6.5.2

Create Heatmaps for Weather Parameters

To implement the heatmap feature on your company's website, you'll need to build out the code for the heatmap and test the feature using the weather data with an API call. Time to write some code.

Creating heatmaps is going to be easy and fun. We'll feel like software developers when we add information to geological maps and can visualize the data based on density. The amount of coding needed to create a heatmap is small. For a basic heatmap, all we need to do is provide the following:

- Latitudes and longitudes for the locations
- A measurement value for each latitude and longitude in the form of arrays that have the same number of items in each array

Begin by importing our dependencies and Google API key, and then add our `cities.csv` file to a DataFrame. Create a new Jupyter Notebook file named `VacationPy.ipynb`. In the first cell, add our dependencies and API key.

```
# Import the dependencies.
import pandas as pd
import gmaps
import requests
# Import the API key.
from config import g_key
```

Let's review how we will use our dependencies. We'll use Pandas to read our CSV file and create the locations and measurements from the DataFrame. We'll use gmaps and the API key to create heatmaps and the locations map, and we'll use the requests dependency to make a request to the Google Places JSON file. This will allow us to get hotel locations from the latitude and longitude of the city.

Next, we'll read our `cities.csv` file into a DataFrame. Add the following code and run it to create the DataFrame.

```
# Store the CSV you saved created in part one into a DataFrame.
city_data_df = pd.read_csv("weather_data/cities.csv")
city_data_df.head()
```

One caveat to using gmaps: The data we use for any mapping must be either an integer or a floating-point decimal number. Let's check the data types for the columns of our DataFrame.

REWIND

Recall that you use the `dtypes` method to get the data types of a DataFrame.

Confirm the data types for the data columns are integers or floating-point decimal numbers.

```
# Get the data types.
city_data_df.dtypes
```

```
City_ID          int64
City            object
Country         object
Date            object
Lat            float64
Lng            float64
Max Temp       float64
Humidity         int64
Cloudiness       int64
Wind Speed     float64
dtype: object
```

Create a Maximum Temperature Heatmap

First, tell gmaps to use your API key. You only need to configure gmaps to use your API key once.

Add the following code to a new cell and run the cell.

```
# Configure gmaps to use your Google API key.
gmaps.configure(api_key=g_key)
```

Next, create the heatmap for the maximum temperature. The general syntax for creating a heatmap is as follows.

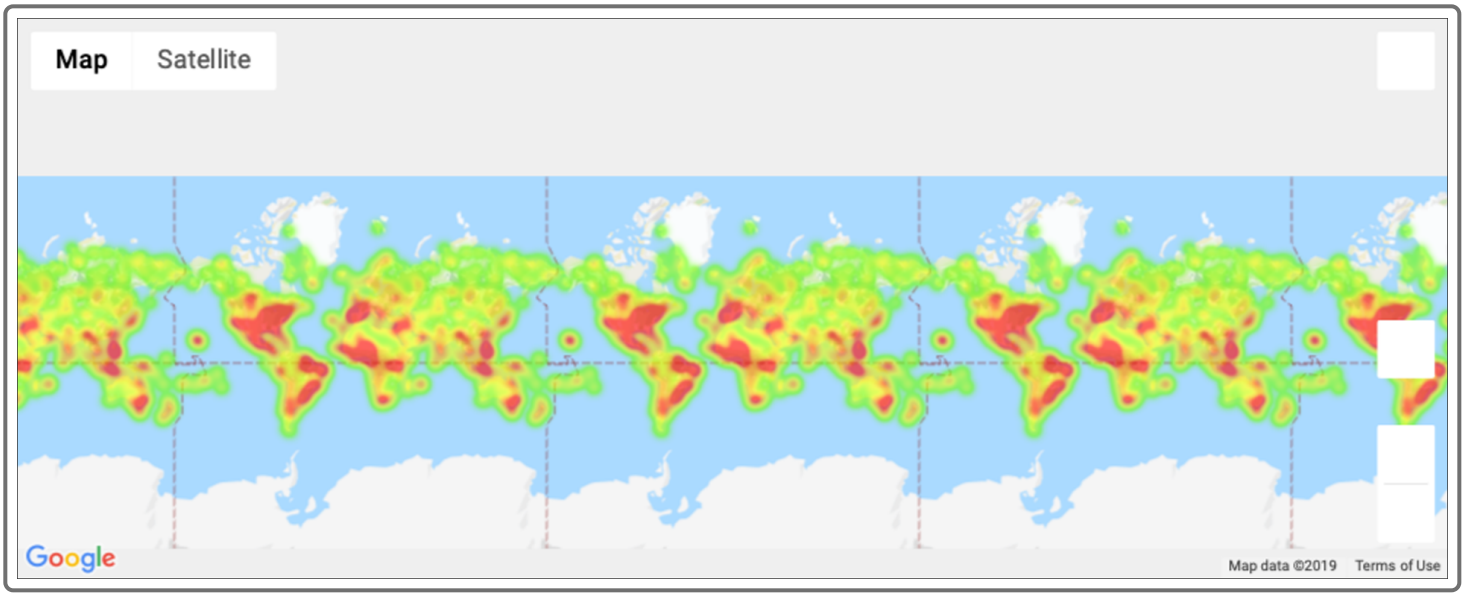
```
# 1. Assign the locations to an array of latitude and longitude pairs.
locations = [latitude, longitude]
# 2. Assign the weights variable to some values.
temperatures = # an array of length equal to the locations array length
# 3. Assign the figure variable to the gmaps.figure() attribute.
fig = gmaps.figure()
# 4. Assign the heatmap_layer variable to the heatmap_layer attribute and
heatmap_layer = gmaps.heatmap_layer(locations, weights=temperatures)

# 5. Add the heatmap layer.
fig.add_layer(heatmap_layer)
# 6. Call the figure to plot the data.
fig
```

Add the locations array and the values from the maximum temperature from our `city_data_df` DataFrame for Steps 1 and 2 and the rest of the code above.

```
# Heatmap of temperature
# Get the latitude and longitude.
locations = city_data_df[["Lat", "Lng"]]
# Get the maximum temperature.
max_temp = city_data_df["Max Temp"]
# Assign the figure variable.
```

```
fig = gmaps.figure()
# Assign the heatmap variable.
heat_layer = gmaps.heatmap_layer(locations, weights=max_temp)
# Add the heatmap layer.
fig.add_layer(heat_layer)
# Call the figure to plot the data.
fig
```

When we run this cell, we get the following large landscape map in the output window.



Google heatmaps do not plot negative numbers. If you have a maximum temperature that is less than 0 °F, then you will get an `InvalidWeightException` error for this line of code:

```
heat_layer = gmaps.heatmap_layer(locations, weights=max_temp)
```

To remove the negative temperatures we can use a `for` loop to iterate through the `max_temp` and add the temperatures that are greater than 0 °F to a new list.

Add a new cell above our previous code block, and then add the following code in the cell and run the cell.

```
# Get the maximum temperature.
max_temp = city_data_df["Max Temp"]
temps = []
for temp in max_temp:
    temps.append(max(temp, 0))
```

In the `for` loop, we're using the `max()` function to get the largest value between the `temp` and 0. If the `temp` is less than 0, then 0 will be added to the list in its place. Otherwise, the `temp` is added to the list.

Now change the following code in the `heat_layer`:

```
heat_layer = gmaps.heatmap_layer(locations, weights=max_temp)
```

To look like the following:

```
heat_layer = gmaps.heatmap_layer(locations, weights=temps)
```

Rerun the cell. You should see a heatmap in the output window.

How would you convert the following for loop using list comprehension?

```
for temp in max_temp:
    temps.append(max(temp, 0))
```

- `[for temp in max_temp max(temp, 0)]`
- `[for temp in max_temp(temps.append(temp)]`
- `[max(temp, 0) for temp in max_temp]` ✔

Feedback
Correct. Nice work!

Retake

Instead of using the `for` loop, we can perform a list comprehension within the `heatmap_layer()` function.

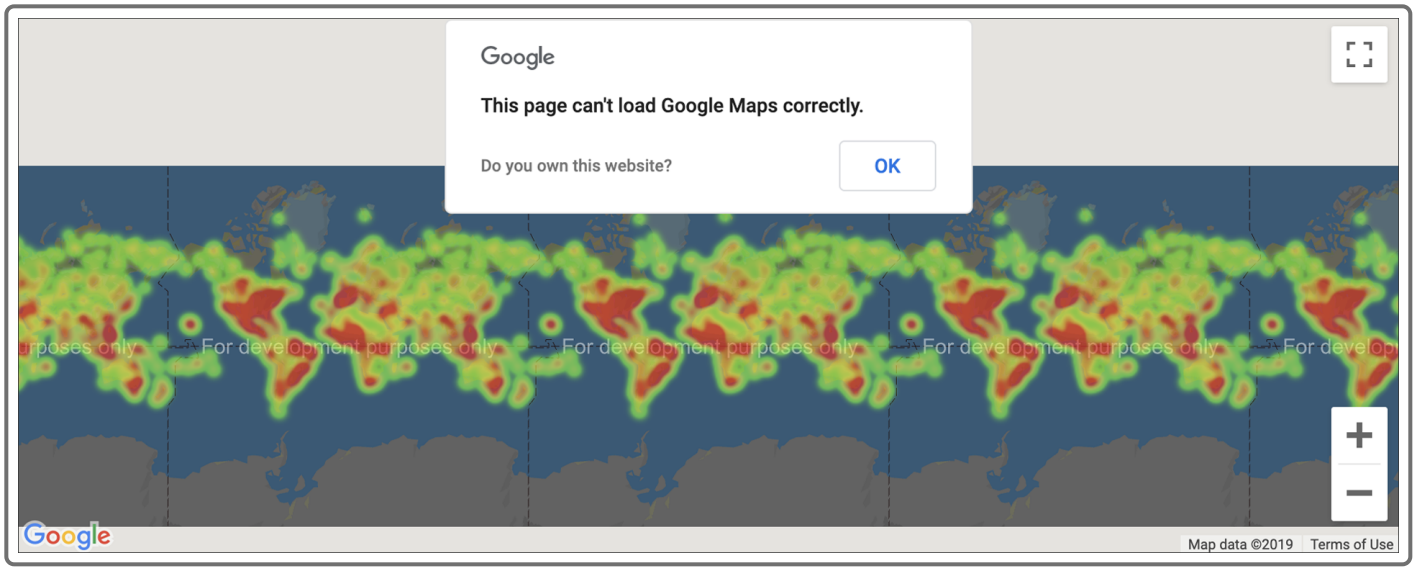
Replace `temps` with our code for the list comprehension so our `heat_layer` code looks like the following:

```
heat_layer = gmaps.heatmap_layer(locations, weights=[max(temp, 0) for temp in max_temp])
```

Rerun the cell.

IMPORTANT

Does your map look like the following image with a pop-up message saying "The page can't load Google Maps correctly"?



The common errors for this are:

- Not configuring gmaps to use your API key with `gmaps.configure(api_key=g_key)`
- Not importing your API key from the correct folder
- Not having or using the correct API key for gmaps
- Not adding a credit card to your account for billing

Other options can be found by clicking on "Do you own this website?" in the pop-up message window.

For more information, see the **documentation on error messages (https://developers.google.com/maps/documentation/javascript/error-messages?utm_source=maps_js&utm_medium=degraded&utm_campaign=keyless#api-key-and-billing-errors)** .

This map is too large. We will need to make some adjustments to the `gmaps.figure()` attribute.

Adjust Heatmap Zoom, Intensity, and Point Radius

First, add the geographic center of Earth in the form of latitude and longitude (30.0° N and 31.0° E). Also, add a zoom level so that only one map of Earth is shown.

When we add a center and zoom level to the `gmaps.figure()` attribute, it will look like this:

```
fig = gmaps.figure(center=(30.0, 31.0), zoom_level=1.5)
```

You might have to play with the zoom level to get things right.

When we run the cell, our map is centered and shows all the inhabitable places on Earth.

If we scroll and zoom in on North America, we can see that the circles for the temperatures need to be modified so that they are larger and show temperature gradient differences.

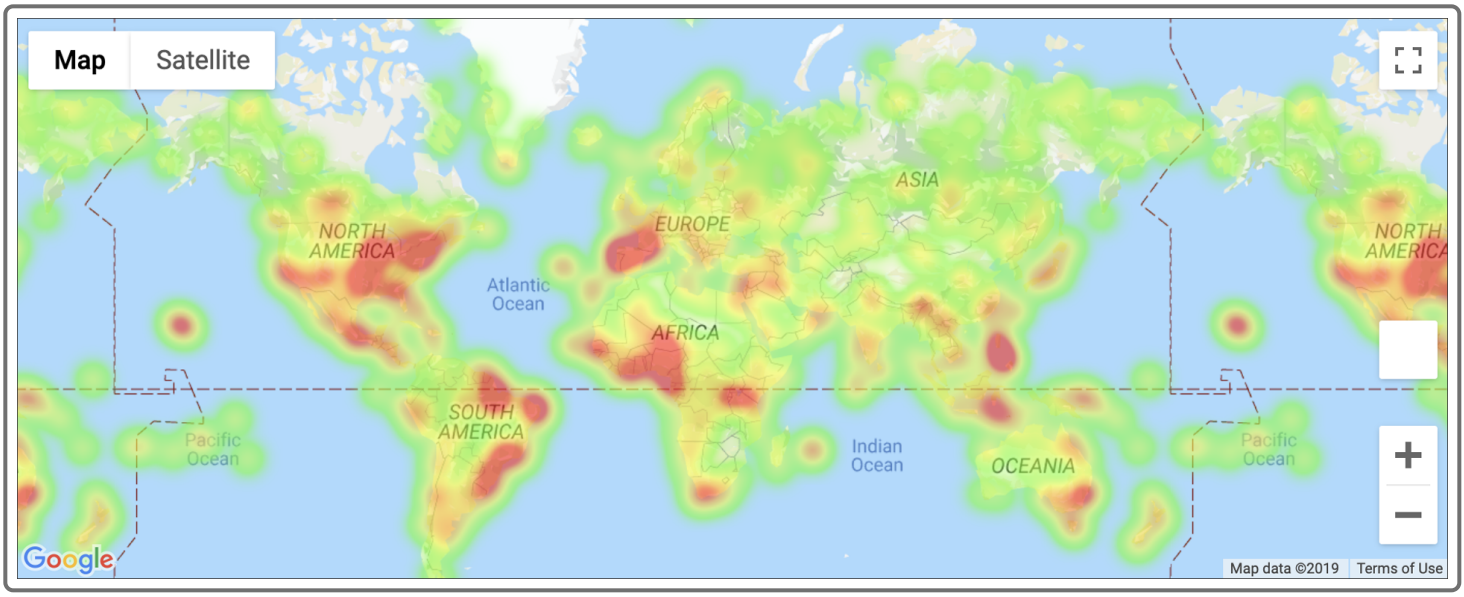
If you review the gmaps documentation, you may notice there is a dissipation option for creating heatmaps that can be added to the `gmaps.heat_layer()` attribute.

The options for this are:

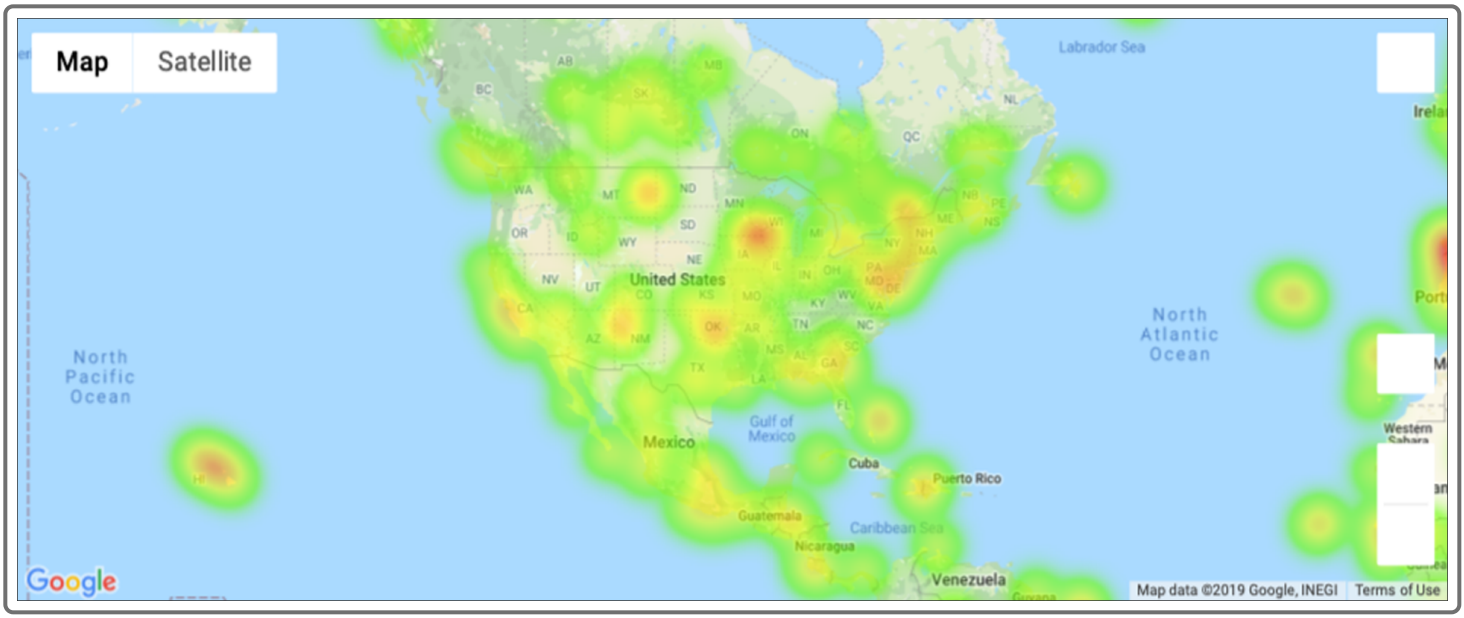
1. The default option for the dissipation is "True," so we need to set our "dissipation" to "False."
2. We can add `max_intensity` to make each measurement have a better gradient variance.
3. We can add `point_radius` to make each measurement radius larger.

Also, you'll have to tweak these values until you are satisfied with the final map.

Let's modify our `gmaps.heat_layer()` attribute to `heat_layer = gmaps.heatmap_layer(locations, weights=[max(temp, 0) for temp in max_temp], dissipating=False, max_intensity=300, point_radius=4)` and rerun our code.



If we scroll and zoom in on North America, we can see that the circles for the temperature are larger and the gradients show differences.



Now our maximum temperature heat map looks a lot better!

NOTE

If you see the following message in the output window below your code block, when you open your Jupyter Notebook file that had the Google map, you will have rerun the cell to create the map.

```
# Add the heatmap layer
fig.add_layer(heat_layer)

fig
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

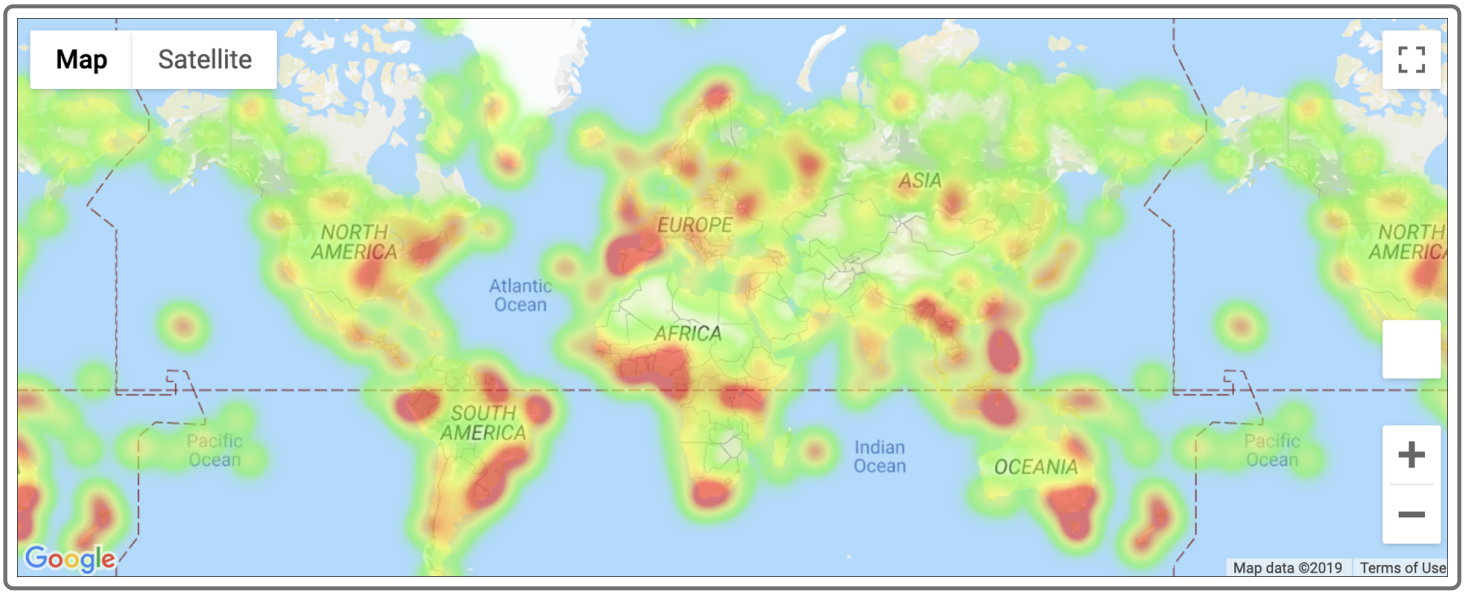
Create a Percent Humidity Heatmap

Now that we have created our maximum temperature heatmap, let's create the heatmap for humidity. We can reuse the code and use the humidity values for the measurements.

Add the following code to a new cell and run the cell.

```
# Heatmap of percent humidity
locations = city_data_df[["Lat", "Lng"]]
humidity = city_data_df["Humidity"]
fig = gmaps.figure(center=(30.0, 31.0), zoom_level=1.5)
heat_layer = gmaps.heatmap_layer(locations, weights=humidity, dissipating

fig.add_layer(heat_layer)
# Call the figure to plot the data.
fig
```

Our humidity heatmap should look like the following.



Create a Percent Cloudiness Heatmap

Next, we will create the heatmap for percent cloudiness.

Complete the code to create the heatmap for percent cloudiness.

```
# Heatmap of percent cloudiness

locations = city_data_df[["Lat", "Lng"]]

clouds = city_data_df["Cloudiness"]

fig = gmaps.figure(center=(30.0, 31.0), zoom_level=1.5)

heat_layer = gmaps.heatmap_layer(locations, weights=

dissipating=False, max_intensity=300, point_radius=4)

fig.add_layer(heat_layer)

# Call the figure to plot the data.

fig
```

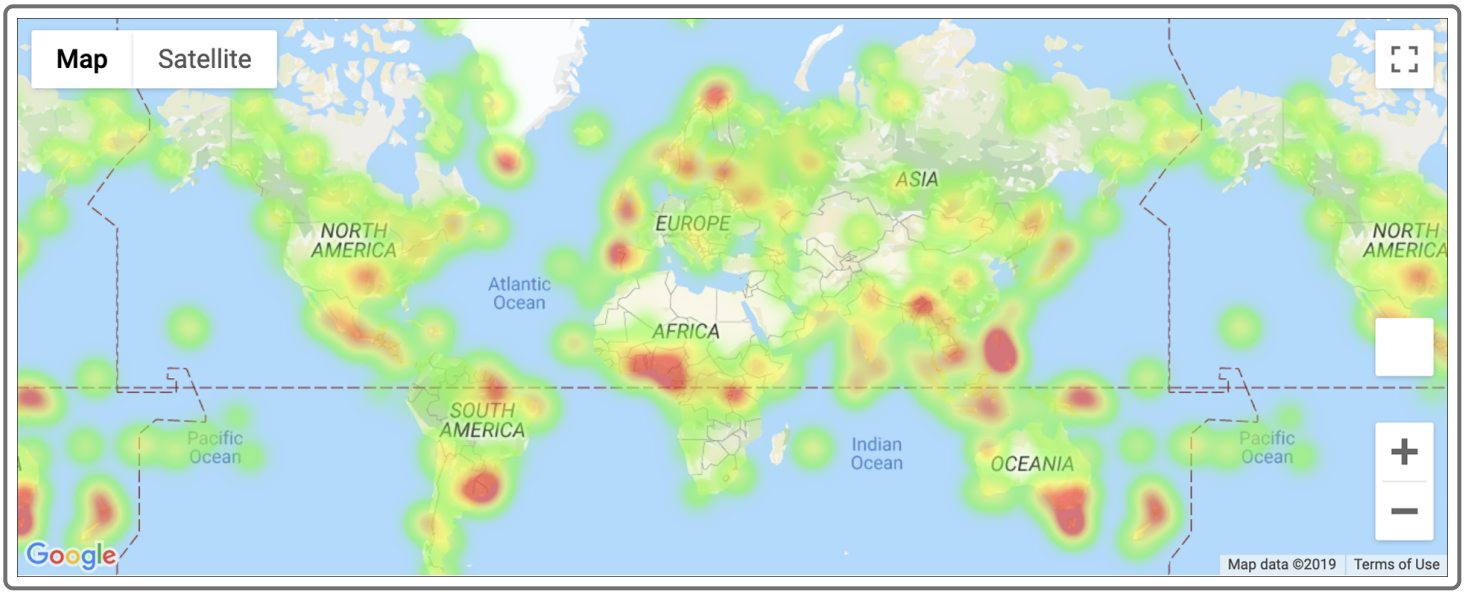
cloudiness | clouds=weights

Feedback
Correct. Good job!

Finish ▶

To create the heatmap for percent cloudiness, replace the measurement from the humidity values to cloudiness values. Copy the code that created the previous heatmap and edit your code with the line `clouds = city_data_df["Cloudiness"]` so that the percent cloudiness replaces the percent humidity, and set the variable `weights=clouds`, and run your cell.

Your percent cloudiness heatmap should look like the following.



Create a Wind Speed Heatmap

Now we can create the final heatmap. Copy the code that created the previous heatmap and edit your code with the line `wind = city_data_df["Wind Speed"]` so that the wind speed replaces the percent cloudiness, and set the variable `weights=wind`, and run your cell.

Your wind speed heatmap should look similar to the following.

Congratulations on creating the heatmaps for each weather parameter!