DAYMET PRECIPITATION ANALYSIS

Presented by: Sam Shuster

INTRO & PURPOSE

Exploratory data analysis of NASA's DAYMET precipitation data

Long-term goal: determine areas with the highest number of instances where precipitation exceeds 50.8 mm/day or 2 in/day during summer months from 2000-2020

Primary Goal: programmatically subset DAYMET data according to selected variable of interest, study area, and time range.

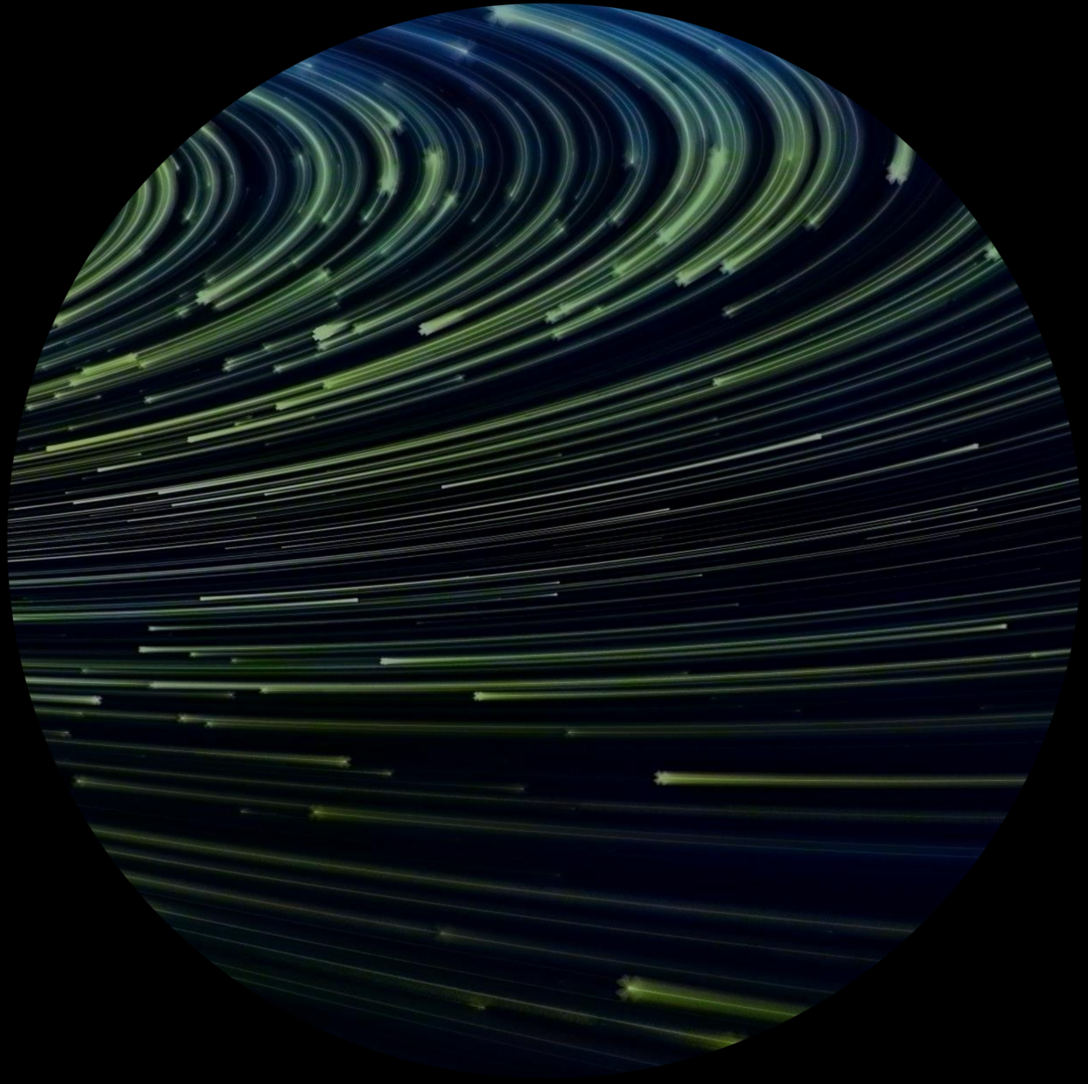
Study Area: State of Georgia – Census Tract level resolution.

ENVIRONMENT

Utilized a combination of Jupyter notebooks and ArcGIS to perform analysis and visualization

Main packages used:

- Geopandas – data manipulation and cleanup
- Rasterio – used for processing raster data
- Xarray – data manipulation
- Shapely – for manipulation and analysis of planar geometric objects
- Pydap – used to inspect data from any scientific dataset available on the internet from OPeNDAP servers.



PROCEDURE

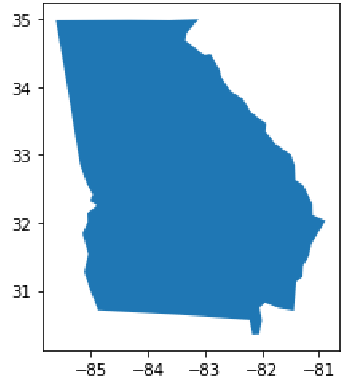
1. Import required modules into Jupyter notebook environment
2. Set study area and subset parameters
   a. Read in Georgia state boundary shapefile

In [3]:

```
ga_shape = gpd.read_file(r'C:\Users\samsh\OneDrive\Documents\Productivity\GIS\Research Projects\Marshep\Data\GA_stateBou
# ga_shape = gpd.read_file(r'C:\Users\samsh\OneDrive\Documents\Productivity\GIS\Research Projects\Marshep\Data\tl_2019_1
ax = ga_shape.plot()

ga_shape.crs
```

Out[3]:

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```



Procedure

- Set geographic bounding box used for searching NASA’s metadata repository
- Create bounding box that aligns with DAYMET’s CRS

In [4]:

```
# ga_shape_4269 = ga_shape.to_crs(epsg=4326) # Dont need to reproject because already in WGS84 and no projection applied
xy1 = ga_shape.bounds
print(xy1) # Look at creating complex polygon so that the geometry does not change subsetting upon DAYMET overlay
```

```
        minx       miny       maxx      maxy
0 -85.606675  30.356734 -80.885553  35.00118
```

In [5]:

```
xy1 = ga_shape.bounds.values.tolist()[0] # We'll need the bounding box as a Python list
                                          # to server as a subsetting parameter
# xy1 = [-85.605165, 30.357851, -80.839729, 35.000659]
print(xy1)
```

```
[-85.606675, 30.356734, -80.885553, 35.00118]
```

In [6]:

```
#defining Daymet proj - we'll use this in a later step
daymet_proj = "+proj=lcc +ellps=WGS84 +a=6378137 +b=6356752.314245 +lat_1=25 +lat_2=60 +lon_0=-100 +lat_0=42.5 +x_0=0 +y_
ga_shape_lcc = ga_shape.to_crs(daymet_proj) # to_crs re-projects from NAD 1983 to LCC
lccbounds = ga_shape_lcc.bounds # Bounds in LCC projection
lccbounds.round(2)
```

Out[6]:

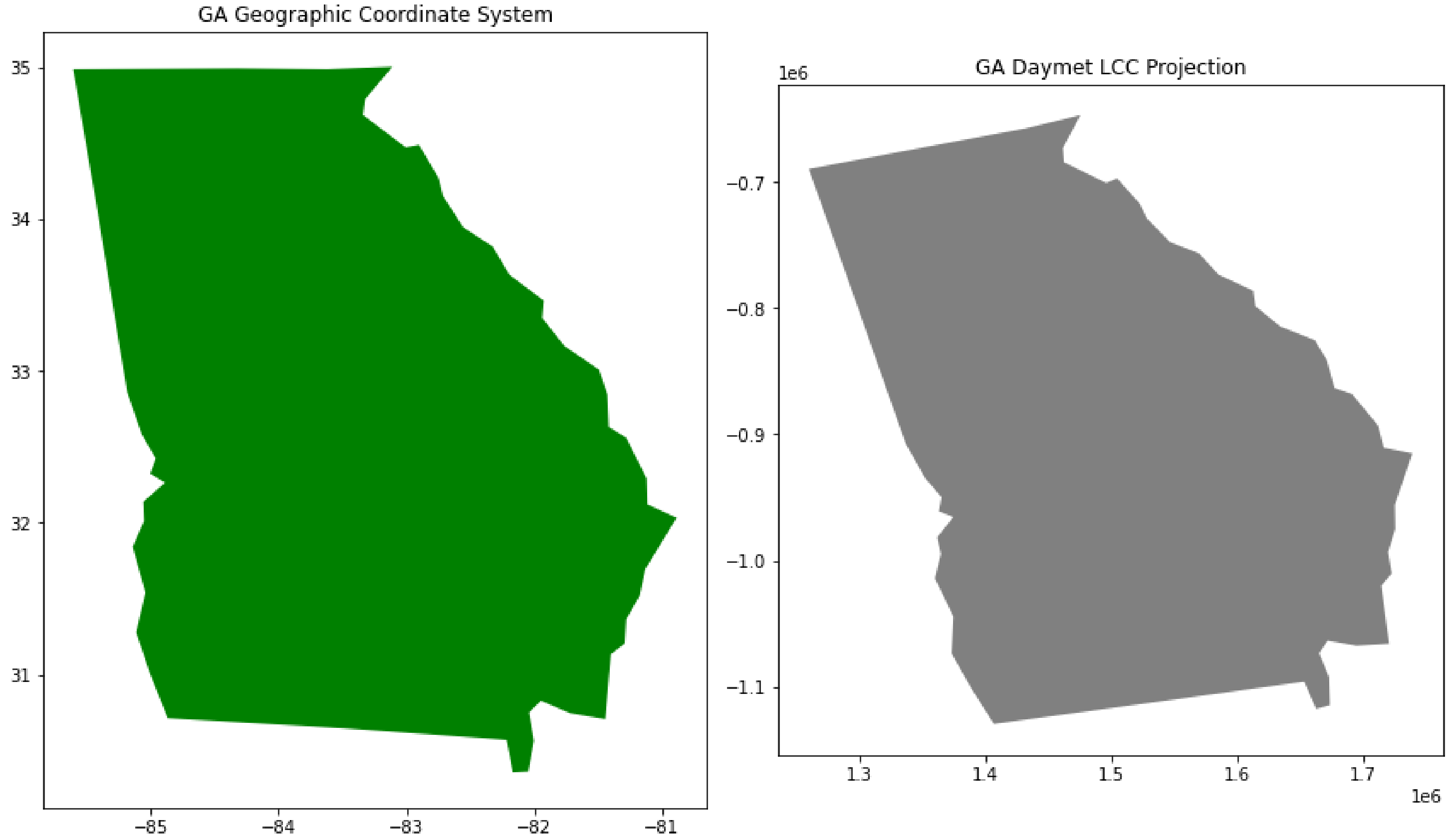
	minx	miny	maxx	maxy
0	1259685.37	-1129705.22	1738456.68	-646896.28

Procedure

- Picture shapefile in original and transformed CRS

In [7]:

```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))
#ga_shape.plot(ax=ax1, facecolor='blue');
ga_shape.plot(ax=ax1, facecolor='green');
ax1.set_title("GA Geographic Coordinate System");
ga_shape_lcc.plot(ax=ax2, facecolor='gray');
ax2.set_title("GA Daymet LCC Projection");
plt.tight_layout()
```



Procedure

- Specify time range of interest:

```
In [8]:
start_date = dt.datetime(2015, 1, 1) # specify your own start date
end_date = dt.datetime(2018, 12, 31)  # specify your end start date

dt_format = '%Y-%m-%dT%H:%M:%SZ' # format requirement for datetime search
temporal_str = start_date.strftime(dt_format) + ',' + end_date.strftime(dt_format)

var = 'prcp' # select a Daymet variable of interest
print(temporal_str)
print(var)
```

```
2015-01-01T00:00:00Z,2018-12-31T00:00:00Z
prcp
```

Procedure

- Create URL used to enact data access request:

In [9]:

```
daymet_doi = '10.3334/ORNLDAAC/1840' # define the Daymet V4 Daily Data DOI as the variable `daymet_doi`
cmrurl='https://cmr.earthdata.nasa.gov/search/' # define the base url of NASA's CMR API as the variable `cmrurl`
doisearch = cmrurl + 'collections.json?doi=' + daymet_doi # Create the Earthdata Collections URL
print('Earthdata Collections URL: Daymet V4 Daily -->', doisearch)
```

```
Earthdata Collections URL: Daymet V4 Daily --> https://cmr.earthdata.nasa.gov/search/collections.json?doi=10.3334/ORNLDAAC/1
840
```

In [10]:

```
# From the doisearch, we can obtain the ConceptID for the Daymet V4 Daily data
# We'll search the json response of the Daymet metadata for "id" within the 'entry' dictionary key
response = requests.get(doisearch)
collection = response.json()['feed']['entry'][0]
#print(collection)
concept_id = collection['id']
print('NASA Earthdata Concept_ID --> ' , concept_id)
```

```
NASA Earthdata Concept_ID -->  C2031536952-ORNL_CLOUD
```

`C2031536952-ORNL_CLOUD` is the unique NASA-given Concept ID for the Daymet V4 Daily data Collection. We'll use this to search for Daymet V4 Daily files (granules) that match our search criteria.

Procedure

- Create URL used to enact data access request:

2.2.a. We'll build a Request URL `granulesearch` to create a listing of all the granules (files) in NASA's Earthdata holdings that fit the search criteria we defined.

In [12]:

```
granulesearch = cmrurl + 'granules.json?collection_concept_id=' + concept_id + \
                '&page_size=1000' + '&temporal=' + temporal_str + \
                '&bounding_box[]=' + ','.join(map(str, xy1))
print(granulesearch)
```

```
https://cmr.earthdata.nasa.gov/search/granules.json?collection_concept_id=C2031536952-ORNL_CLOUD&page_size=1000&temporal=201
5-01-01T00:00:00Z,2018-12-31T00:00:00Z&bounding_box[]=-85.606675,30.356734,-80.885553,35.00118
```

2.2.b. Again using Python's `requests` library, we can provide the URL `granulesearch` to create a listing of all the granules (files) in NASA's Earthdata holdings that fit the search criteria we defined.

In [13]:

```
response = requests.get(granulesearch)
granules = response.json()['feed']['entry']  # Michele, look at url feed, 'entry' key in the granulesearch URL {}
granule_names = []  # create an empty array
for g in granules:
    granule_name = g['title'] # fill the array with granule names that match our search parameters
    if var in granule_name:
        granule_names.append(granule_name)
        print(granule_name)
```

```
Daymet_Daily_V4.daymet_v4_daily_na_prcp_2015.nc
Daymet_Daily_V4.daymet_v4_daily_na_prcp_2016.nc
Daymet_Daily_V4.daymet_v4_daily_na_prcp_2017.nc
Daymet_Daily_V4.daymet_v4_daily_na_prcp_2018.nc
```

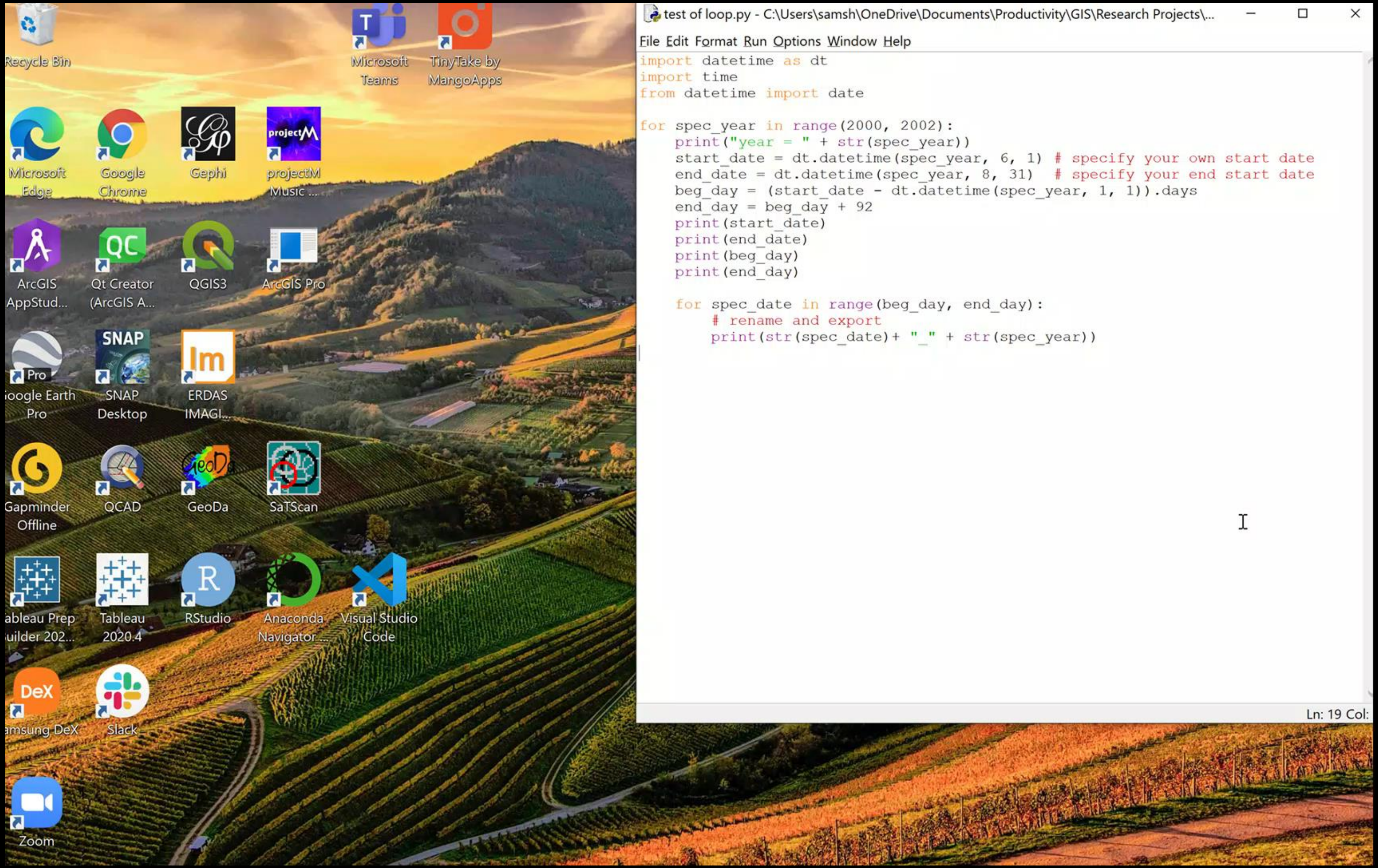
Procedure

- Enact request and retrieve data:

```
In [14]:
#from pydap.client import open_url
#import xarray as xr
#import time

thredds_url = 'https://thredds.daac.ornl.gov/thredds/dodsC/ornldaac/1840/' # ORNL DAAC TDS OPeNDAP URL
                                                                           # for Daymet V4 Daily Files

before = time.time()
cnt = 0
for g_name in granule_names:
    print(' ***GRANULE_NAME*** ---->', g_name)
    granule_dap = thredds_url + g_name.replace('Daymet_Daily_V4.','')
    print(granule_dap)

    # Using pydap's open_url
    thredds_ds = open_url(granule_dap)

    # Xarray DataSet - opening dataset via remote OPeNDAP
    ds = xr.open_dataset(xr.backends.PydapDataStore(thredds_ds), decode_coords="all")

    temp=ds['prcp'].sel(x=slice(lccbounds.minx[0],lccbounds.maxx[0]), y=slice(lccbounds.maxy[0],lccbounds.miny[0]))

    if cnt==0:
        prcp = temp
    else:
        prcp = xr.concat([prcp, temp], dim="time")

    cnt += 1


# save to netcdf
prcp.to_netcdf(var + '_tdssubset.nc')
print("Processing Time: ", time.time() - before, 'seconds')
# Processing Time:  50.4509379863739 seconds
```

PROCEDURE

- Enact request and retrieve data:

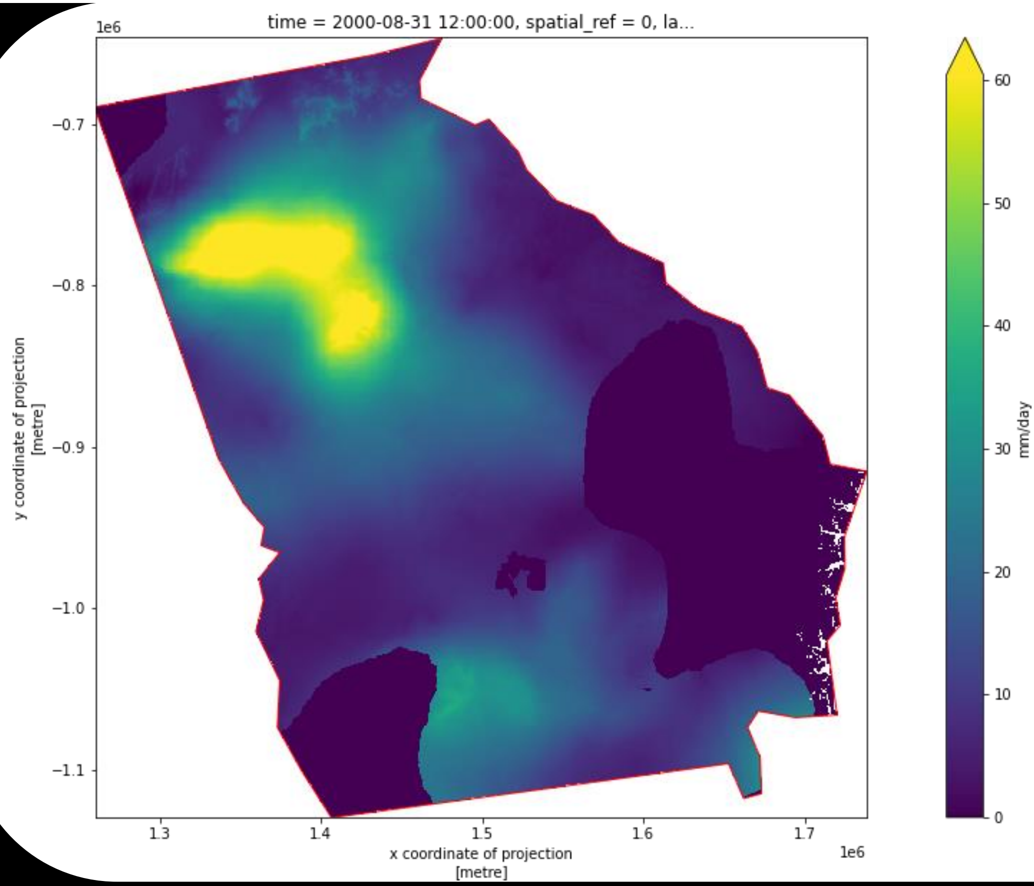


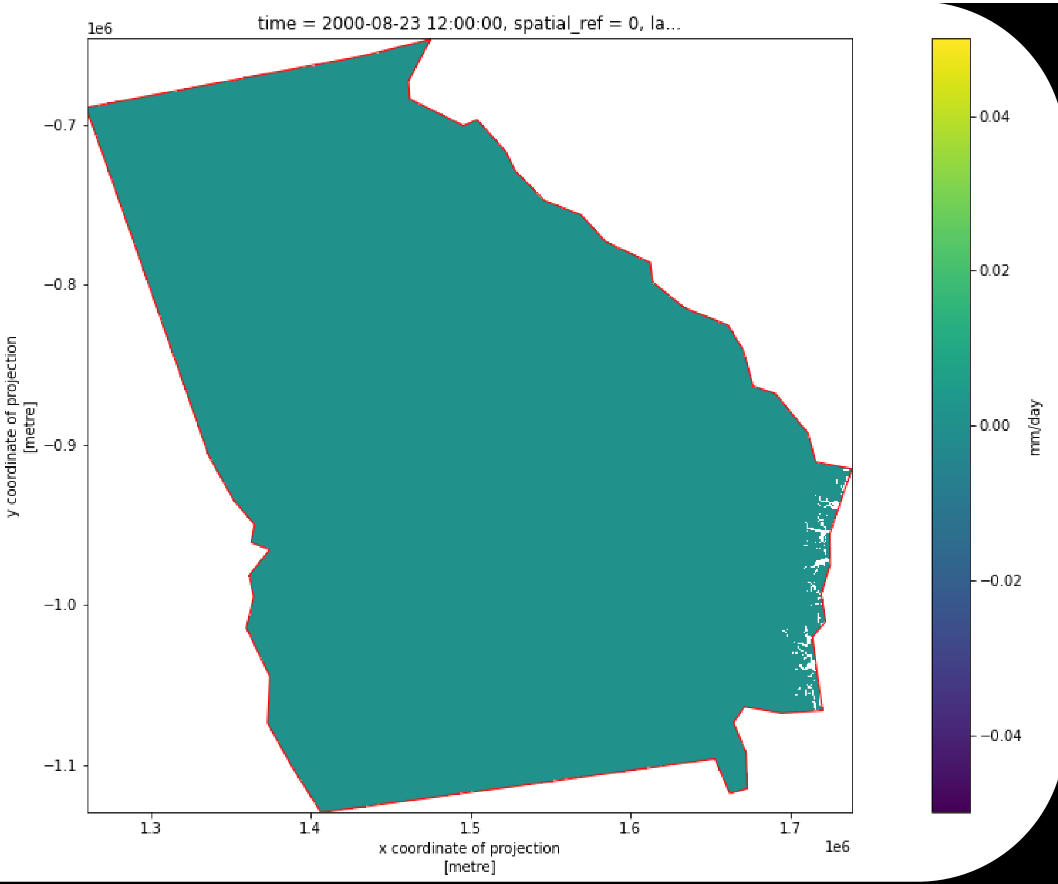
Procedure

- Visualize and Clip Daymet Precipitation Subset:

In [15]:

```
prcp_ga = prcp.isel(time=6) # isel = xarray index selection (python index start from 0, so time is Jan 7, 2010)
fig, ax = plt.subplots(figsize = (20,10))
prcp_ga.plot(ax=ax, robust=True, cbar_kwargs={'label': 'mm/day'})
ga_shape_lcc.plot(ax = ax, color = 'none', edgecolor = 'red')
```





Procedure

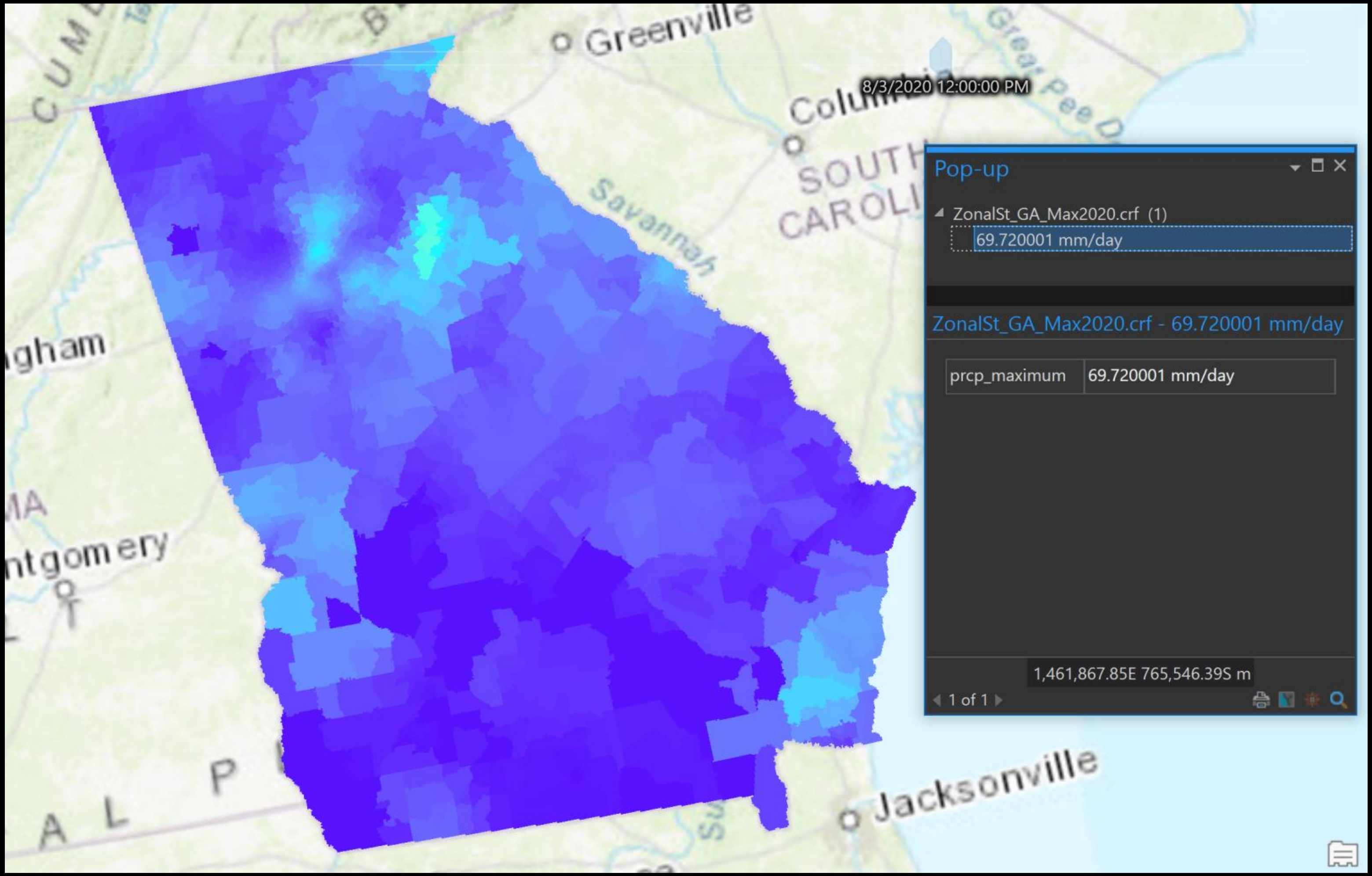
- Create function that automatically subsets and exports data for all days of interest
  - Interested in summer months from 2000-2020

In [15]:

```
for spec_year in range(2016, 2018):
        print("year = " + str(spec_year))
        start_date = dt.datetime(spec_year, 6, 1) # specify your own start date
        end_date = dt.datetime(spec_year, 7, 1)  # specify your end start date
        beg_day = (start_date - dt.datetime(spec_year, 1, 1)).days
        end_day = beg_day + 92
        print(start_date)
        print(end_date)
        print(beg_day)
        print(end_day)

        for spec_day in range(beg_day, end_day):
            # rename and export
#             print(str(spec_day)) #spec_day will be the value to use to specify what time equals a particular day
            # The line below is the likely result of the inaccurate specification of year when automating the subset.
#             prcp_1day_export = prcp_clip.isel(time=spec_day)
            prcp_1day_export = prcp_clip.sel(time=slice(start_date, end_date))
            print(str(prcp_1day_export))
            prcp_1day_export.rio.to_raster("prcp_1day_export" + "_" + str(spec_day) + "_" + str(spec_year) + ".tif")
            fig, ax = plt.subplots(figsize = (20,10))
            prcp_1day_export.plot(ax=ax, robust=True, cbar_kwargs={'label': 'mm/day'})
            ga_shape_lcc.plot(ax = ax, color = 'none', edgecolor = 'red')
            fig.savefig("prcp_1day_export" + "_" + str(spec_day) + "_" + str(spec_year) + ".png")


print("Finished")
```

Procedure

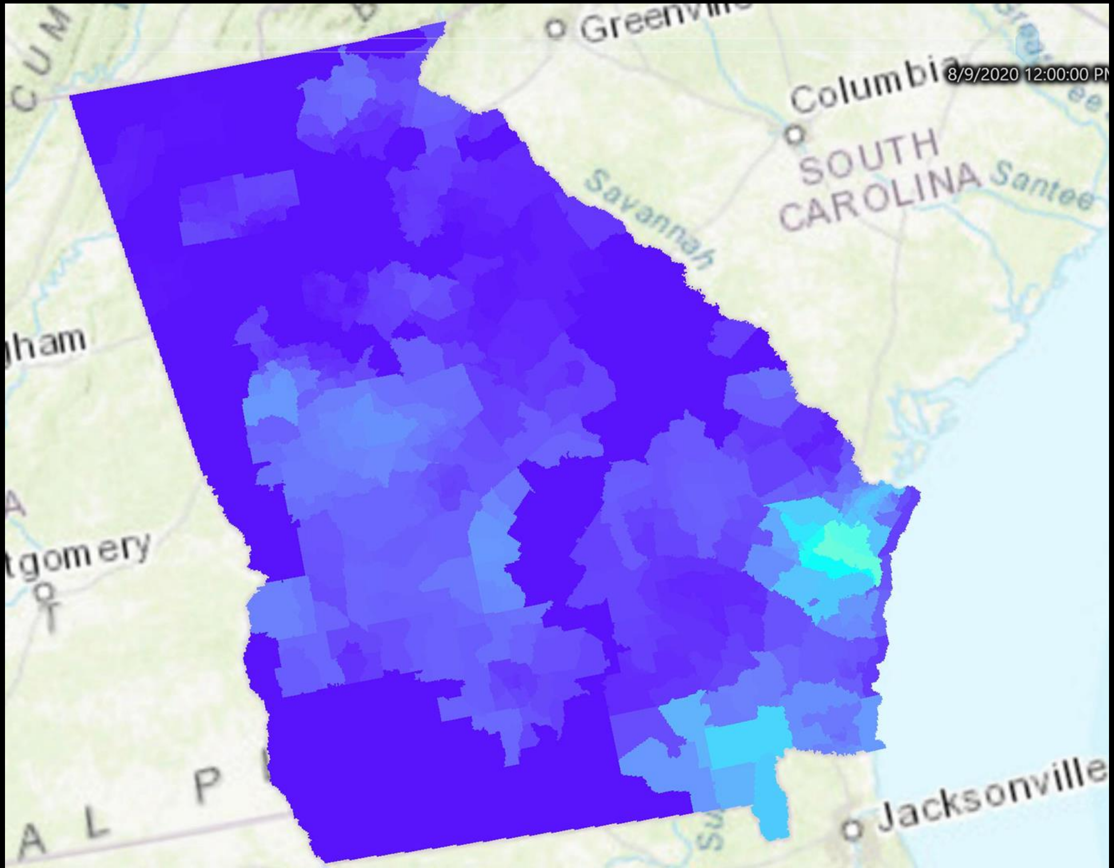
- Bring data into ArcGIS Pro
- Conduct zonal statistics for each year at Census tract level



Pictured: 8/3/2020

Procedure

- Final step is to aggregate daily data to provide summary statistics for summer months of each year
- <u>Currently underway</u>: ArcGIS has ability to process multidimensional data but working with data that has a time dimension must be done by hand presently



QUESTIONS?