For our final project, your task will be to pull together all of the pieces from the previous weeks' practice projects and draw a map of the USA overlaid by the county level cancer-data. In particular, you will take your solution to the practice project from the first week of this specialization and add code that loads and plots the cancer-risk data at the corresponding county centers (using the CSV file from week three).

To provide a reasonable visualization of this county data , you will use the size of the scatter points to indicate the population of the county and the color of the scatter points to indicate the level of risk. To this end, you will need to experiment with the various options for the `scatter()` method in the `matplotlib.pyplot` module.

Practice Project: Visualizing Cancer-risk Data on the USA Map

Read the cancer-risk data from week three and create a preliminary plot

For the first part of this project your task is to implement a function `draw_cancer_risk_map(joined_csv_file_name, map_name, num_counties)` that takes the name of the joined cancer-risk CSV file (from week three) and the name of the USA map and draws a scatter plot with scatter points of fixed size and color at the center of the `num_counties` counties with highest cancer risk. Omitting the final optional argument `num_counties` should default to drawing all counties.

To begin this task, we suggest that you start from your solution to the practice project for week one and then add code that reads the CSV file from week three's practice project. Next, you should sort the rows of the resulting table by cancer-risk if necessary. Finally, you should add code that iterates through the rows of the cancer-risk table and plot the appropriate scatter points. When your plot is done, visually verify that the centers for each county are plotted at an appropriate location. As done in the practice project for week one, your code should work on any of the various resolution PNG versions of the USA map that are available.

Compute appropriate sizes for the scatter points based on population

To make your resulting plot more informative, you should scale the size of the scatter points to reflect the population of the county. To this end, we suggest that you review the documentation for the `scatter()` method and experiment with various methods for calculating appropriate values for the `s` option from the county's population. One key fact about this option is that the "size" of the scatter point corresponds to the area of the point (not it's diameter).

To isolate this portion of your code from the main body of the plotting code, we suggest that you implement a function `compute_county_cirle(county_population)` that takes the population of a county and returns an appropriate area for the corresponding scatter point. Note that your implementation of this function will almost certainly involve the inclusion of a "magic" constant that yields scatter points of an aesthetically pleasing size.

Compute appropriate colors for the scatter points based on cancer-risk

To complete this project, your final task is to assign an RGB color to each scatter point using the `c` option. This color should reflect the level of cancer-risk for each county and vary from red (indicating high risk) to green/blue (indicating low risk). Surprisingly, this seemingly simple task is the hardest part of this project (in our opinion) since it will require you to learn a modest amount about how `matplotlib` handles colors.

To this end, we suggest that you begin by reviewing the documentation on how colors are specified in `matplotlib`. Although there are a number of possible ways to specify colors, our task is complicated by the fact that the cancer-risks are all small floating point numbers in the range $8.60E-06$ (small risk) to $1.50E-04$ (large risk). One approach would to use the logarithm function to scale these values into a range suitable for use with gray-scale colors. However, we suggest a more colorful approach that leverages some of the more advanced features of matplotlib.

`matplotlib` includes a powerful tool for mapping a varying range of floating point numbers to a range of colors called a colormap. Colormaps can be used in matlpotlib to create a function that maps a floating point number in a specified range into a corresponding color in a specified colormap. We suggest that you review the documentation for working with colors and colormaps.

Your task for the final part of this project is to create a function `create_riskmap(colormap)` that take a colormap from the module `matplotlib.cm` module and returns a **function** that maps a cancer risk to an RGB value from the given colormap. Our code for this task includes the following steps:

1. Use the **`matplotlib.colors.Normalize()`** method to define a function that the normalizes numbers between the logarithm of the minimum and maximum cancer risks to the range [0, 1]. Here, we use the logarithm of the risks to provided a better visualization of the range of values.
2. Create a scalar mapping from floats to colors using the **`matplotlib.cm.ScalarMappable()`** method using the normalization function computed in step one and the given colormap.
3. Return a function (created using **`lambda`**) that takes the logarithm of the given cancer risk, plugs the value into the scalar mapping. and finally converts the result to an RGB color using the **`matplotlib.cm.ScalarMappable.to_rgba`** method.

Once your implementation of **`create_riskmap()`** is complete, use **`create_riskmap()`** to create a function that maps cancer-risks to colors. We suggest using the color map **`matplotlib.cm.jet`** to generate a nice range of colors from red to green/blue. Finally, call this function inside the calls to the **`scatter()`** to compute appropriate values for the **`c`** option.

Now run your code on the joined CSV file created in week 3. If your code works similar to ours, the resulting scatter plot should look something like this image. If you are interested in more details concerning our solution, feel free to examine our code.

Mark as completed