Eco-tourism Project-GSOC 2025

Mariam Iftikhar

2025-03-23

Contents

Project information

- Project title: Ecotourism Data Package: Tourism Records and Endangered Wildlife Reports
- Project short title (30 characters): Ecotourism Data Package
- URL of project idea page:

Bio of Contributor

I am a graduate student pursuing MS in Business Analytics at Roosevelt University, Chicago, while also working as a Graduate Assistant for the Heller College of Business for almost a year. My academic knowledge combined with real-world experience has sharpened my analytical and technical skills, which are crucial for this project.

I started my professional journey as an intern and later transitioned into a full-time Business Intelligence Analyst at a Canada-based analytics and consulting company. Over the two and a half years, I created more than 25 Power BI reports and utilized various other technologies for clients both locally and internationally across various industries, including FMCG, Healthcare, Financial Services, EdTech, and E-commerce. These projects not only helped identify business opportunities and risks but also assisted clients in cutting costs and making better decisions.This experience solidified my belief that **"correct decisions taken at the right time can rule the entire world."**

My educational background has been the backbone of my professional growth. During my master's program, I've broadened my technical expertise by diving into AWS, Tableau, Python, and version control, with a special focus on R—covering everything from coding and developing R packages to working with R Markdown. Along the way, I've tackled a variety of assignments and completed certifications that strengthen my qualifications for this project.

Additionally, I've successfully completed evaluation projects that showcase my ability to manage ecotourism-related datasets, which aligns perfectly with what this opportunity requires. With my strong analytical skills, hands-on experience with R, and a proven track record of working with large datasets, I truly believe I'm a great fit for this project.

Contact Information

- Contributor name: Mariam Iftikhar
- Contributor postal address:
- Telephone(s): +1 (773) 410-7087
- Email(s): mariamiftikhar127@gmail.com
- Other communications channels: https://www.linkedin.com/in/mariamiftikhar/

mariamiftikhar97@gmail.com

Contributor affiliation

- Institution: Roosevelt University
- Program: Business Analytics

- Stage of completion: Graduate
- Contact to verify:

Schedule Conflicts

I'm dedicated to giving this project like a full-time job throughout the GSoC period. I don't expect any major conflicts, but I'll make sure to plan ahead just in case any small commitments come up.

Mentors

Evaluating mentor name and email: Di Cook (dicook@monash.edu)

Co-mentor name(s) and email(s): Lyn Cook (l.cook@uq.edu.au)

Have you been in touch with the mentors? When and how?

I've emailed the mentors to share my enthusiasm for the project before 24th March, 2025.

Coding Plan and Methods

Describe in detail your plan for completing the work. What functions will be written, how and when will you do design, how will you verify the results of your coding? The sub-section headings below are examples only. Each project is different, please make your application appropriate to the work you propose.

Project Execution Plan

1. Data Acquisition, Preparation, Standardization & Validation

Timeline: Weeks 1–5

Functions to be Developed:

- fetch_data(): This function pulls raw data from three different databases.
- clean_data(): It takes care of missing values, formats timestamps, and standardizes spatial attributes.
- subset_data(): This filters records based on specific criteria we set.

Design Approach:

Start by establishing connections to the databases and writing queries to get the data we need. Then, develop a preprocessing pipeline to make sure everything is consistent.

Verification:

Conduct exploratory data analysis (EDA) to spot any anomalies and cross-check a sample of records with the original source data.

2. Spatial and Temporal Matching

Timeline: Weeks 6–7

Functions to be Developed:

- match_spatially(): This assigns records to their geographic locations using spatial joins.
- match_temporally(): It aligns records based on timestamps and the intervals that will be defined.

Design Approach:

Utilize geospatial libraries (like `sf` in R) for the spatial operations and implement time-based joins to match records within specific time frames.

Verification:

Plot some sample spatial matches on maps and manually inspect test cases to validate the temporal alignment.

3. Data Structuring and Table Generation

Timeline: Weeks 8–9

Functions to be Developed:

- generate_summary_tables(): This creates summary tables from the matched records.
- aggregate_statistics(): It computes key metrics such as counts, averages, and trends.

Design Approach:

To ensure that the tables are structured for efficient querying and format the outputs for easy integration into reports.

Verification:

Compare the aggregated statistics with the distributions of the raw data and validate the results against expected outcomes using the dataset.

4. Documentation and Vignette Writing

Timeline: Weeks 10–11

Tasks:

- Document all functions with clear explanations and examples.
- Write a vignette that showcases real-world use cases.

Verification:

Contact the mentors to review the documentation and test the vignette code in a fresh R session.

5. Package Testing and CRAN Submission

Timeline: Weeks 12–14

Tasks:

- Set up unit tests using the testthat package.
- Debug any issues and optimize the code for better performance.
- Get the package ready and submit it to CRAN.

Verification:

- Run automated tests for every function to catch any problems.
- Review the CRAN submission logs for any errors or warnings that might pop up.

Describe perceived obstacles and challenges, and how you plan to overcome them.

As we dive into developing the Ecotourism Data Package, we might encounter a few bumps along the way. Here's a rundown of the main challenges we could face, along with some strategies to keep everything on track:

1. Data Acquisition and Integration:

Challenge:

Pulling records from three different databases can lead to inconsistencies in data formats, missing values, or limitations with APIs.

Mitigation Strategy:

- Start with an initial data exploration phase to get a handle on the different data formats.
- Set up automated processes for data validation and cleaning to tackle any missing or inconsistent data.
- Use caching mechanisms to cut down on API calls and ensure that results can be reproduced easily.

2. Spatial and Temporal Matching:

Challenge:

Accurately matching wildlife sightings with tourism records across different locations and times can be tricky, especially with sparse data or inaccuracies in geolocation.

Mitigation Strategy:

- Employ spatial joins and proximity-based matching to effectively link the datasets.
- Use time-windowing techniques to connect sightings with the relevant tourism data.
- Conduct exploratory data analysis (EDA) to uncover and address any potential biases in the matching process.

3. Package Development & CRAN Submission:

Challenge:

Making sure the package meets CRAN's strict requirements, such as code structure, documentation, and testing.

Mitigation Strategy:

- Stick to best practices in R package development, ensuring that DESCRIPTION and NAMESPACE files are properly maintained.
- Regularly run devtools::check() and R CMD check to catch any potential issues early on.
- Keep the documentation current with roxygen2 and create a comprehensive vignette for users.

4. Time Management & Scope Control:

Challenge:

Juggling multiple project tasks while keeping everything on schedule.

Mitigation Strategy:

- Set clear milestones and deliverables for each phase of the project.

- Utilize GitHub project boards and issues to monitor tasks and progress.
- Schedule weekly check-ins with mentors to tackle any roadblocks as they arise.

5. Ensuring Code Quality & Testing:

Challenge:

Creating a solid package that features thoroughly tested functions and dependable outputs.

Mitigation Strategy:

- Use unit tests with testthat for the essential functions.
- Establish a continuous integration (CI) pipeline through GitHub Actions to streamline the testing process.
- Validate accuracy by comparing results against known datasets.

By tackling these challenges head-on with organized mitigation strategies, I'm committed to achieving a successful and well-documented project outcome.

Timeline

Provide a detailed timeline of how you plan to spend your summer.

Week	TimeFrame	Objectives	Tasks
Week 1	May 8 - June 1	Community Bonding Period	Dive into research on datasets, set up a GitHub repository, and chat with mentors about the project roadmap
Week 2	June 2 - 9 June	Data Cleaning & Acquisition	Pull raw data from three different databases and transform it into structured formats.
Week 3	June 9 - 16 June	Data Standardization	Clean up the data by removing duplicates, addressing any missing values, and standardizing timestamps and geolocation information.
Week 4	June 16 - 23 June	Data Organization	Break down the data according to specific requirements and sort it into categories.
Week 5	June 23 - 30 June	Data Validation along with documentation	Check the integrity of the data and document all the cleaning and processing steps taken.
Week 6	June 30 - July 7	Implementing Spatial Matching	Set up spatial matching functions and align the geolocation data accordingly.
Week 7	July 7 - July 14	Implementing Temporal Matching	Create temporal matching functions and establish the criteria for matching.
Week 8	July 14 - July 21	Data Merging & Summarization	Automate the process of merging data and flag any mismatches or inconsistencies
Week 9	July 21 - July 28	Summary Tables & Data Validation	Produce summary tables and verify the accuracy of the matches.
Week 10	July 28 - August 4	Code Documentation	Use roxygen2 to document the functions and provide clear examples of how to use them.
Week 11	August 4 - 11 August	Creating Vignette	Write a vignette that includes use cases and develop a case study.
Week 12	August 11 - 18 August	Internal Testing & Modifications	Perform unit testing, fix any bugs, and optimize the code for better performance

Week	TimeFrame	Objectives	Tasks
Week 13	August 18 - 25 August	Final Testing & CRAN Preparation	Go through the CRAN checklist, debug any issues, and refine the documentation.
Week 14	August 25 - 1 September	CRAN Submission	Submit the package to CRAN and respond to any feedback from reviewers.

What is your contingency plan for things not going to schedule? We understand things change, but how are you planning to address changes and setbacks?

To make sure this project wraps up successfully, I've pinpointed some potential risks and laid out strategies to tackle them:

1. Data Availability Issues Risk:

Some databases might have limited access, incomplete records, or inconsistent formats.

Mitigation Strategy:

- Early on, identify alternative data sources.
- If there are delays in access, work with publicly available datasets.
- When necessary, simulate sample data to keep the workflow on track until the real data comes in.

2. Challenges in Data Integration & Preprocessing Risk:

Variations in data formats, missing values, or inconsistencies could slow down the data preparation phase.

Mitigation Strategy:

- Utilize strong data-cleaning pipelines that automatically handle missing values.
- Use standard R packages like dplyr, tidyr, and lubridate to make transformations smoother.
- Keep clear documentation to track any changes made during data preprocessing.

3. Spatial & Temporal Matching Complexity Risk:

Some records might not have exact geographic coordinates or timestamps.

Mitigation Strategy:

- Apply geospatial techniques like nearest-neighbor matching and spatial interpolation.
- Use external sources (like OpenStreetMap) to help infer locations.
- If needed, collaborate with mentors to fine-tune spatial matching methods.

4. Delays in Function Development & Testing Risk:

Some functions might take longer to optimize, which could impact the overall timeline.

Mitigation Strategy:

- Focus on core functionalities first and then gradually enhance additional features.
- Conduct parallel testing—write unit tests while developing functions.
- Use version control to keep track of changes and easily roll back if necessary.

5. Documentation & Vignette Writing Delays Risk:

If documentation is left until the end, it may end up being rushed and incomplete.

Mitigation Strategy:

- Write documentation progressively as you code.
- Regularly update the README, function descriptions, and vignettes.
- Conduct peer reviews or seek feedback from mentors to improve explanations.

6. CRAN Submission & Compliance Issues Risk:

There's a chance the package might not meet CRAN's submission standards right off the bat.

Mitigation Strategy:

- Start by following the best practices for R package development.
- Utilize tools like devtools::check(), lintr, and roxygen2 to ensure everything is up to par.
- Run mock submissions and tackle any feedback before making the final submission to CRAN.

7. Unexpected Personal or Schedule Conflicts Risk:

Unforeseen commitments could throw a wrench in the planned development schedule.

Mitigation Strategy:

- Embrace an agile approach—get as much work done as possible in the early weeks to create a buffer.
- Keep your work hours flexible to handle any unexpected changes in the schedule.
- Stay in touch with mentors and communicate any necessary adjustments proactively.

By putting these contingency plans into action, I'm committed to keeping the project on track and delivering a top-notch ecotourism data package within the GSoC timeline.

Management of Coding Project

How do you propose to ensure code is submitted / tested?

To make sure that the code is submitted and tested properly, I'll stick to a clear workflow:

- **Version Control & Commit Workflow:** I'll manage all the code through GitHub, making regular commits that come with detailed messages and will create feature branches for any new functionality before merging them into the main branch.
- **Automated Testing & Continuous Integration (CI):** Setting up a CI pipeline using GitHub Actions to automatically run tests with every commit, ensuring everything is stable before merging.
- **Use of GitHub for Code Review & PRs:** Code changes will go through a review process via pull requests (PRs), allowing for peer feedback and insights from mentors before they're merged into the main branch.
- **Unit Tests & Code Quality Checks:** Every function in the package will have unit tests to ensure they work correctly and using tools like R CMD check and lintr to keep the code quality high and maintain a consistent style.

How often do you plan to commit? What changes in commit behavior would indicate a problem?

Commit Schedule

I aim to commit code at least 3 times a week to keep the momentum going. Each commit will be small, self-contained, and thoroughly documented.

Indicators of Potential Issues:

- *A sudden drop in commits:* This might suggest I'm facing challenges with implementation and could benefit from some guidance from my mentor.
- *Infrequent commits:* This could indicate that I'm not making incremental progress and might run into integration problems.
- *Frequent bug fixes on the same issue:* This could point to deeper design flaws or misunderstandings in how I'm implementing things.
- *Unexplained delays in PR reviews:* This might mean there are blockers or that I need to have more discussions with my mentors.

By sticking to a consistent commit routine and tracking my progress on GitHub, I'll make sure the development process runs smoothly.

Test

Describe the qualification test that you have submitted to you project mentors. If feasible, include code, details, output, and example of similar coding problems that you have solved.

As part of my qualification test for the Ecotourism Data Package, I tackled two important tasks to showcase my skills in data retrieval, spatial analysis, and visualization using R. These tasks involved diving into real-world ecological data, utilizing geospatial processing techniques, and employing data visualization methods to draw out meaningful insights.

Project 1: Identifying Weather Patterns Near Platypus Sightings in Victoria, Australia

Objective:

The aim of this task was to gather occurrence data for the Platypus (Ornithorhynchus anatinus) from the Atlas of Living Australia (ALA) and create a visual representation of where these sightings occurred in 2024.

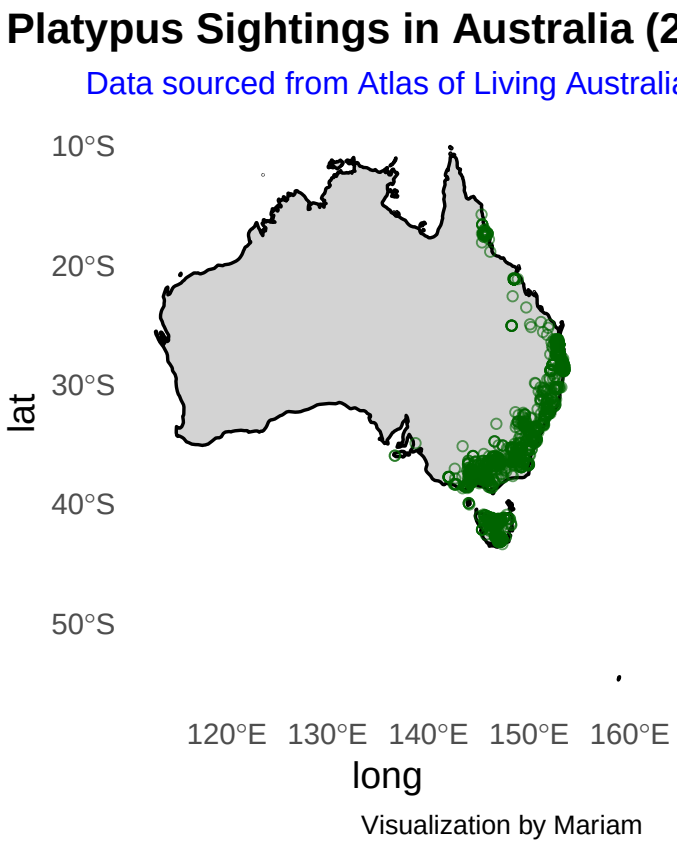
Code:

```
# Set up Galah configuration with your email
galah_config(email = "mariamiftikhar127@gmail.com")

# Search for Platypus occurrences
platypus_taxon <- search_taxa("Ornithorhynchus anatinus")  # Platypus scientific name
```

```r
# Applying filter for 2024
platypus_data <- galah_call() |>
  galah_identify(platypus_taxon) |>
  galah_filter(year == 2024) |>
  galah_select("decimalLatitude", "decimalLongitude") |>
  atlas_occurrences()
```

```
## Request for 1650 occurrences placed in queue
## Current queue length: 1
## Downloading
```

```r
platypus_sf <- platypus_data %>%
  filter(!is.na(decimalLatitude) & !is.na(decimalLongitude)) %>%
  st_as_sf(coords = c("decimalLongitude", "decimalLatitude"), crs = 4326)

# Plotting the graph
ggplot() +
  borders("world", regions = "Australia", fill = "lightgray", colour = "black", size = 0.6) +
  geom_sf(data = platypus_sf, aes(geometry = geometry),
          color = "darkgreen", alpha = 0.6, shape = 21) +
  labs(
    title = "Platypus Sightings in Australia (2024)",
    subtitle = "Data sourced from Atlas of Living Australia",
    caption = "Visualization by Mariam"
  ) +
  theme_minimal(base_size = 14) +  # Larger base font for better readability
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5, size = 16),
    plot.subtitle = element_text(hjust = 0.5, size = 12, color = "blue"),
    plot.caption = element_text(size = 10, hjust = 1, color = "black"),
    panel.grid = element_blank()  # Remove unnecessary grid lines
  )
```



Approach & Implementation:

- Set up and configured the necessary packages: galah, ggplot2, sf, rnaturalearth, and ggspatial to manage data retrieval, spatial processing, and mapping.
- Using galah_config(), I accessed the ALA API and pulled in the Platypus occurrence records for 2024.
- Cleaned up the dataset by removing any missing latitude and longitude values and transformed it into a spatial format (sf object).
- Then, plotted the sightings on a base map of Australia with ggplot2, adding enhancements like:Geospatial layers showing Platypus locations as green points using minimalistic theme to keep it clear and straightforward.

Outcome:

This project produced a weather dataset linked to Platypus sightings, paving the way for deeper analysis of how climate factors influence their distribution. The insights gained can play a significant role in ecological studies and conservation efforts.

Project 2: Mapping Platypus Sightings in Australia (2024)

Objective:

The goal of this task was to utilize GSODR to gather daily weather data—specifically temperature and precipitation—from a weather station in Victoria, Australia, situated close to areas where Platypus sightings are frequent.

Code:

```
load(system.file("extdata", "isd_history.rda", package = "GSODR"))

# Set up Galah configuration with your email
galah_config(email = "mariamiftikhar127@gmail.com")

# Search for Platypus occurrences
platypus_taxon <- search_taxa("Ornithorhynchus anatinus")  # Platypus scientific name


# Applying filter for 2024
platypus_data <- galah_call() |>
  galah_identify(platypus_taxon) |>
  galah_filter(year == 2024) |>
  galah_select("decimalLatitude", "decimalLongitude") |>
  atlas_occurrences()
```

```
## Request for 1650 occurrences placed in queue
## Current queue length: 1
## Downloading
```

```
platypus_sf <- platypus_data %>%
  filter(!is.na(decimalLatitude) & !is.na(decimalLongitude)) %>%
  st_as_sf(coords = c("decimalLongitude", "decimalLatitude"), crs = 4326)
```

```
# Convert weather station data into spatial format for Australia
stations_sf <- isd_history |>
  filter(COUNTRY_NAME == "AUSTRALIA") |>
  sf::st_as_sf(coords = c("LON", "LAT"), crs = 4326)

# Victoria's  boundary

victoria_sf <- ozmaps::ozmap_states |>
  filter(str_detect(NAME, "Victoria")) |>
  sf::st_transform(crs = 4326)

victoria_stations_data <- sf::st_intersection(stations_sf, victoria_sf)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout
## all geometries
```

```r
distance_matrix <- geosphere::distm(
  st_coordinates(platypus_sf),
  st_coordinates(victoria_stations_data),
)

# Nearest station
nearest_station <- apply(distance_matrix, 1, which.min)

platypus_stations <- platypus_sf |>
  mutate(
    station_id = victoria_stations_data$STNID[nearest_station],
    station_name = victoria_stations_data$NAME[nearest_station]
  )

# Stations with the most Platypus occurrences
Stations <- platypus_stations |>
  group_by(station_id, station_name) |>
  summarise(n = n(), .groups = "drop") |>
  arrange(desc(n))  |>
  slice_head(n = 5)

# Finding data for 2024
weather_data <- GSODR::get_GSOD(years = 2024, station = Stations$station_id)
```

```
## Warning:
## This station, 949999-00194, only provides data for years 1956 to 1963.
## Please send a request that falls within these years.
```

```r
# Exporting data in csv
write.csv(weather_data,"weatherdata.csv")
```

Approach & Implementation:

- Installed and loaded the GSODR and geosphere packages to help with retrieving weather data and performing spatial calculations.
- Extracted Australian weather stations from the isd_history dataset and filtered them to focus on those located in Victoria.
- Then, converted the station coordinates into spatial features and conducted a spatial intersection to keep only those within the state.
- Using Haversine distance calculations (geosphere::distm()), I matched Platypus occurrences with the nearest weather station.
- Identified the top stations with the highest number of Platypus sightings and pulled 2024 weather data for these stations using GSODR::get_GSOD().
- Finally, the resulting dataset in CSV format for further analysis.

Outcome:

The final spatial visualization offers a clear view of Platypus distribution across Australia, aiding researchers and ecologists in understanding their habitat preferences and movement patterns.