Reptile Radar: a Python GIS Approach to Studying Species Distribution Near Major Roadways

GeoComputation 1 GTECH 73100 01[7140]

Erin Witt

Abstract

Human activity, especially through the construction and expansion of road networks, poses significant threats to the habitats of various reptile species all over the United States. Highways and other major roadways fragment ecosystems, impede animal movement, and increase mortality rates due to vehicle collisions (Paterson, 2019). This project introduces a novel Python-based GIS tool designed to analyze and provide comprehensive statistics on how roads endanger the lives of many different reptile species. Utilizing a combination of spatial data from the Global Biodiversity Information Facility of reptile sightings and several different Python libraries and tools, this code assesses the proximity of roads to confirmed sightings of different animals, shows the most likely species to be threatened, and identifies high-risk zones for reptile mortality. The tool aims to be highly user-accessible, catering to researchers, conservationists, and the general public. By providing an easy-to-understand interface and detailed map showing the user's location, it facilitates a deeper understanding of the ecological impacts of road infrastructure on the user's surroundings. The paper discusses the development process of the tool, including data integration, data design, and user interface considerations.

The purpose of Reptile Radar is to raise awareness about the environmental consequences of road development and to promote informed decision-making to mitigate these impacts. By highlighting the urgent need for conservation efforts and habitat protection, this tool serves as a crucial resource in the ongoing efforts to preserve reptile biodiversity amidst increasing human encroachment.

Introduction

The term "herping" refers to the activity of searching for reptiles and amphibians in their natural environment. The idea is to observe the animals living in their ecosystems while disturbing the wildlife as little as possible. There are some challenges associated with this pastime. It can be difficult to know what animals are in your area, what characteristics to look out for in order to spot them and how long they might be. Additionally, if you are located in an urban environment, some of those reptile sightings may be located near dangerous roadways. So, to aid in my and others herping endeavors I developed Reptile Radar, a Python tool that uses geopandas, plotly and other libraries to develop and create useful, user-facing information that can be interpreted by anyone. When you input your coordinates and the distance you want, the tool begins by creating a buffer around your point. It then creates the map displaying the buffer, the point that you are located at and the points surrounding you, color coded to match the animal they represent.

Reptile Radar uses data sourced from the Global Biodiversity Information Facility (GBIF), an internationally curated database of animal sightings, created and managed by a collective of scientists and experts from around the world. As of right now, there are no tools using GBIF data that do what I am attempting to do. They have a built-in map feature that shows a hotspot analysis map but this information is very vague and cannot display more than one species at a time with any specificity. This is not a very useful feature for someone who wants to learn about a specific location with any granularity. A mapping tool like Reptile Radar would be helpful as an addition to governmental websites such as for parks or townships. They could help to convey where dangerous animals are located and where they are most likely to be seen.

Using their website I was able to download the sightings of eleven different reptiles. I then cleaned and managed the information using PostgreSQL. I added information to the table about each

species of animal such as their typical length, color and when they are most likely to be active. I also added a true or false boolean value on whether or not they pose a threat to people and pets. This will later be addded as an additional sentence of each animal in the final output of the tool.

Research Questions

1. How can Python be used to visualize where species are located in at risk areas?
2. How can we enhance the accessibility of public sighting data for at-risk animal populations near roadways and create engaging mapping tools to mobilize environmentally conscious individuals in community conservation efforts?

Literature Review

Animals of all kinds are having their natural habitats fragmented by human activities. This has led to a decrease in the populations of thousands of species who have to cross busy roadways to access needed resources. Donaldson and Weber (2006) developed a GIS to identify extensive regions of natural land cover that could serve as connectors for animal habitats, facilitating easier movement of animals across areas separated by roadways. Another study by Atesoglu (2018) used a GIS and measured how far out noise from roadways extended into the natural habitats of different animals. They developed a software that could predict with some accuracy how noisy a particular area may be based on the usual levels of traffic on the nearby highways.

Kautz et al. (1993) examines the proposed Northern Extension of the Florida Turnpike (NEFT) and the Suncoast Expressway, which aim to link South Florida and Tampa Bay to U.S. 19 in North Florida's Big Bend region. Recognizing the potential traffic increase and its impact on local ecosystems, a mediation process led to a multi-agency agreement to protect U.S. 19 and mitigate development pressures. The Big Bend area is rural, with diverse habitats supporting fisheries and wildlife like the Florida black bear. The project used GIS data to map sensitive ecological resources, including seagrass beds, freshwater buffers, endangered species habitats, and public lands. This data was compiled into a GIS database and distributed for conservation planning. Recommendations for protecting these resources will be submitted to the Florida Governor’s Office to guide land use and development away from environmentally sensitive areas. This application of GIS is just one example of how it can be used to inform the public and professionals of the benefits and drawbacks of human intervention in wildlife.

While these are very compelling uses of GIS to determine the risk that roads play in the lives of different species, the outputs of these tools are meant to be used by professionals. The maps are typically complex and are hard to understand from the perspective of a non-expert. However, this insight could be useful or interesting to those that want to know more about their environment or have a stake in the preservation of reptile species in their communities.

Methodology & Data Collection

My data comes from the Global Biodiversity Information Facility or GBIF. GBIF is an internationally maintained and updated open data source created by professionals in environmental science and other related fields. iNaturalists are certified and add animal sightings of different species to websites all the time. The purpose of this database is to understand species distributions and especially to track those that are at risk or endangered. Their data is very reliable because in order to make a contribution you have to have credentials as a professional or be in a field related to environmental science. My other data source was the United States Census Bureau for my road line files and my county shapefiles. Since this data is from a government agency and maintained and updated often, the information is very likely to be accurate and the best source for this kind of data.

Currently, GBIF's website does not offer many tools for viewing their data online. It is very challenging to discern much from the hotspot maps they display on a species main page. It shows vaguely what continent the species is on and some areas where they are concentrated but there is no granularity to the information displayed. Figure 1 shows the map as is on their website. It clearly does not show the distribution to the data with any serious granularity. This is a hexagon hot-spot analysis map which can be useful in many scenarios but you cannot query this data or use this display for anything except for seeing generally where they congregate. It also should be noted that you cannot filter the data before displaying it here. Many GBIF datasets contain older data points, as far back as the 80s and 90s. For some people's purposes, these sightings may not be relevant to their uses. This map is also fairly misleading. The color scale is very subtle so actual hotspots of sightings are pretty poorly represented.

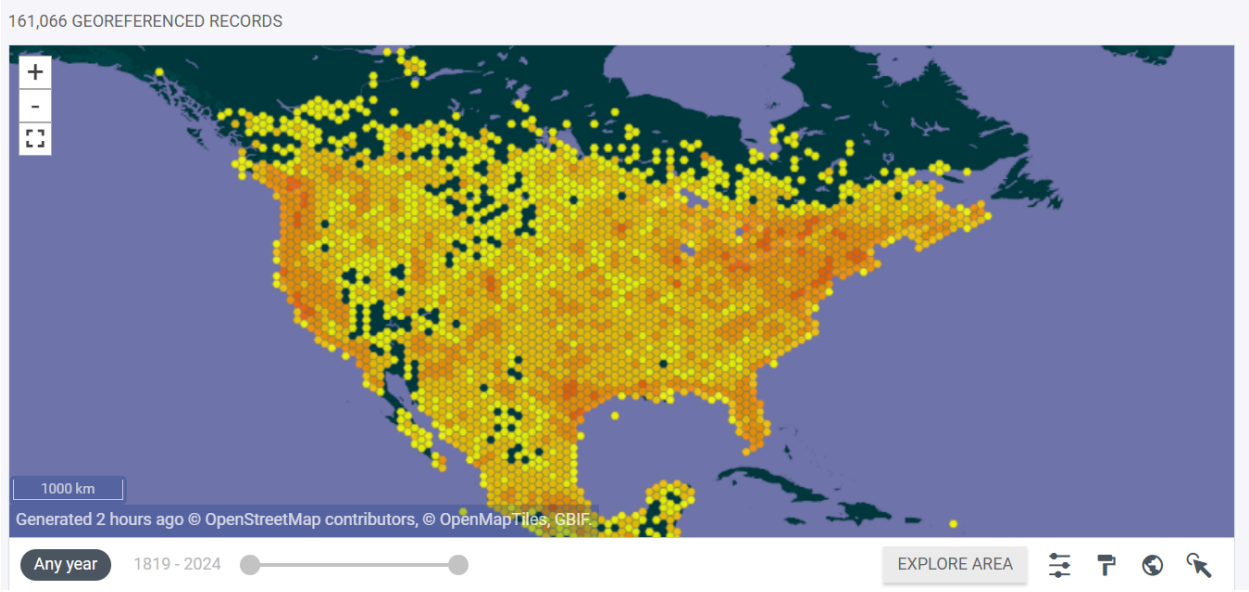
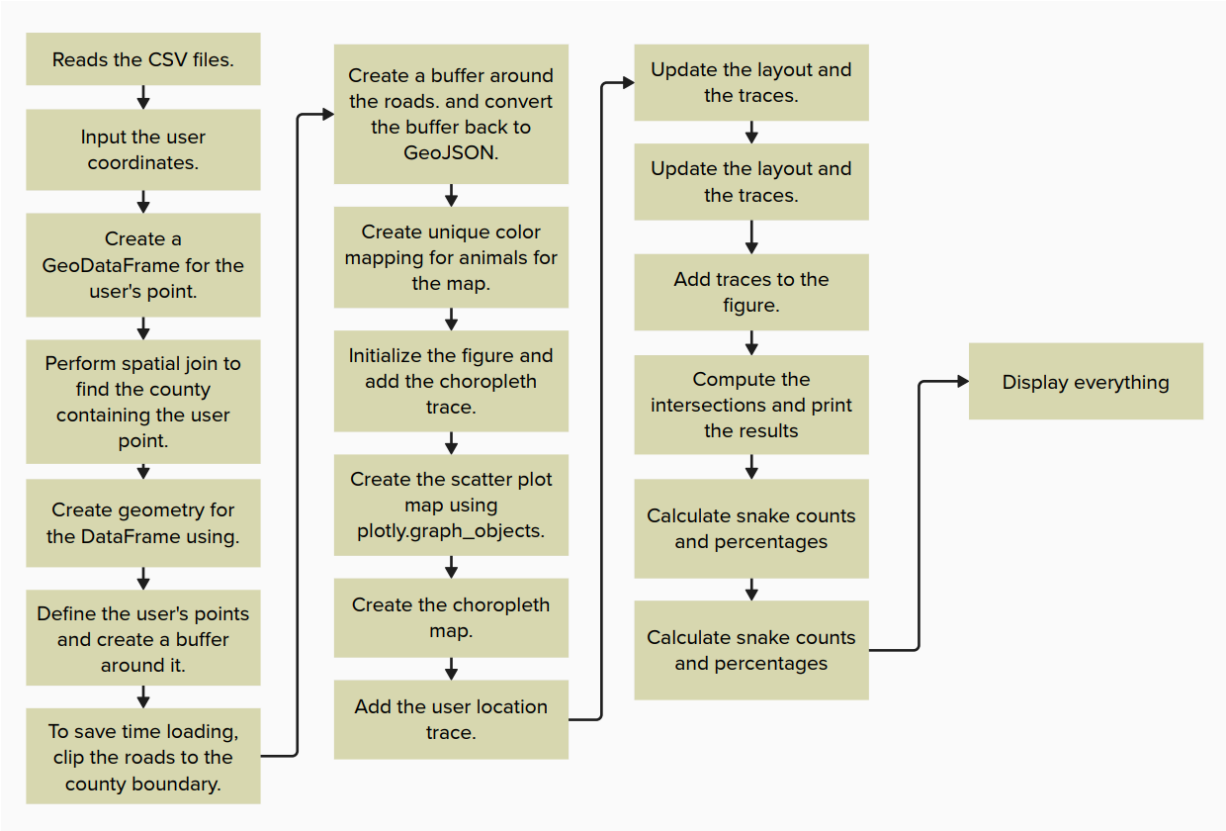


Figure 1. The current mapping tool available on GBIF's website.

WorkFlow Diagram



Expected Results

To measure the success of my project, I could use a survey for people who have used Reptile Radar. I could receive feedback on whether or not they found the generated map useful and clear and whether or not they would use it again. This tool is meant to solve the problem of usability and accessibility so these would be the ideal ways to test if I had reached my goal. Positive survey results showing user satisfaction and repeat usage would be indicators of success.

Another measure of success would be how long the code takes to run. Reptile Radar was developed with efficiency in mind since the point of it is to be a user-facing mapmaking tool. That is why the county file clips the road boundaries so the code runs faster. As it is, Reptile Radar takes less than 10-15 seconds to run from scratch with all new variables. Consistently fast execution times would confirm the tool's efficiency and practicality for real-time use.

Timeline and Budget

The timeline for Reptile Radar's development was a few weeks. The research into the topic took a few days and the data cleaning process took 3 hours. I removed repeat entries and removed excess columns in the tables in order to cut down on the amount of data my code had to process. The cost to develop the code was minimal since Visual Studio Code is free and the Global Biodiversity Information

Facility, and the United States Census Bureau's data is all open source and free to download. To implement my tool into GBIF's website may be challenging and require some serious processing power to use on a mass scale.

Python Libraries Used

The following is a brief summary of what each library was used for in my code.

1. **Plotly (plotly.graph_objects and plotly.express)**

Plotly is a graphing library for making interactive maps using Python. You can make maps, plots, bar charts and other useful visualizations. plotly.graph_objects is specifically for making low-level edits to Plotly graphics, giving users way more control over what they look like. I also used plotly.express to generate unique colors for each of the different animal types. This was important for making the graphs visually appealing to users.

2. **Pandas (pd)**

Pandas is for data manipulation and analysis in Python. It offers data structures like Series and GeoDataFrame which make it possible to use Python for enormous data sets. I used it to read the sighting data CSV files and turn them into a DataFrame which I later used in my code to analyze the results.

3. **Geopandas (gpd)**

Geopandas is an extension of Pandas specifically made for handling geospatial information. It allows users to make spatial joins, geometric and geographic data changes and to transform data to new coordinate reference systems. In my code, I used it to read my geospatial data such as my shapefiles for the roads and counties and to create buffers, clip geometries and other spatial operations.

4. **Shapely**

Shapely is generally used in Python for geometric operations. It allows users to create and alter geometric shapes and perform operations such as union, intersection and buffers. I used it to represent the user's location and animal sighting locations. Those points were then imported into GeoPandas for spatial analysis later.

Important Steps

Step 1: Create User Point GeoDataFrame

Next we make a GeoDataFrame, 'user_point' for the user's location. Then we project it to the right coordinate reference system (CRS). The code uses CRS 32618 (UTM Zone 18N) for actual spatial analysis. This is because this reference system is better suited for smaller, more localized calculations. It offers more accuracy for precise distance and area calculations. However, when we display the data, we show it in 4326 or the World Geodetic System 1984.

```
user_lon = input("What is your longitude?")
user_lat = input("What is your latitude?")
```

```
    user_point = gpd.GeoDataFrame(geometry=[Point(user_lon,
user_lat)], crs="EPSG:4326")
    user_point_projected = user_point.to_crs(epsg=32618)
    point_gdf =
gpd.GeoDataFrame(geometry=gpd.points_from_xy([user_lon], [user_lat]),
crs=county_gdf.crs)
```

Step 2: Create a Spatial Join to Find County Containing User Point
Then I do a spatial join to find which county contains the user point. I want to limit the county file down to just the county the user is located in.

```
    joined_gdf = gpd.sjoin(point_gdf, county_gdf,
predicate='within')
    filtered_county_gdf = county_gdf[county_gdf['NAME'] ==
joined_gdf['NAME'].iloc[0]]
    filtered_county_gdf = filtered_county_gdf.to_crs(epsg=32618)
```

Step 3: Create Buffer Around User Point
Generates an 8000 meter buffer around the user's point and converts it to GeoJSON format so it can be displayed on the final map at a later time.

```
    buffer_user_meters = 8000
    buffer = user_point_projected.buffer(buffer_user_meters)
    buffer_user = buffer.to_crs(epsg=4326).__geo_interface__
```

Step 4: Clip Roads to County Boundary and Convert to GeoJSON
Project the roads to 32618 so it can be analyzed spatially and I clip the roads to the county.

```
    roads = roads.to_crs(epsg=32618)
    clipped_roads = gpd.clip(roads, filtered_county_gdf)
    clipped_roads_geojson =
clipped_roads.to_crs(epsg=4326).__geo_interface__
```

Step 5: Create Buffer Around Roads
We make a 600-meter buffer around the roads and then convert them to a GeoJSON format.

```
    buffer_roads_meters = 600
    buffer_roads =
clipped_roads.to_crs(epsg=32618).buffer(buffer_roads_meters)
    road_buffer_geojson =
buffer_roads.to_crs(epsg=4326).__geo_interface__
```

Step 6: Add Animal Traces to Plot
This portion of the code generates a scatter plot for every type of animal and makes a list of them.

```
animal_traces = []
for animal in df['animalname'].unique():
    animal_df = df[df['animalname'] == animal]
    trace = go.Scattermapbox(
        lon=animal_df['decimallongitude'],
        lat=animal_df['decimallatitude'],
        mode='markers',
        marker=dict(size=9, color=animal_df['color2'].iloc[0]),
        text=animal_df['animalname'],
        name=animal
    )
    animal_traces.append(trace)
```

Step 7: Add User Location Trace
Add a trace to add the user's location to the map.

```
user_location_trace = go.Scattermapbox(
    lon=[user_lon],
    lat=[user_lat],
    marker=dict(size=20, color='red'),
    text=["You are here."],
    name='User Location',
    showlegend=True)
```

Step 8: Add Dummy Trace for Roads to Legend
To ensure that the roads show up on the legend, make a dummy trace to add them so you can see the symbology.

```
road_dummy_trace = go.Scattermapbox(
    lon=[None],
    lat=[None],
    mode='lines',
    line=dict(color='black', width=2),
    name='Roads')
```

Step 9: Update Layout
Update the layout of the figure to include map layers and adjust appearance settings.

```
fig.update_layout(
    mapbox=dict(
        style="open-street-map",
        zoom=12,
        center=dict(lat=user_lat, lon=user_lon),
        layers=[
            {
                "sourcetype": "geojson",
```

```
                "source": clipped_roads_geojson,
                "type": "line",
                "color": "black",
                "line": {"width": 1.5},
                "opacity": 0.7,
                "below": "traces",
            },
            {
                "sourcetype": "geojson",
                "source": road_buffer_geojson,
                "type": "fill",
                "color": "red",
                "opacity": 0.1,
                "below": "traces",
            },
            {
                "sourcetype": "geojson",
                "source": buffer_user,
                "type": "fill",
                "color": "blue",
                "opacity": 0.3,
                "below": "traces",
            }
        ]
    ),
    margin={"r":0, "t":50, "l":0, "b":10},
    width=800,
    height=600,
    title='Reptile Radar Results',
    showlegend=True)
```

Step 10: Add Traces to Figure
Next we add all the animal scatter plot traces, the dummy road trace and the user's location to the figure.

```
for trace in animal_traces:
    fig.add_trace(trace)

fig.add_trace(road_dummy_trace)
fig.add_trace(user_location_trace)
```

Step 11: Compute Intersections and Print the Results
This step finds all the points that are within the user's buffer zone.

```
buffer_new_union = buffer.unary_union
buffer_roads_union = buffer_roads.unary_union
```

```
        intersection =
    buffer_new_union.intersection(buffer_roads_union)
```

Step 12: Calculate Snake Counts and Percentages
Calculate the counts and percentages of each animal in the list of those that intersect and filter those with significant occurrences.

```
    snake_counts = df['animalname'].value_counts()
    total = snake_counts.sum()
    snake_percentages = snake_counts / total * 100

    significant_snakes = snake_counts[snake_percentages >= 5]
    significant_labels = significant_snakes.index.tolist()
```

Step 13: Print Number of Nearby Sightings and Create a GeoDataFrame
Next step was to print the number of nearby sightings for the first line of the output.

```
    print("Number of nearby sightings:", len(intersecting_points))
    print()

    unique_animals_df =
intersecting_points.drop_duplicates(subset='animalname',
keep='first')
    intersection_gdf = gpd.GeoDataFrame(geometry=[intersection],
crs=buffer.crs)
    road_reptiles =
gdf[gdf.intersects(intersection_gdf.unary_union)]
    print("There are", len(road_reptiles), "reptiles near major
roadways near you.")
```

Step 14: Print Number of Nearby Sightings and Create a GeoDataFrame
Print the number of sightings for the first line of the output and create a GeoDataFrame using Geopandas for the reptiles near major roadways portion of the output.

```
    if len(intersecting_points) > 0:
        percentage = len(road_reptiles) / len(intersecting_points)
* 100
        print(f"That is {percentage:.2f}% of all sightings near
you. Be mindful of their habitat.")
        print()
    else:
        print("There are no sightings near roads nearby you.")
        print()
```

Step 21: Print Information about the Animals Near the Road

Print the percentage of sightings near roads and any additional information you might need.

```
    if len(intersecting_points) > 0:
        percentage = len(road_reptiles) / len(intersecting_points)
* 100
        print(f"That is {percentage:.2f}% of all sightings near
you. Be mindful of their habitat.")
        print()
    else:
        print("There are no sightings near roads nearby you.")
        print()
```

Step 15: Generate and Show the Pie Chart
Next step is to generate and show the pie chart of significant reptile sightings near the user and display it as part of the outputs. This step uses Plotly to display the pie chart. Previously, I had used matplotlib but it was incompatible with Plotly and caused an issue on the backend.

```
    fig_pie = go.Figure(data=[go.Pie(labels=significant_labels,
values=significant_snakes)])

    fig_pie.update_layout(
        title='Reptiles Near Roads',
        width=700,
        height=400)

    fig_pie.show()
```

Step 16: Print Information Statements About the Animals that are Nearby
Print detailed information about the animals that are nearby the person using the data that accompanies each sighting in the table.

```
    if len(unique_animals_df) == 0:
        first_sentence = "There are no animals nearby.\n"
    elif len(unique_animals_df) == 1:
        first_sentence = f"There is a
{unique_animals_df.iloc[0]['animalname']}s nearby.\n"
    else:
        first_sentence = "There are "
        for i in range(len(unique_animals_df)):
            if i == len(unique_animals_df) - 1:
                first_sentence += f"and
{unique_animals_df.iloc[i]['animalname']}s nearby.\n"
            else:
                first_sentence +=
f"{unique_animals_df.iloc[i]['animalname']}, "
```

```
            print(first_sentence)
        for index, row in unique_animals_df.iterrows():
            if row['dangerous'] == 't':
                danger_sentence = "They are dangerous, so keep your
distance and stay safe."
            else:
                danger_sentence = ""
            second_sentence = (
                f"The {row['animalname']} is {row['color']} in color
and can grow up to "
                f"{row['size']} centimeters long.\nThey are active
{row['active']}. {row['fact']}\n{danger_sentence}"
            )
            print(second_sentence)
```

Expected Results

From running my code with an example user's location, a few things are displayed:

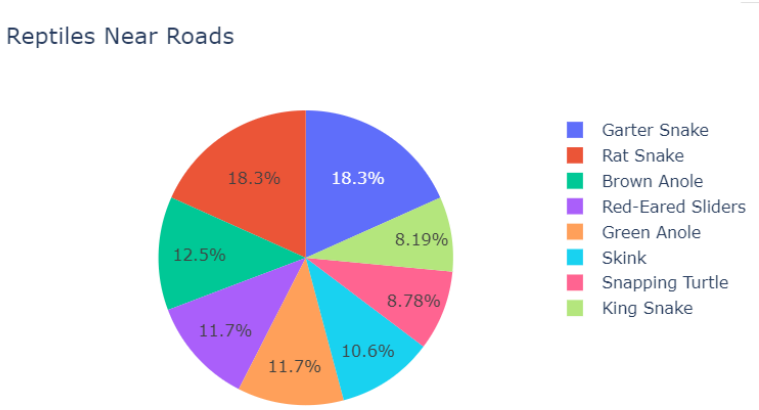
1. A map highlighting the user's location as well and their surrounding area. A light blue buffer surrounds the point and the reptile sightings are smaller color coded points. The roads all have light red buffers around them.
2. A pie chart showing the distribution of species closest to the road by type. I chose to show only those that make up greater than 5% of the population.
3. Sentences based on the animals that have been spotted nearby. Using SQL, I added information to my database about each animal. Their color, size, and other details about each animal accompany ecah entry in the table.

Together, these things make up the output of Reptile Radar.

```
  _____                 _   _ _         _____                 _
 |  __ \               | | (_) |       |  __ \               | |
 | |__) |___ _ __  | |_ _| | ___   | |__) |__ _  __| | __ _ _ __
 |  _  // _ \ '_ \| __| | |/ _ \ |  _  // _` |/ _` |/ _` | '__|
 | | \ \  __/ |_) | |_| | |  __/ | | \ \ (_| | (_| | (_| | |
 |_|  \_\___| .__/ \__|_|_|\___| |_|  \_\__,_|\__,_|\__,_|_|
            | |
            |_|
```

```
Number of nearby sightings: 72

There are 6 reptiles near major roadways near you.
That is 8.33% of all sightings near you. Be mindful of their habitat.
```




```
                _    _ _                                                  _
 _ _ ___ _ __| |_(_) |___ ___  _ _  ___ __ _ _ _   _  _ ___ _  _  (_)
| '_/ -_) '_ \  _| | / -_|_-< | ' \/ -_) _` | '_| | || / _ \ || |  _
|_| \___| .__/\__|_|_\___/__/ |_||_\___\__,_|_|    \_, \___/\_,_| (_)
        |_|                                        |__/
```

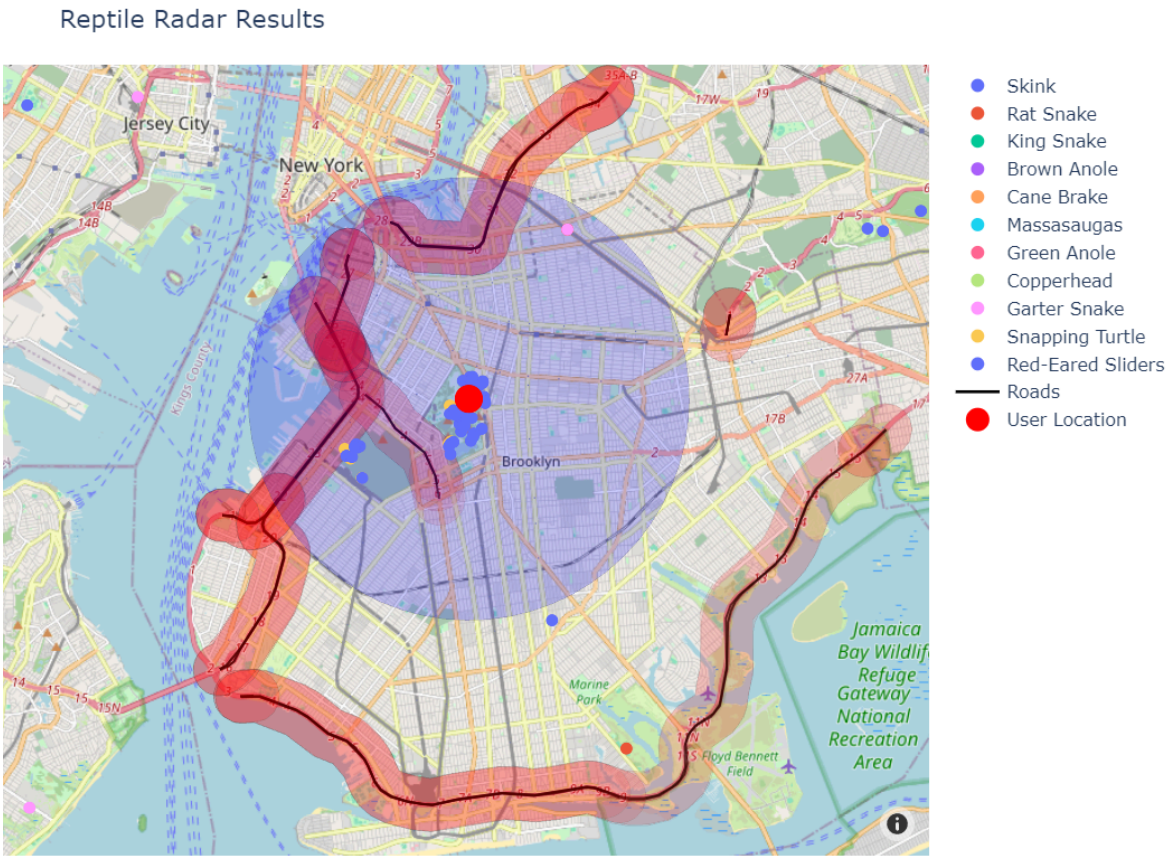
There are Rat Snake, Garter Snake, Red-Eared Sliders, and Snapping Turtles nearby.

The Rat Snake is black and white in color and can grow up to 91-152.4 centimeters long.
They are active during the day time. They are exceptionally good at climbing trees.

The Garter Snake is brown with a yellow stripe down their back in color and can grow up to 51-76.2 centimeters long.
They are active during the day time. From birth they are able to hunt by themselves.

The Red-Eared Sliders is green and red in color and can grow up to 12.7-20.32 centimeters long.
They are active during the day. They are a very popular pet but very invasive when released.

The Snapping Turtle is brown in color and can grow up to 40-70 centimeters long.
They are active during the day. To hunt fish, they stay underwater and keep their mouths open.
They are dangerous, so keep your distance and stay safe.



The Future of Reptile Radar

In the future, Reptile Radar could be integrated into GBIF's website, either as a replacement for their current insufficient mapping tool or as an additional feature available when users download data. This integration would enhance the user experience by providing more detailed and specific information about the proximity of animals to major roadways. The tool could analyze geospatial data to identify patterns and trends in animal locations relative to infrastructure, offering valuable insights for conservationists, researchers, and urban planners alike.

Or, Reptile Radar could be implemented as a web browser extension, allowing users to easily access its functionalities without needing to visit a website. From presenting in class, I received feedback about a tool called Marimo which is an open-source Python notebook that lets users create interactive tools that lets you take input directly from a user. In the future, I would love to expand Reptile Radar to use this notebook. These extensions could provide real-time data and notifications about the presence of animals near roadways in the user's vicinity, and take their exact longitude and latitude. Such a tool would be particularly useful for herping enthusiasts, wildlife experts, and community members who are concerned about conservation.

Conclusion

The habitats of reptiles and other species all over the world face significant threats from the expansion of roadways and escalating human activity. Roads fragment natural landscapes, disrupt important reptile habitats, and lead to the shrinkage and fragmentation of their habitats. Reptiles often require highly specific conditions to survive so they struggle to adapt to altered landscapes, facing increased mortality rates due to vehicle collisions and loss of their food source. Human activity introduces additional stressors such as pollution, habitat destruction for new development. These impacts endanger reptile populations worldwide, highlighting the urgent need for conservation efforts and sustainable land use practices to mitigate these threats and preserve reptile biodiversity. My intention with Reptile Radar is to create an accessible tool for the average person to learn more about their surroundings and the biodiversity around them.

From this project I got to learn a ton about user accessibility, Python programming, its diverse libraries, and how to use them to craft engaging visuals for user interaction. Reptile Radar uses several libraries to make comprehensive and accessible maps and charts.

I believe there is significant potential in utilizing the Global Biodiversity Information Facility's dataset to involve environmentally conscious individuals in protecting the habitats of animals in their communities. However, with the existing tools, this is very difficult. Using Reptile Radar, I aim to empower more individuals to interact with open-source data, enhancing their comprehension of human activity's effects on the environment.

References

Atesoglu, A., (March 2019). Noise Effects of Roads on Wildlife using GIS: A case study of Bartin-Karabuk Highway in Turkey. European Journal of Advances in Engineering and Technology. doi:5(7):493-499

Paterson, J.E. (2019 Aug 13). Road avoidance and its energetic consequences for reptiles. Ecol Evol. 2019 Sep; 9(17): 9794–9803. doi: 10.1002/ece3.5515

Common Garter Snake
Thamnophis Fitzinger, 1843 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

King Snakes or Milk Snakes
Lampropeltis Fitzinger, 1843 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Ratsnakes Dataset
Pantherophis Fitzinger, 1843 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Skink Dataset
Plestiodon Duméril & Bibron, 1839 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Canebrake Rattlesnake
Crotalus horridus Linnaeus, 1758 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Massasauga Rattlesnake
Sistrurus catenatus (Rafinesque, 1818) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Copperhead Rattlesnake
Agkistrodon contortrix (Linnaeus, 1766) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Brown Anole
Anolis sagrei Duméril & Bibron, 1837 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Green Anole
Anolis carolinensis Voigt, 1832 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

Snapping Turtles
Chelydra serpentina (Linnaeus, 1758) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/39omei accessed via GBIF.org on 2023-12-12.

National Park Service. (2022 March 25). Garter Snake Fast Facts. Retrieved vfrom https://www.nps.gov/articles/000/garter-snake-fast-facts.htm

Szalay, J., (2016 Feburary 29) Kingsnake Facts. Retrieved May 15th from https://www.livescience.com/53890-kingsnake.html

Greer, H., (2022). Pantherophis obsoletus Eastern Rat Snake. Animal Diversity Web. Retrieved May 15th from https://animaldiversity.org/accounts/Pantherophis_obsoletus/#economic_importance_positive

Hall, H., (2023 March 29). Skink Lizard. A-Z Animals. Retrieved May 15th from https://a-z-animals.com/animals/skink-lizard/

Wilson, J.D., (2005). Canebrake / Timber Rattlesnake (Crotalus horridus) - Venomous. Retrieved May 15th from https://srelherp.uga.edu/snakes/crohor.htm

Biggs. B., (2022 31 July). Copperhead snakes: Facts, bites & babies. Live Science. Retrieved May 15th from https://www.livescience.com/43641-copperhead-snake.html

Vallie. S., (2022 24 November). What to Know About Brown Anoles. WebMD. Retrieved May 15th from https://www.webmd.com/pets/what-to-know-about-brown-anoles

McLeod. L., (2021 3 March). Green Anole: Species Profile Characteristics, Housing, Diet, and Other Information. The Spruce Pets. Retrieved May 15th from https://www.thesprucepets.com/green-anoles-pets-1236900

The Editors of Encyclopedia Britanica. (2023 3 May). snapping turtle. Encyclopedia Britanica. Retrieved May 15th from https://www.britannica.com/animal/snapping-turtle

Pace. L., (2022 21 July). Red-Eared Slider Trachemys scripta elegans. A-Z Animals. Retrieved May 15th from https://a-z-animals.com/animals/red-eared-slider/

US Census Bureau, Geography Division. (2024). American Road Shapefile. Retrieved