Introduction

When it comes down to visual data, an extremely powerful tool to analyze and understand the implications of data is a choropleth map. A choropleth map is the mapping of data over a mapped location using colors to represent the degrees of data being represented.

By studying the data of a choropleth map, we can make conclusions about the geographical location. We can make a wide array of conclusions from comparing the data in different locations to comparing data across time.

One of the most pressing topics of discussion and concern in the country currently is the topic of unemployment. It is of such importance that it is often used as a metric of the state of the economy and one of the qualities people often use to determine if a president's term was "good" or "bad".

[So far, so good. You should end this intro with a brief statement of specifically what you're going to do.]

Obtaining the data

In order to create a choropleth map, we need the data to map onto the choropleth. We can obtain sets of unemployment data from the Bureau of Labor Statistic's website. When downloaded, the files come named uniformly named so we can easily gather all of the files up.

In [305]:
```
from glob import glob
files = sorted(glob('laucnty*.xlsx'))
```

We have read in the list of files with the code above

Now we need the information inside the files themselves. When taking a look inside the files, we notice that the data in the xlsx files were ordered in a column-row fashion that also makes it simple to parse through to find what we need. We only need the data from the columns of State FIPS code, County FIPS code, and Rate.

Reading and storing the data

In [306]:
```
import xlrd #a library that helps read xlsx files

countyInfo = {} #creation of dictionary to hold data, one key for each year

for file in files:
    with open(file) as f:
        wb = xlrd.open_workbook(f.name)
        sheet = wb.sheet_by_index(0)
        year = "20" + f.name[7:-5]
        countyInfo[year] = {}
        # for each key year, holds a dictionary for each FIPS code with the va
lue being the unemployment rate
        for i in range(6,sheet.nrows):
            countyInfo[year]["FIPS_" + sheet.cell_value(i,1) + sheet.cell_valu
e(i,2)] = sheet.cell_value(i,9)
```

In this code, we created a dictionary with keys of the years of data, the values of which are dictionaries with the key being the FIPS code and the values of those keys are the unemployment rates.

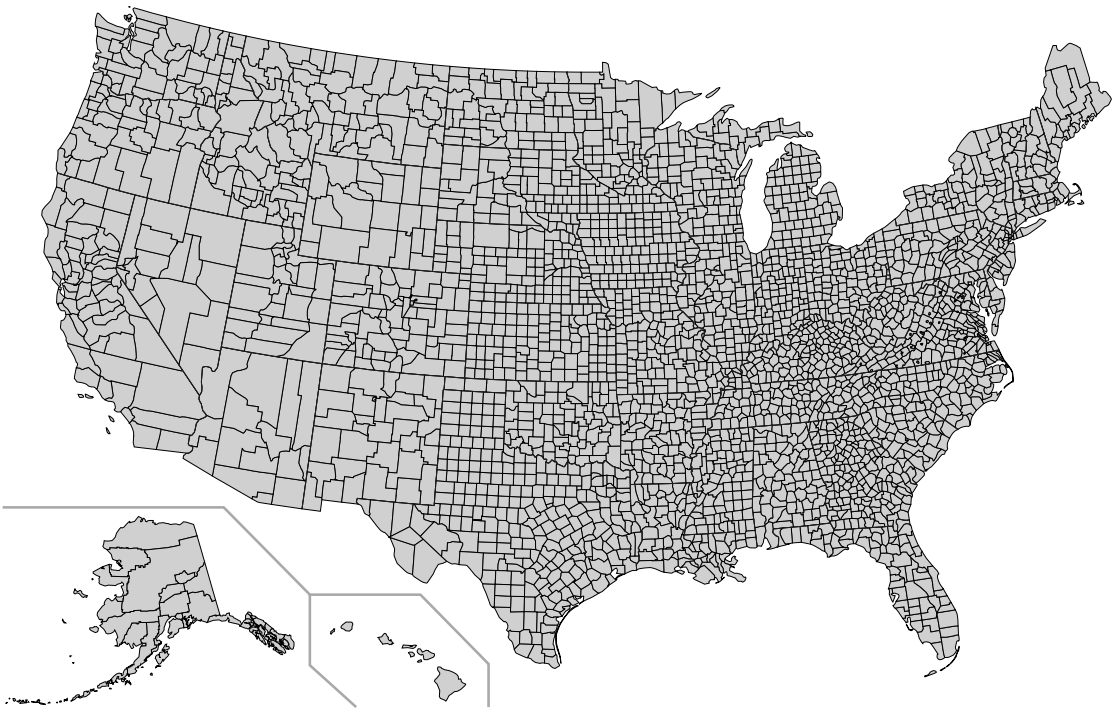
Displaying the map

Now that we have the data stored, we need to incorporate it into an SVG(Scalable Vector Graphic) file that displays a map of the United States by counties that looks like this

In [307]:
```
from lxml import etree

with open('USA_Counties_with_FIPS_and_names.svg') as f: #opens SVG file
    map = etree.fromstring(f.read().encode('utf-8'))
SVG(etree.tostring(map)) #renders and displays the svg graphic
```

Out[307]:



We are now able to visulize the map portion of the choropleth, we just need to develop some kind of scale in order to match the data with.

Please install spelling-checker plugin.

In [308]:
```
#Function by Dr. John Ringland

def add_scale(svg_map_in, colors, labels, font_size=6, stroke=False, save_file
=None):

    def set_props(elt, d):

        for k, v in d.items():
            elt.set(k,v)

    # coordinates of the upper left corner of the scale
    scale_x = 515
    scale_y = 180
    # scale dimensions
    scale_height = 150
    scale_width = 10
    # x-coordinates of scale labels
    label_x = 528

    # convert labels to strings if needed
    minus_sign = '−'
    labels = [str(x).replace('-',minus_sign) for x in labels]

    Nc = len(colors)
    hc = scale_height/Nc
    Nl = len(labels)
    hl = (scale_height)/(Nl-1)

    # attributes of scale rectangles
    rect_props = {"x": f"{scale_x}",
                  "width": f"{scale_width}",
                  "height": f"{hc}",
                  "stroke": "black",
                  "stroke-width": "0.25"}

    # attributes of scale labels
    label_props = {"x": f"{label_x}",
                   "text-anchor": "start",
                   "dominant-baseline": "central",
                   "font-size": f"{font_size}",
                   "font-family": "sans-serif"}

    # make a copy of the tree so that the input object will remain unaltered
    strmap = str(etree.tostring(svg_map_in).decode("utf8"))
    svg_map = etree.fromstring(strmap.encode("utf8")) # using string form to o
bviate deepcopy

    # create group element for the scale
    scale = etree.SubElement(svg_map, "g")
    scale.set("id", "color_scale")

    for i in range(Nc):
        rect_y = f"{scale_y + scale_height -(i+1)*hc}"
        rect_props["id"] = f"scale_rect_{i}"
        rect_props["y"] = rect_y
        rect_props["fill"] = colors[i]
```

```
        if not stroke:
            rect_props["stroke"] = "None"
            rect_props["shape-rendering"]="crispEdges"
        rect = etree.SubElement(scale, "rect")
        set_props(rect, rect_props)

    for i in range(Nl):
        label_y = f"{scale_y + scale_height - i*hl}"
        label =  etree.SubElement(scale, "text")
        label_props["id"] = f"scale_label_{i}"
        label_props["y"] = label_y
        set_props(label, label_props)
        label.text = labels[i]

    if save_file is not None:
        new_map = etree.tostring(svg_map).decode("utf8")
        with open(save_file, "w") as foo:
            foo.write(new_map)

    return svg_map
```

With this function we will be able to create a map with a scale placed onto it. It takes an input of our existing map and adds on a colormap scale to contextualize what the colors of the map means.

In [309]:
```
from IPython.display import SVG
import matplotlib.cm as cm

def makemap(c0,c1,xlo,xhi): #c0,c1 are tuples made of three numbers ranging fr
om [0-255] which represent RGV values
    def themap(x):
        ac0 = np.array(c0) #converts tuples to arrays to make vector operation
s possible; therefore manipulating colors
        ac1 = np.array(c1)
        h = (x-xlo)/(xhi-xlo)
        c = np.array( (1-h)*ac0 + h*ac1, dtype=int)
        return '#{:02x}{:02x}{:02x}'.format(*list(c))
    return themap
```

This function is used to create the range of colors.

In [310]:
```
def midpoints(xlo,xhi,n):
    dx = (xhi-xlo)/n #Return an array of the midpoints of n equal subintervals
of the interval [xlo,xhi]
    return np.linspace(xlo+dx/2,xhi-dx/2,n)
```

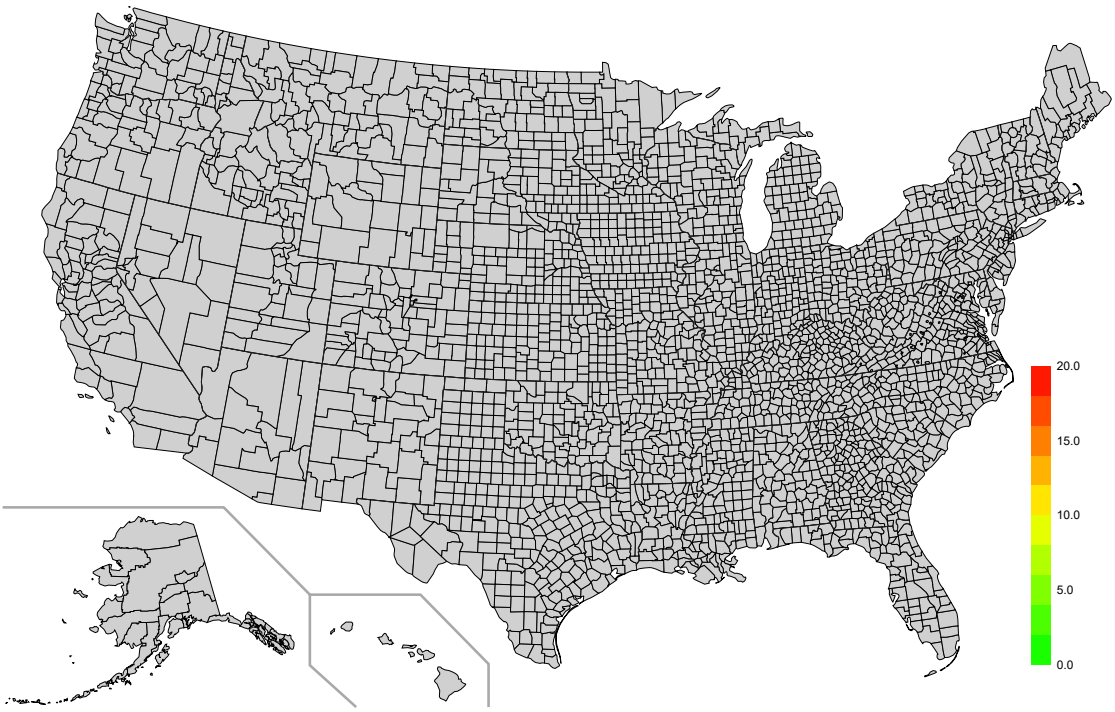
Color scale

We can now use these two functions to create our graph with a proper color scale by choosing the values for our function parameters.

In [311]:
```
mymap1 = makemap([0,255,0],[255,255,0],0, 10)
mymap2 = makemap([255,255,0],[255,0,0],10,20)
def mymap(x):
    if x<10: return mymap1(x)
    else:   return mymap2(x)
num_colors = 10
num_labels = 5
colors = [mymap(x) for x in midpoints(0,20,num_colors)]
labels = np.linspace(0,20,num_labels)

svg_map_with_scale = add_scale(map, colors, labels, font_size=6, stroke=False,
save_file="justscale.svg")
SVG(etree.tostring(svg_map_with_scale))
```

Out[311]:

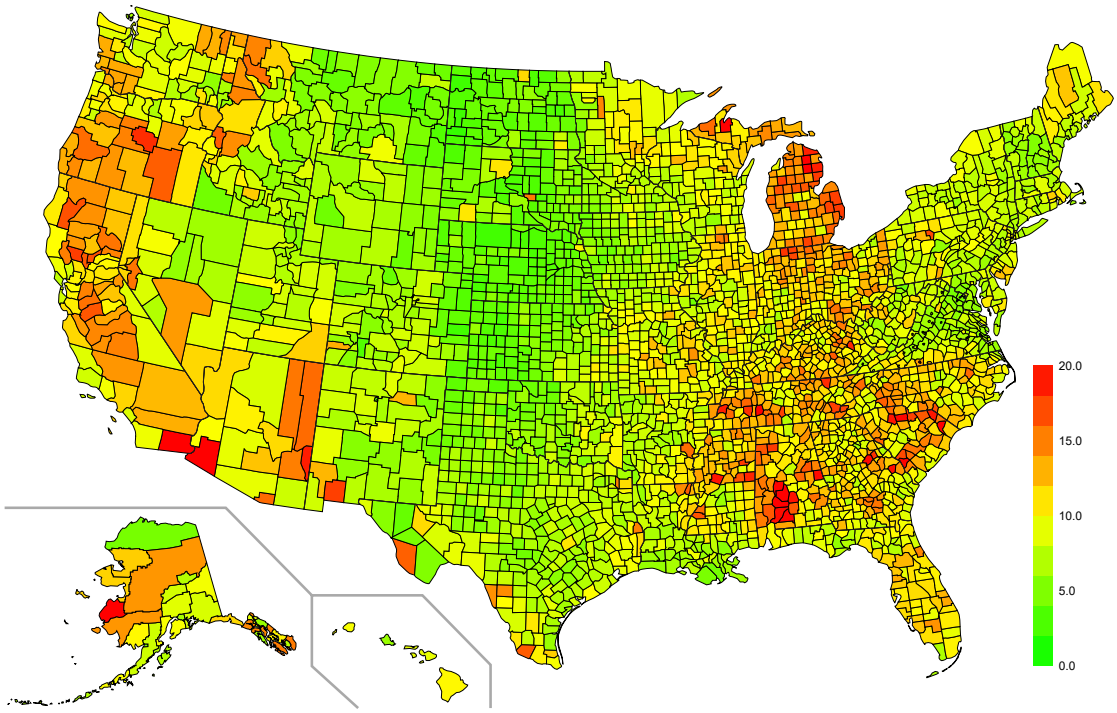


Filling in the map

We can start filling in the map by utilizing the data we stored earlier with the corresponding FIPS codes.

In [312]:
```
svg_map_with_scale = add_scale(map, colors, labels, font_size=6, stroke=False,
save_file="next.svg")
with open('next.svg') as f: #opens SVG file
    map2 = etree.fromstring(f.read().encode('utf-8'))
    default_val = 20
for item in map2[0]: #for each SVG entry for a county, assign it a color corre
sponding to our color scale
    if item.attrib["id"] in countyInfo["2009"]:
        item.attrib["fill"] = mymap(countyInfo["2009"][item.attrib["id"]])
        if countyInfo["2009"][item.attrib["id"]] >= 20: #caps unemployment rat
e at 20%
            item.attrib["fill"] = mymap(20)
    else:
        item.attrib["fill"] = "#000000" # in the case that there is a discrepa
ncy with the FIPS codes
SVG(etree.tostring(map2)) #renders and displays the svg graphic
```

Out[312]:



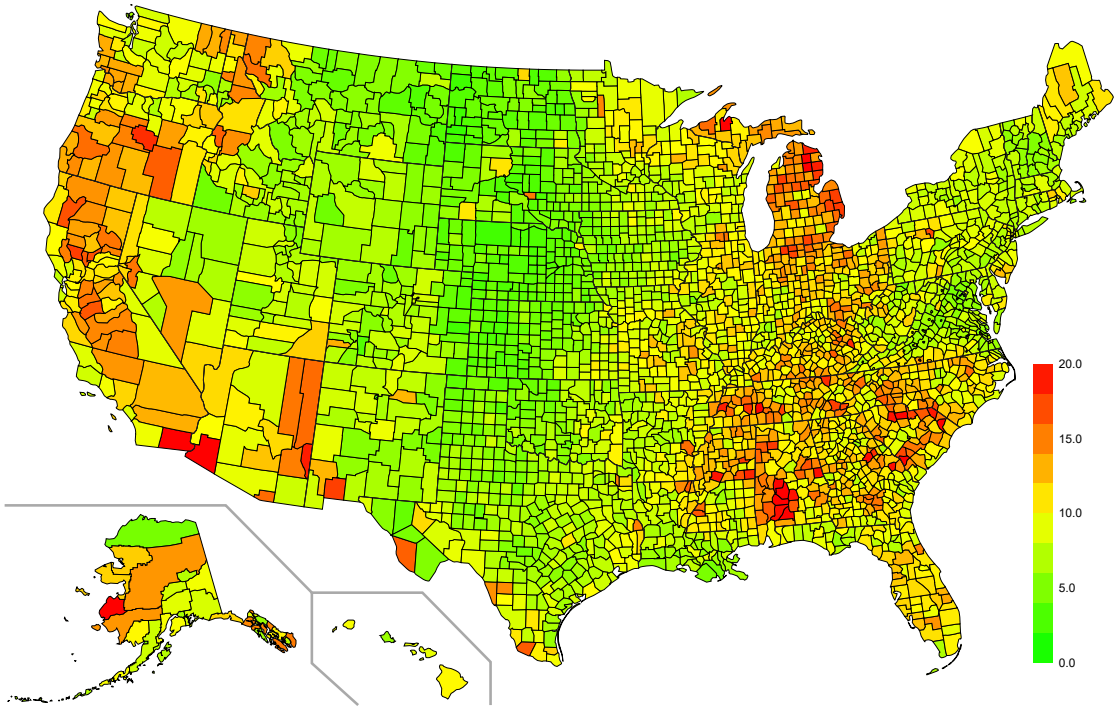
From the map, we can see that in the year 2009, there were many pockets of the country whose unemployment rates were disateriously high. We can see that in some counties during that time, almost for every 5 people, one person was unemployed. This seems to be due to the fact that in 2009 the country was going through an economic crisis.

Choropleth through time

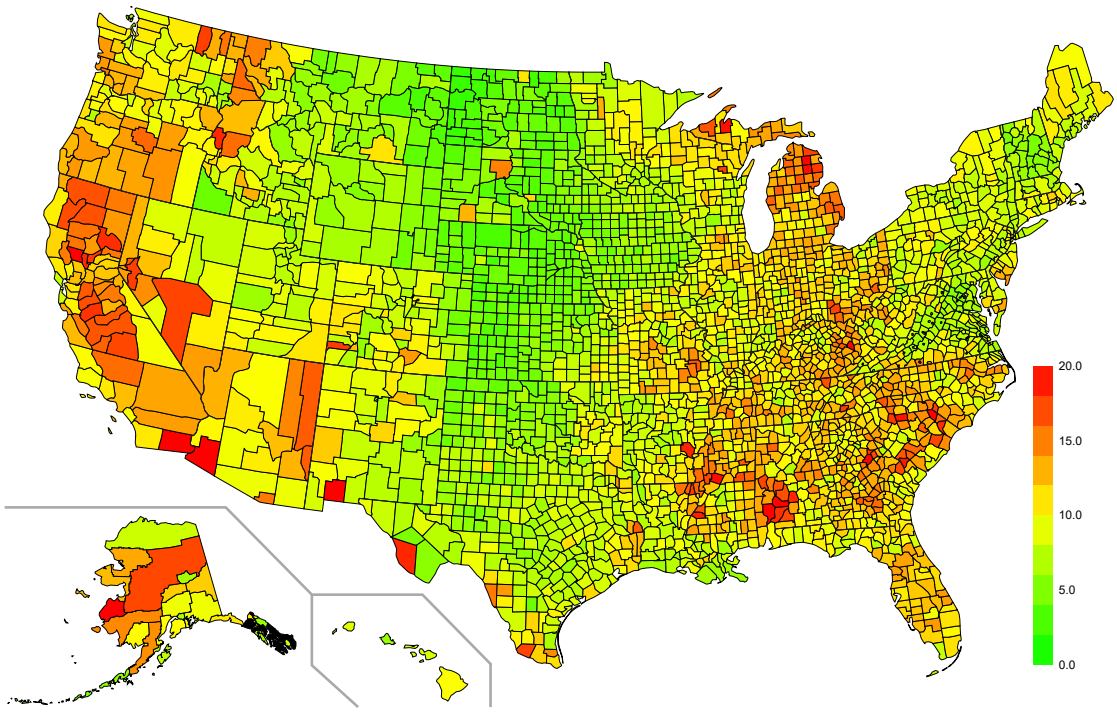
Since we have data and the means to create a choropleth for many years, we can put them together as a means to see how things have either change or stayed the same in regards to unemployment.

In [315]:
```
for i in range(9,17):
    year = "20" + str(i).zfill(2)
    print("Unemployment Percentage in " + year)
    svg_map_with_scale = add_scale(map, colors, labels, font_size=6, stroke=False, save_file= None)
    with open('next.svg') as f: #opens SVG file
        map2 = etree.fromstring(f.read().encode('utf-8'))
        default_val = 20
    for item in map2[0]: #for each SVG entry for a county, assign it a color corresponding to our color scale
        if item.attrib["id"] in countyInfo[year]:
            item.attrib["fill"] = mymap(countyInfo[year][item.attrib["id"]])
            if countyInfo[year][item.attrib["id"]] >= 20: #caps unemployment rate at 20%
                item.attrib["fill"] = mymap(20)
        else:
            item.attrib["fill"] = "#000000" # in the case that there is a discrepancy with the FIPS codes
    display(SVG(etree.tostring(map2)))
```

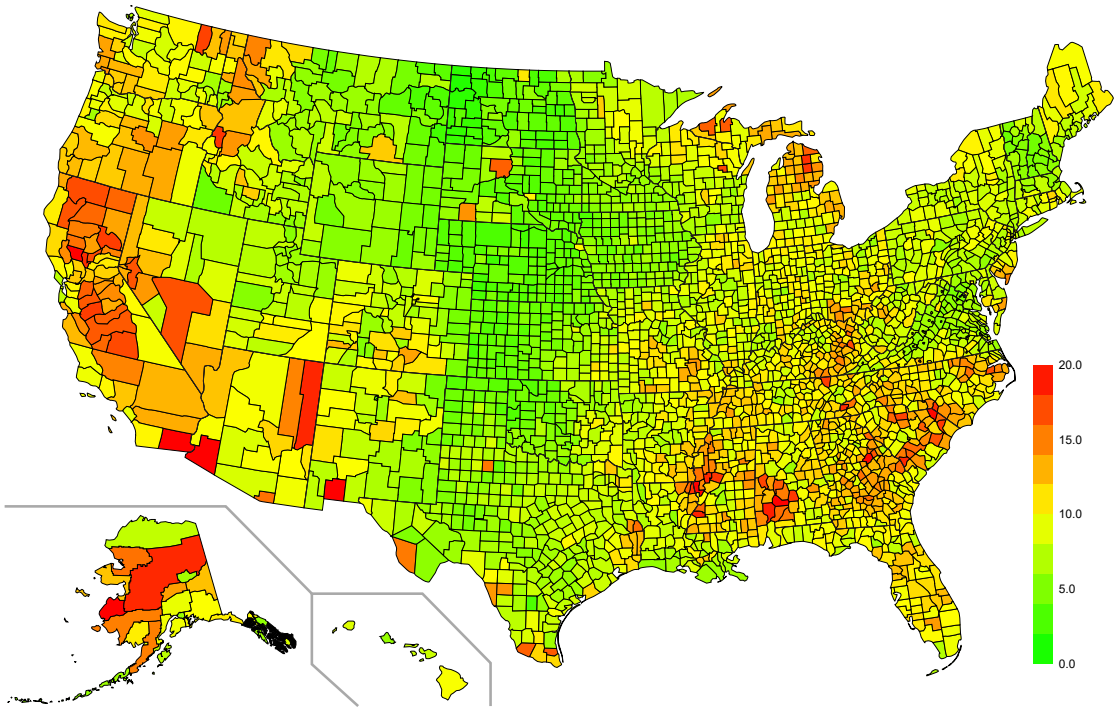
Unemployment Percentage in 2009



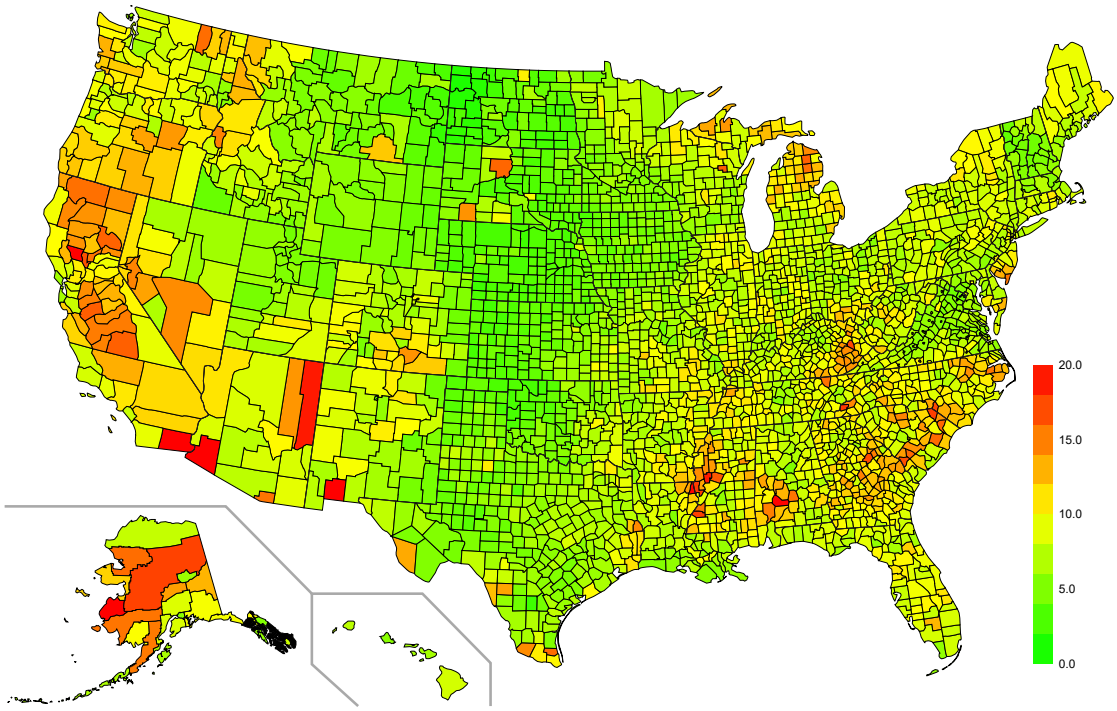
Unemployment Percentage in 2010



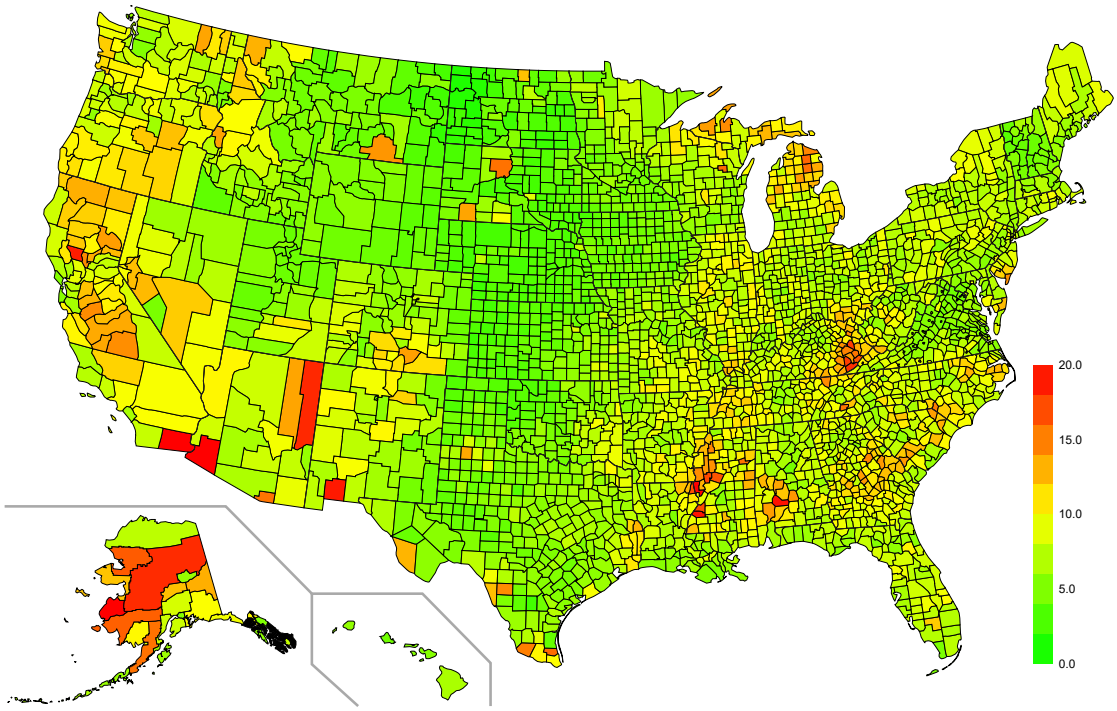
Unemployment Percentage in 2011



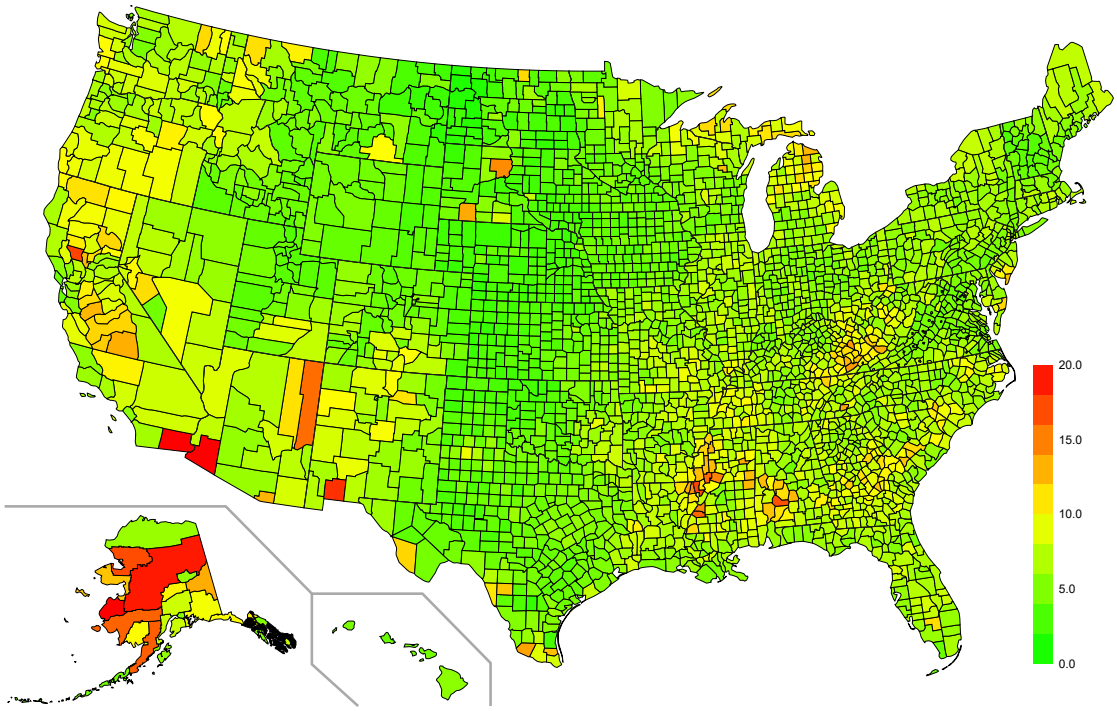
Unemployment Percentage in 2012



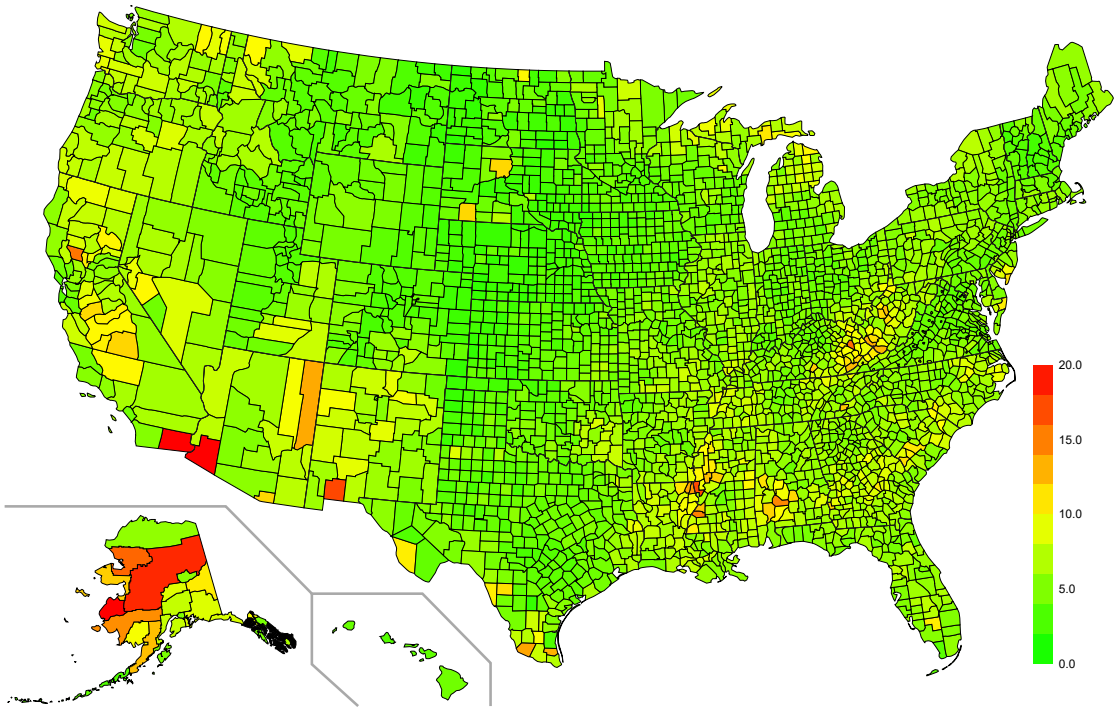
Unemployment Percentage in 2013



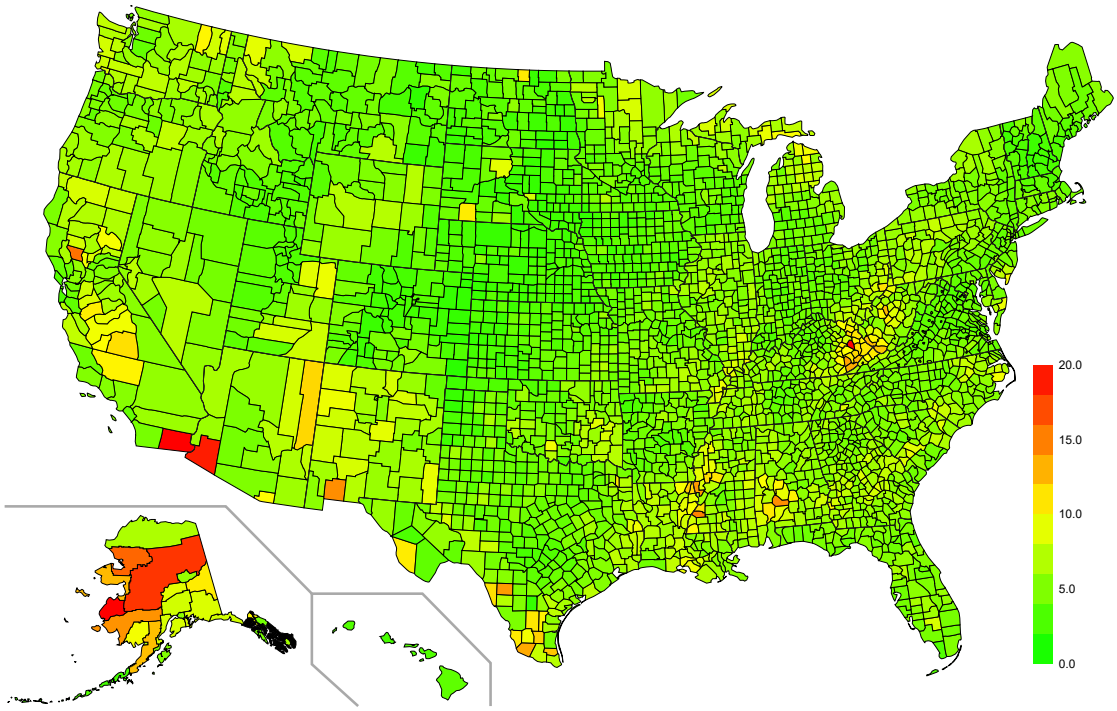
Unemployment Percentage in 2014



Unemployment Percentage in 2015



Unemployment Percentage in 2016



From the transition of choropleths through the 8 years of former president Obama's terms in office, we are able to see that the unemployment rate of the country dramatically lowered. [Yes, it is quite striking!] This is probably due to two reason. One was that the financial crisis that was ravaging the country at the time was starting to die down and the country was recovering. The other reason being former president Obama's policies that he inacted that provided more jobs for the people but also helped booster the economy after the end of the crisis.

Conclusion

The reason for the selection of those specific years is due to the fact that one of the most significant economic crisises of the country's recent history occured in 2008 and 2009. By observing the unemployment rate around those years, greatly puts into perspective just how much of a disaster the financial crisis was and how it affect the lives of so many Americans. With the red counties signifying greater or equal to 20% unemployment rate, it almost depicts the country on fire during this crisis.

Another interesting observation we were able to make was the unemployment rate of the country at the beginning of the first term of former president Barack Obama. Because economics and the financial well being of the country is a pressing debate and a concern for many citizens, it is a heavily discussed and observed metric during a president's term. Being able to witness the change in the country's unemployment rate and on a larger scale, the shape of the economy throughout his 8 years of being president allows us at least a little insight to help us conclude on into whether Obama was a "good" or "bad" president.

Sources

https://www.bls.gov/lau/#tables (https://www.bls.gov/lau/#tables)

http://blue.math.buffalo.edu/448/f19_day07_class.html (http://blue.math.buffalo.edu/448/f19_day07_class.html)

http://blue.math.buffalo.edu/448/day07_f19.html (http://blue.math.buffalo.edu/448/day07_f19.html)

http://blue.math.buffalo.edu/448/add_color_scale.html (http://blue.math.buffalo.edu/448/add_color_scale.html)

https://www.thebalance.com/what-has-obama-done-11-major-accomplishments-3306158 (https://www.thebalance.com/what-has-obama-done-11-major-accomplishments-3306158)