20	1.44	1.440	0.12
2020-05-26	0.23	3.13	0.36	0.72	1.68	2.10	0.36	0.960	0.00
2020-06-02	0.20	1.21	0.36	0.84	1.68	0.30	0.12	0.480	0.12
2020-06-09	0.00	1.22	0.12	0.72	1.32	0.20	0.12	0.000	0.00
2020-06-16	0.00	1.32	0.24	0.72	0.00	0.10	0.12	0.625	0.00

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2020-06-23	0.01	1.41	0.36	0.84	0.24	0.10	0.24	1.250	0.12
2020-06-30	0.00	1.39	0.12	0.72	0.24	0.10	0.12	1.875	0.00
2020-12-01	4.53	6.96	0.12	0.12	11.80	3.30	1.80	2.500	0.00
2020-12-08	4.35	7.06	0.24	0.72	11.81	3.30	1.80	2.600	0.00
2020-12-15	5.83	8.11	0.24	1.20	0.63	5.10	3.24	3.600	0.60
2020-12-22	7.66	8.62	2.40	2.64	1.47	5.80	4.08	4.200	1.32
2020-12-29	8.98	9.65	0.84	4.44	2.69	7.00	5.40	4.900	2.76
2021-01-05	9.65	9.36	1.32	6.36	2.52	7.80	6.48	6.000	5.40
2021-01-12	10.20	9.13	1.44	7.68	2.45	8.00	6.72	5.900	6.60
2021-01-19	8.43	9.33	0.48	7.80	0.96	7.50	5.40	6.100	6.48
2021-01-26	9.93	9.78	0.84	8.28	2.62	10.95	6.12	6.200	6.48
2021-02-02	17.68	17.56	2.52	12.48	9.51	14.40	15.24	10.000	13.68
2021-02-09	17.33	17.47	2.40	12.96	10.06	14.50	15.84	10.500	15.36
2021-02-16	19.56	19.12	2.88	15.36	12.11	16.50	17.52	12.000	17.16
2021-02-23	20.31	19.11	1.56	17.04	12.52	17.10	17.64	12.500	18.48
2021-03-02	19.90	19.01	0.36	16.92	11.91	17.50	17.28	12.300	18.84
2021-03-09	20.07	18.96	0.12	18.48	10.72	17.50	17.64	13.000	19.44
2021-03-16	22.11	20.61	0.96	20.64	13.46	19.60	19.68	13.700	21.60
2021-03-23	23.81	21.95	0.72	22.20	12.93	21.10	20.52	14.600	23.28
2021-03-30	23.23	20.14	0.24	22.68	9.50	21.70	20.52	14.100	22.80
2021-04-06	18.13	19.08	0.24	21.12	3.98	19.90	18.96	11.000	20.52
2021-04-13	12.40	17.94	0.12	18.84	0.31	17.10	17.16	8.400	17.40
2021-04-20	8.01	16.82	0.24	15.48	0.33	15.20	15.00	7.100	12.84
2021-04-27	3.34	15.44	0.12	12.36	0.45	13.40	13.20	2.500	8.28
2021-05-04	0.31	11.94	0.24	5.04	0.42	9.40	8.64	1.875	0.00
2021-05-11	0.02	5.91	0.24	0.12	0.44	2.30	0.60	1.250	0.00
2021-05-18	0.00	1.10	0.24	0.12	0.54	0.40	0.24	0.625	0.00
2021-05-25	0.13	0.86	0.12	0.12	0.36	0.40	0.24	0.000	0.00
2021-06-01	0.01	1.04	0.24	0.00	0.40	NaN	0.24	NaN	0.00

station_id	CDEC:ALP	CDEC:BCB	CDEC:BCH	CDEC:BFL	CDEC:BLC	CDEC:BLK	CDEC:BLS	CDEC:BMW	CDEC:BNK
2021-06-08	0.00	1.12	0.12	0.00	0.34	NaN	0.00	NaN	0.00
2021-06-15	0.00	1.04	0.24	0.12	NaN	NaN	0.00	NaN	0.00
2021-06-22	0.00	1.17	0.24	0.00	NaN	NaN	0.12	NaN	0.00
2021-06-29	0.00	1.25	0.36	0.12	NaN	NaN	0.24	NaN	0.00

57 rows × 673 columns

In [81]:
```
#Look at percentage of null values
(gm_test_t.isna().sum().sort_values(ascending=False)[:50])*100/57
```

executed in 81ms, finished 18:29:31 2022-05-30

Out[81]:
```
station_id
CDEC:INN               12.280702
CDEC:EP5               10.526316
CDEC:HVN                8.771930
CDEC:BSK                8.771930
CDEC:CSL                8.771930
CDEC:SDW                8.771930
CDEC:CDP                8.771930
CDEC:DSS                8.771930
CDEC:BMW                8.771930
CDEC:VRG                8.771930
CDEC:BLK                8.771930
SNOTEL:1084_OR_SNTL     8.771930
CDEC:FDC                7.017544
CDEC:BLC                5.263158
CDEC:IDP                3.508772
CDEC:SQV                3.508772
CDEC:CWF                3.508772
SNOTEL:734_WA_SNTL      3.508772
CDEC:FLL                1.754386
SNOTEL:640_AZ_SNTL      0.000000
SNOTEL:643_UT_SNTL      0.000000
SNOTEL:637_ID_SNTL      0.000000
SNOTEL:646_MT_SNTL      0.000000
SNOTEL:644_WA_SNTL      0.000000
SNOTEL:645_ID_SNTL      0.000000
SNOTEL:629_CO_SNTL      0.000000
SNOTEL:647_OR_SNTL      0.000000
SNOTEL:648_WA_SNTL      0.000000
SNOTEL:649_MT_SNTL      0.000000
SNOTEL:650_ID_SNTL      0.000000
SNOTEL:651_OR_SNTL      0.000000
SNOTEL:652_NV_SNTL      0.000000
SNOTEL:654_ID_SNTL      0.000000
SNOTEL:633_CA_SNTL      0.000000
SNOTEL:623_ID_SNTL      0.000000
SNOTEL:628_UT_SNTL      0.000000
SNOTEL:615_NV_SNTL      0.000000
SNOTEL:604_MT_SNTL      0.000000
SNOTEL:605_OR_SNTL      0.000000
SNOTEL:606_WA_SNTL      0.000000
SNOTEL:607_CO_SNTL      0.000000
SNOTEL:610_ID_SNTL      0.000000
SNOTEL:612_UT_SNTL      0.000000
SNOTEL:613_MT_SNTL      0.000000
SNOTEL:614_OR_SNTL      0.000000
SNOTEL:616_WY_SNTL      0.000000
SNOTEL:627_ID_SNTL      0.000000
SNOTEL:617_AZ_SNTL      0.000000
SNOTEL:619_OR_SNTL      0.000000
SNOTEL:620_ID_SNTL      0.000000
dtype: float64
```

In [82]:
```
#Check for patterns in the missing values

dump_list = list((gm_test_t.isna().sum().sort_values(ascending=False)[:9]).index)

gm_test_t[dump_list]
```

executed in 131ms, finished 18:29:35 2022-05-30

Out[82]:

station_id	CDEC:INN	CDEC:EP5	CDEC:HVN	CDEC:BSK	CDEC:CSL	CDEC:SDW	CDEC:CDP	CDEC:DSS	CDEC:BMW
2020-01-07	5.0	15.6	7.400000	12.400000	12.000000	6.700000	6.100000	11.70	9.000
2020-01-14	4.3	16.5	7.600000	12.600000	14.100000	6.700000	9.300000	16.20	9.200
2020-01-21	5.6	17.4	8.900000	13.600000	16.300000	7.500000	9.900000	18.50	10.200
2020-01-28	5.1	17.9	9.500000	14.200000	17.500000	7.900000	12.100000	22.80	11.200
2020-02-04	4.9	18.3	9.800000	14.400000	17.300000	7.800000	12.300000	23.30	11.200
2020-02-11	4.3	19.5	10.000000	14.000000	16.800000	8.100000	12.400000	23.60	10.700
2020-02-18	6.8	19.6	9.800000	14.400000	16.400000	8.300000	12.500000	25.20	10.300
2020-02-25	6.4	19.3	9.900000	14.900000	15.800000	8.500000	12.500000	25.60	10.400
2020-03-03	4.8	18.3	10.500000	13.700000	14.000000	8.700000	12.600000	25.20	10.000
2020-03-10	2.5	17.3	9.900000	13.100000	12.700000	9.900000	12.500000	25.10	12.000
2020-03-17	4.1	22.5	13.200000	17.200000	18.500000	11.700000	13.100000	26.80	14.900
2020-03-24	5.0	23.9	13.800000	17.500000	19.000000	11.700000	15.200000	27.90	16.200
2020-03-31	6.2	26.0	16.000000	17.800000	20.000000	11.800000	16.700000	30.70	17.300
2020-04-07	7.3	28.7	16.900000	19.700000	23.300000	15.300000	17.300000	30.80	17.900
2020-04-14	5.4	27.5	16.200000	19.600000	19.600000	15.900000	13.600000	28.80	17.000
2020-04-21	0.9	24.3	14.100000	17.500000	15.800000	14.800000	10.300000	27.00	15.000
2020-04-28	0.1	15.2	8.600000	12.000000	8.600000	10.900000	4.200000	22.30	9.800
2020-05-05	0.0	4.8	1.100000	5.400000	1.800000	5.400000	4.010000	17.30	2.400
2020-05-12	0.1	0.0	0.000000	0.100000	2.166667	0.000000	3.820000	12.00	1.920
2020-05-19	0.2	1.1	0.500000	0.050000	2.533333	0.100000	3.630000	13.60	1.440
2020-05-26	0.1	0.0	0.000000	0.000000	2.900000	0.000000	3.440000	9.20	0.960
2020-06-02	0.1	0.0	0.000000	0.533333	3.266667	0.000000	3.250000	0.00	0.480
2020-06-09	0.1	0.0	0.000000	1.066667	3.633333	0.000000	3.060000	0.88	0.000
2020-06-16	1.6	0.0	0.000000	1.600000	4.000000	0.000000	2.870000	1.76	0.625
2020-06-23	0.1	0.0	1.433333	2.133333	4.366667	0.433333	2.680000	2.64	1.250
2020-06-30	0.0	0.0	2.866667	2.666667	4.733333	0.866667	2.490000	3.52	1.875
2020-12-01	0.9	6.3	4.300000	3.200000	5.100000	1.300000	2.300000	4.40	2.500
2020-12-08	1.2	6.2	4.400000	3.200000	5.200000	1.200000	2.200000	4.90	2.600
2020-12-15	1.9	8.7	5.600000	4.500000	7.400000	2.200000	3.500000	6.00	3.600
2020-12-22	2.6	10.1	6.500000	5.000000	8.500000	3.100000	4.300000	7.30	4.200
2020-12-29	3.2	11.3	6.900000	6.300000	10.700000	3.600000	5.200000	8.00	4.900
2021-01-05	3.8	13.1	7.300000	6.900000	13.300000	3.900000	7.300000	10.60	6.000
2021-01-12	3.7	13.3	7.400000	6.800000	13.100000	3.800000	7.600000	11.30	5.900
2021-01-19	4.7	13.5	7.200000	6.500000	12.500000	3.700000	7.500000	11.40	6.100
2021-01-26	3.7	13.9	8.200000	6.900000	12.600000	4.100000	8.000000	11.60	6.200
2021-02-02	6.7	19.9	11.500000	13.200000	20.000000	9.900000	8.800000	14.90	10.000
2021-02-09	6.8	21.7	11.800000	13.100000	21.400000	9.800000	9.800000	16.00	10.500
2021-02-16	6.8	25.8	13.500000	15.100000	25.500000	11.000000	12.900000	19.80	12.000
2021-02-23	6.5	27.3	13.900000	14.900000	26.600000	11.100000	13.900000	21.80	12.500
2021-03-02	7.3	28.0	13.700000	14.700000	26.200000	10.900000	14.000000	22.20	12.300

station_id	CDEC:INN	CDEC:EP5	CDEC:HVN	CDEC:BSK	CDEC:CSL	CDEC:SDW	CDEC:CDP	CDEC:DSS	CDEC:BMW
2021-03-09	7.9	28.1	13.700000	14.500000	25.900000	11.000000	14.400000	22.40	13.000
2021-03-16	9.9	30.3	15.300000	15.900000	27.700000	11.900000	15.400000	23.30	13.700
2021-03-23	8.5	32.5	16.000000	17.200000	29.600000	13.800000	16.300000	24.30	14.600
2021-03-30	6.3	31.2	14.500000	15.350000	27.600000	14.600000	15.050000	23.90	14.100
2021-04-06	4.1	24.6	13.000000	13.500000	21.300000	12.900000	13.800000	20.70	11.000
2021-04-13	0.3	18.2	9.100000	9.600000	14.900000	10.200000	10.200000	18.20	8.400
2021-04-20	0.1	14.6	6.000000	7.500000	10.200000	8.900000	4.600000	14.80	7.100
2021-04-27	0.2	8.7	2.600000	4.000000	5.400000	6.900000	3.200000	16.40	2.500
2021-05-04	0.1	0.5	1.975000	0.000000	0.200000	2.000000	0.000000	12.40	1.875
2021-05-11	0.2	0.0	1.350000	0.000000	0.100000	1.000000	0.033333	7.20	1.250
2021-05-18	NaN	0.0	0.725000	0.000000	0.100000	0.000000	0.066667	0.00	0.625
2021-05-25	NaN	NaN	0.100000	0.000000	0.200000	0.000000	0.100000	0.00	0.000
2021-06-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-15	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2021-06-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

So the remaining stations that have missing values are missing values in the tail end of 2021's melt season. We'll leave this be and impute these values with a more robut imputation method in our modeling workflow.

In [83]:
```
#Transpose back to our original dataframe shape.
gm_test_feat=gm_test_t.T
gm_test_feat.reset_index(inplace=True)
gm_test_feat
```

executed in 48ms, finished 18:29:38 2022-05-30

Out[83]:

	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: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	
1	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	
2	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	
3	CDEC:BFL	5.76	7.08	9.72	10.32	10.20	10.80	10.08	10.20	10.56	...	12.36	5.04	0.12	0.12	
4	CDEC:BLC	3.24	5.40	7.44	6.84	6.36	6.12	5.04	3.36	1.44	...	0.45	0.42	0.44	0.54	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
668	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	
669	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
670	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	
671	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
672	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

673 rows × 58 columns

1.4.2 Testing labels

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 [86]:
```
test_labels = pd.read_csv("../data/labels_2020_2021.csv")
test_labels = test_labels.melt(id_vars=["cell_id"]).dropna()
test_labels.rename({'variable':'dates'},axis=1,inplace=True)
test_labels
```

executed in 287ms, finished 18:30:50 2022-05-30

Out[86]:

	cell_id	dates	value
29	00c4db22-a423-41a4-ada6-a8b1b04153a4	2020-01-07	8.0
64	018cf1a1-f945-4097-9c47-0c4690538bb5	2020-01-07	11.4
68	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2020-01-07	18.5
93	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2020-01-07	3.9
95	02cf33c2-c8e2-48b9-bf72-92506e97e251	2020-01-07	13.0
...	...	...	...
516655	fd4492f2-8aa9-4279-bdc0-73991786943f	2021-06-29	0.0
516678	fde3221a-9ce3-45a9-857f-bd196b07aa05	2021-06-29	0.0
516680	fdeb8912-f9d1-445d-aadb-e943534f67fe	2021-06-29	0.0
516693	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2021-06-29	0.0
516724	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2021-06-29	0.0

45241 rows × 3 columns

1.4.2.1 Add nearest neighboring ground stations to test set

Add neighbors values to our dataframe.

In [87]:
```
#Function to help us pull nearest neighbor list

def get_neighb_list(df_idx,label_df, grid_df):
    """
    Get the neareset neighbor list for the row
    """
    cell_id = label_df['cell_id'].iloc[df_idx]
    nn_list = grid_df[grid_df['cell_id'] == cell_id]['nearest_neighb'].values[0]

    return nn_list

def get_gm_value(df_idx, station_idx, gm_df,label_df,nn_list):
    value = gm_df[gm_df['station_id'] == nn_list[station_idx]][label_df['dates'].iloc[df_idx]].valu
    return value
```

executed in 29ms, finished 18:30:52 2022-05-30

In [107]:
```
#Test function
nn_list = get_neighb_list(0,test_labels,grid_cell_df)
get_gm_value(0,0,gm_test_feat,test_labels,nn_list)
```

executed in 16ms, finished 18:34:54 2022-05-30

Out[107]: 1.2

In [108]:

```
for k in range (0,20):
    print (f"Working on neighbor {k+1}")
    neighbor=[]
    for i in range(0,len(test_labels)):
        nn_list = get_neighb_list(i,test_labels,grid_cell_df)
        try:
            value = get_gm_value(i,k,gm_test_feat,test_labels,nn_list)
        except:
            value = np.NaN
        neighbor.append(value)
#         print (neighbor)
    col_name = f'neighbor_{k+1}'
    test_labels[col_name] = neighbor
```

executed in 37m 19s, finished 19:12:30 2022-05-30

```
Working on neighbor 1
Working on neighbor 2
Working on neighbor 3
Working on neighbor 4
Working on neighbor 5
Working on neighbor 6
Working on neighbor 7
Working on neighbor 8
Working on neighbor 9
Working on neighbor 10
Working on neighbor 11
Working on neighbor 12
Working on neighbor 13
Working on neighbor 14
Working on neighbor 15
Working on neighbor 16
Working on neighbor 17
Working on neighbor 18
Working on neighbor 19
Working on neighbor 20
```

In [109]:
```
test_labels
```
executed in 88ms, finished 19:21:20 2022-05-30

Out[109]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
29	00c4db22-a423-41a4-ada6-a8b1b04153a4	2020-01-07	8.0	1.2	2.8	3.7	6.7	2.1	1.6	2.8
64	018cf1a1-f945-4097-9c47-0c4690538bb5	2020-01-07	11.4	1.2	2.8	3.7	6.7	2.1	1.6	2.8
68	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2020-01-07	18.5	1.2	2.8	3.7	6.7	2.1	1.6	2.8
93	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2020-01-07	3.9	1.2	2.8	6.7	3.7	2.1	1.6	2.8
95	02cf33c2-c8e2-48b9-bf72-92506e97e251	2020-01-07	13.0	1.2	2.8	6.7	3.7	2.1	1.6	2.8
...	...	...	...	...	...	...	...	...	...	...
516655	fd4492f2-8aa9-4279-bdc0-73991786943f	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516678	fde3221a-9ce3-45a9-857f-bd196b07aa05	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516680	fdeb8912-f9d1-445d-aadb-e943534f67fe	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516693	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516724	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

45241 rows × 23 columns

Let's check for null is our new dataframe.

In [110]:
```
test_labels.isna().sum()
```
executed in 46ms, finished 19:21:29 2022-05-30

Out[110]:
```
cell_id        0
dates          0
value          0
neighbor_1     0
neighbor_2     0
neighbor_3     0
neighbor_4     0
neighbor_5     0
neighbor_6     0
neighbor_7     0
neighbor_8     0
neighbor_9     0
neighbor_10    0
neighbor_11    0
neighbor_12    0
neighbor_13    0
neighbor_14    0
neighbor_15    0
neighbor_16    0
neighbor_17    0
neighbor_18    0
neighbor_19    0
neighbor_20    0
dtype: int64
```

Cool!! No nulls for our test labels. Let's proceed to adding geospatial data to our dataframe.

1.4.2.2 Geospatial Data

In [115]:
```
test_labels['latitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr_lat'].valu
test_labels['longitude'] = [round(grid_cell_df[grid_cell_df['cell_id'] == cell_id]['centr_lon'].val
test_labels['region'] = [grid_cell_df[grid_cell_df['cell_id'] == cell_id]['region'].values[0] for c
test_labels
```

executed in 4m 50s, finished 19:28:32 2022-05-30

Out[115]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7
29	00c4db22-a423-41a4-ada6-a8b1b04153a4	2020-01-07	8.0	1.2	2.8	3.7	6.7	2.1	1.6	2.8
64	018cf1a1-f945-4097-9c47-0c4690538bb5	2020-01-07	11.4	1.2	2.8	3.7	6.7	2.1	1.6	2.8
68	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2020-01-07	18.5	1.2	2.8	3.7	6.7	2.1	1.6	2.8
93	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2020-01-07	3.9	1.2	2.8	6.7	3.7	2.1	1.6	2.8
95	02cf33c2-c8e2-48b9-bf72-92506e97e251	2020-01-07	13.0	1.2	2.8	6.7	3.7	2.1	1.6	2.8
...	...	...	...	...	...	...	...	...	...	...
516655	fd4492f2-8aa9-4279-bdc0-73991786943f	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516678	fde3221a-9ce3-45a9-857f-bd196b07aa05	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516680	fdeb8912-f9d1-445d-aadb-e943534f67fe	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516693	fe33672e-7ea7-4c5d-8639-96b2cc7edb0c	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
516724	ff01e8c2-19a2-4a89-af0e-608b8f40ad5f	2021-06-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

45241 rows × 26 columns

In [116]:
```
#Saving dataframe to file for now, just in case.
test_labels.to_csv('../data/test_labels_final.csv')
```

executed in 932ms, finished 19:54:48 2022-05-30

Type *Markdown* and LaTeX: α^2

1.5 EDA

Let's visualize the locations of our data so we know where in the map we are working in and if there's any pattern we could catch.

1.5.0.1 Ground Measure stations

I would like to explore the historical mean of SWE of our ground measure stations and see it on a map.

In [27]:
```
# Transpose and set station id as columns
gm_t=gm_train_feat.T
gm_t.columns = gm_t.iloc[0]
gm_t = gm_t[1:]

# Get the mean & transform into a dataframe.
gm_t_mean = gm_t.mean()
gm_t_mean = pd.DataFrame(gm_t_mean).reset_index()
gm_t_mean
```

executed in 129ms, finished 08:05:25 2022-06-01

Out[27]:

	station_id	0
0	CDEC:ADM	4.665962
1	CDEC:AGP	9.787230
2	CDEC:ALP	13.884484
3	CDEC:BCH	3.173521
4	CDEC:BFL	6.302254
...	...	...
680	SNOTEL:989_ID_SNTL	7.514554
681	SNOTEL:990_WA_SNTL	21.769014
682	SNOTEL:992_UT_SNTL	3.905164
683	SNOTEL:998_WA_SNTL	55.166197
684	SNOTEL:999_WA_SNTL	29.232394

685 rows × 2 columns

Let's merge the mean dataframe with our ground measure metadata dataframe.

In [28]:
```
gm_md_mean = gm_md.merge(gm_t_mean,on='station_id')
gm_md_mean = gm_md_mean.rename(columns = {0:'historical_mean'})
gm_md_mean = gm_md_mean.set_crs('epsg:4326')
gm_md_mean
```

executed in 53ms, finished 08:05:27 2022-06-01

Out[28]:

	station_id	name	elevation_m	latitude	longitude	state	geometry	historical_mean
0	CDEC:ADM	Adin Mountain	1889.760000	41.237000	-120.792000	California	POINT (-120.79200 41.23700)	4.665962
1	CDEC:AGP	Agnew Pass	2880.360000	37.726631	-119.141731	California	POINT (-119.14173 37.72663)	9.787230
2	CDEC:ALP	Alpha (Smud)	2316.480000	38.804192	-120.215652	California	POINT (-120.21565 38.80419)	13.884484
3	CDEC:BCH	Beach Meadows	2331.720000	36.126095	-118.293457	California	POINT (-118.29346 36.12609)	3.173521
4	CDEC:BFL	Big Flat	1554.480000	41.077599	-122.942230	California	POINT (-122.94223 41.07760)	6.302254
...	...	...	...	...	...	...	...	...
680	SNOTEL:989_ID_SNTL	Moscow Mountain	1432.560059	46.805000	-116.853500	Idaho	POINT (-116.85350 46.80500)	7.514554
681	SNOTEL:990_WA_SNTL	Beaver Pass	1106.423950	48.879299	-121.255501	Washington	POINT (-121.25550 48.87930)	21.769014
682	SNOTEL:992_UT_SNTL	Bear River RS	2675.229492	40.885201	-110.827698	Utah	POINT (-110.82770 40.88520)	3.905164
683	SNOTEL:998_WA_SNTL	Easy Pass	1606.296021	48.859329	-121.438950	Washington	POINT (-121.43895 48.85933)	55.166197
684	SNOTEL:999_WA_SNTL	Marten Ridge	1072.895996	48.762920	-121.698227	Washington	POINT (-121.69823 48.76292)	29.232394

685 rows × 8 columns

1.5.0.2 Grid cells

Let's explore the grid cells as well. We'll begin with transforming our grid cell as geodataframes.

In [29]:
```
#transform grid cell data to geopandas df

geometry = [Polygon(eval(str(xy_string))) for xy_string in grid_cell_df['coordinates']]
grid_cell_df = gpd.GeoDataFrame(grid_cell_df,geometry=geometry)
grid_cell_df = grid_cell_df.set_crs('epsg:4326')
```

executed in 1.17s, finished 08:05:30 2022-06-01

In [30]:
```
grid_cell_df
```
executed in 43ms, finished 08:05:30 2022-06-01

3	000ba8d9-d6d5-48da-84a2-1fa54951fae1	[[-119.320824, 37.431707], [-119.320824, 37.43...	sierras	-119.324418	37.434560	POLYGON ((-119.32082 37.43171, -119.32082 37.4...
4	00146204-d4e9-4cd8-8f86-d1ef133c5b6d	[[-118.521324, 36.657353], [-118.521324, 36.66...	sierras	-118.524917	36.660236	POLYGON ((-118.52132 36.65735, -118.52132 36.6...
...	...	...	...	...	...	...
18125	ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	[[-118.620138, 37.117184], [-118.620138, 37.12...	sierras	-118.623732	37.120049	POLYGON ((-118.62014 37.11718, -118.62014 37.1...
18126	ffe43514-2c92-43b6-bd84-d183806aca65	[[-123.49799, 47.901318], [-123.49799, 47.9073...	other	-123.501584	47.903727	POLYGON ((-123.49799 47.90132, -123.49799 47.9...
18127	ffeabc13-7c6f-4b63-b043-19c8f15e0345	[[-119.644218, 37.879756], [-119.644218, 37.88...	sierras	-119.647811	37.882592	POLYGON ((-119.64422 37.87976, -119.64422 37.8...
18128	fff95195-ccc9-40b7-b302-a0d8570c86bc	[[-123.372226, 47.732416], [-123.372226, 47.73...	other	-123.375819	47.734833	POLYGON ((-123.37223 47.73242, -123.37223 47.7...

Then, we'll fine the historical mean (2013-2019) of SWE of our grid cells

In [31]:
```
#Load clean dataframe
train_labels = pd.read_csv("../data/train_labels.csv")
labels_df_T = train_labels.T
labels_df_T.columns = labels_df_T.iloc[0]
labels_df_T = labels_df_T[1:]

#Find mean and turn into dataframe
labels_df_T = pd.DataFrame(labels_df_T.mean()).reset_index()
labels_df_T
```
executed in 2.90s, finished 08:05:34 2022-06-01

Out[31]:

	cell_id	0
0	0003f387-71c4-48f6-b2b0-d853bd4f0aba	35.300000
1	000617d8-8c14-43e2-b708-7e3a69fe3cc3	0.700000
2	000ba8d9-d6d5-48da-84a2-1fa54951fae1	4.414286
3	0017d1c4-64cb-426d-9158-3f6521d2dd22	2.000000
4	0020c632-3d5c-4509-b4ee-6b63a89bf2ff	0.000000
...	...	...
10873	ffdc53d2-5565-496a-b849-4fcf33f33a36	13.080000
10874	ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	29.200000
10875	ffe43514-2c92-43b6-bd84-d183806aca65	0.100000
10876	fff95195-ccc9-40b7-b302-a0d8570c86bc	12.250000
10877	fffb4d40-5947-4922-9f05-5d8b5a243d84	0.000000

10878 rows × 2 columns

In [32]:
```
#Merge to grid cell metadata dataframe
grid_cell_mean = grid_cell_df.merge(labels_df_T,on='cell_id')
grid_cell_mean = grid_cell_mean.rename(columns = {0:'historical_mean'})
grid_cell_mean = grid_cell_mean.set_crs('epsg:4326')
grid_cell_mean
```

executed in 87ms, finished 08:05:35 2022-06-01

Out[32]:

	cell_id	coordinates	region	centr_lat	centr_lon	geometry	historical_mean
0	0003f387-71c4-48f6-b2b0-d853bd4f0aba	[[-118.718953, 37.074192], [-118.718953, 37.08...	sierras	-118.722546	37.077059	POLYGON ((-118.71895 37.07419, -118.71895 37.0...	35.300000
1	000617d8-8c14-43e2-b708-7e3a69fe3cc3	[[-107.076787, 37.780424], [-107.076787, 37.78...	central rockies	-107.080380	37.783264	POLYGON ((-107.07679 37.78042, -107.07679 37.7...	0.700000
2	000ba8d9-d6d5-48da-84a2-1fa54951fae1	[[-119.320824, 37.431707], [-119.320824, 37.43...	sierras	-119.324418	37.434560	POLYGON ((-119.32082 37.43171, -119.32082 37.4...	4.414286
3	0017d1c4-64cb-426d-9158-3f6521d2dd22	[[-119.428622, 37.23886], [-119.428622, 37.246...	sierras	-119.432215	37.241720	POLYGON ((-119.42862 37.23886, -119.42862 37.2...	2.000000
4	0020c632-3d5c-4509-b4ee-6b63a89bf2ff	[[-118.898616, 36.851682], [-118.898616, 36.85...	sierras	-118.902209	36.854557	POLYGON ((-118.89862 36.85168, -118.89862 36.8...	0.000000
...	...	...	...	...	...	...	...
10873	ffdc53d2-5565-496a-b849-4fcf33f33a36	[[-119.096246, 37.045518], [-119.096246, 37.05...	sierras	-119.099839	37.048386	POLYGON ((-119.09625 37.04552, -119.09625 37.0...	13.080000
10874	ffdfb5a4-91a0-41a9-a4d5-501b04ef6326	[[-118.620138, 37.117184], [-118.620138, 37.12...	sierras	-118.623732	37.120049	POLYGON ((-118.62014 37.11718, -118.62014 37.1...	29.200000
10875	ffe43514-2c92-43b6-bd84-d183806aca65	[[-123.49799, 47.901318], [-123.49799, 47.9073...	other	-123.501584	47.903727	POLYGON ((-123.49799 47.90132, -123.49799 47.9...	0.100000
10876	fff95195-ccc9-40b7-b302-a0d8570c86bc	[[-123.372226, 47.732416], [-123.372226, 47.73...	other	-123.375819	47.734833	POLYGON ((-123.37223 47.73242, -123.37223 47.7...	12.250000
10877	fffb4d40-5947-4922-9f05-5d8b5a243d84	[[-123.794435, 47.520516], [-123.794435, 47.52...	other	-123.798028	47.522942	POLYGON ((-123.79443 47.52052, -123.79443 47.5...	0.000000

10878 rows × 7 columns

1.5.0.3 Plotting

The code cell below is commented out but works just fine to produce a static map of grid cells and ground measure stations

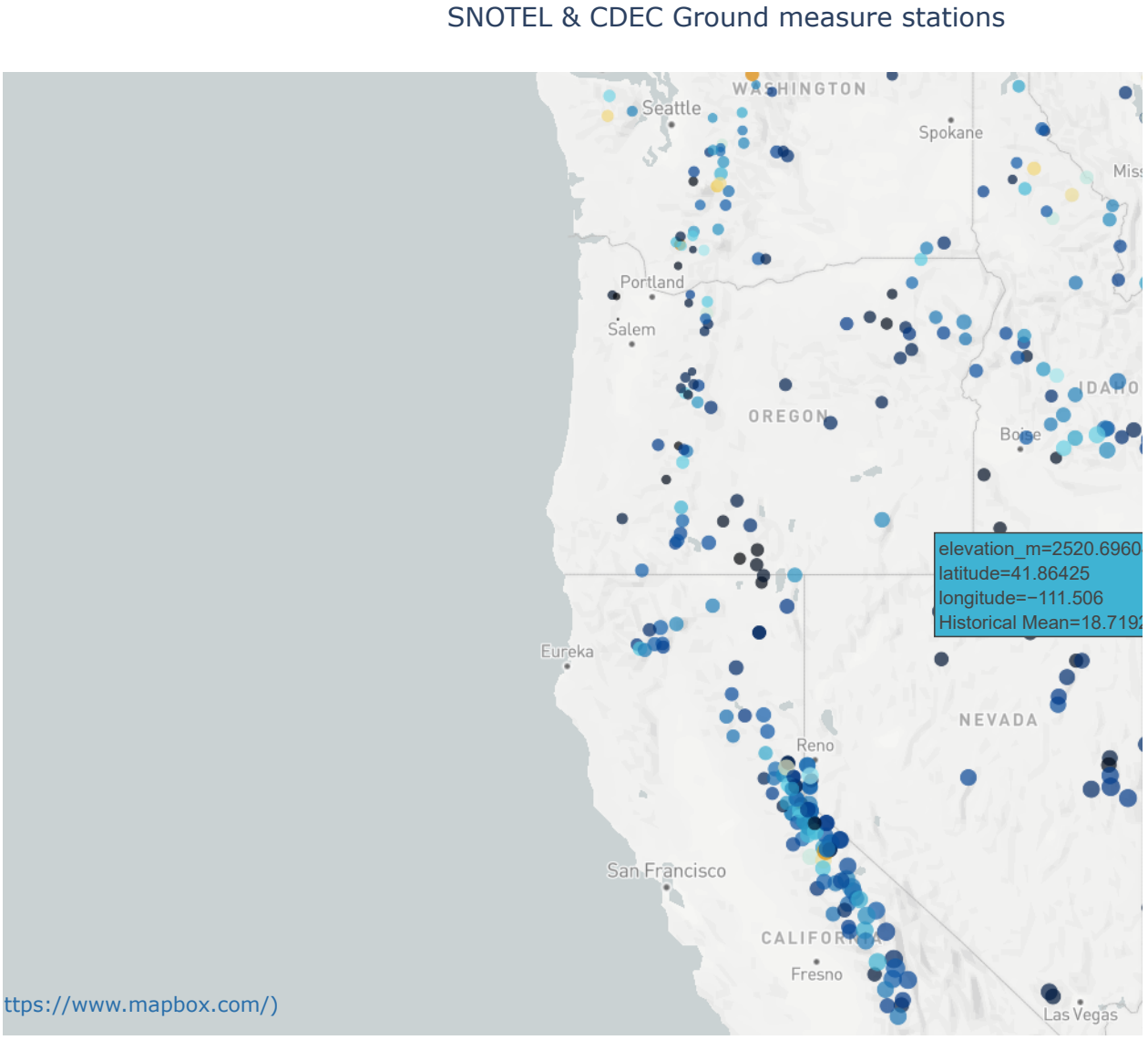
In [254]:
```
# fig, ax = plt.subplots(figsize=(20,20))

# f = grid_cell_df.plot(ax=ax, markersize=40)
# gm_md.plot(ax=ax,c='r')
# cx.add_basemap(f, crs='EPSG:4326', source=cx.providers.CartoDB.Voyager)
```

executed in 6ms, finished 22:12:07 2022-05-31

In [54]:

```
px.set_mapbox_access_token(open("./mapbox_access_token.txt").read())
fig = px.scatter_mapbox(gm_md_mean, lat="latitude", lon="longitude",color="historical_mean",size='e
                        size_max=10,
                       color_continuous_scale=px.colors.cyclical.IceFire,
                        labels={"historical_mean":'Historical Mean'},
                        width=1200, height=800)
fig.update_layout({"plot_bgcolor": "rgba(0, 0, 0, 0)",
                   "paper_bgcolor": "rgba(0, 0, 0, 0)"},
                  title_text = 'SNOTEL & CDEC Ground measure stations',
                  title_font_size = 18,
                  title_xref = 'container',
                  title_y = 0.98,
                  title_x = 0.5,
                  showlegend = False,
                  hovermode = 'closest')
fig.show()
```

executed in 473ms, finished 14:08:16 2022-06-01

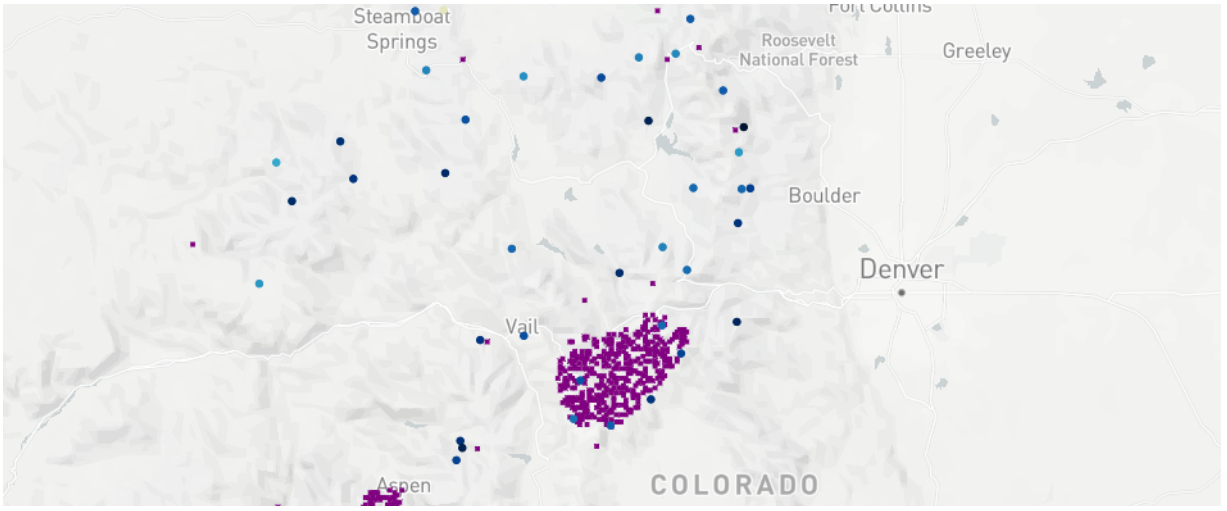


Plotting grid cells on the same map

In [53]:

```
px.set_mapbox_access_token(open("./mapbox_access_token.txt").read())
fig = px.scatter_mapbox(gm_md_mean, lat="latitude", lon="longitude",color="historical_mean",
                  color_continuous_scale=px.colors.cyclical.IceFire,
                       labels={"historical_mean":'Historical Mean'},
                        width=1200, height=800)

# fig.update_traces(marker={"size":8 })

# fig.add_trace(px.scatter_mapbox(grid_cell_mean, lat="centr_lat", lon="centr_lon").update_traces(m

fig.update_layout(
        mapbox={
#             "style": "open-street-map",
#             "zoom": 5,
            "layers": [
                {
                    "source": json.loads(grid_cell_mean.geometry.to_json()),
                    "below": "traces",
                    "type": "line",
                    "color": "purple",
                    "line": {"width": 1.5},
                }
            ],
        },
        margin={"l": 0, "r": 0, "t": 0, "b": 0},
    )


fig.update_layout({"plot_bgcolor": "rgba(0, 0, 0, 0)",
                   "paper_bgcolor": "rgba(0, 0, 0, 0)"},
#                   title_text = 'SNOTEL & CDEC Ground measure stations',
                  title_font_size = 18,
                  title_xref = 'container',
                  title_y = 1,
                  title_x = 0.5,
                  showlegend = False,
                  hovermode = 'closest')

fig.show()
```

executed in 5.00s, finished 11:36:04 2022-06-01



2 Step-up/Level ups

2.1 MODIS data

We'll be incorporating MODIS satellite images as a feature for our model. We will be loading the numpy files holding 7 bands of 21 by 21 images of each grid cell at each dates. The process of attaining the satellite images into this form is done and elaborated in this notebook ('./MODIS-DEM-Preprocessing_colab.ipynb').

We have 2 numpy files that holds the images from 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.

We also have pickled file of `cell_id` and `dates` where their index corresponds to the index of the satellite images in the numpy files.

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).

To turn these satellite images into a usable format for our model, we will be calculating the mean and variance of pixels of band 4 and band 6.

2.1.1 Load MODIS for training set

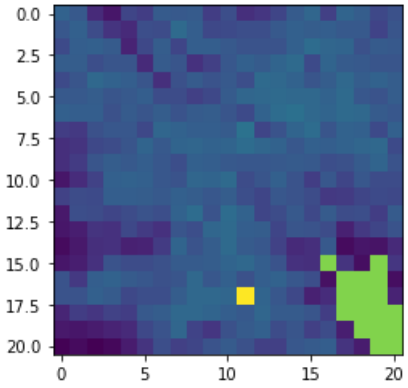
In [2]:
```
#Open file
with open('../sat_images/ModisSnowImagesT.npy','rb') as f:
    modis_np_train = np.load(f)


#Sniff test - print second image in the array
modis_np_train[1]
```

executed in 33.4s, finished 10:01:12 2022-05-27

Out[2]:
```
array([[[  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        ...,
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.]],

       [[  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        ...,
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.]],

       [[  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        ...,
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.],
        [  0.,   0.,   0., ...,   0.,   0.,   0.]],

       ...,

       [[125., 125., 125., ..., 125., 125., 125.],
        [125., 125., 125., ..., 125., 125., 125.],
        [125., 125., 125., ..., 125., 125., 125.],
        ...,
        [125., 125., 125., ..., 125., 125., 125.],
        [125., 125., 125., ..., 125., 125., 125.],
        [125., 125., 125., ..., 125., 125., 125.]],

       [[  1.,   1.,   1., ...,   1.,   1.,   1.],
        [  1.,   1.,   1., ...,   1.,   1.,   1.],
        [  1.,   1.,   1., ...,   1.,   1.,   1.],
        ...,
        [  1.,   1.,   1., ...,   1.,   1.,   1.],
        [  1.,   1.,   1., ...,   1.,   1.,   1.],
        [  1.,   1.,   1., ...,   1.,   1.,   1.]],

       [[  2.,   2.,   2., ...,   2.,   2.,   2.],
        [  2.,   2.,   2., ...,   2.,   2.,   2.],
        [  2.,   2.,   2., ...,   2.,   2.,   2.],
        ...,
        [  2.,   2.,   2., ...,   2.,   2.,   2.],
        [  2.,   2.,   2., ...,   2.,   2.,   2.],
        [  2.,   2.,   2., ...,   2.,   2.,   2.]]])
```

In [4]:
```
# Plot the band 4 of the first image in array

fig, ax
plt.imshow(modis_np_train[0,4,:,:]) #(image, band, row, column)
```
executed in 168ms, finished 10:10:57 2022-05-27

Out[4]: <matplotlib.image.AxesImage at 0x14ff8d134c0>



In [40]:
```
with open('../sat_images/cell_snow_idsT.pkl','rb') as f:
    cell_id_T_train = pickle.load(f)

#Take a peek
cell_id_T_train[10060]
```
executed in 34ms, finished 10:44:18 2022-05-27

Out[40]: ('f4c7b6ea-f881-4855-bc62-46876d67b7ee', '2018152')

In [46]:
```
# daynum = '2014236'
def daynum_gen(date_time):
    '''converts date time objects to filename'''
    date_time = pd.to_datetime(date_time)
    doy = date_time.timetuple().tm_yday #dayofyear
    year = date_time.year
    return str(year) + '{:03d}'.format(doy)
```
executed in 14ms, finished 10:47:19 2022-05-27

In [17]:
```
train_df = pd.read_csv('../data/train_labels_final.csv',index_col=[0])
train_df.head()
```

executed in 228ms, finished 10:33:07 2022-05-27

Out[17]:

	cell_id	dates	value	neighbor_1	neighbor_2	neighbor_3	neighbor_4	neighbor_5	neighbor_6	neighbor_7	...
43	00c4db22-a423-41a4-ada6-a8b1b04153a4	2013-01-01	12.7	2.0	1.6	6.4	3.2	2.7	2.0	2.8	...
77	018cf1a1-f945-4097-9c47-0c4690538bb5	2013-01-01	20.4	2.0	1.6	6.4	3.2	2.7	2.0	2.8	...
85	01be2cc7-ef77-4e4d-80ed-c4f8139162c3	2013-01-01	37.0	2.0	1.6	6.4	3.2	2.7	2.0	2.8	...
120	02c3ec4a-8de4-4284-9ec1-5a942d3d098e	2013-01-01	2.3	2.0	1.6	3.2	6.4	2.7	2.0	2.8	...
121	02cf33c2-c8e2-48b9-bf72-92506e97e251	2013-01-01	8.0	2.0	1.6	3.2	6.4	2.7	2.0	2.8	...

5 rows × 26 columns

In [36]:
```
daynum_gen(train_df.iloc[6000,1])
```

executed in 10ms, finished 10:42:24 2022-05-27

Out[36]: '2013351'

This takes too long to run. Have to figure out a way to do this more efficiently computationally.

In [70]:
```
list_idx = []

for idx, element in enumerate(cell_id_T_train):
    list_idx = [idx if element == (cell_id,daynum_gen(date)) else np.nan for cell_id, date in zip(t
```

executed in 1h 42m 55s, finished 13:08:58 2022-05-27

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [70], in <cell line: 3>()
      1 list_idx = []
      3 for idx, element in enumerate(cell_id_T_train):
----> 4     list_idx = [idx if element == (cell_id,daynum_gen(date)) else np.nan for cell_id, date i
n zip(train_df['cell_id'],train_df['dates'])]

Input In [70], in <listcomp>(.0)
      1 list_idx = []
      3 for idx, element in enumerate(cell_id_T_train):
----> 4     list_idx = [idx if element == (cell_id,daynum_gen(date)) else np.nan for cell_id, date
in zip(train_df['cell_id'],train_df['dates'])]

Input In [46], in daynum_gen(date_time)
      5 doy = date_time.timetuple().tm_yday #dayofyear
      6 year = date_time.year
----> 7 return str(year) + '{:03d}'.format(doy)

KeyboardInterrupt:
```

2.1.2 Load MODIS for test set

In [ ]:
```
```