In [ ]:
```
##Covid-19 Analysis with Geocoded Data
```

In [1]:
```
# !pip install GDAL-3.3.0-cp38-cp38-win_amd64.whl
# !pip install Fiona-1.8.20-cp38-cp38-win_amd64.whl
# !pip install Shapely-1.7.1-cp38-cp38-win_amd64.whl
# !pip install datadotworld
```

In [2]:

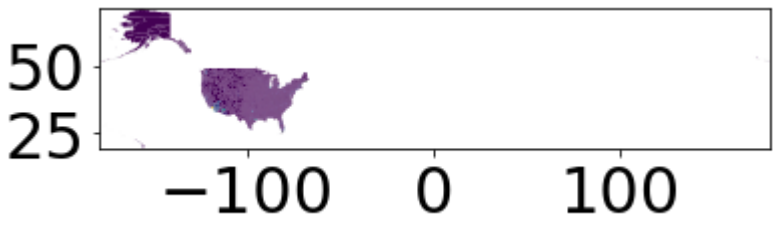
```
#COVID19Map.py
### import all modules that we will use here:
import geopandas
import numpy as np
import pandas as pd
# We won't actually use datetime directly. Since the dataframe index will use
# data formatted as datetime64, I import it in case I need to use the datetime
# module to troubleshoot later
import datetime
# you could technically call many of the submodules from matplotlib using mpl.,
#but for convenience we explicitly import submodules. These will be used for
# constructing visualizations
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import cm
# If we choose to make a dynamic visualization for the homework
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_
    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': '
    '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\JLCat\AppData\Roaming\Python\Python38\site-packages\datadotworld\model
s\dataset.py:206: UserWarning: Unable to set data frame dtypes automatically us
ing 2_cases_and_deaths_by_county_timeseries schema. Data types may need to be a
djusted manually. Error: Integer column has NA values in column 2
  warnings.warn(
```

In [3]:
```
map_data.plot(column = "Population")
```

Out[3]: <AxesSubplot:>



In [4]:
```
map_data
```

Out[4]:

fips_code	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND	A
21007	21	007	00516850	0500000US21007	Ballard	06	639387454	6
21017	21	017	00516855	0500000US21017	Bourbon	06	750439351	
21031	21	031	00516862	0500000US21031	Butler	06	1103571974	1
21065	21	065	00516879	0500000US21065	Estill	06	655509930	
21069	21	069	00516881	0500000US21069	Fleming	06	902727151	
...	...	...	...	...	...	...	...	
31073	31	073	00835858	0500000US31073	Gosper	06	1186616237	1
39075	39	075	01074050	0500000US39075	Holmes	06	1094405866	
48171	48	171	01383871	0500000US48171	Gillespie	06	2740719114	
55079	55	079	01581100	0500000US55079	Milwaukee	06	625440563	245
26139	26	139	01623012	0500000US26139	Ottawa	06	1459502408	276

3142 rows × 11 columns

In [5]:
```
covid_data
```

Out[5]:

fips_code	date	uid	location_type	location_name	state	total_population	cumulative_ca
	2020-01-22	84001001	county	Autauga	Alabama	55200.0	
	2020-01-23	84001001	county	Autauga	Alabama	55200.0	
1001	2020-01-24	84001001	county	Autauga	Alabama	55200.0	
	2020-01-25	84001001	county	Autauga	Alabama	55200.0	
	2020-01-26	84001001	county	Autauga	Alabama	55200.0	
...	...	...	...	...	...	...	
	2021-11-18	84056045	county	Weston	Wyoming	7100.0	1
	2021-11-19	84056045	county	Weston	Wyoming	7100.0	1
56045	2021-11-20	84056045	county	Weston	Wyoming	7100.0	1
	2021-11-21	84056045	county	Weston	Wyoming	7100.0	1
	2021-11-22	84056045	county	Weston	Wyoming	7100.0	1

2109624 rows × 17 columns

In [6]:

```python
#COVID19Map.py
# . . .

def create_merged_geo_dataframe(data, map_data, dates):
    data_frame_initialized = False
    # use groupby to generate a df with only the fips_code index
    # then save that index as the variable counties
    counties =data.groupby("fips_code").mean().index
    for date in dates:
        # select county observations from each date in dates
        # select only the subset of counties in the map that
        # are also present in the covid_data
        agg_df = map_data[map_data.index.isin(counties)]
        agg_df["date"] = date
        if data_frame_initialized == False:
            #Create new dataframe
            matching_gpd = geopandas.GeoDataFrame(
                agg_df, crs = map_data.crs)
            data_frame_initialized = True

        else:
            # or stack thenew  data frame and the dataframe that was initialized
            # i = 0
            matching_gpd = matching_gpd.append(agg_df, ignore_index = False)
    # reset index, then set to [fips, date]
    matching_gpd.reset_index(inplace=True)
    matching_gpd.set_index(["fips_code", "date"], inplace = True)
    matching_gpd.drop("state", axis = 1, inplace = True)
#     for key, val in data.items():
#         matching_gpd[key] = val
    matching_gpd = pd.concat([matching_gpd, data], axis=1)

    return matching_gpd

# dates will be used to create a geopandas DataFrame with multiindex
dates = sorted(list(set(covid_data.index.get_level_values("date"))))
covid_data = create_merged_geo_dataframe(covid_data, map_data, dates)
```

In [7]:
```
covid_data
```

Out[7]:

fips_code	date	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAN
	2020-01-22	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-23	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
1001	2020-01-24	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-25	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-26	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
...	...	...	...	...	...	...	...	
	2021-11-18	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-19	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
56045	2021-11-20	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-21	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-22	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0

2109624 rows × 27 columns

In [8]:

```
#COVID19Map.py
# . . .
def create_new_vars(covid_data, moving_average_days):
    # use a for loop that performs the same operations on data for cases and for
    for key in ["cases", "deaths"]:
        # create a version of the key with the first letter capitalized
        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)
        # generate daily data normalized per million population by taking the dai
        # entity (covid_data.index.names[0]), dividing this value by population d
        # by 1 million 10 ** 6
        covid_data["Daily " + cap_key] = covid_data["new_" + key ]

        covid_data["Daily " + cap_key + " " + str(moving_average_days) + " Day MA
            "new_" + key + "_7_day_rolling_avg"]

        # taking the rolling average; choice of number of days is passed as movir
        covid_data["Daily " + cap_key + " per Million " + str(moving_average_days
            covid_data["Daily " + cap_key + " " + str(moving_average_days) + " Da
            .div(covid_data["total_population"]).mul(10 ** 6)

# . . .
moving_average_days = 7
create_new_vars(covid_data, moving_average_days)
start_date = "03-15-2020"
end_date = dates[-1]
```

In [9]:

```
#COVID19Map.py
#. . .

# only include observations within these boundaries
# this will shrink the size of the map
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["m
    miny = covid_data[covid_data.index.get_level_values("date")== date].bounds["m
    maxx = -58
    maxy = covid_data[covid_data.index.get_level_values("date")== date].bounds["m
    # 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_lev
    counties = bounded_data.groupby("fips_code").mean().index
    covid_map_data =covid_data[covid_data.index.get_level_values("fips_code").isi
    map_bounded = True
```

In [10]:
```
%matplotlib inline
covid_map_data.fillna(0, inplace = True)
covid_map_data
```

```
C:\Users\JLCat\anaconda3\lib\site-packages\pandas\core\frame.py:4462: SettingWi
thCopyWarning:
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/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  return super().fillna(
```

Out[10]:

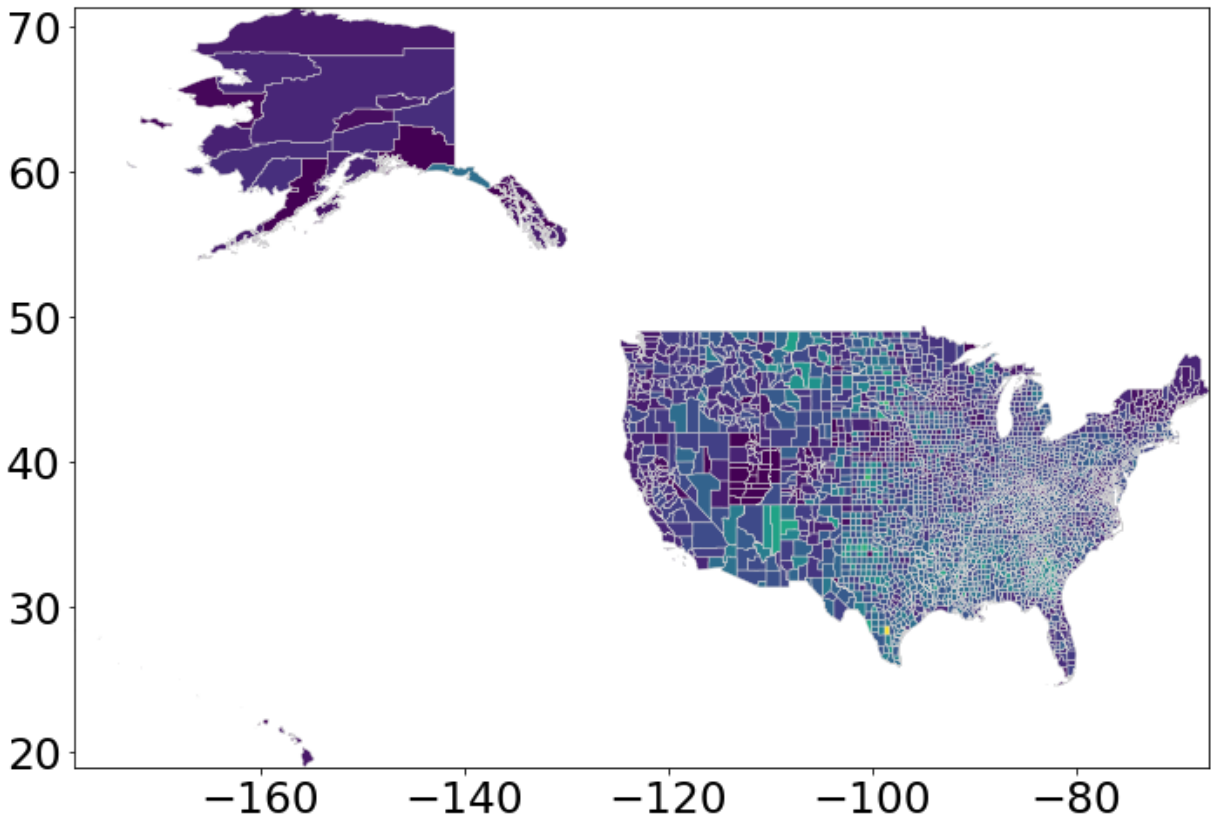
		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAN
fips_code	**date**							
	2020-01-22	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-23	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
1001	**2020-01-24**	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-25	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2020-01-26	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
...	**...**	...	...	...	...	...	...	
56045								
	2021-11-18	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-19	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0

		STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAN
fips_code	**date**							
	2021-11-20	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-21	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0
	2021-11-22	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+0

2107611 rows × 35 columns

In [11]:
```
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"
df = covid_map_data[covid_map_data.index.get_level_values("date")== date]
df.plot(ax=ax, cax = ax, column=key, linewidth=.5,
             edgecolor='lightgrey')
```

Out[11]: <AxesSubplot:>



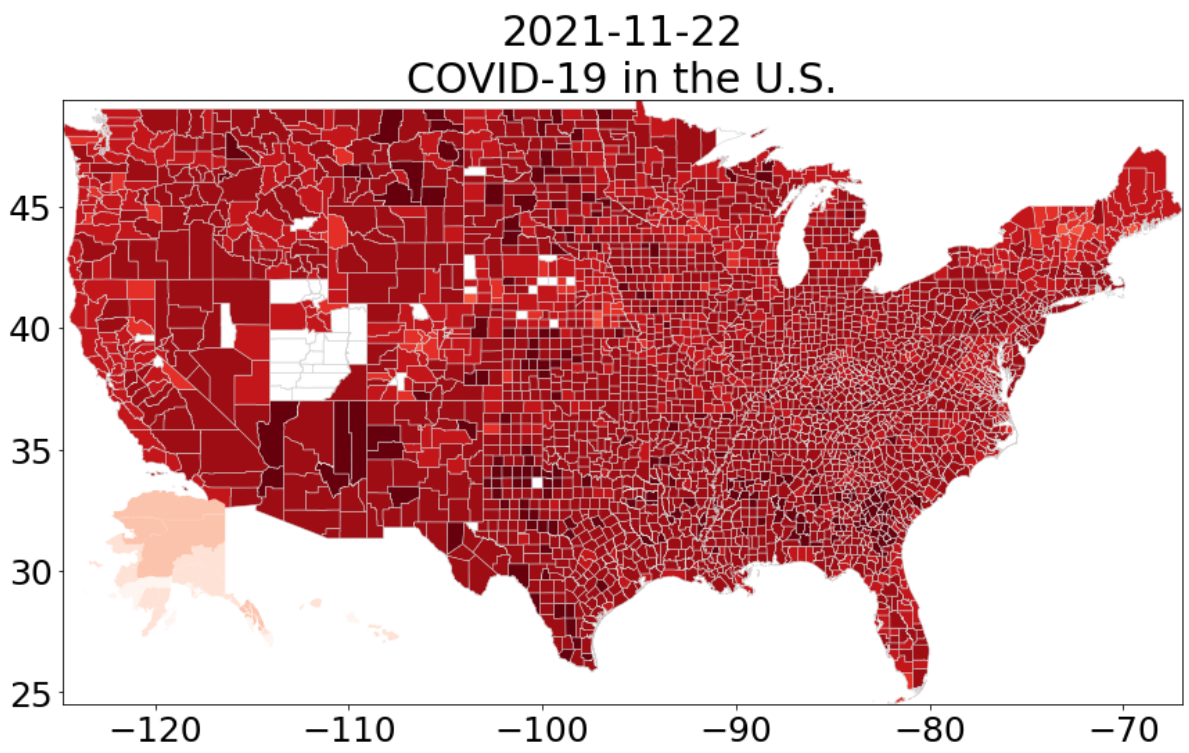
In [12]:
```
map_data["state"]
```

Out[12]:
```
fips_code
21007      Kentucky
21017      Kentucky
21031      Kentucky
21065      Kentucky
21069      Kentucky
            ...
31073      Nebraska
39075          Ohio
48171         Texas
55079     Wisconsin
26139      Michigan
Name: state, Length: 3142, dtype: object
```

In [13]:
```
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-13-a99b906bcdb6>:1: MatplotlibDeprecationWarning:
The mpl_toolkits.axes_grid module was deprecated in Matplotlib 2.1 and will be
removed two 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[13]:
```
(53.0, 71.0)
```


2021-11-22
COVID-19 in the U.S.
45
40
35
30
25
−120
−110
−100
−90
−80
−70

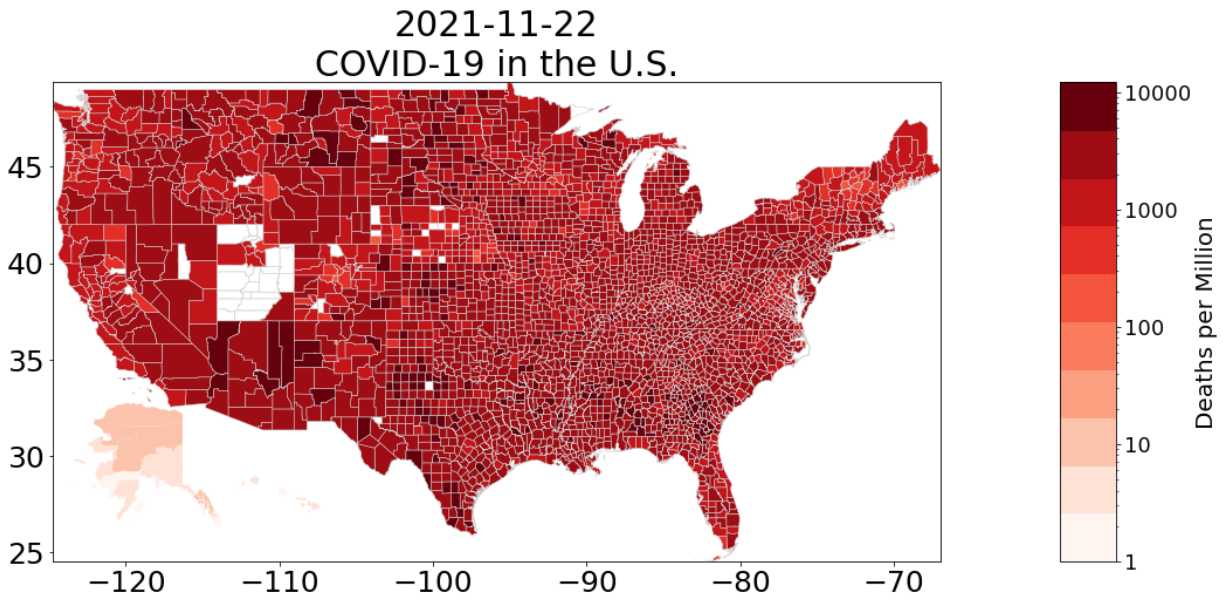
In [14]:

```
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(sm, cax=cax, cmap = 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)
```

```
<ipython-input-14-aaa07216ea92>:27: MatplotlibDeprecationWarning: The 'cmap'
parameter to Colorbar has no effect because it is overridden by the mappable;
it is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-14-aaa07216ea92>:32: UserWarning: FixedFormatter should only b
```

```
e used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```

Out[14]: (53.0, 71.0)

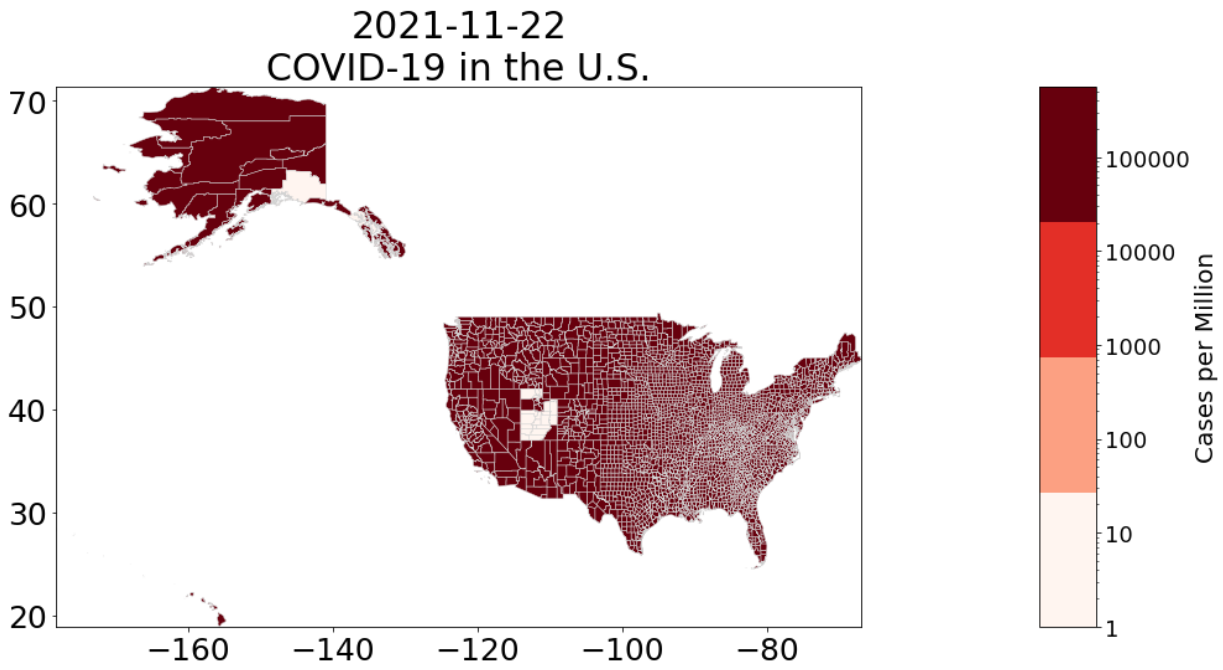


Congratulations! You've made a quality map. Next, let's use a for loop to create maps for all for categories that we plotted in the previous post. This is simple. First we define a list of keys, and iterate over each key.

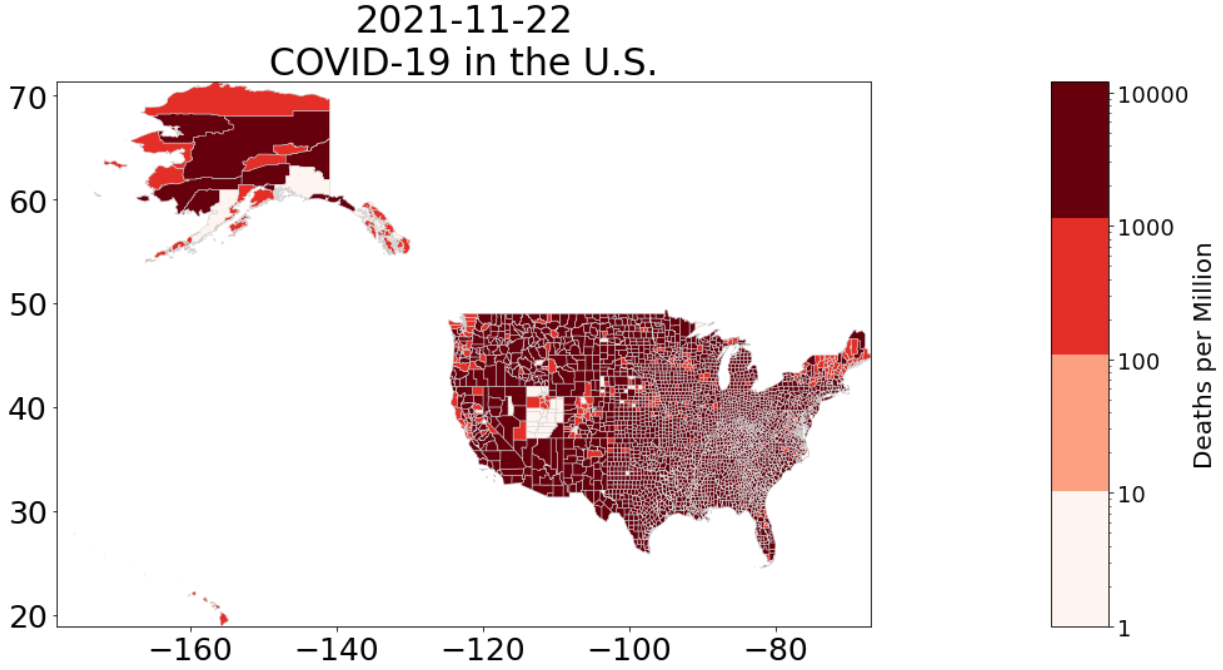
In [15]:
```
keys = ["Cases per Million", "Deaths per Million",
        "Daily Cases per Million 7 Day MA", "Daily Deaths per Million 7 Day MA"]
for key in keys:
    # identify whether or not to log values for color axis
    # if daily rates, do not log. Only log totals.
    log = False if "Daily" in key else True
    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)
    # 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
    df = covid_map_data[covid_map_data.index.get_level_values("date")== date].rep
    # set range of colorbar
    vmin = 1 if log else 0
    vmax = df[key].max()
    # choose colormap
    cmap = cm.get_cmap('Reds', 4)
    # format colormap
    if log:
        norm = cm.colors.LogNorm(vmin=vmin, vmax =vmax)
    else:
        norm = cm.colors.Normalize(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(sm, cax=cax, cmap = 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)

    plt.show()
    plt.close()
```

```
<ipython-input-15-ed397f5f4d4e>:34: MatplotlibDeprecationWarning: The 'cmap'
parameter to Colorbar has no effect because it is overridden by the mappable;
it is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-15-ed397f5f4d4e>:40: UserWarning: FixedFormatter should only b
```

```
e used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```

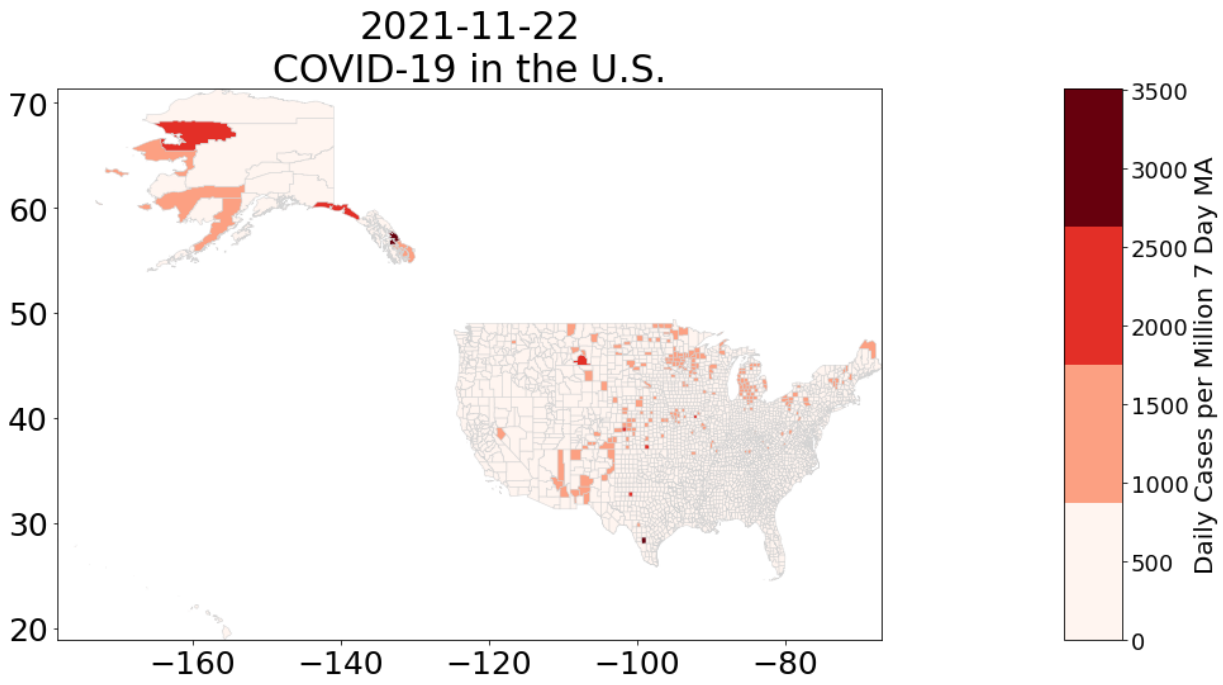



```
<ipython-input-15-ed397f5f4d4e>:34: MatplotlibDeprecationWarning: The 'cmap' pa
rameter to Colorbar has no effect because it is overridden by the mappable; it
is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-15-ed397f5f4d4e>:40: UserWarning: FixedFormatter should only be
used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```

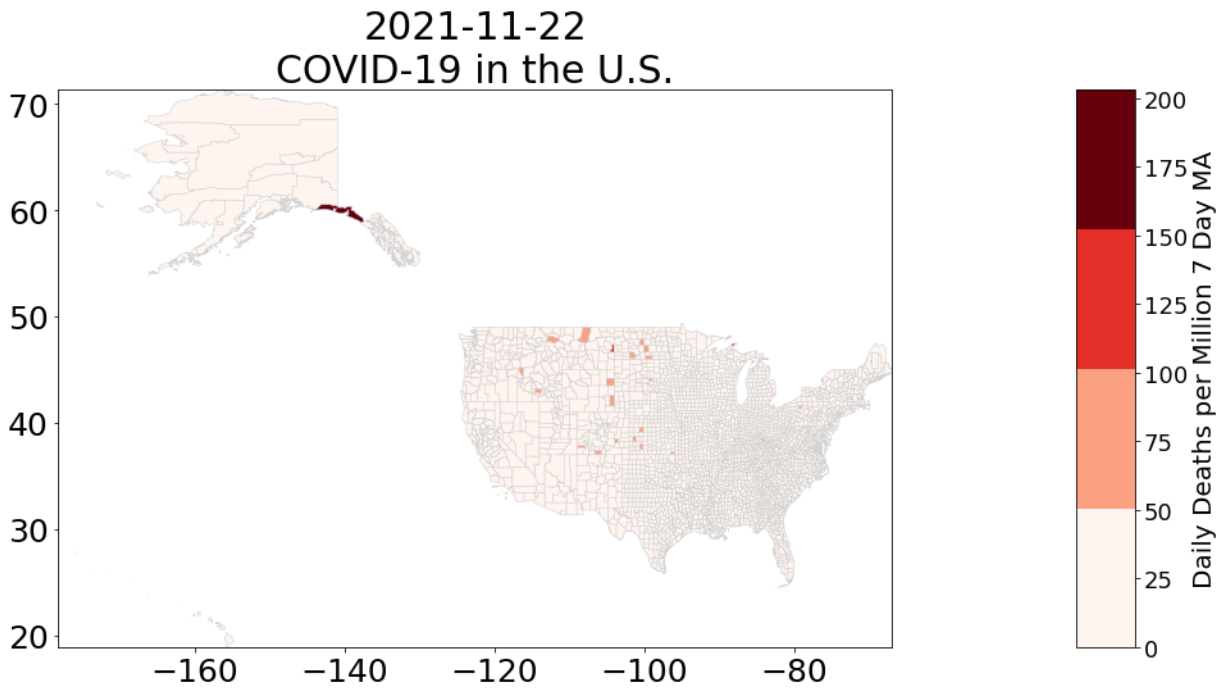



```
<ipython-input-15-ed397f5f4d4e>:34: MatplotlibDeprecationWarning: The 'cmap'
parameter to Colorbar has no effect because it is overridden by the mappable;
it is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-15-ed397f5f4d4e>:40: UserWarning: FixedFormatter should only b
```

```
e used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```




```
<ipython-input-15-ed397f5f4d4e>:34: MatplotlibDeprecationWarning: The 'cmap' pa
rameter to Colorbar has no effect because it is overridden by the mappable; it
is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-15-ed397f5f4d4e>:40: UserWarning: FixedFormatter should only be
used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```

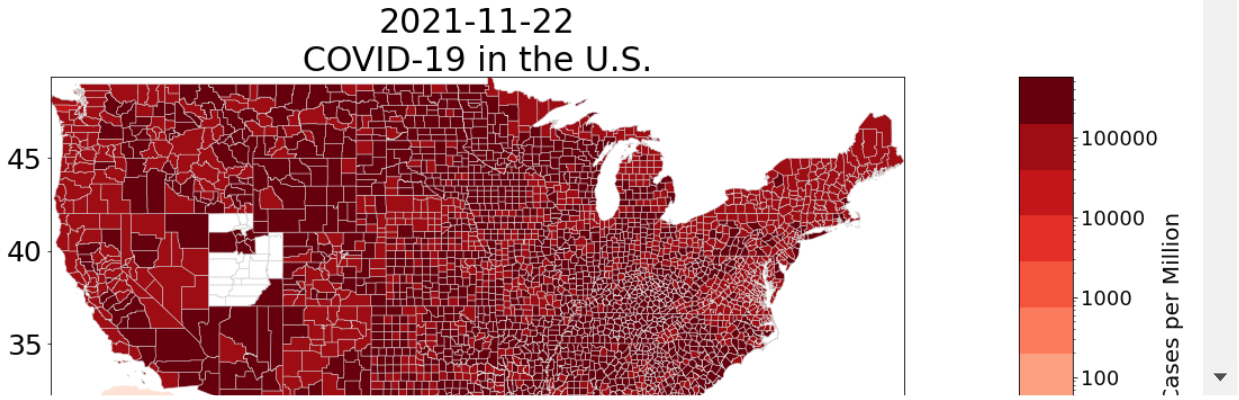


In [16]:

```
keys = ["Cases per Million", "Deaths per Million",
        "Daily Cases per Million 7 Day MA", "Daily Deaths per Million 7 Day MA"]
for key in keys:
    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)
    # 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")== dat
    df = map_data[~map_data["state"].str.contains("Alaska|Hawaii")]
    vmin = 1
    vmax = df[key].max()

    cmap = cm.get_cmap('Reds', 10)
    # only log level values
    if "Daily" not in key:
        norm = cm.colors.LogNorm(vmin=vmin, vmax =vmax)
    # if daily rate, do not log values
    else:
        norm = cm.colors.Normalize(vmin = vmin, vmax = vmax)

    sm = cm.ScalarMappable(cmap=cmap, norm=norm)
    # empty array for the data range
    sm._A = []
    # add the colorbar to the figure
    divider = make_axes_locatable(ax)
    size = "5%"
    # prepare space for the color bar
    cax = divider.append_axes("right", size = size, pad = 0.1)
    # add colorbar to figure
    cbar = fig.colorbar(sm, cax=cax, cmap = 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)

    plt.show()
    plt.close()
```

```
<ipython-input-16-a08a33da1581>:34: MatplotlibDeprecationWarning: The 'cmap'
parameter to Colorbar has no effect because it is overridden by the mappable;
it is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap = cmap)
<ipython-input-16-a08a33da1581>:40: UserWarning: FixedFormatter should only b
e used together with FixedLocator
  cbar.ax.set_yticklabels([int(x) for x in vals])
```



In [17]:
```
import pandas as pd
#import unemployment data
u_data = pd.read_csv(
    "countyUnemploymentData.csv", encoding = "latin1", parse_dates = True, index_
# drop observations with missing fips codes
u_data.reset_index(inplace = True)
index = u_data["fips_code"].dropna(axis = 0).index
u_data = u_data.loc[index]

u_data["fips_code"] = u_data["fips_code"].astype(int)
u_data.set_index(["fips_code", "date"], inplace = True)
```

```
C:\Users\JLCat\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:316
5: DtypeWarning: Columns (7) have mixed types.Specify dtype option on import or
set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [18]:
```
u_data
```

Out[18]:

fips_code	date	state_fips	county_fips	Location	Labor Force	Employed	Unemployed	Unemployment Rate
1001	**2019-08-01**	1	1	Autauga County AL	26,079	25,368	711	2.7
1003	**2019-08-01**	1	3	Baldwin County AL	97,939	95,367	2,572	2.6
1005	**2019-08-01**	1	5	Barbour County AL	8,652	8,322	330	3.8
1007	**2019-08-01**	1	7	Bibb County AL	8,670	8,403	267	3.1
1009	**2019-08-01**	1	9	Blount County AL	25,309	24,641	668	2.6
...	**...**	...	...	...	...	...	...	...
72145	**2021-06-01**	72	145	Vega Baja Municipio PR	12,993	11,580	1,413	10.9
72147	**2021-06-01**	72	147	Vieques Municipio PR	2,610	2,288	322	12.3
72149	**2021-06-01**	72	149	Villalba Municipio PR	6,940	6,152	788	11.4
72151	**2021-06-01**	72	151	Yabucoa Municipio PR	8,352	7,480	872	10.4
72153	**2021-06-01**	72	153	Yauco Municipio PR	9,212	8,127	1,085	10.4

74037 rows × 7 columns

In [19]:
```
map_data = map_data.reset_index().set_index("fips_code")
map_data.drop("date", axis = 1, inplace = True)
map_data
```

Out[19]:

fips_code	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND
1001	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+09
1003	1.0	003	00161527	0500000US01003	Baldwin	06	4.117547e+09
1005	1.0	005	00161528	0500000US01005	Barbour	06	2.292145e+09
1007	1.0	007	00161529	0500000US01007	Bibb	06	1.612167e+09
1009	1.0	009	00161530	0500000US01009	Blount	06	1.670104e+09
...	...	...	...	...	...	...	..
56037	56.0	037	01609192	0500000US56037	Sweetwater	06	2.700575e+10
56039	56.0	039	01605083	0500000US56039	Teton	06	1.035178e+10
56041	56.0	041	01605084	0500000US56041	Uinta	06	5.391632e+09
56043	56.0	043	01605085	0500000US56043	Washakie	06	5.798139e+09

fips_code	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAND
56045	56.0	045	01605086	0500000US56045	Weston	06	6.210804e+09

3141 rows × 35 columns

In [20]:
```
# choose the dates
dates = u_data.groupby("date").mean().index
u_data = create_merged_geo_dataframe(u_data, map_data, dates)
```

```
C:\Users\JLCat\anaconda3\lib\site-packages\geopandas\geodataframe.py:1322: Sett
ingWithCopyWarning:
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/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  super(GeoDataFrame, self).__setitem__(key, value)
```

In [21]:
```
u_data
```

Out[21]:

fips_code	date	STATEFP	COUNTYFP	COUNTYNS	AFFGEOID	NAME	LSAD	ALAN
	2019-08-01	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2019-09-01	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
1001	2019-10-01	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2019-11-01	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
	2019-12-01	1.0	001	00161526	0500000US01001	Autauga	06	1.539602e+0
...	...	...	...	...	...	...	...	
	2021-02-01	NaN	NaN	NaN	NaN	NaN	NaN	Na
	2021-03-01	NaN	NaN	NaN	NaN	NaN	NaN	Na
72153	2021-04-01	NaN	NaN	NaN	NaN	NaN	NaN	Na
	2021-05-01	NaN	NaN	NaN	NaN	NaN	NaN	Na
	2021-06-01	NaN	NaN	NaN	NaN	NaN	NaN	Na

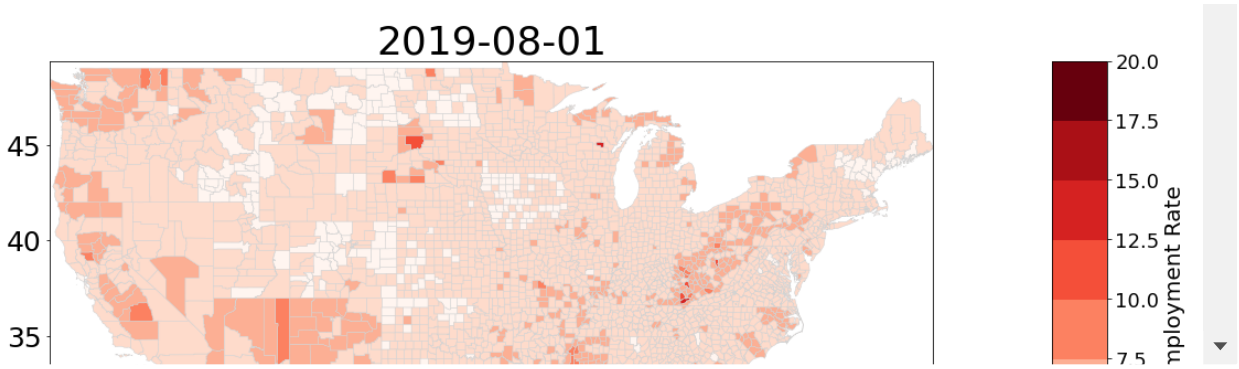
74037 rows × 41 columns

In [22]:
```
# choose map bounds
#if "u_map_bounded" not in locals():
minx = -127
miny = 23
maxx = -66
maxy = 50
u_data = select_data_within_bounds(u_data, minx, miny, maxx, maxy)
u_map_bounded = True
```

In [23]:
```
counties = u_data.groupby("fips_code").mean().index
```

In [24]:
```
key = "Unemployment Rate"
# csv saved data as string, transform to float
u_data[key] = u_data[key].astype(float)
# create new pdf
pp = PdfPages("County Unemployment Rate.pdf")
for date in dates:
    fig, ax = plt.subplots(figsize=(19,9),
                           subplot_kw = {"aspect":"equal"})
    plt.rcParams.update({"font.size": 30})

    plt.xticks(fontsize = 25)
    plt.yticks(fontsize = 25)

    vmin = 0
    #vmax = u_data[key].fillna(0).max()
    vmax = 20
    cmap = cm.get_cmap("Reds", 8)
    norm = cm.colors.Normalize(vmin=vmin, vmax=vmax)
    ### add colorbar
    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 = .1)
    # add colorbar to figure
    cbar = fig.colorbar(sm, cax=cax, cmap= cmap)
    cbar.ax.tick_params(labelsize=18)
    vals = list(cbar.ax.get_yticks())
    vals.append(vmax)
    cbar.ax.set_yticklabels(vals)#[int(x) for x in vals])
    cbar.ax.set_ylabel(key, fontsize=20)

    # select data only from date
    df = u_data[u_data.index.get_level_values("date") == date]#.dropna(axis=0)
    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])
    plt.show()
#     pp.savefig(fig, bbox_inches = "tight")
    plt.close()

# close your pdf
# pp.close()
```

```
<ipython-input-24-c7b47e8bd807>:28: MatplotlibDeprecationWarning: The 'cmap'
parameter to Colorbar has no effect because it is overridden by the mappable;
it is deprecated since 3.3 and will be removed two minor releases later.
  cbar = fig.colorbar(sm, cax=cax, cmap= cmap)
<ipython-input-24-c7b47e8bd807>:32: UserWarning: FixedFormatter should only b
e used together with FixedLocator
  cbar.ax.set_yticklabels(vals)#[int(x) for x in vals])
```



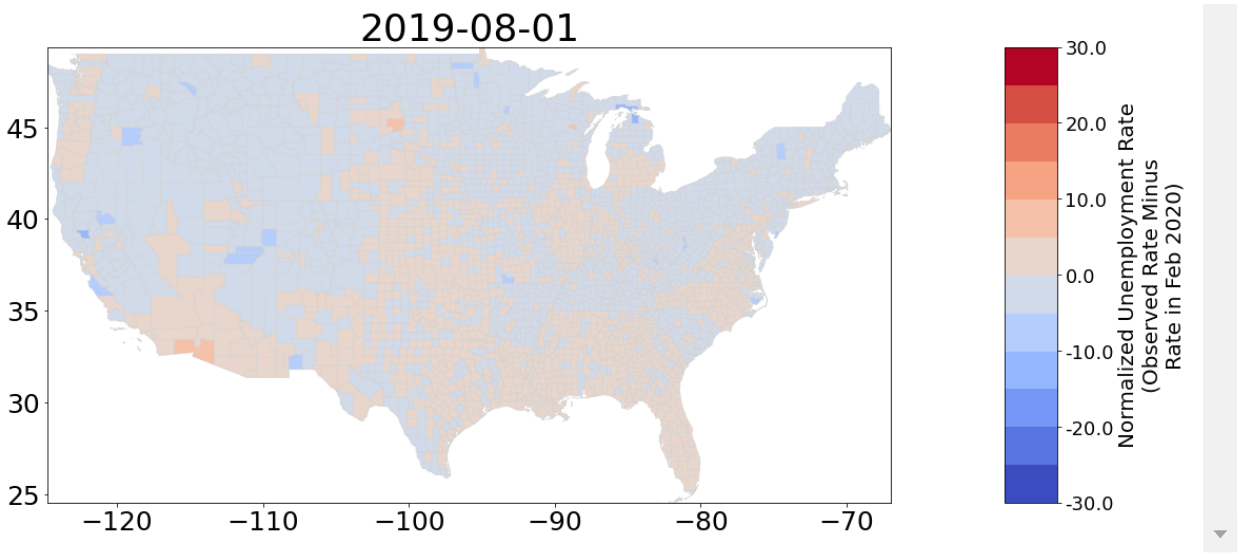
In [25]:
```
# matplotlib will give us warning because we are setting the value a slice
import warnings
warnings.filterwarnings("ignore")
# Normalize Unemployment Feb-20 == 1
key = "Unemployment Rate"
new_key = "Normalized " + key + " (Feb 2020)"
# df.copy() makes a copy of the dataframe
n_u_data = u_data.copy()
# set date first to greatly reduce computation costs
n_u_data[new_key] =  0.
n_u_data = n_u_data.reset_index().set_index(["date", "fips_code"])
# go through data for every county for the key and divide the value by the value
# at the date that you would to normalize to 1
#for county in counties:
#    n_u_data[key][county] = n_u_data.loc[county, key].div(n_u_data.loc[county, '
# take the difference between the observed rate and the Feb rate
for date in dates:
    n_u_data[new_key] =n_u_data[key].sub(n_u_data.loc[datetime.datetime(2020,2,1)
```

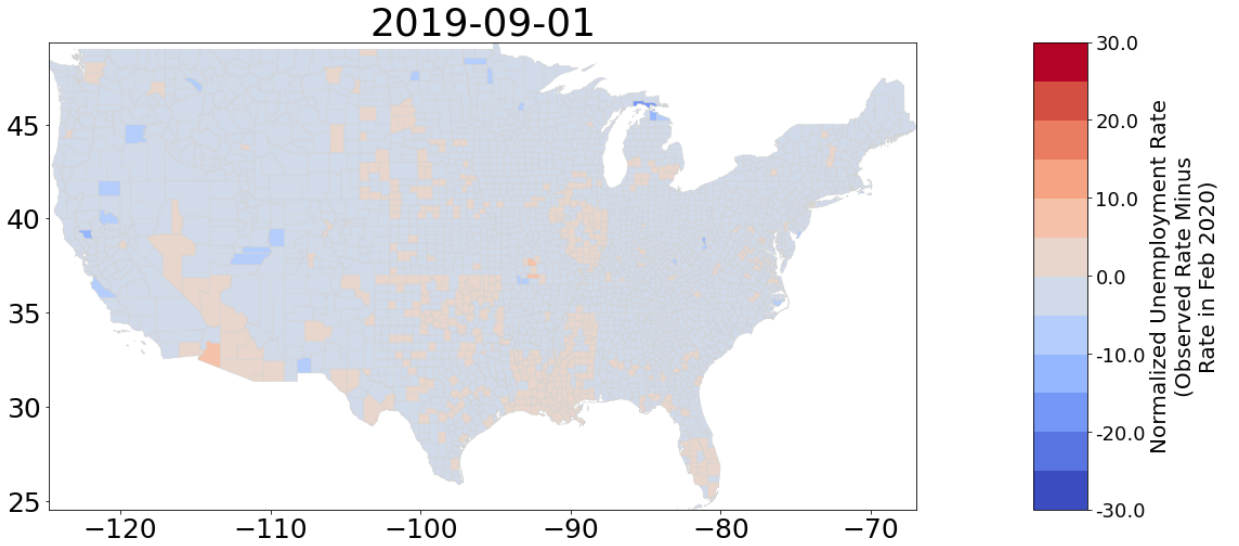
In [26]:
```
n_u_data[new_key]
```

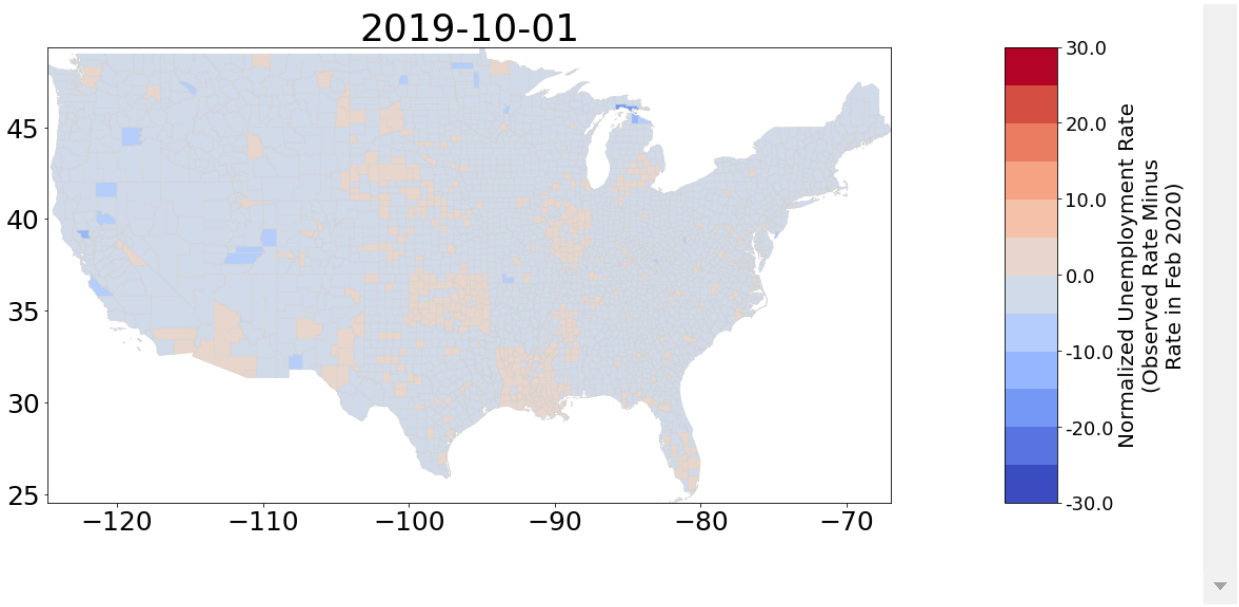
Out[26]:
```
date        fips_code
2019-08-01  1001         0.0
2019-09-01  1001        -0.3
2019-10-01  1001        -0.4
2019-11-01  1001        -0.5
2019-12-01  1001        -0.3
                        ...
2021-02-01  56045         1.5
2021-03-01  56045         1.0
2021-04-01  56045         0.7
2021-05-01  56045         0.8
2021-06-01  56045         0.6
Name: Normalized Unemployment Rate (Feb 2020), Length: 71484, dtype: float64
```

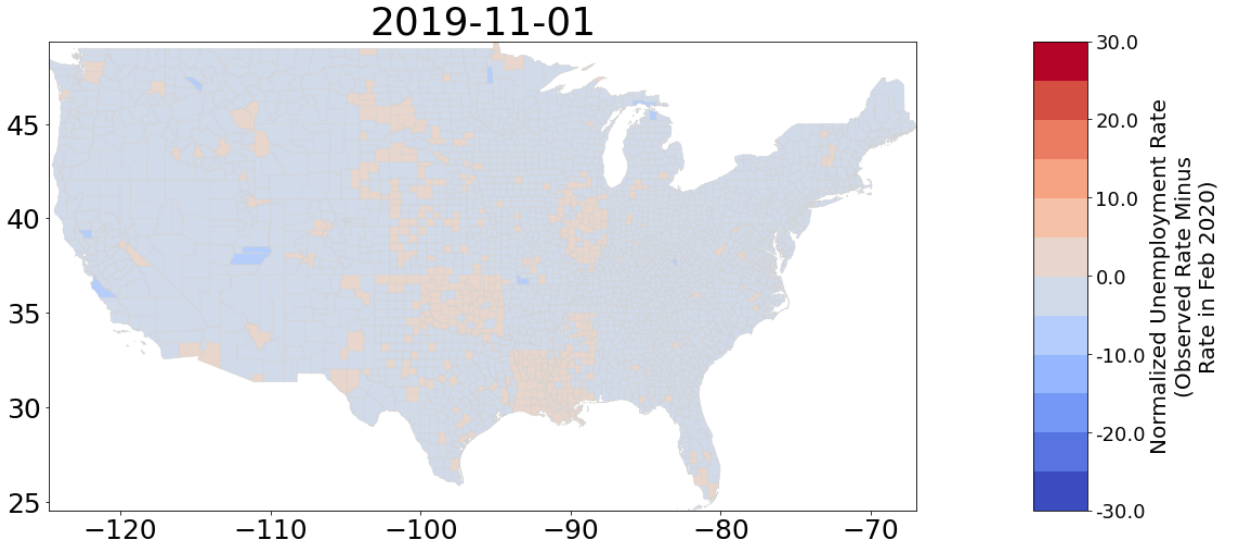
In [27]:

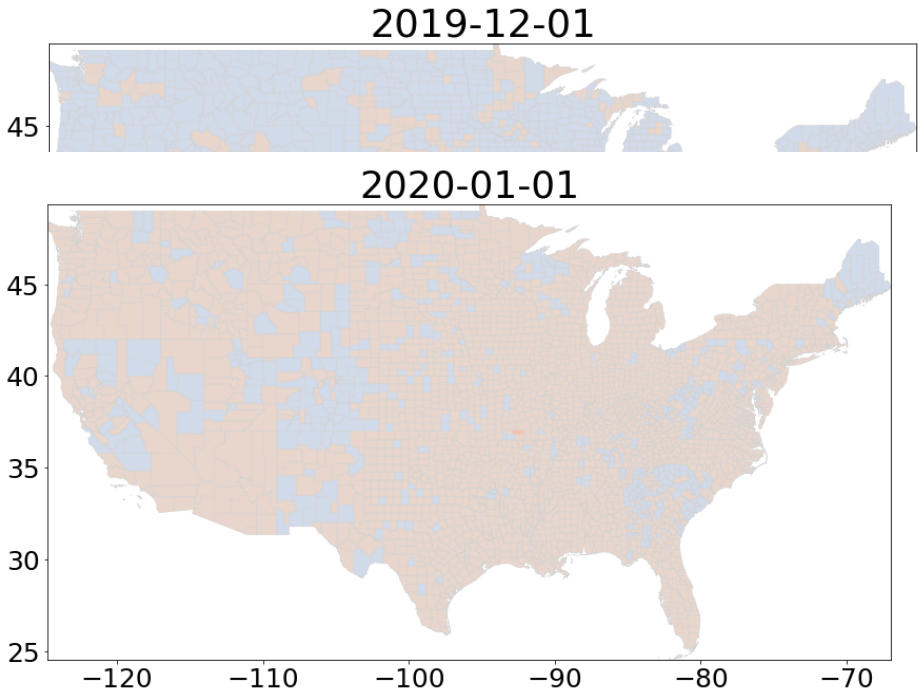
```
# pp = PdfPages("Normalized County Unemployment Rate.pdf")

for date in dates:
    # accomplishes same outcome as date.replace("-", " 20")
    title_date = str(date)[:10]
    fig, ax = plt.subplots(figsize=(19,9),
                           subplot_kw = {"aspect":"equal"})
    plt.rcParams.update({"font.size": 30})

    plt.xticks(fontsize = 25)
    plt.yticks(fontsize = 25)

    # set range color bar values
    vmin = -30# n_u_data["Unemployment Rate"].min()
    vmax = 30
    cmap = cm.get_cmap("coolwarm", 12)
    norm = cm.colors.Normalize(vmin=vmin, vmax=vmax)
    ### add colorbar
    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 = .1)
    # add colorbar to figure
    cbar = fig.colorbar(sm, cax=cax, cmap= cmap)
    cbar.ax.tick_params(labelsize=18)
    vals = list(cbar.ax.get_yticks())
    vals.append(vmax)
    cbar.ax.set_yticklabels(vals)
    cbar.ax.set_ylabel("Normalized "+key + "\n(Observed Rate Minus\nRate in Feb 2

    df = n_u_data.loc[date]#.dropna(axis=0)
    df.plot(ax=ax, cax=ax, column = new_key,
            vmin=vmin, vmax=vmax,
            cmap=cmap, legend=False,
            linewidth = .5, edgecolor="lightgrey",norm=norm)
    ax.set_title(title_date)
    plt.show()
#     pp.savefig(fig, bbox_inches = "tight")
    plt.close()
# pp.close()
```


2019-08-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

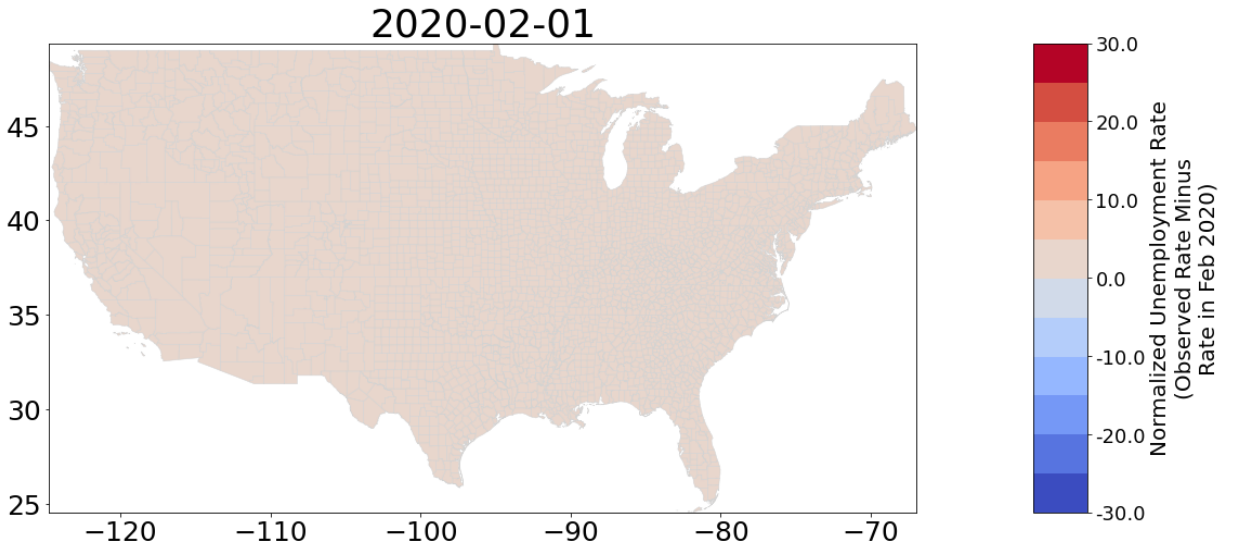

2019-09-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

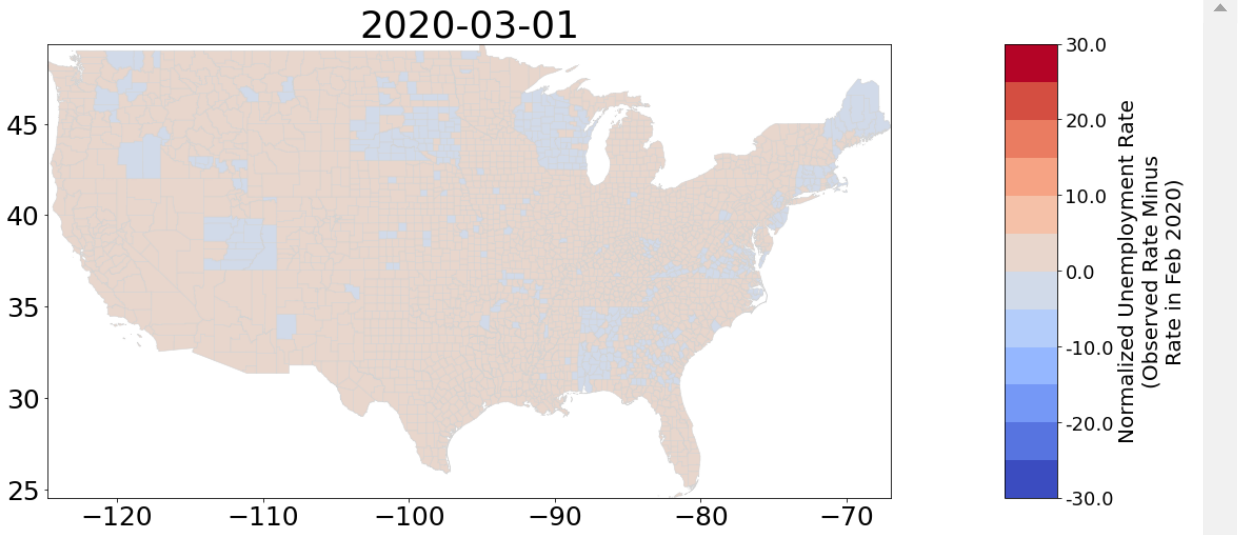

2019-10-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

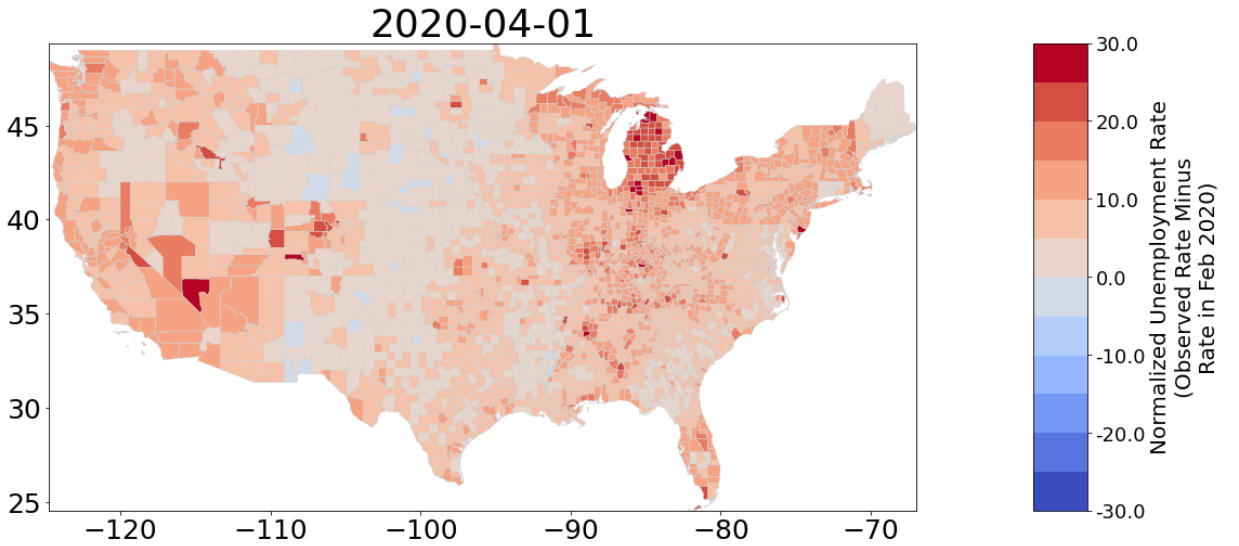

2019-11-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

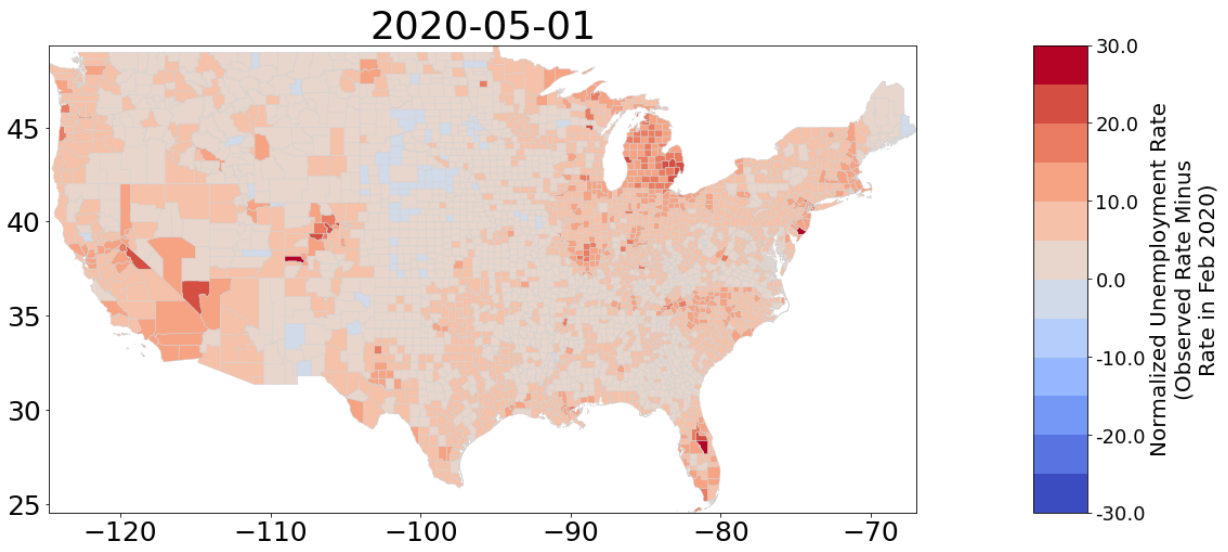

2019-12-01
45
30.0
20.0
Rate

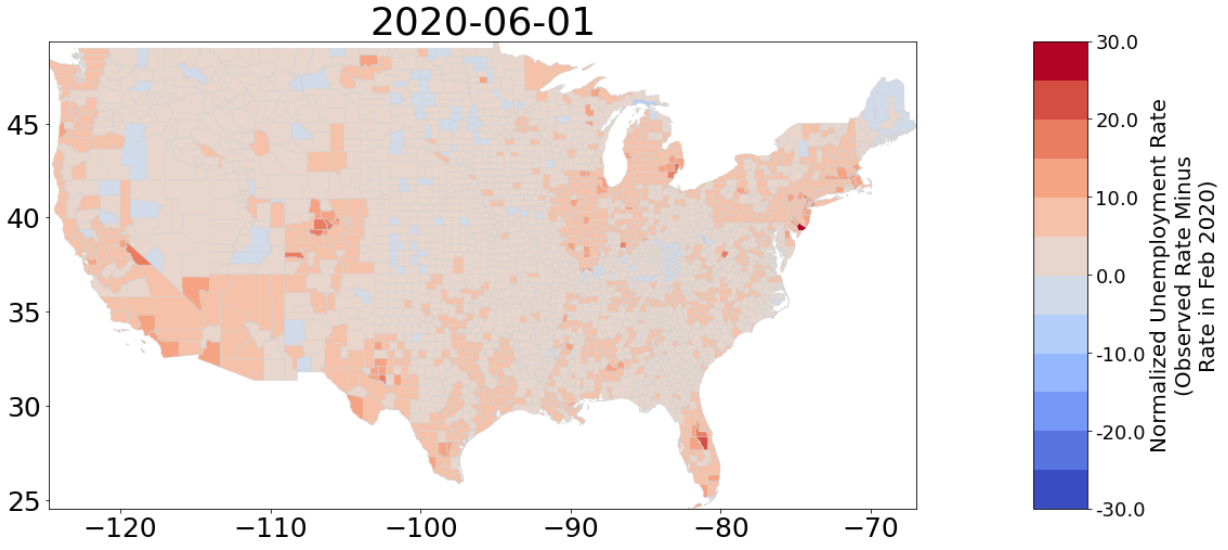
2020-01-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

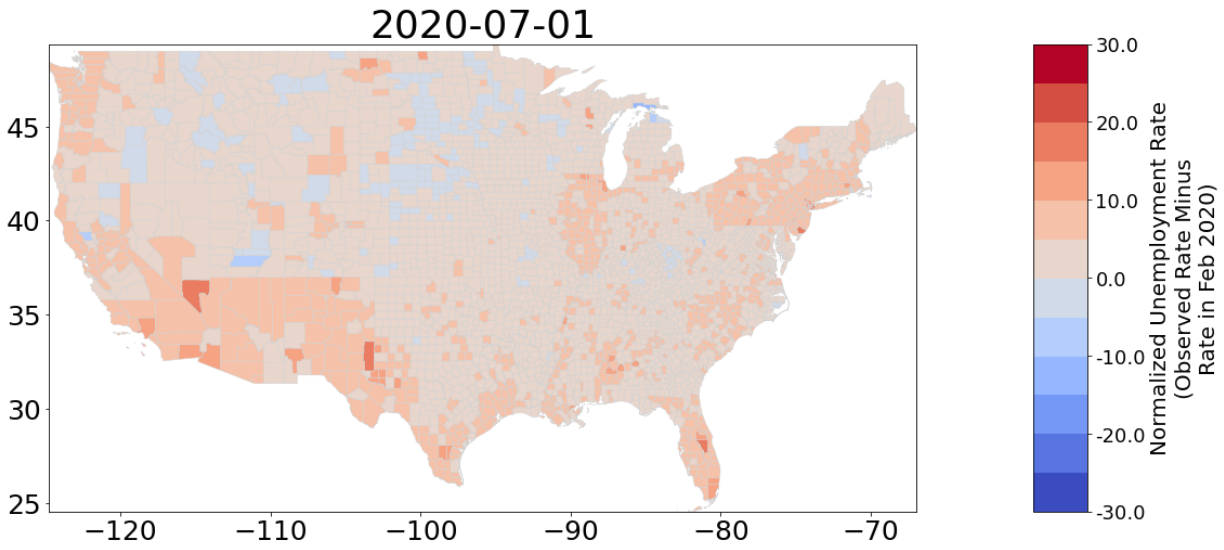

2020-02-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

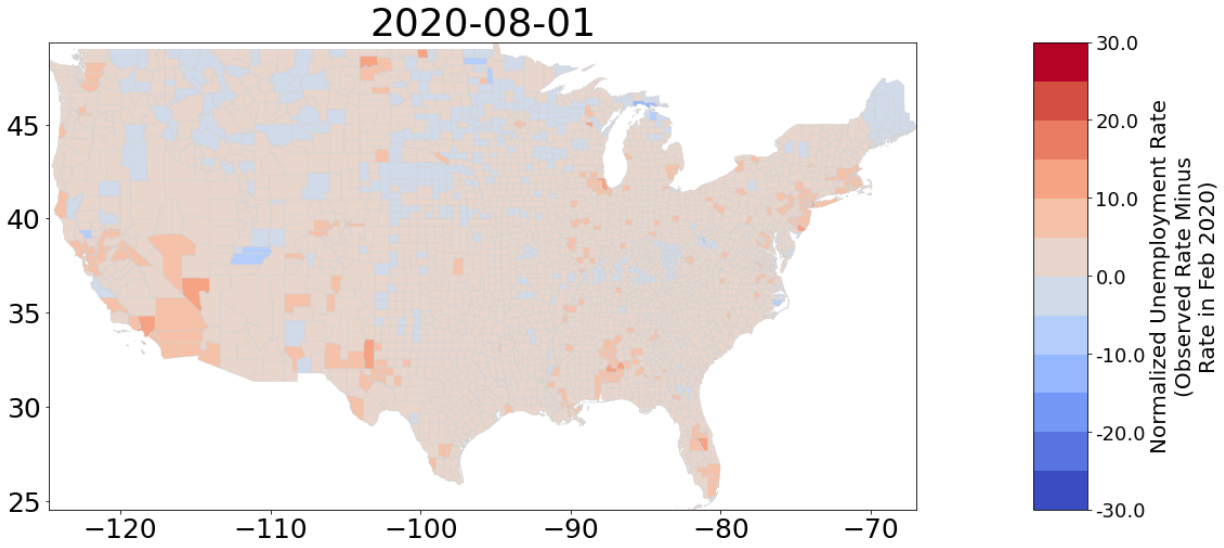

2020-03-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

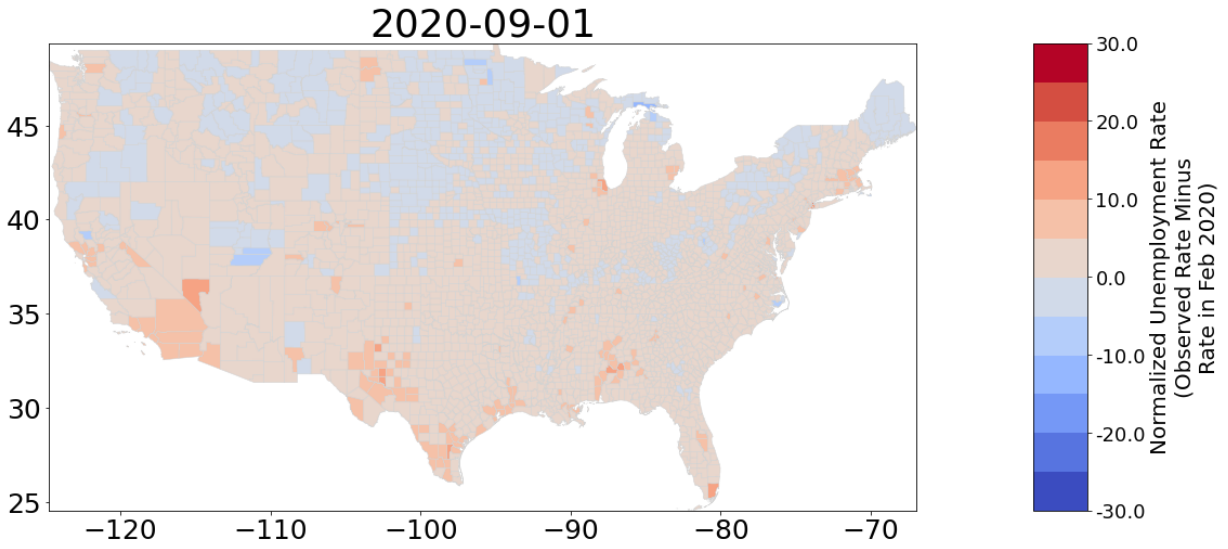

2020-04-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

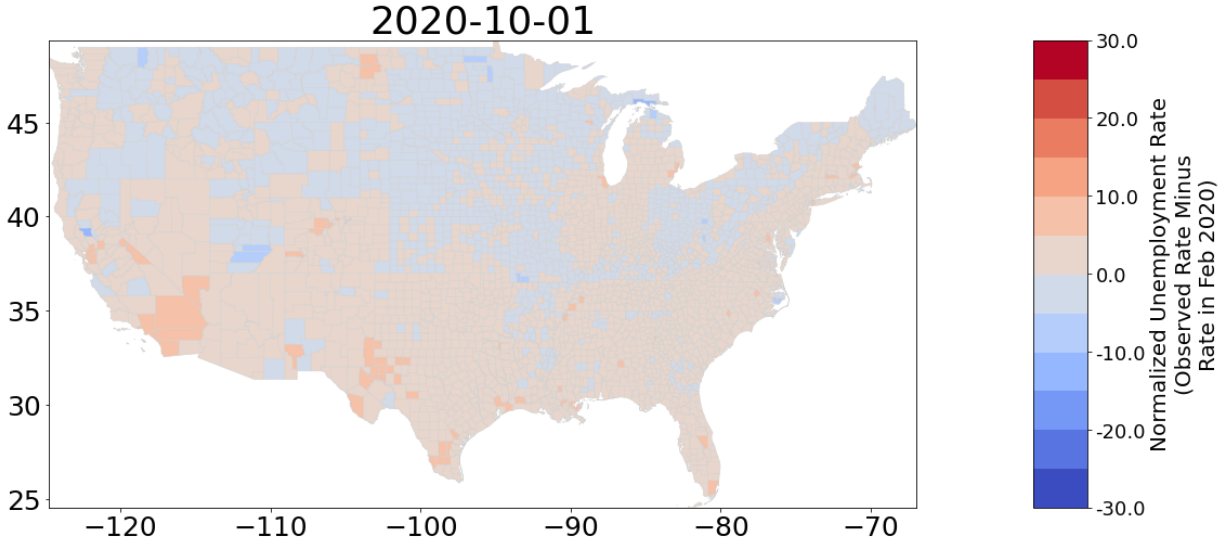

2020-05-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

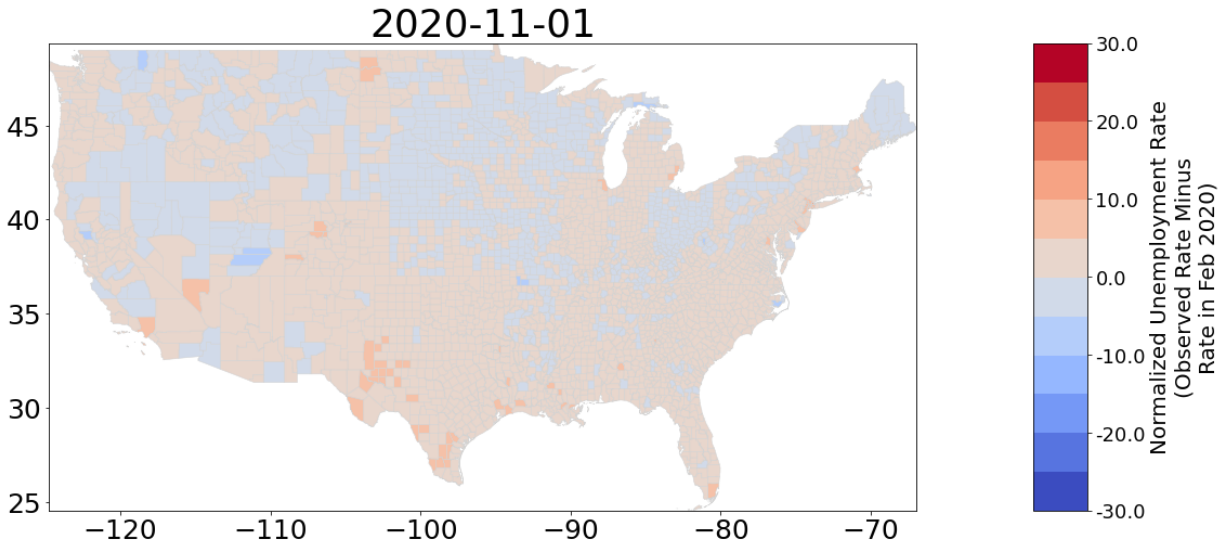

2020-06-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

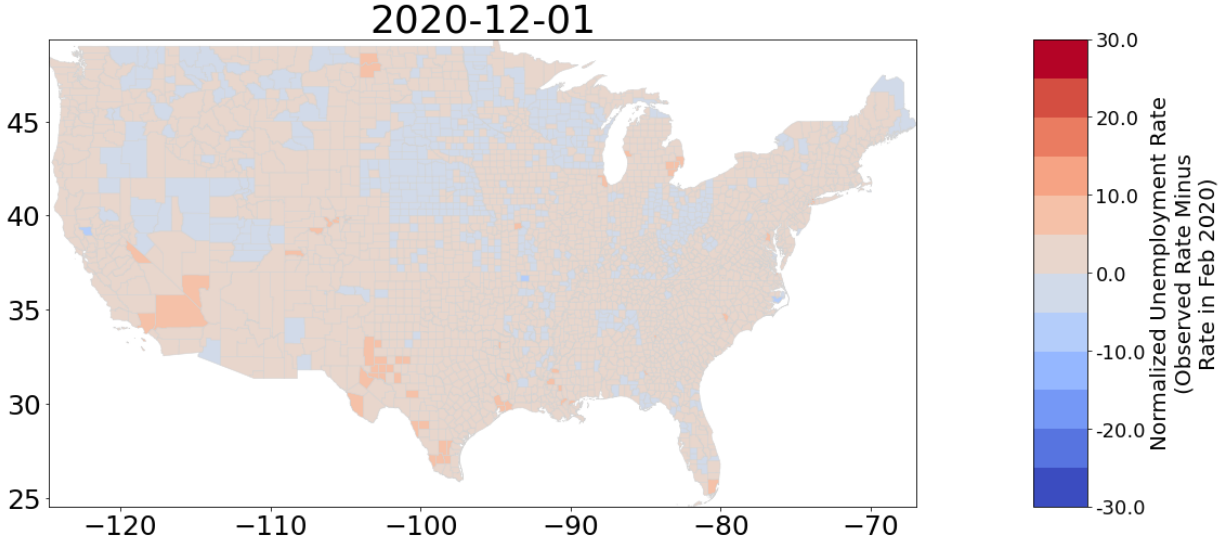

2020-07-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

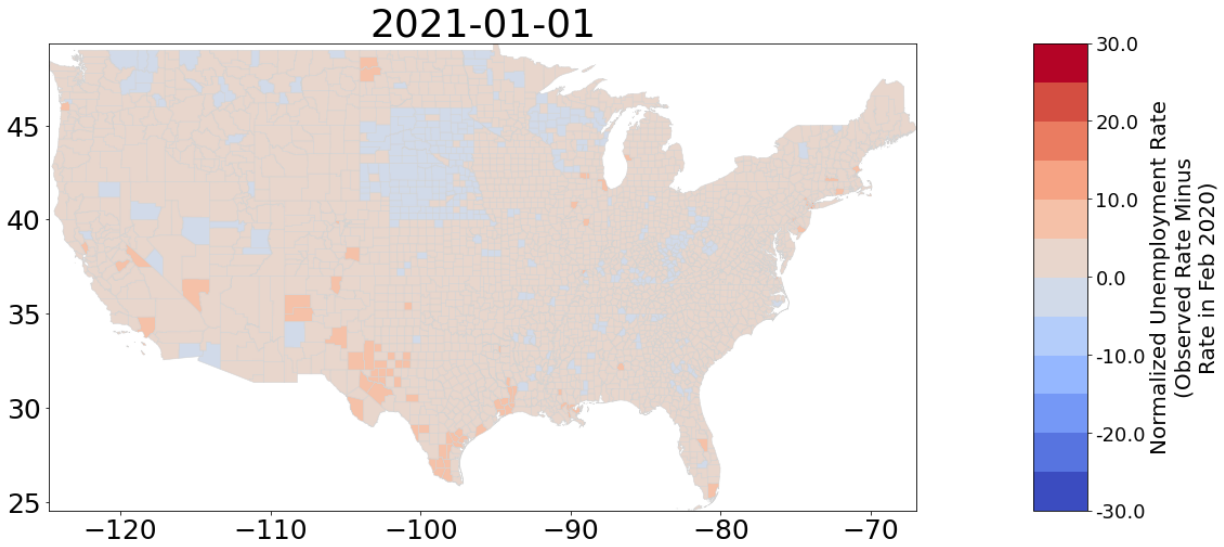

2020-08-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

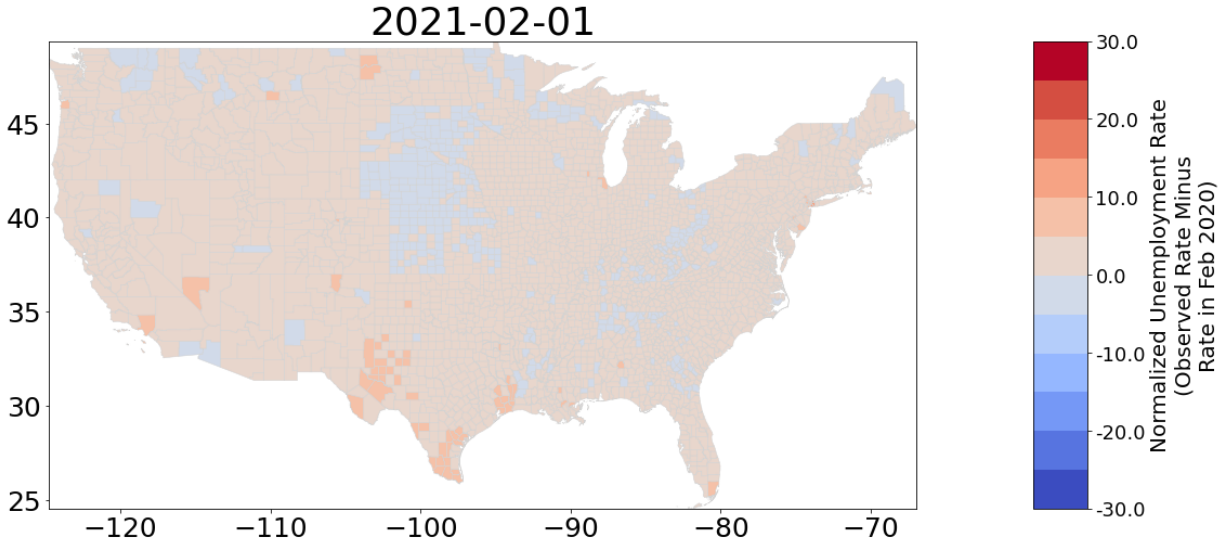

2020-09-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

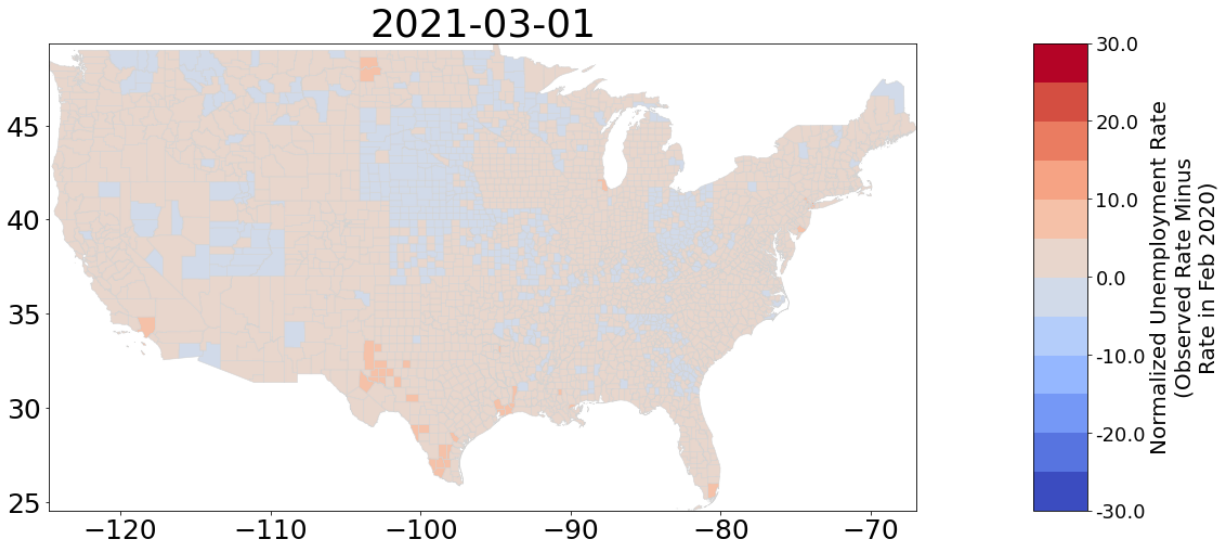

2020-10-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

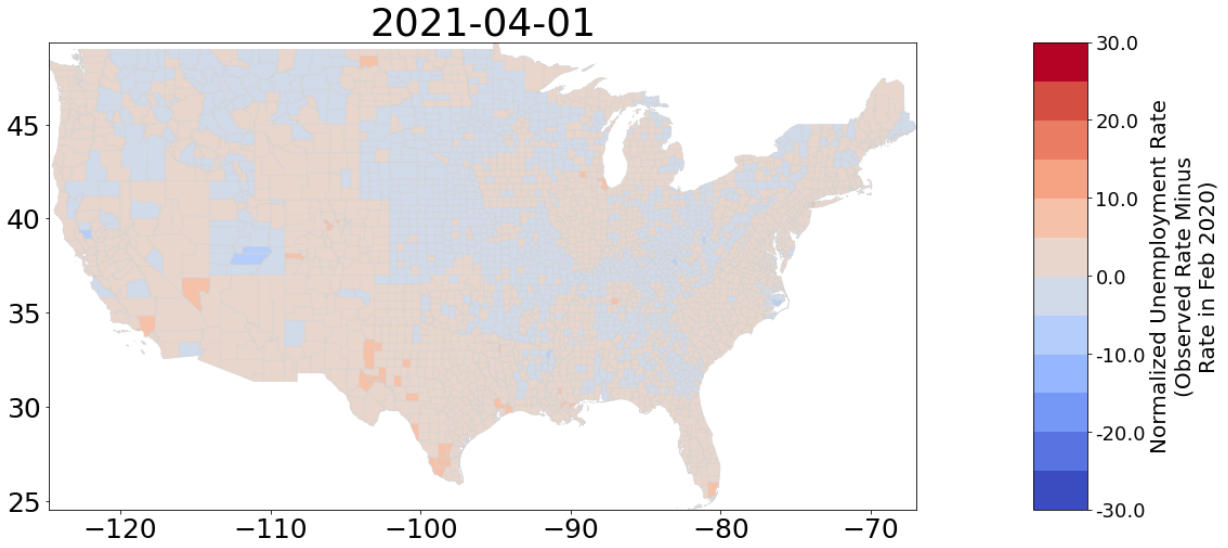

2020-11-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)

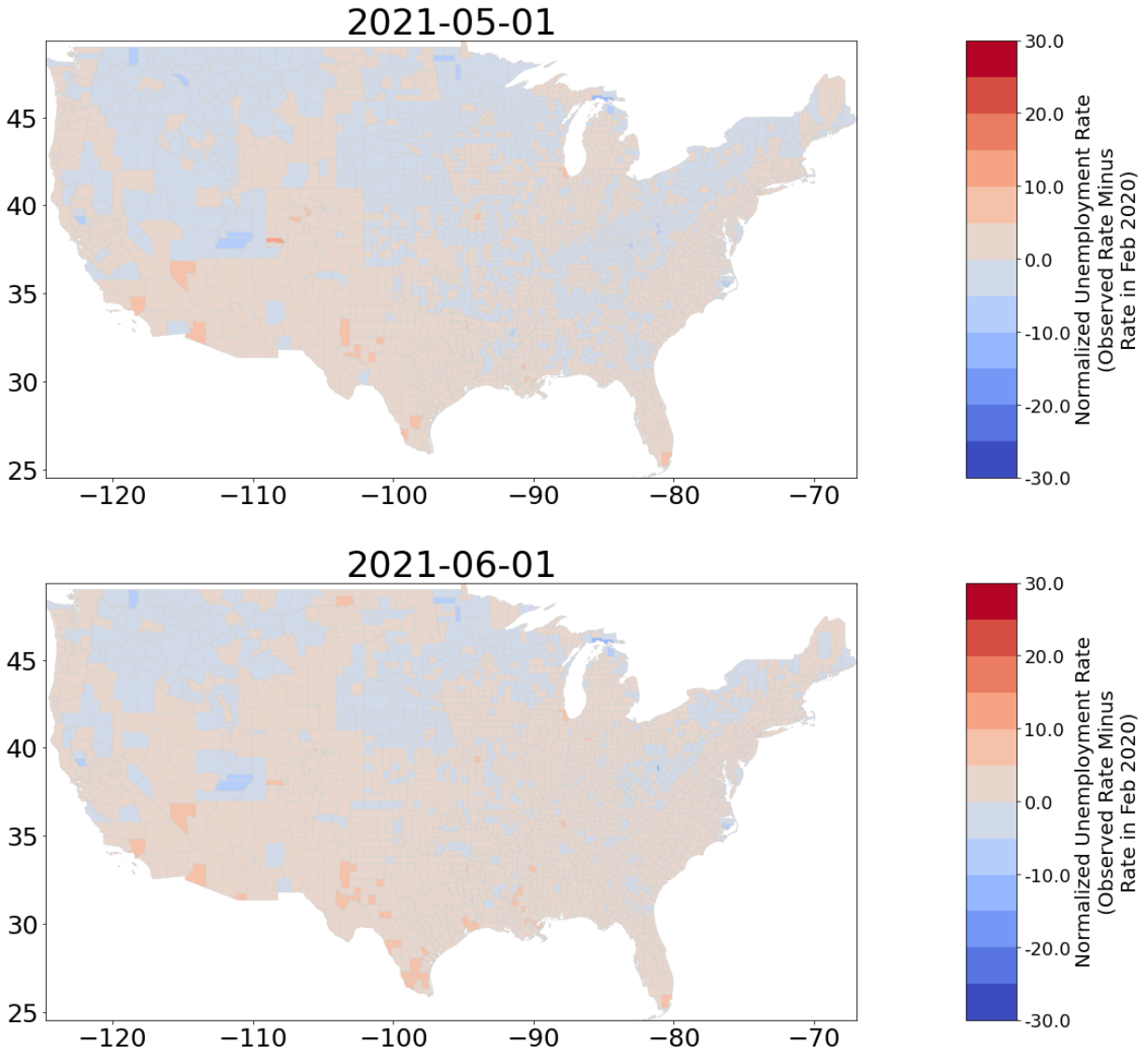

2020-12-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)


2021-01-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)


2021-02-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)


2021-03-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)


2021-04-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)


2021-05-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)
2021-06-01
45
40
35
30
25
−120
−110
−100
−90
−80
−70
30.0
20.0
10.0
0.0
-10.0
-20.0
-30.0
Normalized Unemployment Rate
(Observed Rate Minus
Rate in Feb 2020)