The World Checklist of Vascular Plants: A Resource for Global Plant Diversity

Developing interoperable Python libraries for loading, querying, and analyzing ethnobotanicaldata from multiple resources using WCVP database

Authors: Xiaoxuan Yu, Samana Fatima, Vaishnavee Thote, and Maria Martinez

28/02/2025

Abstract

This project develops a Python library within a multi-container Docker setup to manage and analyze data from the World Checklist of Vascular Plants (WCVP). The system integrates MySQL and phpMyAdmin for efficient data management while providing a FastAPI interface for querying and analyzing plant information. Key features include plant retrieval, taxon management, and statistical analysis. To enhance interoperability, the system utilizes unique identifiers, ensuring accurate cross-referencing of plant species across datasets. The containerized architecture ensures portability and scalability, making it a versatile tool for biodiversity researchers.

Introduction

The WCVP, developed by the Royal Botanic Gardens, Kew, is a comprehensive, globally curated database documenting over 1.38 million plant names, including 342,953 accepted species. It serves as a critical resource for biodiversity research, conservation, and policy implementation by integrating data from the International Plant Names Index (IPNI), herbaria, literature, and expert reviews.

Unlike other databases, WCVP undergoes rigorous expert validation, receives dynamic weekly updates, and adheres to international nomenclatural standards. It supports initiatives like the Global Biodiversity Information Facility (GBIF) and has been instrumental in research efforts such as Indonesia's vascular plant checklist.
Despite its strengths, WCVP faces challenges, including gaps in geographic distribution data and regional taxonomic inconsistencies, particularly in tropical ecosystems. To address these, WCVP continues to expand collaborations with botanical institutions, refine taxonomic reconciliation workflows, and enhance data validation. Strengthening these efforts is crucial to maintaining WCVP's role as a foundational global plant database.

This project aims to develop a system that integrates MySQL and phpMyAdmin for efficient data management while also providing a FastAPI interface for querying and analyzing WCVP plant data. Additionally, we will explore and discuss some of the database's limitations, including potential improvements in data accuracy, taxonomic updates, and integration with other biodiversity platforms to enhance its usability for researchers, conservationists, and policymakers.

Methodology

The development of this project followed a structured methodology to ensure efficient data processing, database management, and seamless API interactions. The system was designed with a modular architecture, integrating **FastAPI**, **SQLAlchemy**, and **Pandas** to handle large-scale taxonomic data from the WCVP. The methodology encompasses data processing, database structuring, API development, and deployment strategies. This section details the steps involved in cleaning and importing WCVP data, defining relational database schemas, implementing API endpoints, and deploying the system using containerized environments.

1. **Data Processing:Pandas and SQLAlchemy for Managing Dataset Imports**

The data processing phase involved preparing the WCVP dataset for integration into a **MySQL** database. The process ensured data cleanliness, consistency, and proper structuring for efficient querying and analysis.

1.1 Dataset Analysis and Cleaning:

- Each WCVP dataset was analyzed and cleaned using **Pandas.**
- Duplicates were removed, and missing or inconsistent values were handled by replacing `NaN` values with appropriate defaults (e.g., `"Unknown"` for text fields and `0` for boolean flags).
- Individual columns were checked for duplicate values and inconsistencies to maintain data integrity.

1.2 Database Schema Structuring:

The wcvp_names and wcvp_distributions tables were retrieved directly from the WCVP database, accessible from their website, and subsequently processed and structured to ensure seamless integration into a **MySQL** database while maintaining compatibility with the system's relational model.

- **wcvp_names**: Contains taxonomic and nomenclatural data for plant species
- **wcvp_distributions**: Stores geographic distribution data for plant species.

To enable validation and testing prior to full-scale implementation, the processed data was condensed into a smaller test dataset (**test_data.csv**), containing only a representative subset of the full dataset. This approach ensured efficient testing while maintaining the integrity of the data structure.

2. **Database Design: SQLAlchemy ORM with SQLite/MySQL**

During development, SQLite was used for its simplicity in building data models with SQLAlchemy, making early testing and prototyping easier. Since SQLAlchemy abstracts the database layer, transitioning to MySQL for deployment was seamless. MySQL was chosen for its scalability, robust queries, and structured data management, ensuring better data integrity, concurrency control, and system integration, ultimately enhancing the database's performance for biodiversity research and analysis.

2.1 Data Integration:

Once cleaned, the dataset was structured for integration into the **MySQL** database using **SQLAlchemy** ORM

- **SQLAlchemy** was used to manage database interactions, including table creation, data insertion, and querying.
- A **DbManager** class was implemented to handle database operations, such as creating/dropping tables and importing data.
- The `get_or_create` method ensured efficient handling of database entries by either fetching existing records or creating new ones.

2.2 Data Import into MySQL:

The cleaned dataset was imported into **MySQL** while preserving data integrity and entity relationships:

- Each row of the dataset was processed to:
  - Fetch or create related entities (e.g., Family, Genus, Species, Taxon).
  - Handle date formatting and missing values.
  - Link entities (e.g., linking a plant to its family, genus, and geographic area).
- To preserve data quality, rows missing essential fields ((e.g., `plant_name_id`, `family`, `genus`, `species`) were excluded from the dataset.

2.3 Database Structure and Relationships:

To ensure an organized and efficient representation of plant-related data, an Entity-Relationship (ER) diagram was developed. This diagram offers a visual overview of the database structure, illustrating key entities, attributes, and relationships:

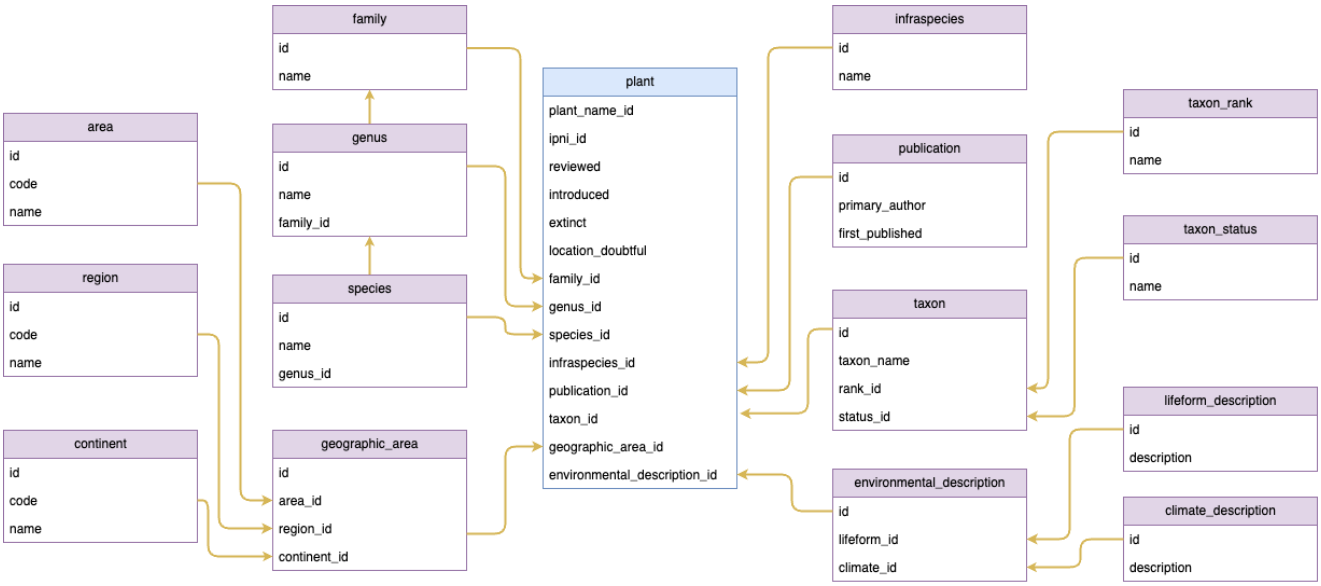

Figure 1: ER Diagram

- Taxonomic classification is represented through hierarchical associations among family, genus, species, and infraspecies.
- Geographical distribution is structured across multiple levels, including continent, region, area, and geographic area.
- Taxonomic attributes such as taxon, taxon rank, and taxon status provide detailed classification information.
- Environmental descriptors, including climate, lifeform, and general environmental conditions, help contextualize plant habitats.

- Infraspecies classification and publication records support further scientific documentation and research validation.

This relational model ensures a comprehensive and structured framework for storing, managing, and analyzing plant-related data efficiently.

2.4 Key Scripts and Libraries:

Category	Name	Description
Key Scripts	`models.py`	Defines the database schema, including table structures, fields, and relationships.
	`manager.py`	Manages core business logic, handling data retrieval, updates, and interactions between the API and the database.
	`test_software.py`	Contains unit and integration tests to validate database operations, ensuring reliability and performance.
Key Tools & Libraries	`Pandas`	Used for data cleaning, transformation, and analysis.
	`SQLAlchemy`	Handles database schema management and data insertion.
	`Python`	Powers the entire data processing pipeline through scripting.

3. **API: FastAPI for Handling Data Requests**

FastAPI is used to expose endpoints for interacting with the database. The API enables users to create, retrieve, update, and delete records efficiently.

3.1 Core Functionality:

- **FastAPI** supports asynchronous processing, improving performance and responsiveness.
- Automatic generation of interactive documentation (Swagger UI) facilitates user exploration of available endpoints.

3.2 Key Components:

Key Component	Description
`main.py`	Launches the **FastAPI** application and defines API routes.
`schemas.py`	Specifies Pydantic models for request validation and response formatting, ensuring data consistency.
`tags.py`	Organizes and categorizes API endpoints for clearer navigation and management.

3.3 API Testing:

- **test_api.py** validates the robustness of API endpoints by checking response correctness, data retrieval, submission functionalities, and error-handling mechanisms.

This API structure ensures efficient and secure user interactions with the WCVP database.

4. **Deployment: VS Code and Docker for Containerization**

The deployment process was designed for scalability, portability, and consistent performance using VS Code and Docker.

4.1 Development Environment:

- VS Code was used as the primary development environment, providing powerful tools for coding, debugging, and version control.

4.2 Containerization with Docker:

To ensure the application runs consistently across different environments, Docker was used for containerization. This approach eliminates dependency issues and simplifies deployment by packaging all necessary components within containers.

- Dockerfile – Defines the runtime environment and necessary dependencies for the application.
- docker-compose.yml – Manages the deployment of multiple containers, including:
  - **FastAPI** application for handling API requests
  - **MySQL** database for structured data management
  - **phpMyAdmin** for an intuitive web-based database interface

4.3 Database Management with phpMyAdmin:

To simplify database interactions, **phpMyAdmin** was integrated as a web-based **MySQL** management tool, providing developers with an intuitive interface to execute SQL queries, inspect and manipulate database records, and debug or optimize data transactions in real time. This enhances database accessibility and streamlines development workflows.

4.4 API Documentation and Testing

The **FastAPI** framework provides interactive API documentation through two interfaces:

- **Swagger UI**: Accessible at `/docs`, this interface allows to test API endpoints in real-time and explore available routes interactively [https://localhost:8080/docs] .
- **MySQL client**: Connect to the **MySQL** database using the command:

```
**MySQL** -u db_user -p db_passwd -h 127.0.0.1 -P 3307
```

- **phpMyAdmin**: Manage the **MySQL** database through the web-based interface at [http://localhost:8081].

By combining VS Code, Docker, FastAPI, and phpMyAdmin, this deployment approach ensures a consistent, scalable, and developer-friendly environment for managing the WCVP database and API.

Analysis and results

System Analysis

The analysis of the taxonomic database system focused on evaluating the performance, data integrity, and usability of the API. Various tests were conducted to assess the efficiency of data retrieval, the correctness of database migrations from SQLite to MySQL, and the effectiveness of the implemented validation mechanisms.

Database Analysis

This section explores key patterns in the WCVP database, including missing values, taxonomic representation, species distribution, and introduced species trends. By assessing data completeness and using visualizations like treemaps and choropleth maps, we identify reliable fields for analysis and uncover trends in biodiversity and species introductions.

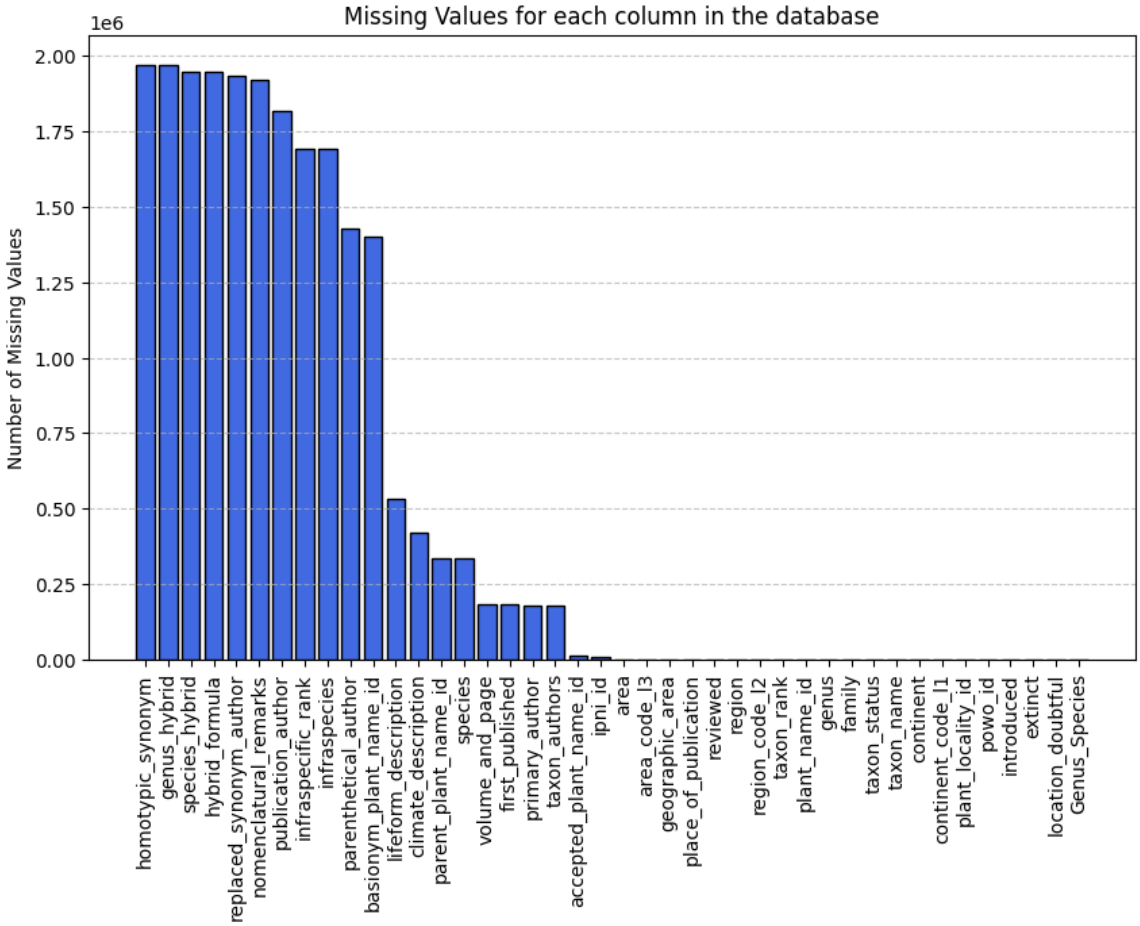

Figure 2. Missing Values in the WCVP Database Variables, Sorted by Highest Frequency.

The graph illustrates the number of missing values for each column in the database, revealing a clear trend. Several columns, appearing at the left of the graph, such as *homotypic_synonym, genus_hybrid, species_hybrid, hybrid_formula, and replaced_synonym_author*, contain the highest amount of missing data, making them less reliable for analysis. Conversely, columns like *taxon_name, genus, family, continent, and region* have significantly fewer missing values, suggesting they are more complete. These columns were prioritized for analysis because well-populated columns will yield more accurate and meaningful results

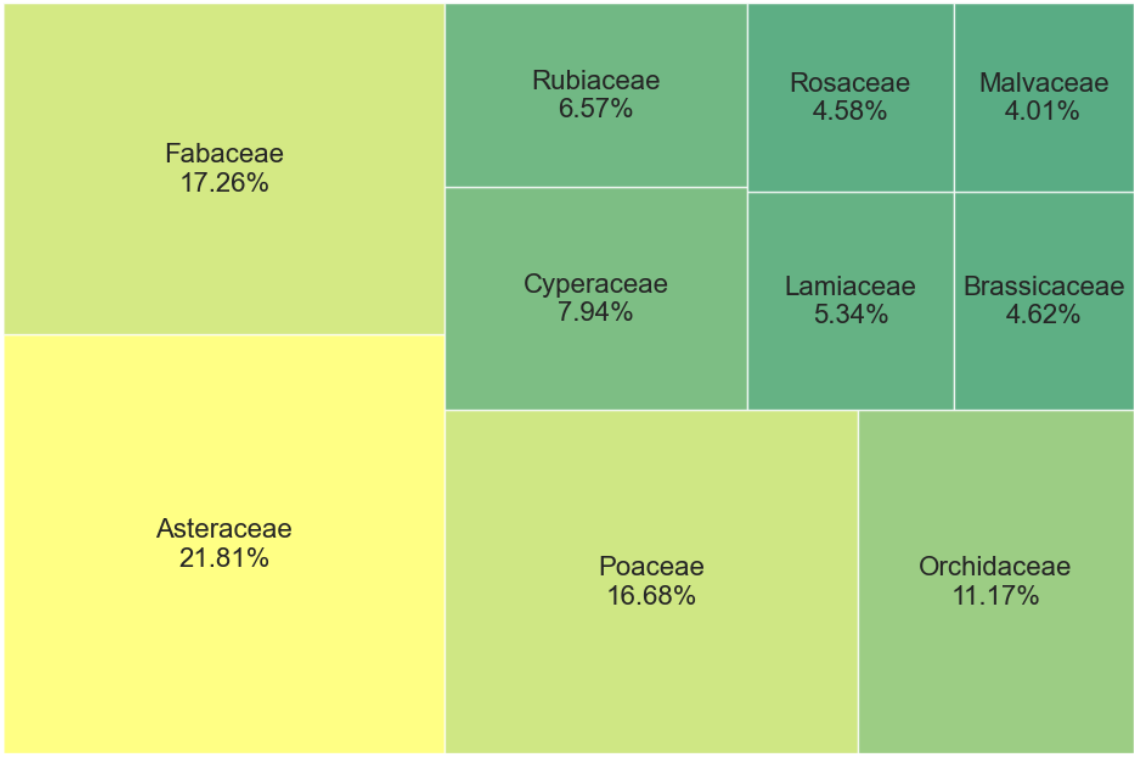

Figure 3. Top 10 Most Frequent Plant Families in the Database.

This treemap provides a hierarchical visualization of the frequency of the top 10 plant families in the WCVP dataset. Each rectangle represents a family, with its size proportional to the number of occurrences in the database. The Asteraceae family appears most frequently, followed by Fabaceae and Poaceae.

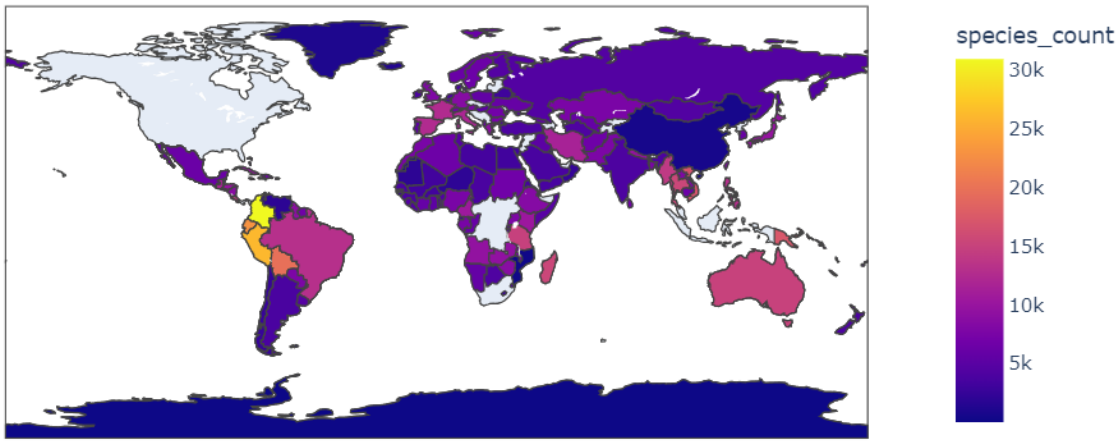

Figure 4. Species Distribution by area of the WCVP database.

The choropleth map illustrates the species distribution by area based on the WCVP database. Regions with higher species counts are represented in yellow and orange, indicating biodiversity hotspots, particularly in parts of South America, Africa, and Australia. Conversely, areas shaded in dark purple have lower species counts. It is important to notice that certain regions appear gray, which suggests a lack of data rather than an absence of species. These gaps might be caused due to data unavailability for those specific locations.

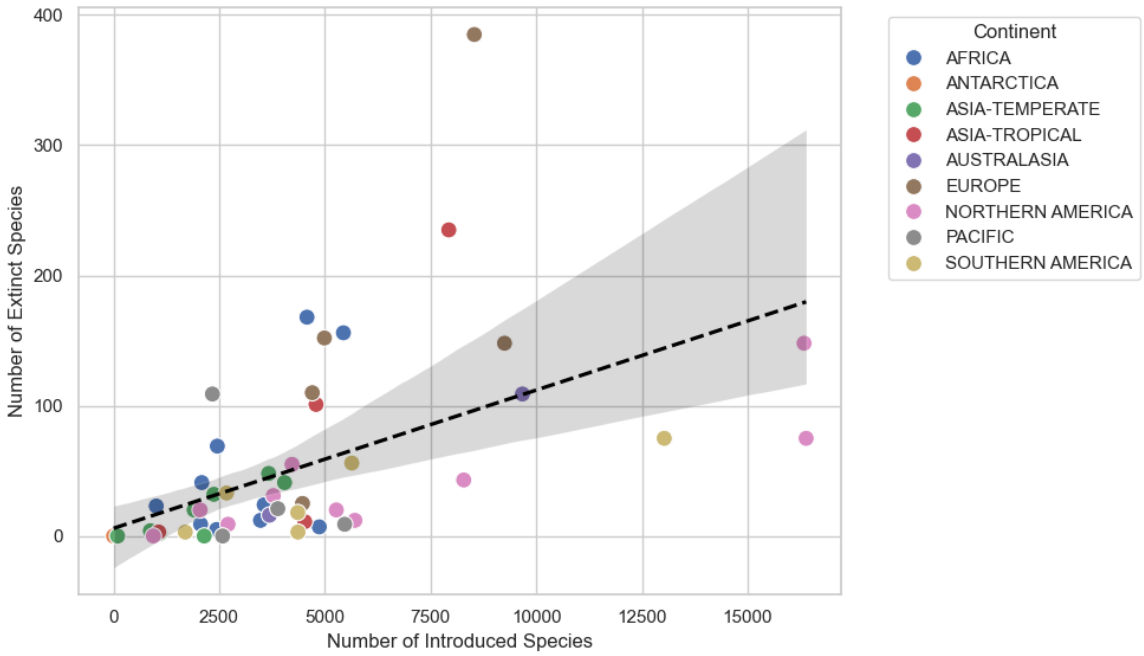

Figure 5. Correlation between the number of introduced species and extinct species across different regions, group and labeled by continent

The positive trend line in the graph suggests a correlation, indicating that continents with a higher number of introduced species tend to have a greater number of extinct species. However, some continents, such as Antarctica, show very few introduced species and minimal extinctions, while others, like Northern America, have a high number of introduced species but a relatively lower extinction count compared to the trend line. The variability in data points suggests that factors beyond introduction alone—such as conservation efforts, ecosystem resilience, and human impact—may influence extinction rates differently across regions.

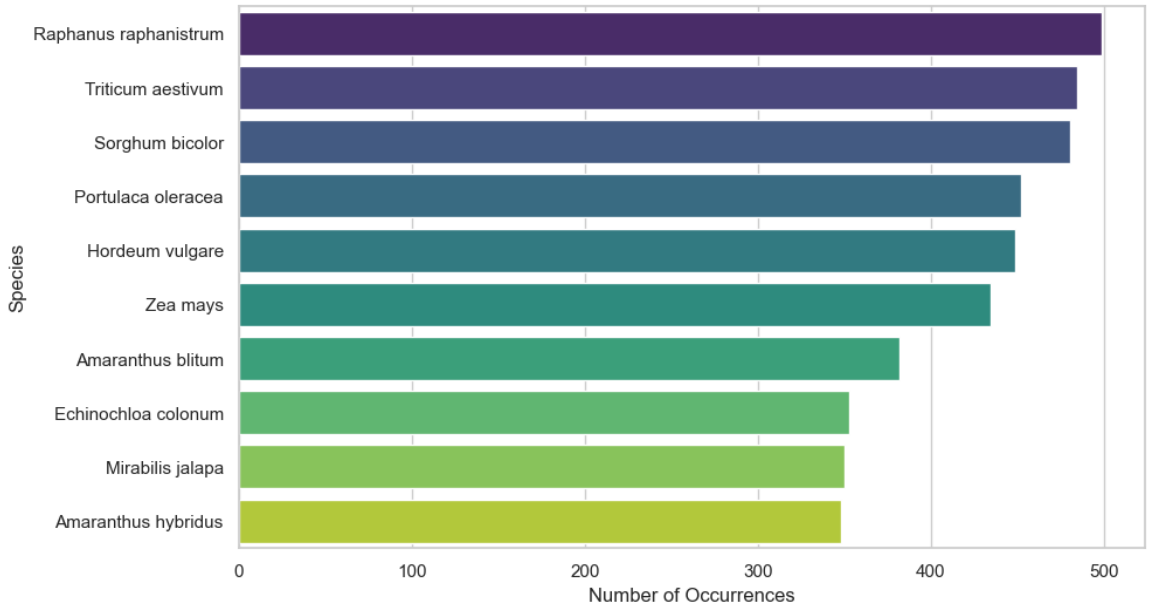

Figure 6. 10 most common introduced species in the dataset, ranked by their number of occurrences.

Raphanus raphanistrum (wild radish) has the highest number of occurrences, closely followed by Triticum aestivum (wheat) and Sorghum bicolor (sorghum) making them the most frequently recorded introduced species in the WCVP database. This chart highlighting a mix of cultivated crops such as Zea mays (maize),

Hordeum vulgare (barley), and Triticum aestivum (wheat), alongside widespread weeds like Portulaca oleracea (purslane) and Amaranthus species. The relatively even distribution of occurrences suggests that these species have successfully established themselves across multiple regions, either due to agricultural practices or their adaptability as invasive plants.

Discussion

The development of this Python library and its integration into a multi-container Docker setup have significantly enhanced data accessibility, interoperability, and usability for researchers working with the WCVP database. By leveraging modern technologies, the project streamlines data management and querying while addressing inconsistencies in taxonomic expertise and species concepts. The implementation of structured data models, unique identifiers, and automated cleaning workflows using Pandas and SQLAlchemy ensures that species concepts are consistently applied, reducing misclassifications and improving taxonomic accuracy. Additionally, the integration of phpMyAdmin and Swagger UI enhances user accessibility, allowing researchers to interact with the database and API without requiring advanced programming skills. These improvements not only strengthen WCVP's reliability but also establish a scalable and robust infrastructure for future data integration, automation, and expansion efforts, ensuring that plant biodiversity research remains precise, accessible, and impactful for conservation and policy-making.

Limitations

Despite its extensive coverage and rigorous validation processes, the WCVP has notable limitations that impact its effectiveness. One major challenge is data accuracy and completeness, particularly regarding geographic distribution and taxonomic consistency. Many plant species, especially those from tropical regions, remain under-documented or misclassified due to gaps in available data and regional taxonomic inconsistencies. Additionally, missing values in key taxonomic columns—such as genus_hybrid (439,847 null values), species_hybrid (431,946 null values), and infraspecific_rank (384,857 null values)—highlight the need for more comprehensive data collection and verification. On the other hand, some species records may not reflect the latest scientific updates, leading to outdated classifications or synonymy issues. These challenges make it difficult for researchers and conservationists to rely fully on WCVP for precise species identification and distribution mapping. Addressing these concerns requires a more dynamic approach to data validation, incorporating machine learning algorithms for automated error detection and improved workflows for taxonomic reconciliation. Regular cross-referencing with national and regional plant databases could further enhance data accuracy and fill existing knowledge gaps.

Another key limitation is the lack of seamless integration with other biodiversity platforms, which restricts its usability for a broader audience, including policymakers and ecological researchers. While WCVP is a critical resource, its potential impact could be significantly expanded by improving interoperability with platforms like GBIF, iNaturalist, and Catalogue of Life. By establishing more robust API connections and data-sharing agreements, WCVP could facilitate real-time taxonomic updates, ensuring that global biodiversity databases remain synchronized and up to date. Additionally, enhancing user engagement through community-driven validation tools—such as crowdsourced taxonomic verification by experts and citizen scientists—could help refine species records more efficiently. These improvements would not only bolster WCVP's reliability but also reinforce its role as a comprehensive reference for biodiversity research, conservation planning, and global policy development.

Conclusion

This project has successfully developed a Python library and multi-container Docker setup to manage and analyze WCVP data, addressing key challenges in biodiversity research. By combining modern tools like **FastAPI**, **MySQL**, and **Pandas**, the system provides a scalable, interoperable, and user-friendly platform for querying and analyzing vascular plant data.

The project's contributions include:

- **Improved Data Accessibility:** Through **phpMyAdmin** and Swagger UI, researchers can easily interact with the database and API.
- **Enhanced Interoperability:** The use of unique identifiers and standardized workflows ensures compatibility with other datasets and systems.
- **Scalability:** The containerized architecture supports future expansion and integration of additional data sources.

By addressing gaps in geographic data coverage, taxonomic expertise, and data integration, the project strengthens WCVP's role as a cornerstone of global plant data infrastructure. It also lays the groundwork for future research, including ethnopharmacological studies and conservation planning.

References

Govaerts, R. (Ed.). (2024). WCVP: World Checklist of Vascular Plants. Facilitated by the Royal Botanic Gardens, Kew [Version 13]. https://doi.org/10.34885/nswv-8994

Govaerts, R., Nic Lughadha, E., et al. (2021). The World Checklist of Vascular Plants, a continuously updated resource for exploring global plant diversity. Scientific Data, 8(215). https://doi.org/10.1038/s41597-021-00997-6

Samadd, Md. A., Hossain, Md. J., Zahan, M. S., Islam, Md. M., & Rashid, M. A. (2024). A comprehensive account on ethnobotany, phytochemistry, and pharmacological insights of genus Celtis. Heliyon, 10(4), e29707. https://doi.org/10.1016/j.heliyon.2024.e297072

Sun, J., Liu, B., Rustiami, H., Xiao, H., Shen, X., & Ma, K. (2024). Mapping Asia plants: Plant diversity and a checklist of vascular plants in Indonesia. Plants, 13(2281). https://doi.org/10.3390/plants13162281

WCVP. (2020). World Checklist of Vascular Plants, version 2.0. Facilitated by the Royal Botanic Gardens, Kew. Published online at http://wcvp.science.kew.org

Acknowledgments

The authors (Xiaoxuan Yu, Samana Fatima, Vaishnavee Thote, and Maria Martinez) extend their sincere gratitude to the WCVP team for their dedication and ongoing efforts in maintaining and enhancing the database. We would like to thank Christian and Fabian for the support and guidance during the course.