In [1]:

```
#!pip install GDAL-3.1.4-cp38-cp38-win_amd64.whl
#!pip install Fiona-1.8.17-cp38-cp38-win_amd64.whl
#!pip install Shapely-1.7.1-cp38-cp38-win_amd64.whl
#!pip install geopandas
#!pip install datadotworld
```

In [ ]:

```
import geopandas
import numpy as np
import pandas as pd

import datetime

import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import cm

from matplotlib.animation import FuncAnimation
from mpl_toolkits.axes_grid1 import make_axes_locatable
from matplotlib.backends.backend_pdf import PdfPages
import matplotlib.ticker as mtick
import datadotworld as dw

def import_geo_data(filename, index_col = "Date", FIPS_name = "FIPS"):
    # import county level shapefile
    map_data = geopandas.read_file(filename = filename,
                                   index_col = index_col)
    # rename fips code to match variable name in COVID-19 data
    map_data.rename(columns={"State":"state"},
                    inplace = True)
    # Combine statefips and county fips to create a single fips value
    # that identifies each particular county without referencing the
    # state separately
    map_data[FIPS_name] = map_data["STATEFP"].astype(str) + \
        map_data["COUNTYFP"].astype(str)
    map_data[FIPS_name] = map_data[FIPS_name].astype(np.int64)
    # set FIPS as index
    map_data.set_index(FIPS_name, inplace=True)

    return map_data

def import_covid_data(FIPS_name):
    # Load COVID19 county data using datadotworld API
    # Data provided by Johns Hopkins, file provided by Associated Press
    dataset = dw.load_dataset(
        "associatedpress/johns-hopkins-coronavirus-case-tracker",
        auto_update = True)
    # the dataset includes multiple dataframes. We will only use #2
    covid_data = dataset.dataframes["2_cases_and_deaths_by_county_timeseries"]
    # Include only oberservation for political entities within states
    # i.e., not territories, etc... drop any nan fip values with covid_data[FIPS_name]
    covid_data = covid_data[covid_data[FIPS_name] < 57000]
    covid_data = covid_data[covid_data[FIPS_name] > 0]

    # Transform FIPS codes into integers (not floats)
    covid_data[FIPS_name] = covid_data[FIPS_name].astype(int)
    covid_data['date'] = pd.to_datetime(covid_data['date'])
    covid_data.set_index([FIPS_name, "date"], inplace = True)
    # Prepare a column for state abbreviations. We will draw these from a
    # dictionary created in the next step.
    covid_data["state_abr"] = ""
    for state, abr in state_dict.items():
        covid_data.loc[covid_data["state"] == state, "state_abr"] = abr
    # Create "Location" which concatenates county name and state abbreviation
    covid_data["Location"] = covid_data["location_name"] + ", " + \
        covid_data["state_abr"]

    return covid_data

# I include this dictionary to convenienlty cross reference state names and
# state abbreviations.
# I include this dictionary to convenienlty cross reference state names and
# state abbreviations.
state_dict = {
    'Alabama': 'AL', 'Alaska': 'AK', 'Arizona': 'AZ',
    'Arkansas': 'AR', 'California': 'CA', 'Colorado': 'CO', 'Connecticut': 'CT',
    'Delaware': 'DE', 'District of Columbia': 'DC', 'Florida': 'FL',
    'Georgia': 'GA', 'Hawaii': 'HI', 'Idaho': 'ID', 'Illinois': 'IL',
    'Indiana': 'IN', 'Iowa': 'IA','Kansas': 'KS', 'Kentucky': 'KY',
    'Louisiana': 'LA', 'Maine': 'ME', 'Maryland': 'MD', 'Massachusetts': 'MA',
    'Michigan': 'MI', 'Minnesota': 'MN', 'Mississippi': 'MS', 'Missouri': 'MO',
    'Montana': 'MT', 'Nebraska': 'NE', 'Nevada': 'NV', 'New Hampshire': 'NH',
    'New Jersey': 'NJ', 'New Mexico': 'NM', 'New York': 'NY', 'North Carolina': 'NC',
    'North Dakota': 'ND', 'Ohio': 'OH', 'Oklahoma': 'OK',
    'Oregon': 'OR', 'Pennsylvania': 'PA', 'Rhode Island': 'RI',
    'South Carolina': 'SC', 'South Dakota': 'SD', 'Tennessee': 'TN', 'Texas': 'TX',
    'Utah': 'UT', 'Vermont': 'VT', 'Virginia': 'VA',
    'Washington': 'WA', 'West Virginia': 'WV', 'Wisconsin': 'WI',
    'Wyoming': 'WY'}

plt.rcParams['axes.ymargin'] = 0
plt.rcParams['axes.xmargin'] = 0
plt.rcParams.update({'font.size': 32})

#if "data_processed" not in locals():
fips_name = "fips_code"
# covid_filename = "COVID19DataAP.csv"
# rename_FIPS matches map_data FIPS with COVID19 FIPS name
map_data = import_geo_data(
    filename = "countiesWithStatesAndPopulation.shp",
    index_col = "Date", FIPS_name= fips_name)
covid_data = import_covid_data(FIPS_name = fips_name)
# dates will be used to create a geopandas DataFrame with multiindex
```

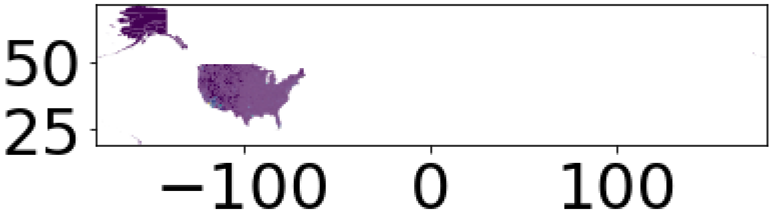
```
C:\Users\jzach\anaconda3\lib\site-packages\datadotworld\models\dataset.py:206: UserWarni
ng: Unable to set data frame dtypes automatically using 2_cases_and_deaths_by_county_tim
eseries schema. Data types may need to be adjusted manually. Error: Integer column has N
A values in column 2
  warnings.warn(
```

In [25]:

```
map_data.plot(column = "Population")
```

Out[25]:

```
<AxesSubplot:>
```



```
In [26]: map_data
```

Out[26]:

	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	AWATE
fips_code								
21007	21	007	00516850	0500000US21007	Ballard	06	639387454	6947332
21017	21	017	00516855	0500000US21017	Bourbon	06	750439351	482977
21031	21	031	00516862	0500000US21031	Butler	06	1103571974	1394304
21065	21	065	00516879	0500000US21065	Estill	06	655509930	651633
21069	21	069	00516881	0500000US21069	Fleming	06	902727151	718279
...	...	...	...	...	...	...	...	
31073	31	073	00835858	0500000US31073	Gosper	06	1186616237	1183182
39075	39	075	01074050	0500000US39075	Holmes	06	1094405866	369523

	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	AWATE
fips_code								
48171	48	171	01383871	0500000US48171	Gillespie	06	2740719114	901276
55079	55	079	01581100	0500000US55079	Milwaukee	06	625440563	245538363
26139	26	139	01623012	0500000US26139	Ottawa	06	1459502408	276583098

3142 rows × 11 columns

In [27]:

```
covid_data
```

Out[27]:

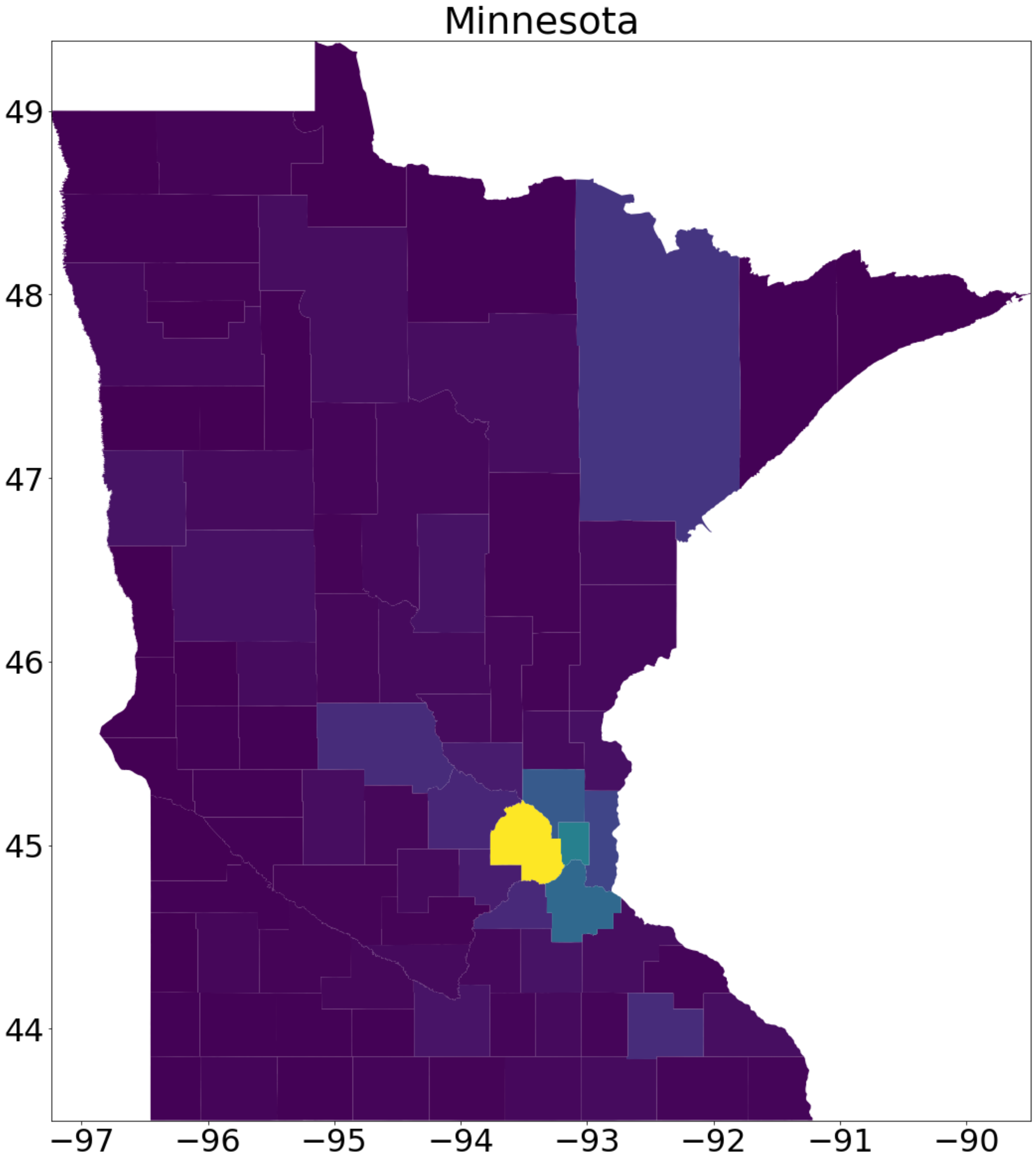
		uid	location_type	location_name	state	total_population	cumulative_cases
fips_code	**date**						
1001	**2020-01-22**	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-23	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-24	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-25	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-26	84001001	county	Autauga	Alabama	55200.0	0
...	**...**	...	...	...	...	...	...
56045	**2021-11-18**	84056045	county	Weston	Wyoming	7100.0	1186
	2021-11-19	84056045	county	Weston	Wyoming	7100.0	1187
	2021-11-20	84056045	county	Weston	Wyoming	7100.0	1187
	2021-11-21	84056045	county	Weston	Wyoming	7100.0	1187

		uid	location_type	location_name	state	total_population	cumulative_cases
fips_code	**date**						
	2021-11-22	84056045	county	Weston	Wyoming	7100.0	1188

2109624 rows × 17 columns

In [29]:
```
fig, ax = plt.subplots(figsize = (30,20))
map_data[map_data["state"] == "Minnesota"].plot(column = "Population", ax = ax)
ax.set_title("Minnesota")
```

Out[29]:
```
Text(0.5, 1.0, 'Minnesota')
```



In [30]:

```
def import_covid_data(FIPS_name):
    dataset = dw.load_dataset(
        "associatedpress/johns-hopkins-coronavirus-case-tracker",
        auto_update = True)
    covid_data = dataset.dataframes["2_cases_and_deaths_by_county_timeseries"]
    covid_data = covid_data[covid_data[FIPS_name] < 57000]
    covid_data = covid_data[covid_data[FIPS_name] > 0]

    covid_data[FIPS_name] = covid_data[FIPS_name].astype(np.int64)
    # format the date columns as datetime
    covid_data["date"] = pd.to_datetime(covid_data["date"])
    covid_data.set_index([FIPS_name, "date"], inplace = True)
    covid_data["state_abr"] = ""
    for state, abr in state_dict.items():
        covid_data.loc[covid_data["state"] == state, "state_abr"] = abr
    covid_data["Location"] = covid_data["location_name"] + ", " + \
                                       covid_data["state_abr"]

    return covid_data
```

In [31]:

```
covid_data = import_covid_data(FIPS_name = fips_name)
```

```
C:\Users\jzach\anaconda3\lib\site-packages\datadotworld\models\dataset.py:206: UserWarni
ng: Unable to set data frame dtypes automatically using 2_cases_and_deaths_by_county_tim
eseries schema. Data types may need to be adjusted manually. Error: Integer column has N
A values in column 2
  warnings.warn(
```

In [32]:

```
covid_data
```

Out[32]:

		uid	location_type	location_name	state	total_population	cumulative_cases
fips_code	**date**						
1001	**2020-01-22**	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-23	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-24	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-25	84001001	county	Autauga	Alabama	55200.0	0
	2020-01-26	84001001	county	Autauga	Alabama	55200.0	0
...	**...**	...	...	...	...	...	...
56045	**2021-11-18**	84056045	county	Weston	Wyoming	7100.0	1186
	2021-11-19	84056045	county	Weston	Wyoming	7100.0	1187
	2021-11-20	84056045	county	Weston	Wyoming	7100.0	1187
	2021-11-21	84056045	county	Weston	Wyoming	7100.0	1187
	2021-11-22	84056045	county	Weston	Wyoming	7100.0	1188

2109624 rows × 17 columns

In [33]:

```
covid_data[covid_data["state"] == "North Dakota"].groupby("date").sum()[["new_cases", "
```

Out[33]:

	new_cases	new_deaths
date		
2020-01-22	0.0	0.0
2020-01-23	0.0	0.0
2020-01-24	0.0	0.0
2020-01-25	0.0	0.0
2020-01-26	0.0	0.0
...	...	...
2021-11-18	641.0	7.0
2021-11-19	470.0	12.0
2021-11-20	487.0	2.0
2021-11-21	148.0	0.0
2021-11-22	221.0	0.0

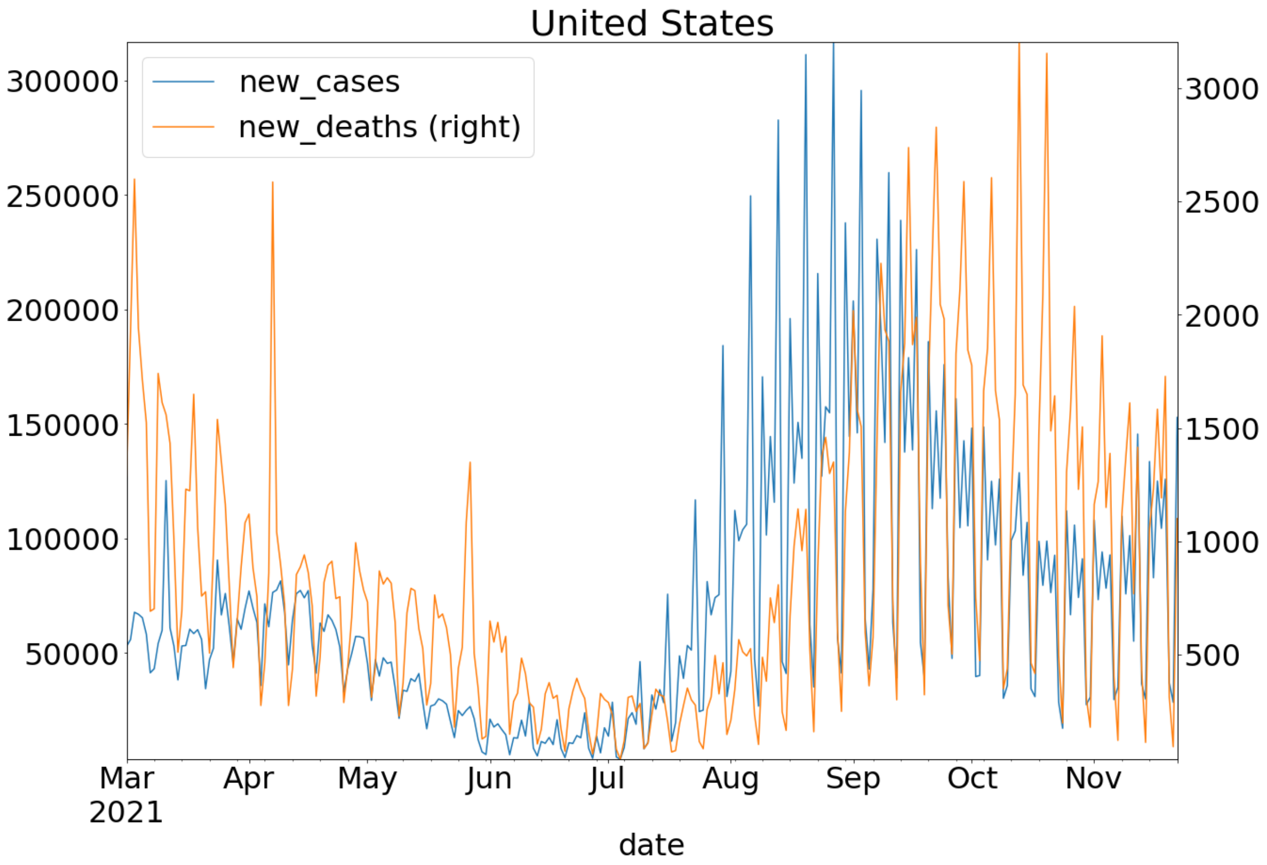
671 rows × 2 columns

In [34]:

```
fig,ax = plt.subplots(figsize = (20,14))
covid_data.groupby("date").sum().loc["2021-03-01":,["new_cases", "new_deaths"]].plot.li
    secondary_y = "new_deaths", ax = ax)
ax.set_title("United States")
```

Out[34]:

```
Text(0.5, 1.0, 'United States')
```

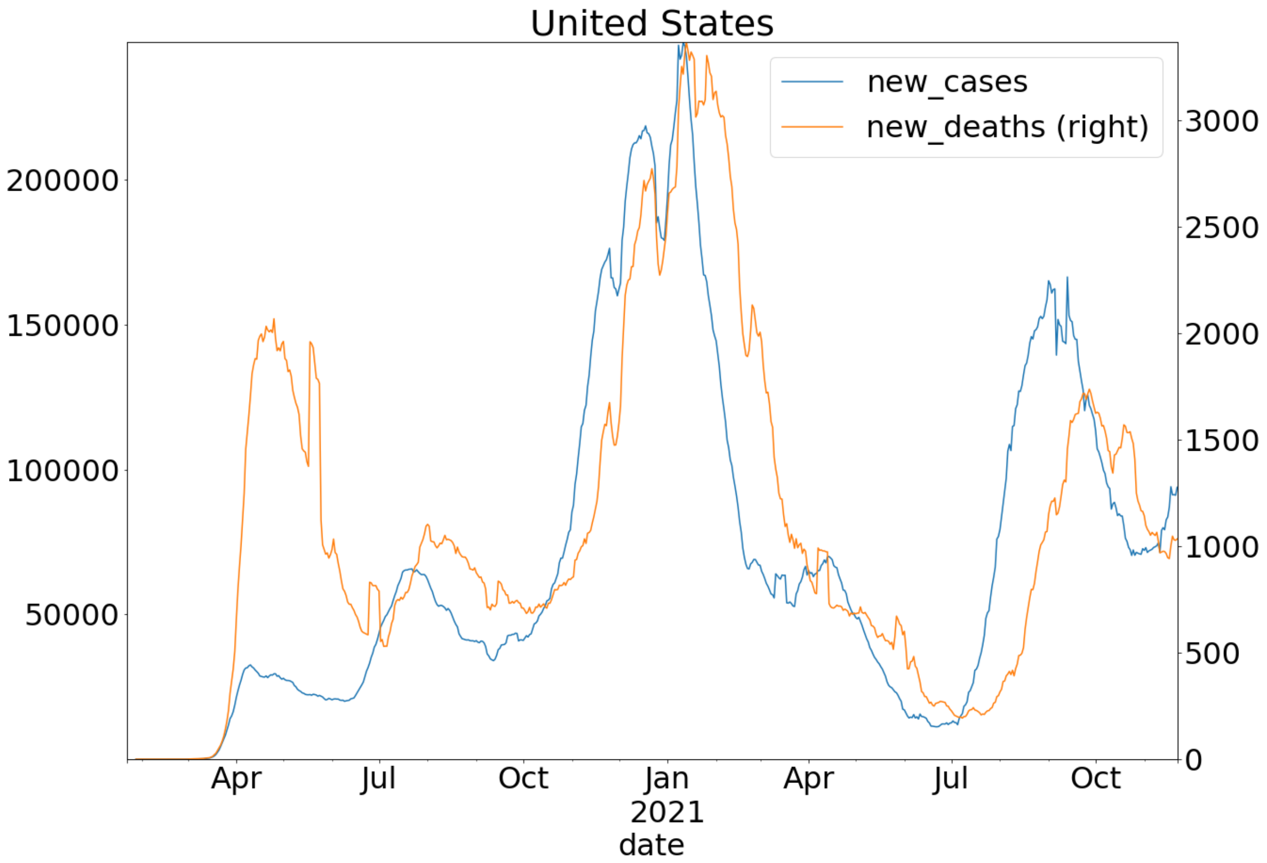


In [35]:

```
fig,ax = plt.subplots(figsize = (20,14))
covid_data.groupby("date").sum().loc[:,["new_cases", "new_deaths"]].rolling(7).mean().p
    secondary_y = "new_deaths", ax = ax)
ax.set_title("United States")
```

Out[35]:

```
Text(0.5, 1.0, 'United States')
```



In [36]:

```
def create_merged_geo_dataframe(data, map_data, dates):
    data_frame_initialized = False

    counties = data.groupby("fips_code").mean().index
    for date in dates:
        agg_df = map_data[map_data.index.isin(counties)]
        agg_df["date"] = date
        if data_frame_initialized == False:
            matching_gpd = geopandas.GeoDataFrame(agg_df,
                                                   crs = map_data.crs)
            data_frame_initialized = True
        else:
            matching_gpd = matching_gpd.append(
                agg_df,
                ignore_index = False)

    matching_gpd.reset_index(inplace = True)
    matching_gpd.set_index(["fips_code", "date"], inplace = True)
    matching_gpd.drop("state", axis = 1, inplace = True)

    matching_gpd = pd.concat([matching_gpd, data], axis = 1)

    return matching_gpd
```

In [48]:

```
dates = sorted(list(set(covid_data.index.get_level_values("date"))))
```

In [39]:

```
def select_data_within_bounds(data, minx, miny, maxx, maxy):
    data = data[data.bounds["maxx"] <= maxx]
    data = data[data.bounds["maxy"] <= maxy]
    data = data[data.bounds["minx"] >= minx]
    data = data[data.bounds["miny"] >= miny]

    return data
```

In [49]:

```
date = dates[-1]
```

We now want to create a geo-dataframe with an entry for every date

In [50]:

```
# dates will be used to create a geopandas DataFrame with multiindex
covid_data = create_merged_geo_dataframe(covid_data, map_data, dates)
covid_data
```

Out[50]:

		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	**date**								
1001	**2020-01-22**	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-23	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-24	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-25	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-26	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
...	**...**	...	...	...	...	...	...	...	
56045	**2021-11-18**	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	date								
	2021-11-19	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-20	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-21	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-22	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

2109624 rows × 27 columns

In [51]:

```
def create_new_vars(covid_data):

    for key in ["cases", "deaths"]:

        cap_key = key.title()
        covid_data.rename(columns={"cumulative_" + key: "Total " + cap_key},
                          inplace=True)
        covid_data[cap_key + " per Million"] = covid_data["Total " + cap_key].fillna(0)
            .div(covid_data["total_population"]).mul(10 ** 6)

        covid_data["Daily " + cap_key] = covid_data[
            "new_" + key]  #.groupby(covid_data.index.names[1])\

        covid_data[
            "Daily " + cap_key + " 7 Day MA"] = covid_data[
                "new_" + key +
                "_7_day_rolling_avg"]  #.rolling(moving_average_days).mean()


        covid_data["Daily " + cap_key + " per Million 7 Day MA"] = \
            covid_data["Daily " + cap_key + " 7 Day MA"]\
            .div(covid_data["total_population"]).mul(10 ** 6)



create_new_vars(covid_data)
start_date = "03-15-2020"
end_date = dates[-1]
```

In [52]:

```
covid_data
```

Out[52]:

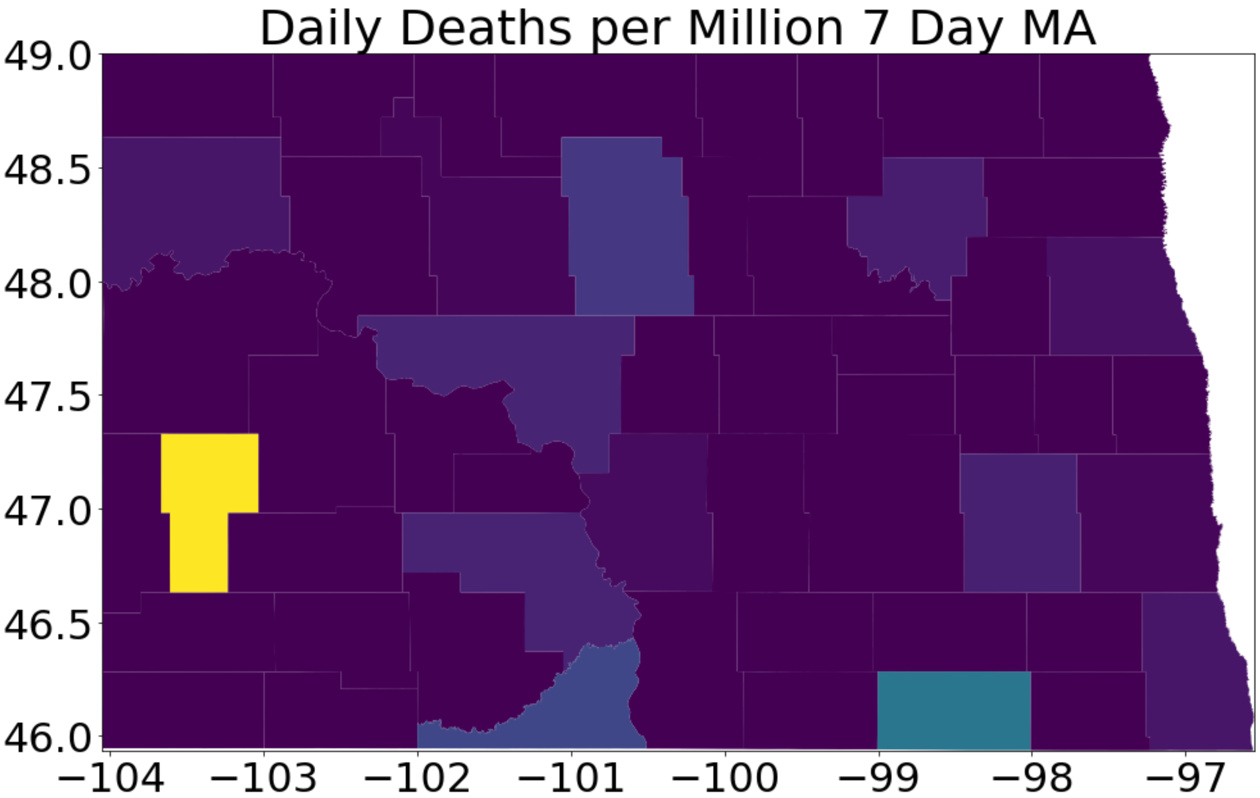
		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	**date**								
1001	**2020-01-22**	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-23	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-24	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-25	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-26	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
...	**...**	...	...	...	...	...	...	...	
56045	**2021-11-18**	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-19	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-20	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	**date**								
	2021-11-21	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-22	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

2109624 rows × 35 columns

```
In [54]: # selected data by state
         nd_data = covid_data[covid_data["state"] == "North Dakota"]

         # select data by date
         nd_data = nd_data[nd_data.index.get_level_values("date") == "2021-11-15"]

         # then plot
         fig, ax = plt.subplots(figsize = (20,10))
         key = "Daily Deaths per Million 7 Day MA"
         nd_data.plot(column = key, ax = ax)#, cmap = "Reds")
         ax.set_title(key)
```

```
Out[54]: Text(0.5, 1.0, 'Daily Deaths per Million 7 Day MA')
```



In [55]:

```
def select_data_within_bounds(data, minx, miny, maxx, maxy):
    data = data[data.bounds["maxx"] <= maxx]
    data = data[data.bounds["maxy"] <= maxy]
    data = data[data.bounds["minx"] >= minx]
    data = data[data.bounds["miny"] >= miny]

    return data


date = dates[-1]

if "map_bounded" not in locals():
    minx = covid_data[covid_data.index.get_level_values("date")== date].bounds["minx"].
    miny = covid_data[covid_data.index.get_level_values("date")== date].bounds["miny"].
    maxx = -58
    maxy = covid_data[covid_data.index.get_level_values("date")== date].bounds["maxy"].
    # find counties using only 1 date, only performs operation once instead of
    # several hundred times
    bounded_data =  select_data_within_bounds(covid_data[covid_data.index.get_level_val
    counties = bounded_data.groupby("fips_code").mean().index
    covid_map_data =covid_data[covid_data.index.get_level_values("fips_code").isin(coun
    map_bounded = True
```

In [56]:

```
covid_map_data
```

Out[56]:

	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND

fips_code	date	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	date								
1001	2020-01-22	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-23	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-24	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-25	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
	2020-01-26	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09	25
...	...	...	...	...	...	...	...	...	
56045	2021-11-18	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-19	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-20	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	
fips_code	date								
	2021-11-21	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5
	2021-11-22	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09	5

2107611 rows × 35 columns

In [57]:

```
covid_map_data.fillna(0, inplace = True)
```

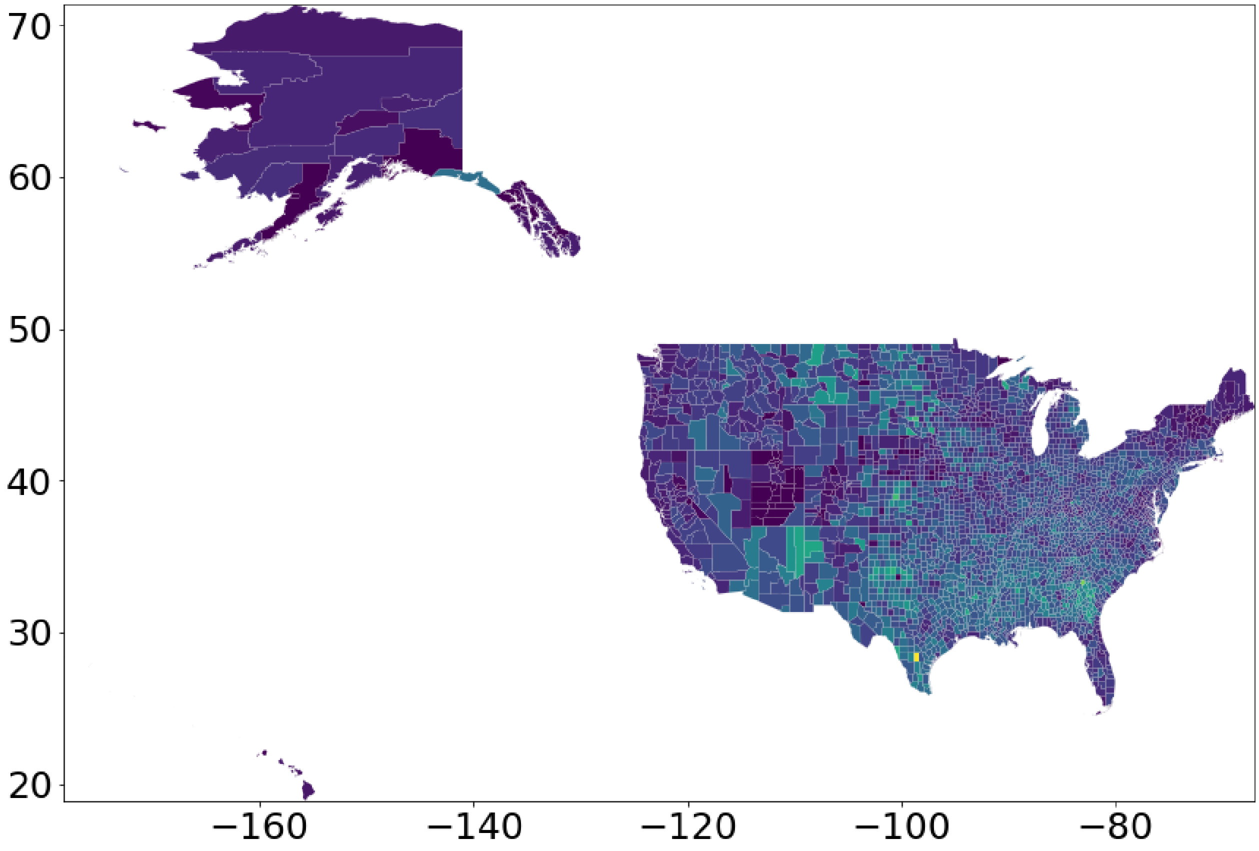
```
C:\Users\jzach\anaconda3\lib\site-packages\pandas\core\frame.py:4462: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().fillna(
```

In [58]:

```
fig, ax = plt.subplots(figsize = (20,10))
plt.rcParams.update({"font.size": 30})
plt.xticks(fontsize = 25)
plt.yticks(fontsize = 25)
key = "Deaths per Million"
df = covid_map_data[covid_map_data.index.get_level_values("date")==date]
df.plot(ax = ax, column = key, linewidth = .1,
       edgecolor = "lightgrey")
```

Out[58]:

```
<AxesSubplot:>
```



This looks good, but alaska and hawaii should be moved

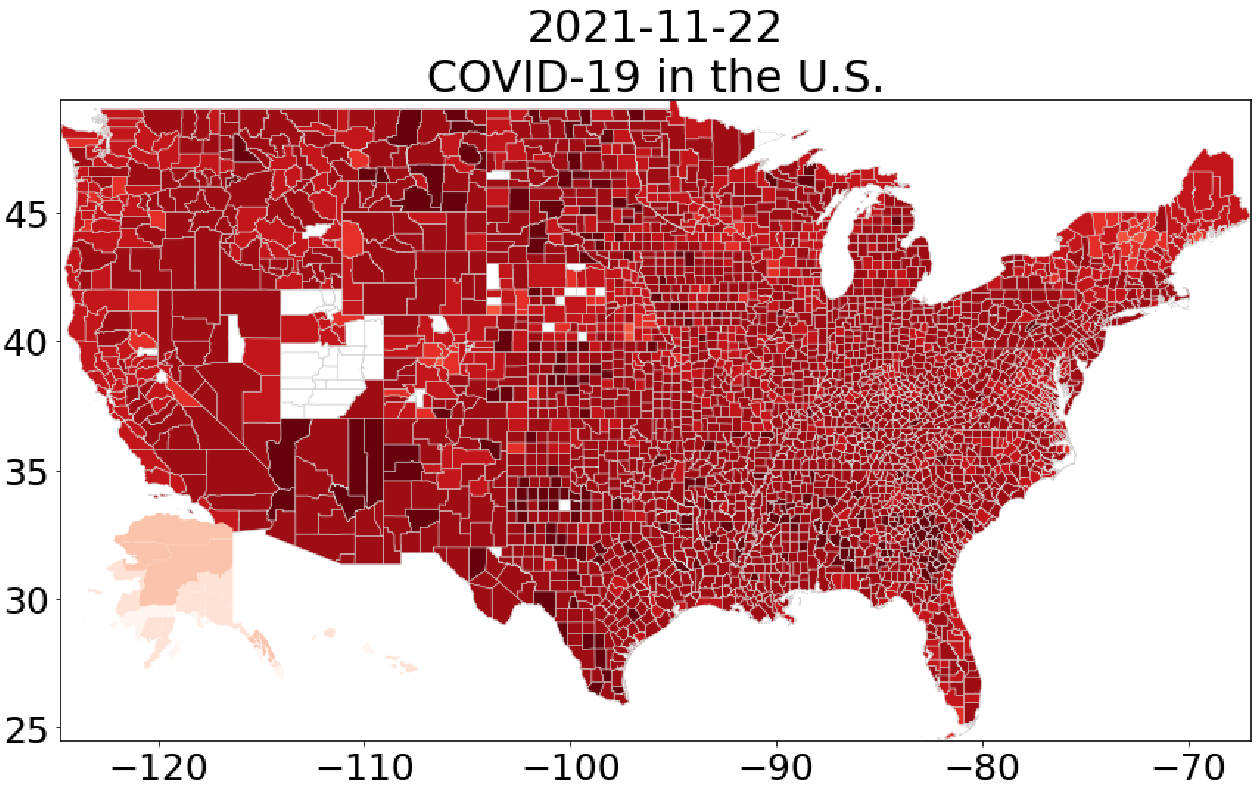
In [59]:

```python
from mpl_toolkits.axes_grid.inset_locator import inset_axes
fig, ax = plt.subplots(figsize=(18,8),
        subplot_kw = {'aspect': 'equal'})
plt.rcParams.update({"font.size": 30})
plt.xticks(fontsize = 25)
plt.yticks(fontsize = 25)
key = "Deaths per Million"
map_data = covid_map_data[covid_map_data.index.get_level_values("date")== date]
df = map_data[~map_data["state"].str.contains("Alaska|Hawaii")]
cmap = cm.get_cmap('Reds', 10)
vmin = 1
vmax = df[key].max()
norm = cm.colors.LogNorm(vmin=vmin, vmax =vmax)
plt.cm.ScalarMappable(cmap=cmap, norm=norm)
df.plot(ax=ax, cax = ax, column=key, vmin=vmin ,vmax = vmax,
             cmap = cmap, legend=False, linewidth=.5, edgecolor='lightgrey',
             norm = norm)
ax.set_title(str(date)[:10] + "\n" + "COVID-19 in the U.S.", fontsize = 30)
axins = {}
axins["Alaska"] = inset_axes(ax, width="17%", height="35%", loc="lower left")
axins["Hawaii"] = inset_axes(ax, width="50%", height="40%", loc="lower left")
for state in axins.keys():
    axins[state].set_xticks([])
    axins[state].set_yticks([])
    axins[state].axis("off")
    map_data[map_data["state"].str.contains(state)].plot(
        ax = axins[state], cax = ax, cmap = cmap, norm = norm)
axins["Hawaii"].set_xlim(-161, -154)
axins["Alaska"].set_ylim(53, 71)
```

```
<ipython-input-59-847b75fe5379>:1: MatplotlibDeprecationWarning: 
The mpl_toolkits.axes_grid module was deprecated in Matplotlib 2.1 and will be removed t
wo minor releases later. Use mpl_toolkits.axes_grid1 and mpl_toolkits.axisartist, which 
provide the same functionality instead.
  from mpl_toolkits.axes_grid.inset_locator import inset_axes
```

Out[59]:

```
(53.0, 71.0)
```



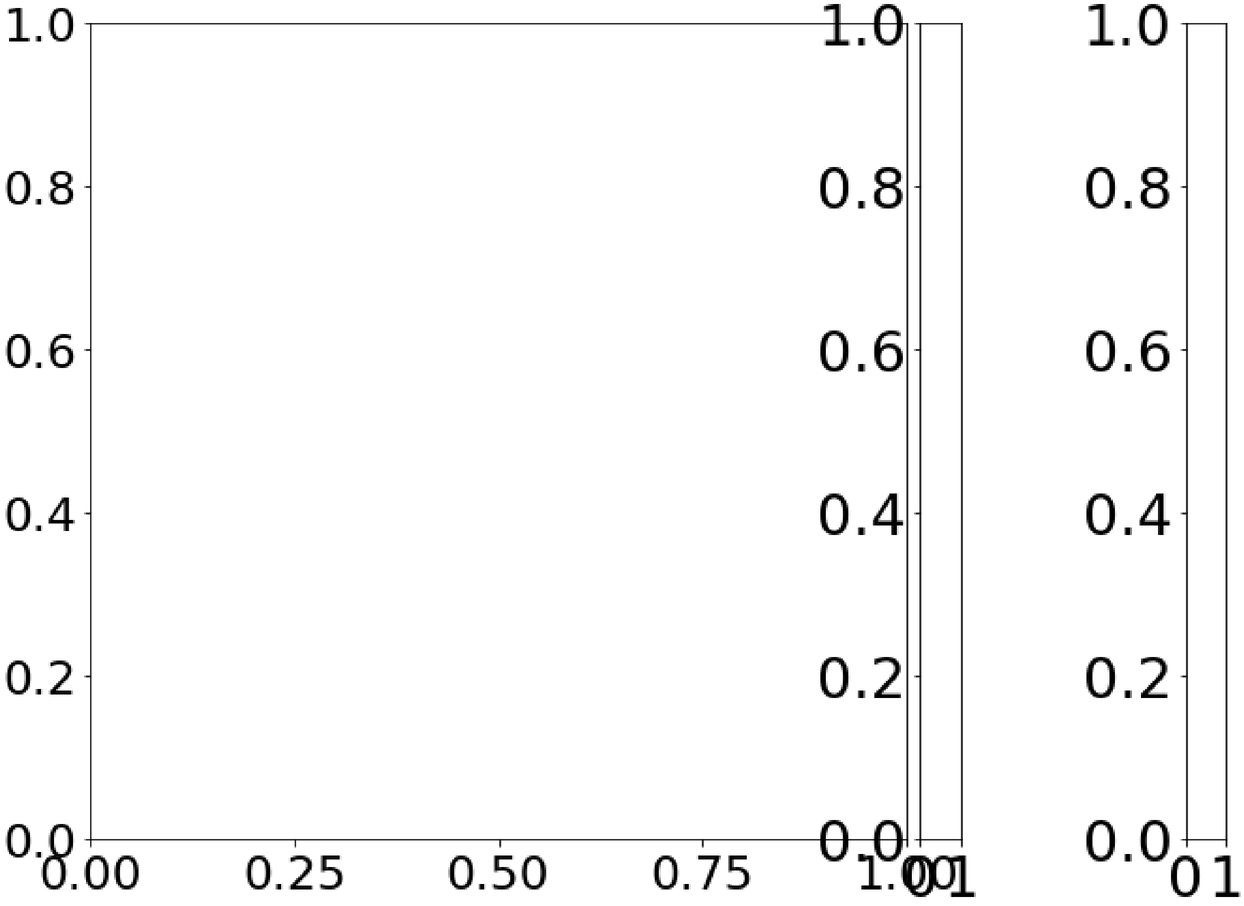
We can also add a colorbar:

In [70]:

```python
fig, ax = plt.subplots(figsize=(18,8),
        subplot_kw = {'aspect': 'equal'})
plt.rcParams.update({"font.size": 30})
plt.xticks(fontsize = 25)
plt.yticks(fontsize = 25)
key = "Deaths per Million"
# this time we replace 0 values with 1
# so that these values show up as beige instead of as white
# when color axis is logged
map_data = covid_map_data[covid_map_data.index.get_level_values("date")== date]
df = map_data[~map_data["state"].str.contains("Alaska|Hawaii")]
# set range of colorbar
vmin = 1
vmax = df[key].max()
# choose colormap
cmap = cm.get_cmap('Reds', 10)
# format colormap
norm = cm.colors.LogNorm(vmin = vmin, vmax = vmax)
sm = cm.ScalarMappable(cmap=cmap, norm=norm)
# empty array for the data range
sm._A = []
# prepare space for colorbar
divider = make_axes_locatable(ax)
size = "5%"
cax = divider.append_axes("right", size = size, pad = 0.1)
```

```
# add colorbar to figure
cbar = fig.colorbar(mappable=cmap)
cbar.ax.tick_params(labelsize=18)
vals = list(cbar.ax.get_yticks())
vals.append(vmax)
# format colorbar values as int
cbar.ax.set_yticklabels([int(x) for x in vals])
cbar.ax.set_ylabel(key, fontsize = 20)


df.plot(ax=ax, cax = cax, column=key, vmin=vmin ,vmax = vmax,
             cmap = cmap, legend=False, linewidth=.5, edgecolor='lightgrey',
             norm = norm)
ax.set_title(str(date)[:10] + "\n" + "COVID-19 in the U.S.", fontsize = 30)
axins = {}
axins["Alaska"] = inset_axes(ax, width="17%", height="30%", loc="lower left")
axins["Hawaii"] = inset_axes(ax, width="50%", height="40%", loc="lower left")
for state in axins.keys():
    axins[state].set_xticks([])
    axins[state].set_yticks([])
    axins[state].axis("off")
    map_data[map_data["state"].str.contains(state)].plot(
        ax = axins[state], cax = ax, cmap = cmap, norm = norm)
axins["Hawaii"].set_xlim(-161, -154)
axins["Alaska"].set_ylim(53, 71)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-70-5c97f0cd3047> in <module>
     25 cax = divider.append_axes("right", size = size, pad = 0.1)
     26 # add colorbar to figure
---> 27 cbar = fig.colorbar(mappable=cmap)
     28 cbar.ax.tick_params(labelsize=18)
     29 vals = list(cbar.ax.get_yticks())

~\AppData\Roaming\Python\Python38\site-packages\matplotlib\figure.py in colorbar(self, m
appable, cax, ax, use_gridspec, **kw)
   1174                               'panchor']
   1175         cb_kw = {k: v for k, v in kw.items() if k not in NON_COLORBAR_KEYS}
-> 1176         cb = cbar.Colorbar(cax, mappable, **cb_kw)
   1177 
   1178         self.sca(current_ax)

~\AppData\Roaming\Python\Python38\site-packages\matplotlib\colorbar.py in __init__(self,
ax, mappable, **kwargs)
   1169         # Ensure the given mappable's norm has appropriate vmin and vmax set
   1170         # even if mappable.draw has not yet been called.
-> 1171         if mappable.get_array() is not None:
   1172             mappable.autoscale_None()
   1173 

AttributeError: 'LinearSegmentedColormap' object has no attribute 'get_array'
```


1.0
0.8
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
1.0
0.8
0.6
0.4
0.2
0.0
0
1
1.0
0.8
0.6
0.4
0.2
0.0
0
1