Visual Analytics of the Impacts of Climate Change on Migratory Bird Habitats
Technical Document

Jacob Vogt, Mihika Krishna, Catherine Kang, Hangyul Yun

Professor Dongyu Liu

University of California, Davis

June 2024

Contents

1 Introduction

Our Senior Capstone project is comprised of two components:

1. A species distribution model (SDM) capable of predicting how climate change will affect future bird habitats up to year 2100
2. A web-app that visualizes SDM output and displays other relevant information such as bird migration patterns and climate trends.

This technical document will overview how each of these components work, as well as the required data and file structure for them to operate correctly.

2 Species Distribution Model

2.1 Datasets

The species distribution model (SDM) uses two datasets, one for climate data and one for species distribution data.

Climate Data

Climate scenarios used were from the NEX-GDDP-CMIP6 dataset, prepared by the Climate Analytics Group and NASA Ames Research Center using the NASA Earth Exchange and distributed by the NASA Center for Climate Simulation (NCCS).

Thrasher, B., Wang, W., Michaelis, A. et al. NASA Global Daily Downscaled Projections, CMIP6. Sci Data 9, 262 (2022). https://doi.org/10.1038/s41597-022-01393-4

Thrasher, B., Wang, W., Michaelis, A. Nemani, R. (2021). NEX-GDDP-CMIP6. NASA Center for Climate Simulation. https://doi.org/10.7917/OFSG3345

NEX-GDDP-CMIP6 datasets are granular by year and in NETCDF (.nc) format.

Species Distribution Data

The occurence datasets for our bird species come from the Global Biodiversity Information Facility (GBIF). The citations for each dataset is below.

- *Numenius americanus Bechstein, 1812 in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset* `https://doi.org/10.15468/39omei` *accessed via GBIF.org on 2024-06-09.*

- *Setophaga striata (J.R. Forster, 1772) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset* `https://doi.org/10.15468/39omei` *accessed via GBIF.org on 2024-06-09.*

- *Anser albifrons (Scopoli, 1769) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset* `https://doi.org/10.15468/39omei` *accessed via GBIF.org on 2024-06-09.*

- *Haliaeetus leucocephalus (Linnaeus, 1766) in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset* `https://doi.org/10.15468/39omei` *accessed via GBIF.org on 2024-06-09.*

- *Numenius phaeopus subsp. phaeopus in GBIF Secretariat (2023). GBIF Backbone Taxonomy. Checklist dataset* `https://doi.org/10.15468/39omei` *accessed via GBIF.org on 2024-06-09.*

GBIF datasets are in .csv format.

2.2 R Preprocessing

Pre-processing is necessary to transform raw datasets into a form factor that is usable by our SDM. This pre-processing step uses R due to its specialty in working with biological and spatial data.

Libraries

- *dismo*: Methods for species distribution modeling, that is, predicting the environmental similarity of any site to that of the locations of known occurrences of a species.

- *sf*: Support for simple features, a standardized way to encode spatial vector data. Binds to 'GDAL' for reading and writing data, to 'GEOS' for geometrical operations, and to 'PROJ' for projection conversions and datum transformations. Uses by default the 's2' package for spherical geometry operations on ellipsoidal (long/lat) coordinates.

- *raster*: Reading, writing, manipulating, analyzing and modeling of spatial data.

- *sp*: Classes and methods for spatial data; the classes document where the spatial location information resides, for 2D or 3D data.

- *ncdf4*: Provides a high-level R interface to data files written using Unidata's netCDF library (version 4 or earlier), which are binary data files that are portable across platforms and include metadata information in addition to the data sets.

Process

There are two different R scripts, one for preprocessing training data and another for preprocessing prediction data. These are named respectively as `training preprocessing.R` and `prediction preprocessing.R`

2.3 Model Training

Libraries

TODO

Input

TODO

Output

TODO

Process

TODO

2.4 Model Prediction

Libraries

TODO

Input

TODO

Output

TODO

Process

TODO

3 Web Application

3.1 APIs

React

Reactjs was used to created the frontend, and the main component is found under src/App.js. In App.js, there are calls to the backend to gather data for the various components located in the src/components folder. In the components folder, there is BirdInfo.js which returns the summary of the desired bird. There are four graphs: ClimateChart.js, HeatMap.js, PolylineMap.js, and SDMchart.js. Those four files contain the code to display the average temperature and prediction graph, the Trajectory graphs discussed more in the Leaflet subsection, and the png of the SDM output. Also in components folder is PredictionControl.js which contains the code to display the year slider and SSP buttons that change the Climate and SDM Charts. In addition to the components, the sidebar and header are created in App.js which allows users to view different birds. All the styling is written up in src/App.css.

FastAPI

The middleware and backend of the website was developed using python and FastAPI, in the file base.py located in the subfolder backend/app. The FastAPI backend primarily serves the purpose of quickly retrieving data and sending it to the front end as requested, allowing the front end user-interface to remain lightweight and easy to load quickly. FastAPI was used to build the backend due to its ease in creating end points for a frontend application to call, natively supporting features such as delayed requests and an intuitive way of passing arguments to the FastAPI backend. The backend is defined predominantly by functions that handle various requests the front end makes in order to retrieve data to display. These functions and their functions are discussed in greater detail in the following section for RestFUL API.

RestFUL

RestFUL API was used to define how the end points were set up, partially due to built-in compatibility with FastAPI, but also due to its simple and intuitive interface. While RestFUL API primarily defines 4 primary methods for GET, PUT, POST, and DELETE, the final application was operational with a stateless backend, meaning only GET and PUT requests were used. In the file base.py, GET and PUT requests were used to define the following end points:

- `get_temperature_data()`: a GET end point that in turn performs a POST request to grid2.rcc-acis.org to obtain temperature data for the state of California throughout a year, which is specified per the user's request.

- `get_precipitation_data()`: a GET end point that in turn performs a POST request to grid2.rcc-acis.org to obtain precipitation data for the state of California throughout a year, which is specified per the user's request.

- `get_predictions()`: a PUT end point that retrieves the pre-computed predictions the Species Distribution Model makes about a specified bird, for a specified year based on specified emission data. While a GET request would have sufficed for its current utility, a PUT request was preferred as this could be expanded to implement a machine learning model to generate predictions dynamically should such a feature be planned in the future.

- `get_trajectory_data()`: a GET end point that retrieves a csv file containing the migration trajectory of a specific bird, identified by its unique bird ID, that can be used to generate individual trajectory maps of an individual bird. The trajectory information is discussed later in the subsection for Leaflet.

- `get_bird_ids()`: a GET end point that returns all unique bird IDs per a given species. This is used to by the front end component PolylineMap.js to search for a specified bird ID.

- `get_heatmap_data()`: a GET end point that retrieves heatmap data of a bird species, to display the aggregated paths of all birds catalogued for that species on the front end with HeatMap.js, as a heat map. This is discussed in further detail in the Leaflet subsection.

Leaflet

Leaflet was used to map bird migration patterns in two ways under the "Trajectory" component. The first map, "Individual Path", is created in src/components/PolylineMap.js. The function fetches data from the csv files under the data folder in backend/app. The function first fetches all the tagged Bird IDs from the csv for the desired bird. Then, all the trajectories for the first Bird ID are fetched from the backend and drawn on OpenStreetMap using Leaflet. Arrows are calculated between two points in calculateBearings function and added onto the map as well. The second map, "Aggregated Path", is created in src/component/HeatMap.js. The function fetches all the latitude and longitude coordinates from the csv of the desired bird in the backend/app/data folder. Using Leaflet and the Leaflet plugin, L.heatLayer, a heat layer is generated showing the combined trajectory path of all the tagged birds on top of OpenStreetMap layer.

3.2 Usage

The website is designed to serve as an interactive tool for ornithologists, researchers, and bird watching enthusiasts. It allows users to:

- Retrieve detailed information about various bird species, including their scientific names, general characteristics, and migration patterns.
- View and interact with climate data visualizations to understand how different climate variables such as temperature and precipitation affect bird migration.
- Predict future distributions of bird species using species distribution models (SDM) based on selected climate scenarios and years.
- Explore bird migration trajectories either as individual paths or aggregated heatmaps, providing a visual understanding of migration routes and densities.

Users can interact with various controls to select different birds, years, and emission scenarios, and the application dynamically updates the information and predictions displayed.

3.3 Design

The website is developed using a React frontend and a FastAPI backend, leveraging modern web technologies and frameworks to ensure a responsive and intuitive user experience. Key design features include:

- **Modular Design:** The application is structured into several independent components like BirdInfo, SDMChart, and PolylineMap. Each component encapsulates a particular feature of the web app, facilitating seperation of concerns, readability, and scalability.
- **Data-Driven Interactions:** By integrating with external APIs and processing data through Python-based FastAPI, the application provides real-time data interactions and updates.
- **User-Centric Interfaces:** Emphasis on usability and accessibility, with interactive charts, maps, and dynamic controls that cater to both novice users and experienced researchers.

The application's architecture is designed to handle high volumes of data efficiently, utilizing caching and asynchronous data fetching to optimize performance and user experience.