Creating Choropleth Maps of US Unemployment Rate

Sun Hyoung Kim

Introduction:

This report focuses on making Chropleth maps of US unemployment rate.

Below is a Table of Contents, showing a list of things I'm interested in analyzing and discussing:

Table of Contents

1. Choropleth maps of 2013 & 2003 unemployment rate

(*more sources: counties2013.py , counties2003.py , lscm.py)

This section focuses on making choropleth maps of unemployment rate (2013 and 2003) of the United States. The maps will show the unemployment rate of each county in the US. And then, we will see how unemployment rates are like, and how they are similar/different to each other.

B) Materials for all sections:

Materials (data): “Labor force data by county” from U.S. Bureau of Labor Statistics.

Data-type: CSV files that include county ID, names, and unemployment rate.

B) Methods for #1:

(a) creating a function of colorbar, hexcolor, and lscm:

By making a definition of colorbar, I'm able to create a colorbar that shows the range of unemployment rate based on the color of the rich.

Next, making a lscm file as a separte file enables me to import it whenever I want to use. And the function of lscm is that it generates a color range (like white to green) based on the range of number.

Also, by creating a function of hexcolor, the lscm enables me to get hex color code (like #FF0000), which I'm going to use later when modifying USA county svg file.

Code for #1 (a) :

```
from numpy import *

def hexcolor( (r,g,b), pallor=0. ):
	s ='#'
	for x in [r,g,b]:
		r = int( 255*pallor + 255*(1-pallor)*x )
		s += hex(r)[2:].zfill(2)
	return s

def lscm( colorlist ):
	n = len(colorlist)
	cl = [array(colorlist[i],dtype=float) for i in range(n)]
	n = float(n)
	def f( x, xmin=0., xmax=1. ):
		xrange = xmax - xmin
		sx = (x-xmin)/xrange
		if    sx<0.: sx=0.
		elif  sx>1.: sx=1.
		segment = int((n-1)*sx)
		h = (n-1)*sx - segment
		if segment == n-1:
			color = cl[-1]
		else:
			color = cl[segment]*(1.-h) + cl[segment+1]*h
		return hexcolor(color)
	return f

def colorbar(xmin,xmax,cm,npatches=200,svgfilename='mycolorbar.svg'):
	x = linspace(xmin,xmax,npatches,endpoint=False)
	dx = x[1]-x[0]
	xc = x + 0.5*dx
	epsilon = 0.0 # a tiny overlap of patches to avoid any gaps
	dx *= (1.+epsilon)
	figure(figsize=(1.5,10))
	hdx = dx/2.
	for xx in xc:
		color = cm(xx,xmin,xmax)
		gca().add_patch(
			Rectangle((0, xx - hdx),1,dx, facecolor=color,ec=color))
		plot([0,0,1,1,0],[xmin,xmax,xmax,xmin,xmin],'k')
	xticks([])
	ylim(xmin,xmax)
	subplots_adjust(left=0.3,right=0.95,bottom=0.05,top=0.95)
	savefig('mycolorbar.svg')
```

After making the definitions, I write codes to generate lscm, and then I can make a color bar. Also, I import numpy, pylab, lscm, and Rectangle at the beginning, so those definitions can work.
I set maxrate = 15, which means I will divide all numbers (unemployment rate) by 15 so that the value can fit into the range between 0 and 1. (lscm range)

Code:

```
from numpy import*
from pylab import*
from lscm import lscm
from matplotlib.patches import Rectangle

maxrate=15
mycm = lscm( ([1,1,1],[0,0.5,0]) )
colorbar(0,maxrate,mycm)
```

(b) creating a dictionary that consists of county id and unemployment rate:

I open a CSV file of 2013 unemployment rate, and then extracting each county's id, and unemployment rate. To do this step efficiently, I separate each line by a comma, and then extracting only id and unemployment rate, and then putting them into a list. For this step, since the id elements were separated by a comma (e.g. id '10300' are separated as '10' {state}, '300' {county}), I merge them in a for loop and putting them into a list.

(*****note that I can make a dictonary of 2003 year data very easily by opening 'unemployment2003.csv' instead of 'unemployment2013.csv')

Code for #1 (b):

```
f = open('unemployment2013.csv')

d=[]
for line in f:
       initial_lines= line.strip().split(',')
       real_lines = [initial_lines[line] for line in (1,2,-1)]
       real_lines[0:2]=[''.join(real_lines[0:2])]
       d.append(real_lines)
```

(c) modifying USA county.svg file and creating my own unemployment rate svg file:

I open the USA county.svg file and then modify the file by using a for loop. If the county id matches to the element in my dictionary, I extract the unemployment rate from the dictionary, and then I get hex color code by using lscm. Lastly, I modify the hexcolor code in the USA county.svg file. After changing all hex color code with these steps, I create my own svg file with changed colors in the end.

Code for #1 (c):

```
from lxml import etree
svg = etree.fromstring( open('USA.svg').read())
```

```
urate = 0
for i,tem in enumerate(svg):
       if tem.tag == '{http://www.w3.org/2000/svg}path':

              county_id = tem.attrib['id']
              if county_id=='State_Lines':
                     break
              for element in d:
                     if element[0]==county_id:
                            urate = float(element[-1])

              newcolor= mycm(urate, 0.,maxrate)
              style = tem.attrib['style']

              beginning,remainder = style.split('fill:')
              beginning +=                        'fill:'
              color = remainder[:7]
              ending= remainder[7:]
              newstyle = beginning + newcolor + ending

              tem.attrib['style']=newstyle

f = open('map2013.svg','w')
print >>f, etree.tostring(svg)
f.close()
```

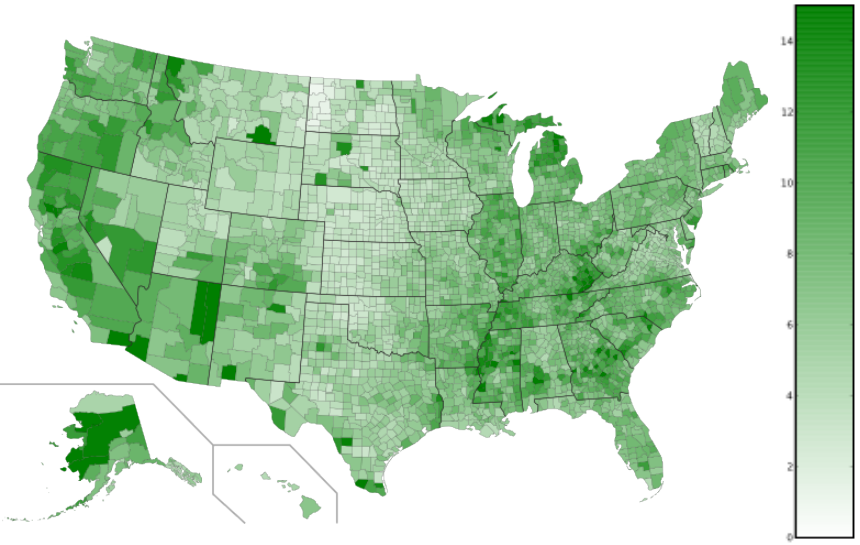
C) Results (output) & Discussion on #1:

Combining all the codes (#1 a, b, c), I could create a choropleth map of 2013 unemployment rate. By changing '2013' to '2003' in the codes above, I also made a choropleth map of 2003 unemployment rate.

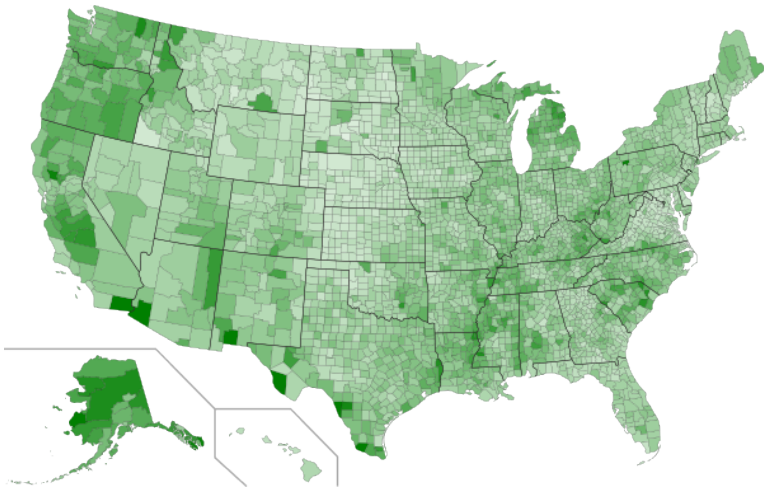
The thicker green, the higher unemployment rate.

Also, in this map, I set 15% (maxrate = 15) as the highest unemployment rate. So if it's either 15% or 20% of unemployment rate, the color will be the same, which is the thickest green.

2013



2003



By comparing those two graphs above, we could observe a few points:

1. central parts tend to have lower unemployment rate than east and west sides.
2. Both west and east sides' unemployment rates seem to have been increased overall.
3. Alaska always seems to have high unemployment rate.

2. Differences of unemployment rate between 2013 and 2003:

(*more sources: urate_difference.py , lscm.py (same as #1))

This section is an extension of #1; in order to see how unemployment rate has changed from 2003 to 2013, I created a choropleth map showing how much unemployment rate increased or decreased.

B) Methods for #2:

Methods are very similar to #1. The only big difference between #1 and #2 is that in #2 section, I make two colors (red, blue) instead of just green alone.

(blue – unemployment rate **increased,** red – unemployment rate **decreased**)

a) importing lscm, making red and blue color bars:

Below the code, I created two separate color bars (same code as the one in #1, but only definition names { e.g. colorbar_blue, colorbar_red } and 'savefig' are different {e.g. savefig('red_colorbar.svg'), savefig('blue_colorbar.svg') }. I just didn't show the code lines cause they are exactly the same as the one in #1 except for these differences.

Also, here I set up maxrate = 10 because the difference of unemployment rate is relatively lower than that of just normal unemployment rate like #1. So I don't have to divide each number by a large number like I did in #1.

Code for #2 (a):

```
from numpy import*
from pylab import*
from lscm import lscm
from matplotlib.patches import Rectangle

maxrate = 10
mycm_red = lscm( ([1,1,1],[1,0,0]) )
mycm_blue = lscm( ([1,1,1],[0,0,1]) )
colorbar_red(0,-maxrate,mycm_red)
colorbar_blue(0,maxrate,mycm_blue)
```

b) creating two dictionaries (2013, 2003):

The same method as #1 (b)

Code for #2 (b):

```
f = open('unemployment2013.csv')
d_13=[] #( county id, unemployment rate )
for line in f:
```

```
	initial_lines= line.strip().split(',')
	real_lines = [initial_lines[line] for line in (1,2,-1)]
	real_lines[0:2]=[''.join(real_lines[0:2])]
	d_13.append(real_lines)

f = open('unemployment2003.csv')
d_03=[]
for line in f:
	initial_lines= line.strip().split(',')
	real_lines = [initial_lines[line] for line in (1,2,-1)]
	real_lines[0:2]=[''.join(real_lines[0:2])] # combining 1,2nd element
	d_03.append(real_lines)
```

(c) modifying USA county.svg file and creating my own unemployment difference rate svg file:

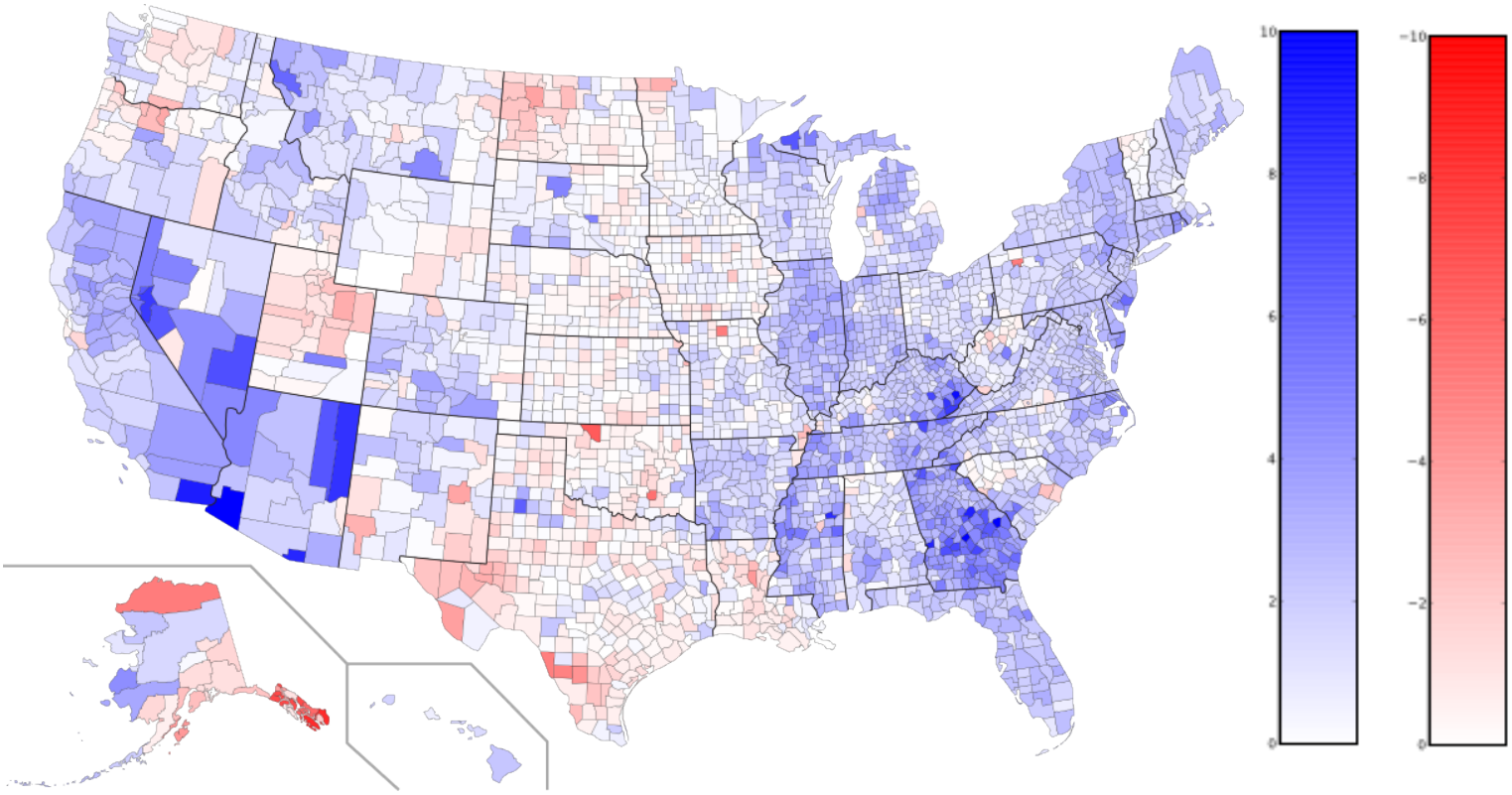
The only difference between this section's method and #1 (c)'s method is that here I am subtracting the unemployment rate of 2013 by that of 2003. And then, if it's positive (unemployment rate increased), I assign a blue color to it. The higher unemployment rate increased, the thicker blue color. In contrast, if the subtracted value is negative (unemployment rate decreased), I assign a red color. The higher unemployment rate decreased, the thicker red color.

Code for #2 (c):

```
from lxml import etree
svg = etree.fromstring( open('USA.svg').read())
urate_13 = 0
urate_03 = 0
diff=0

for i,tem in enumerate(svg):
	if tem.tag == '{http://www.w3.org/2000/svg}path':

		county_id = tem.attrib['id']
		if county_id=='State_Lines':
			break

		for element in d_13:
			if element[0]==county_id:
				urate_13 = float(element[-1])

		for element in d_03:
			if element[0]==county_id:
				urate_03 = float(element[-1])
		diff= urate_13 - urate_03

		if diff>=0:
			newcolor= mycm_blue(diff, 0.,maxrate)

		elif diff<0:
			newcolor= mycm_red(diff, 0.,-maxrate)

		style = tem.attrib['style']
		beginning,remainder = style.split('fill:')
		beginning +=                                'fill:'
		color = remainder[:7]
		ending= remainder[7:]
		newstyle = beginning + newcolor + ending

		tem.attrib['style']=newstyle

f = open('difference_map.svg','w')
print >>f, etree.tostring(svg)
```

```
f.close()
```

C) Results (output) & Discussion on #2:



Blue – unemployment rate increased (from 2003 to 2013)
Red – unemployment rate decreased
White – almost no change

It is noticable that both east and west sides tend to have unemployment rates increased.
Thankfully, I can see some many counties with decreased unemployment rate in central parts.

3. Average unemployment rate (2003 ~ 2013)

(*more sources: urate_average.py , lscm.py {same as #1, #2 })

This section focuses on making a map showing average unemployment rates of counties between 2003 and 2013.
I'm also printing out counties with unemployment rates higher than 15.

B) Methods for #3:

Again, this section's methods are also very similar to #1. For the color bar, I didn't have to make a separate one cause it's the same as the one from #1 (0 ~ 15, from white to green).

(a) making a dictionary consisting of all years (2003 ~ 2013):

I import glob here so I can read all files (2003~2013) easily.

Also, unlike #1 and #2, this time I am adding counties' names, and State names, so that I can use them when printing out them with high unemployment rates later.

Code for #3 (a):

```
import glob
d=[]
for f in glob.glob('unemployment*.csv'):
	lines = open(f).readlines()
	for line in lines:
		initial_lines= line.strip().split(',')
		real_lines = [initial_lines[line] for line in (1,2,3,4,-1)]
		real_lines[0:2]=[''.join(real_lines[0:2])]
		d.append(real_lines)
```

(b) modifying USA county.svg file and creating my own unemployment average rate svg file:

Although the methods were very similar to #1, here are some important differences:

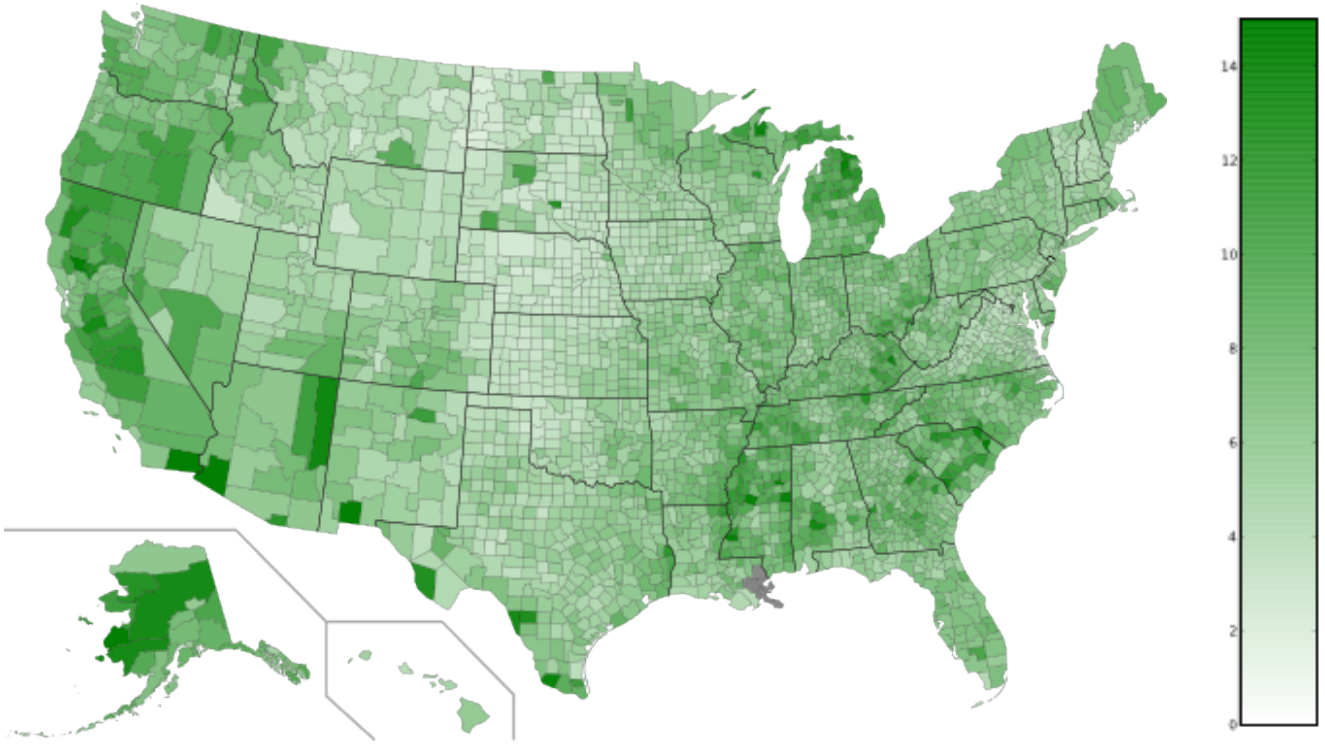
1. maxrate = 15*11 (2003~2011: 11 different values, so dividing by (11*15) results in a proper value fitting to the range between 0 and 1.
2. N.A. : A very few of counties do not have information for unemployment rate (N.A.) In this case, I assign grey color to those counties.
3. When running a for loop, I add up all the values of unemployment rate (2003~2013), and then dividing by maxrate
4. printing out unemployment rates greater than 15. (county name, state name, rate)

Code for #3 (b):

```
from lxml import etree
svg = etree.fromstring( open('USA.svg').read())
maxrate = 15*11
urate = 0
mycm = lscm( ([1,1,1],[0,0.5,0]) )
NAdetector=0
for i,tem in enumerate(svg):
	if tem.tag == '{http://www.w3.org/2000/svg}path':

		county_id = tem.attrib['id']
		if county_id=='State_Lines':
			break

		for element in d:
			if element[0]==county_id:
				if element[-1] != 'N.A.':
					urate+= float(element[-1])
					county_name = element[1]
					county_state = element[2]
				elif element[-1] =='N.A.':
					NAdetector+=1
		if urate>(15*11):
			print county_name,',',county_state,'/ unemployment rate:',
float(urate)/11
```

```
            newcolor= mycm(urate, 0., maxrate)
            style = tem.attrib['style']
            beginning,remainder = style.split('fill:')
            beginning +=                                'fill:'
            color = remainder[:7]
            ending= remainder[7:]
            if NAdetector>0:
                  newcolor='#888888'

            newstyle = beginning + newcolor + ending
            urate=0
            NAdetector=0
            tem.attrib['style']=newstyle

f = open('average_map.svg','w')
print >>f, etree.tostring(svg)
f.close()
```

C) Results (output) & Discussion on #3:




```
Wade Hampton Census Area , AK / unemployment rate: 20.3272727273
Yuma County , AZ / unemployment rate: 20.0909090909
Imperial County , CA / unemployment rate: 22.0181818182
Colusa County , CA /  unemployment rate: 16.0
Marion County , SC / unemployment rate: 15.4545454545
```

Again, west and east sides tend to have higher average unemployment rates than central parts, and Alaska has very high unemployment rates. Also, there were five counties with 15+ average unemployment rates. And it was surprising to see two counties with 20+ unemployment rates.

4. Conclusions:

1. East and West sides are likely to have higher unemployment rates than central parts overall.
2. Compared to the past, the unemployment rates of East and West sides have increased.
3. There were even a few counties with 15+ average unemployment rates.