A Complete Guide to an Interactive Geographical Map using Python

[[](https://towardsdatascience.com/@shivangipatel27?source=post_page-----f4c5197e23e0----------------------)](https://towardsdatascience.com/@shivangipatel27?source=post_page-----f4c5197e23e0----------------------)

[Shivangi Patel](https://towardsdatascience.com/@shivangipatel27?source=post_page-----f4c5197e23e0----------------------)

Follow

[Feb 5, 2019](https://towardsdatascience.com/a-complete-guide-to-an-interactive-geographical-map-using-python-f4c5197e23e0?source=post_page-----f4c5197e23e0----------------------) · 9 min read

Ever wondered how these beautiful geographical maps are created? [Our World in Data](https://ourworldindata.org/" \t "_blank)has an extensive collection of interactive data visualizations on aspects dedicated to the global changes in health, population growth, education, culture, violence, political power, technology and several things that we care about. These visualizations help us understand how and why the world has changed over the last few decades. I was intrigued with this wealth of information and motivated to dive deeper.

A quick google search led me to choropleth maps. Choropleth Maps display divided geographical areas or regions that are coloured, shaded or patterned in relation to a data variable. The geographical area may expanse the entire world, or a country, state or even a county. There are many tools and packages available to make a stand alone or static choropleth map using Python. However, creating a dynamic map is slightly tricky and that is exactly what we are going to learn in this blog. In this step by step guide, we will recreate an interactive global choropleth map on ***Share of Adults who are obese (1975–2016)***using Python libraries and package — Pandas, [Geopandas](http://geopandas.org/" \t "_blank) and [Bokeh](https://bokeh.pydata.org/en/latest/).

In the first section, we will create a static map and then later build on our code to introduce interactivity.

**Just give me the code**

The entire code can also be found at my [github.com](https://github.com/CrazyDaffodils/Interactive-Choropleth-Map-Using-Python).

**Downloads and Installations**

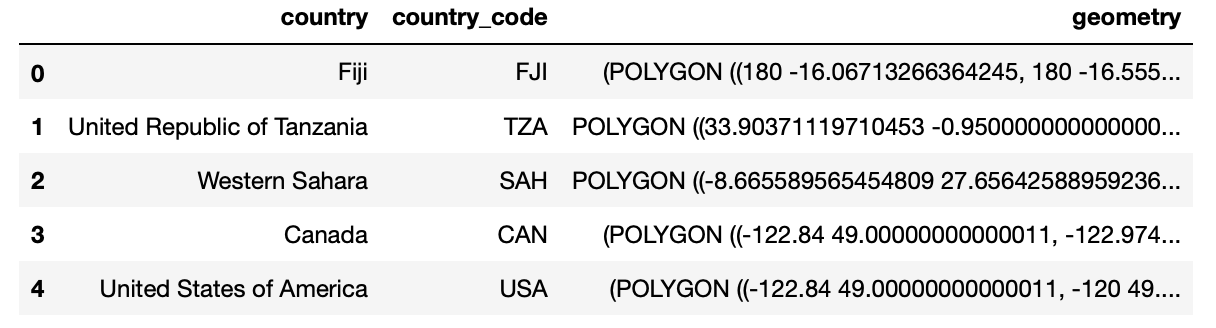
To render a world map, we need a shapefile with world coordinates. [Natural Earth](https://www.naturalearthdata.com/) is a public domain map dataset that provides geospatial data at various resolutions. For our purpose, [1–110m small scale data](https://www.naturalearthdata.com/downloads/110m-cultural-vectors/) is good enough. Click on the green button ***Download Countries v4.1.0***to download the shape folder. Next, go to [Our World in Data](https://ourworldindata.org/obesity) and download the share-of-adults-defined-as-obese.csv by clicking on Data tab on the plot. Alternatively, feel free to download the files from my [github](https://github.com/CrazyDaffodils/Interactive-Choropleth-Map-Using-Python" \t "_blank) repository. Install Geopandas and Bokeh.

**Explore pandas and Geopandas dataframes**

Import geopandas. Geopandas can read almost any vector-based spatial data format including ESRI shapefile using read\_file command which returns a GeoDataframe object. The shapefile consists of many columns, of which only 3 are of any interest to us. The columns are renamed for easy referencing.

import geopandas as gpdshapefile = 'data/countries\_110m/ne\_110m\_admin\_0\_countries.shp'*#Read shapefile using Geopandas*gdf = gpd.read\_file(shapefile)[['ADMIN', 'ADM0\_A3', 'geometry']]*#Rename columns.*  
gdf.columns = ['country', 'country\_code', 'geometry']  
gdf.head()





Geopandas GeoDataFrame

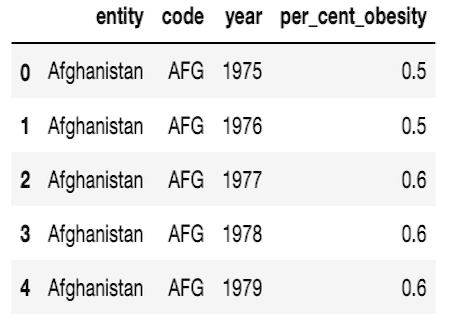
We can drop the row for ‘Antarctica’ as it unnecessarily occupies a large space in our map and is not required in our current analysis.

print(gdf[gdf['country'] == 'Antarctica'])#*Drop row corresponding to 'Antarctica'*gdf = gdf.drop(gdf.index[159])

Next, we import pandas and read in the .csv file.

import pandas as pddatafile = 'data/obesity.csv'*#Read csv file using pandas*  
df = pd.read\_csv(datafile, names = ['entity', 'code', 'year', 'per\_cent\_obesity'], skiprows = 1)df.head()

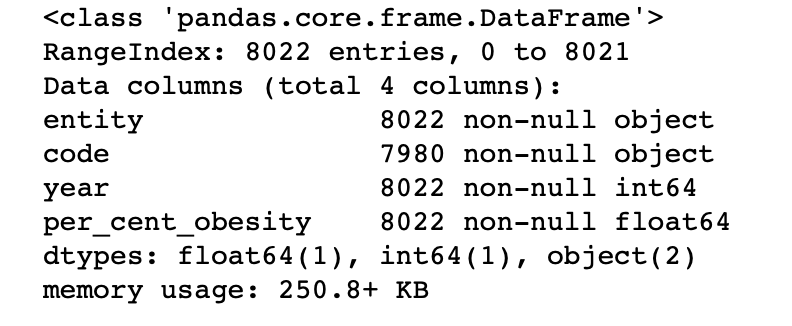




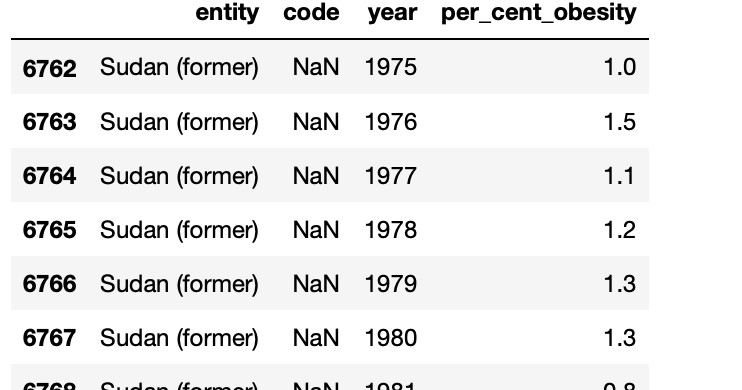
Pandas DataFrame

df.info()  
df[df['code'].isnull()









Investigating missing values in df

Investigating pandas dataframe shows missing values for Sudan. Our data spans the period from 1975–2016; while Sudan split into two countries in July 2011. This resulted in a change of 3-letter ISO code for the newly split country. To keep it simple, let us ignore the missing data for Sudan.

**Static choropleth map for year 2016**

Let us first create a static map representing Share of Obesity in Adults in the year 2016. This requires filtering data for year 2016 from df. The resulting dataframe df\_2016 can then be merged to the GeoDataframe gdf.

*#Filter data for year 2016.*  
df\_2016 = df[df['year'] == 2016]*#Merge dataframes gdf and df\_2016.*  
merged = gdf.merge(df\_2016, left\_on = 'country\_code', right\_on = 'code')

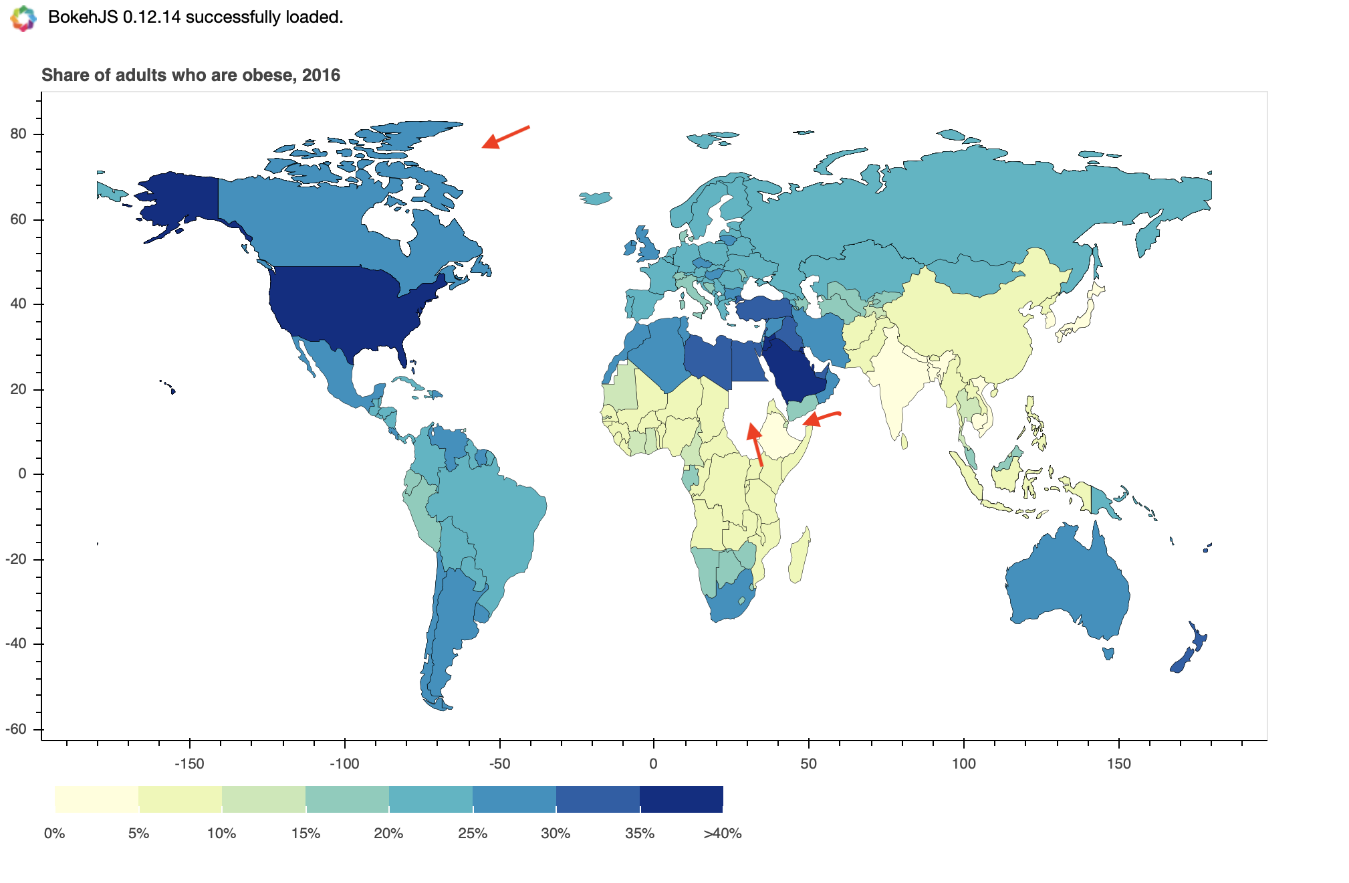
The merged file is a GeoDataframe object that can be rendered using geopandas module. However, since we want to incorporate data visualization interactivity, we will use Bokeh library. Bokeh consumes GeoJSON format which represents geographical features with JSON. GeoJSON describes points, lines and polygons (called Patches in Bokeh) as a collection of features. We therefore convert the merged file to GeoJSON format.

import json#*Read data to json.*  
merged\_json = json.loads(merged.to\_json())#*Convert to String like object.*  
json\_data = json.dumps(merged\_json)

We are now ready to render our choropleth map using Bokeh. Import the required modules. The code is described inline.

from bokeh.io import output\_notebook, show, output\_file  
from bokeh.plotting import figure  
from bokeh.models import GeoJSONDataSource, LinearColorMapper, ColorBar  
from bokeh.palettes import brewer*#Input GeoJSON source that contains features for plotting.*  
geosource = GeoJSONDataSource(geojson = json\_data)*#Define a sequential multi-hue color palette.*  
palette = brewer['YlGnBu'][8]*#Reverse color order so that dark blue is highest obesity.*  
palette = palette[::-1]*#Instantiate LinearColorMapper that linearly maps numbers in a range, into a sequence of colors.*  
color\_mapper = LinearColorMapper(palette = palette, low = 0, high = 40)*#Define custom tick labels for color bar.*  
tick\_labels = {'0': '0%', '5': '5%', '10':'10%', '15':'15%', '20':'20%', '25':'25%', '30':'30%','35':'35%', '40': '>40%'}*#Create color bar.*   
color\_bar = ColorBar(color\_mapper=color\_mapper, label\_standoff=8,width = 500, height = 20,  
border\_line\_color=None,location = (0,0), orientation = 'horizontal', major\_label\_overrides = tick\_labels)*#Create figure object.*  
p = figure(title = 'Share of adults who are obese, 2016', plot\_height = 600 , plot\_width = 950, toolbar\_location = None)  
p.xgrid.grid\_line\_color = None  
p.ygrid.grid\_line\_color = None*#Add patch renderer to figure.*   
p.patches('xs','ys', source = geosource,fill\_color = {'field' :'per\_cent\_obesity', 'transform' : color\_mapper},  
 line\_color = 'black', line\_width = 0.25, fill\_alpha = 1)*#Specify figure layout.*  
p.add\_layout(color\_bar, 'below')*#Display figure inline in Jupyter Notebook.*  
output\_notebook()*#Display figure.*  
show(p)





Cool! Our choropleth map has been generated. You have probably spotted a problem in this map. The African continent looks broken; that’s because Sudan, South Sudan and Somaliland are missing. Also, Greenland is missing in the world map. The reason these countries have not been drawn is that one or more values corresponding to these countries are missing in our .csv file and therefore, dataframe df. Let’s get this sorted as it would put off some people, and rightly so!

**Correct for missing countries**

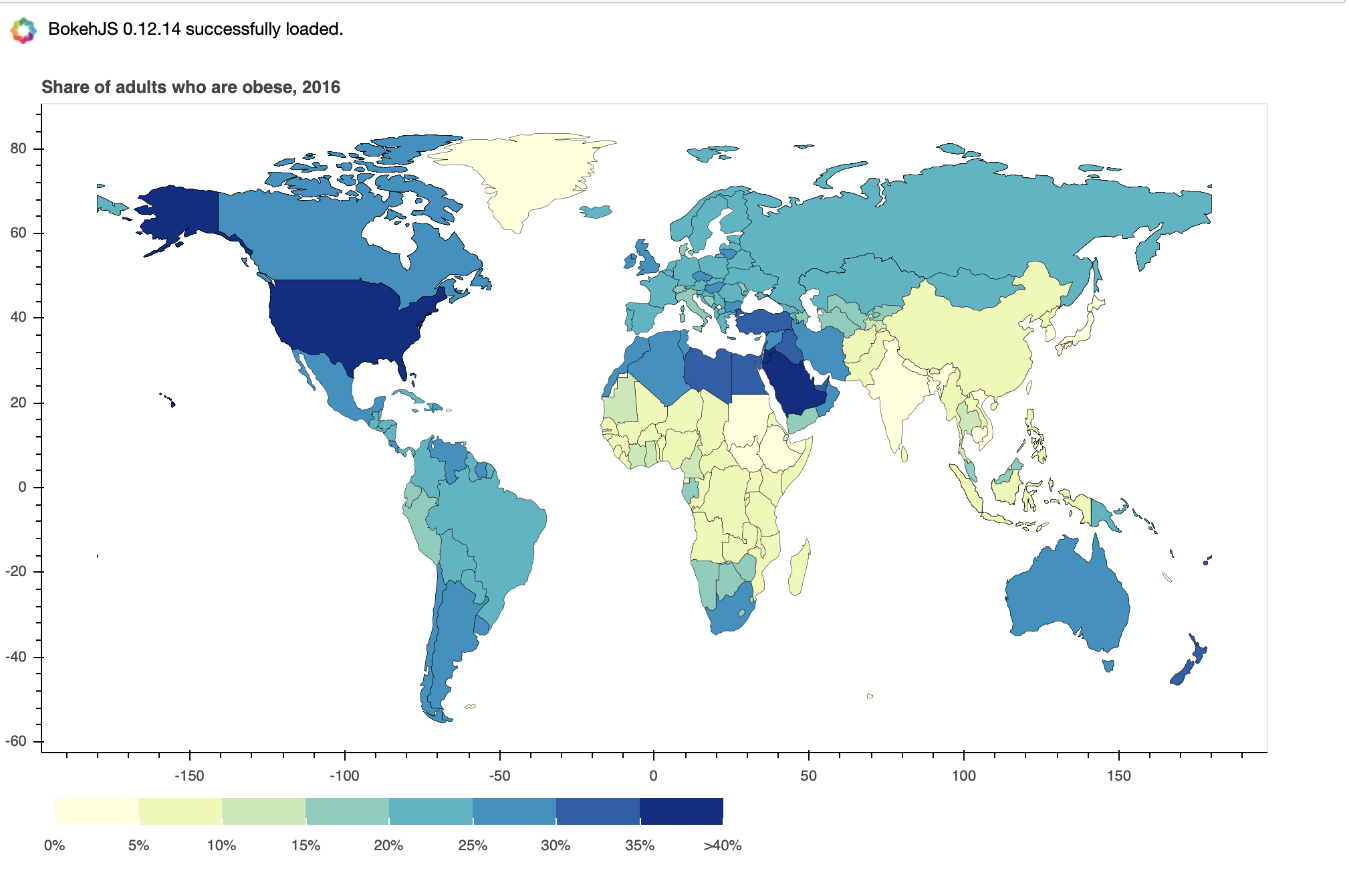
We can ensure the missing countries are drawn on the map and fill them with light grey, indicating missing data. This requires some editing in our previous code.

To preserve the missing countries in our merged data frame, we perform a left merge/left outer join. Thus, every country in GeoDataframe gdf is preserved in the merged dataframe, even if the corresponding row/s are missing in the pandas dataframe df.

df\_2016 = df[df['year'] == 2016]*#Perform left merge to preserve every row in gdf.*  
merged = gdf.merge(df\_yr, left\_on = 'country\_code', right\_on = 'code', how = 'left')

This results in the following figure.





Choropleth map for year 2016 (after left merge)

So now we see the previously missing countries on our map, but there is still a problem. The newly added countries are color coded as defined in our color mapper, indicating obesity in the range of 0–5%. This is misleading, because in reality this data is missing. Ideally, we would want these countries to be colored in a neutral shade eg. light grey which indicates missing data. How can we accomplish that?

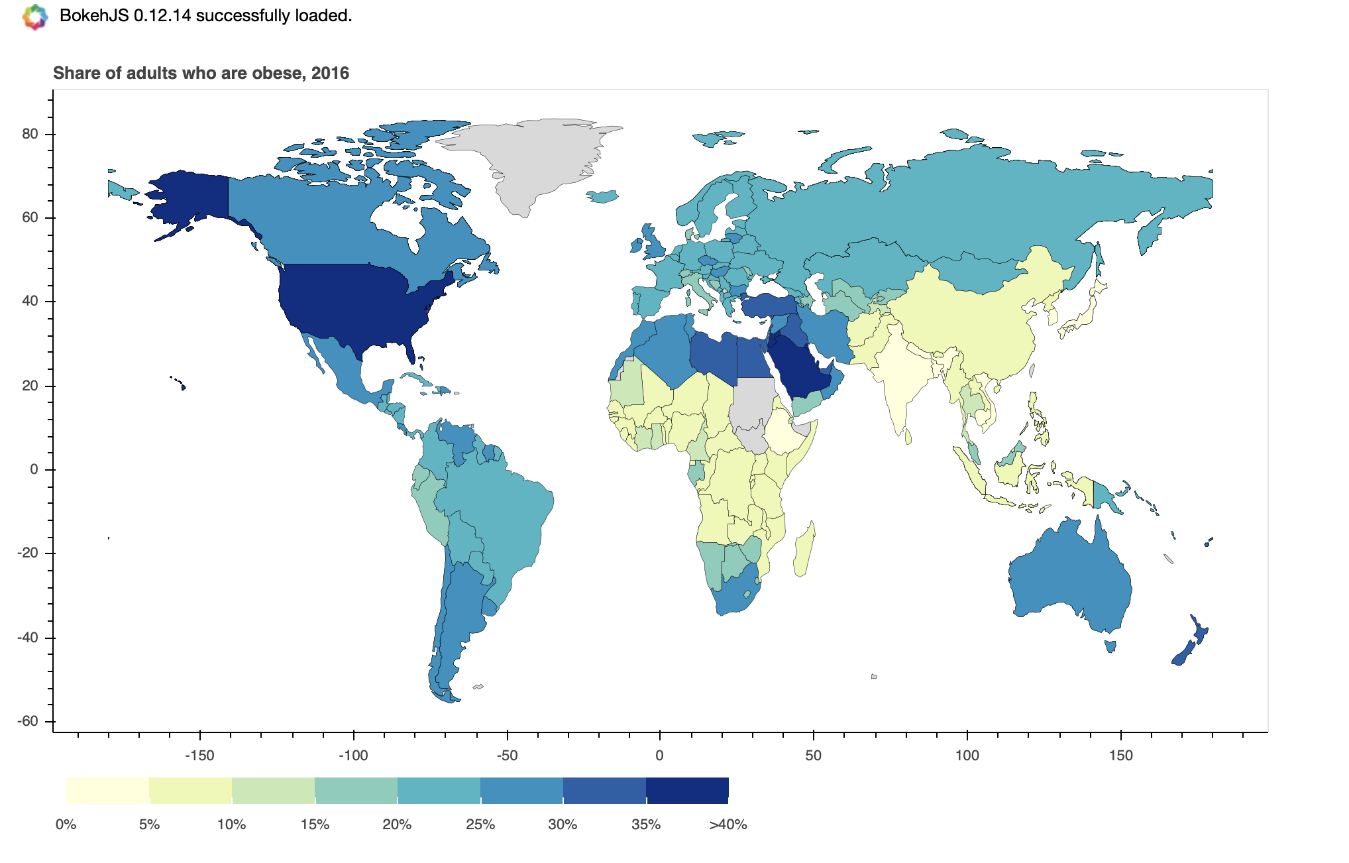
A left merge results in addition of NaN values in the merged dataframe for corresponding missing values in the right dataframe (df\_2016). The problem arises when we convert this merged dataframe into GeoJSON format, as NaN is not a valid JSON object. To circumvent this, we will replace all NaN values in merged dataframe into a string ‘No data’.

df\_2016 = df[df['year'] == 2016]*#Perform left merge to preserve every row in gdf.*  
merged = gdf.merge(df\_2016, left\_on = 'country\_code', right\_on = 'code', how = 'left')*#Replace NaN values to string 'No data'.*  
merged.fillna('No data', inplace = True)

We also input the hex code to color code countries with ‘No data’ as an argument to color mapper.

*#Instantiate LinearColorMapper that maps numbers in a range linearly into a sequence of colors. Input nan\_color.*color\_mapper = LinearColorMapper(palette = palette, low = 0, high = 40, nan\_color = '#d9d9d9')





Choropleth map for year 2016 (after left merge and replacing NaNs)

Great! I am happy with our choropleth map for the year 2016. Time to add some interactivity.

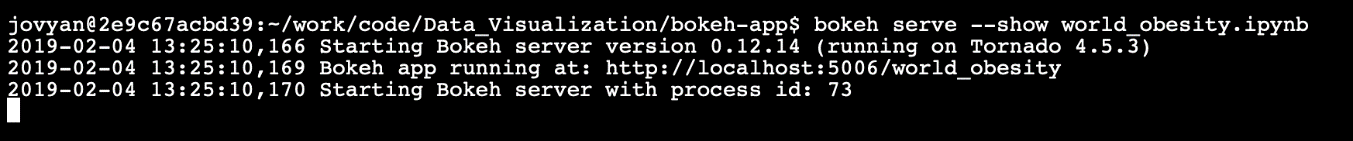
Adding interactivity to our visualization gives the user the ability to extract information they are interested in. Our goal is to create a dynamic map that updates data based on year selected in the range of 1975–2016. We will also add a hover tool which allows user to view details by just hovering the mouse over a specific country/region.

Bokeh provides an extensive set of widgets and tools and makes it very simple to create rich, interactive visualizations. We will define a few functions and reuse a major chunk of code written for creating the static map. The code is described inline.

from bokeh.io import curdoc, output\_notebook  
from bokeh.models import Slider, HoverTool  
from bokeh.layouts import widgetbox, row, column*#Define function that returns json\_data for year selected by user.*  
   
def json\_data(selectedYear):  
 yr = selectedYear  
 df\_yr = df[df['year'] == yr]  
 merged = gdf.merge(df\_yr, left\_on = 'country\_code', right\_on = 'code', how = 'left')  
 merged.fillna('No data', inplace = True)  
 merged\_json = json.loads(merged.to\_json())  
 json\_data = json.dumps(merged\_json)  
 return json\_data*#Input GeoJSON source that contains features for plotting.*  
geosource = GeoJSONDataSource(geojson = json\_data(2016))*#Define a sequential multi-hue color palette.*  
palette = brewer['YlGnBu'][8]*#Reverse color order so that dark blue is highest obesity.*  
palette = palette[::-1]*#Instantiate LinearColorMapper that linearly maps numbers in a range, into a sequence of colors. Input nan\_color.*  
color\_mapper = LinearColorMapper(palette = palette, low = 0, high = 40, nan\_color = '#d9d9d9')*#Define custom tick labels for color bar.*  
tick\_labels = {'0': '0%', '5': '5%', '10':'10%', '15':'15%', '20':'20%', '25':'25%', '30':'30%','35':'35%', '40': '>40%'}*#Add hover tool*  
hover = HoverTool(tooltips = [ ('Country/region','[@country](http://twitter.com/country" \t "_blank)'),('% obesity', '@per\_cent\_obesity')])*#Create color bar.*   
color\_bar = ColorBar(color\_mapper=color\_mapper, label\_standoff=8,width = 500, height = 20,  
 border\_line\_color=None,location = (0,0), orientation = 'horizontal', major\_label\_overrides = tick\_labels)*#Create figure object.*  
p = figure(title = 'Share of adults who are obese, 2016', plot\_height = 600 , plot\_width = 950, toolbar\_location = None, tools = [hover])  
p.xgrid.grid\_line\_color = None  
p.ygrid.grid\_line\_color = None*#Add patch renderer to figure.*   
p.patches('xs','ys', source = geosource,fill\_color = {'field' :'per\_cent\_obesity', 'transform' : color\_mapper},  
 line\_color = 'black', line\_width = 0.25, fill\_alpha = 1)*#Specify layout*  
p.add\_layout(color\_bar, 'below')*# Define the callback function: update\_plot*def update\_plot(attr, old, new):  
 yr = slider.value  
 new\_data = json\_data(yr)  
 geosource.geojson = new\_data  
 p.title.text = 'Share of adults who are obese, %d' %yr  
   
*# Make a slider object: slider*   
slider = Slider(title = 'Year',start = 1975, end = 2016, step = 1, value = 2016)  
slider.on\_change('value', update\_plot)*# Make a column layout of widgetbox(slider) and plot, and add it to the current document*  
layout = column(p,widgetbox(slider))  
curdoc().add\_root(layout)*#Display plot inline in Jupyter notebook*  
output\_notebook()*#Display plot*  
show(layout)

Awesome! We have successfully recreated the interactive choropleth map for **Share of Adults who are obese.**Note that the plot does not update when you change slider value in your Jupyter Notebook. To view this application in interactive mode you need to set up a local Bokeh server. Open a command line window in your current directory and executebokeh serve --show filename.ipynb command.





Setting up local Bokeh server

Click on the link prompted to open the application in your browser. I have inserted a short clip that shows the application in interactive mode.

World obesity Interactive Choropleth map

There are a few cool features in the original map on [Our World in Data](https://ourworldindata.org/obesity" \t "_blank) that I could not incorporate; like highlighting a country upon hovering, highlighting all countries when you hover over a segment on the color bar. I also could’nt figure out the display of ‘light grey’ segment for ‘No data’ in juxtaposition to color bar. If you have any comments or suggestions on how we can further enhance the application interactivity, I would love to hear them!