corner.py: Visualizing Multivariate Distributions with Corner Plots

Author: Soham Ojha
Date: 27th May 2025

0. Title: corner.py – Visualizing Multivariate Distributions with Corner Plots
   Author: Soham Ojha, May 2025

1. Description of the Package:
corner.py is a specialized Python library used to create corner plots, also known as scatterplot matrices. These plots are crucial in scientific data analysis, particularly in Bayesian inference, where they help visualize posterior probability distributions of model parameters. A corner plot provides both one-dimensional histograms and two-dimensional scatter plots of parameter samples, giving insight into the uncertainty and correlation structure of model fits.

2. Reason for Selection:
I chose corner.py due to its critical role in interpreting the results of probabilistic models and Markov Chain Monte Carlo (MCMC) simulations. It is widely used in astrophysics, statistics, and machine learning to evaluate model outputs. Its clean visualizations and user-friendly API make it a powerful yet simple tool for students and professionals alike.

3. Package History and Related Software:
corner.py was developed by Dan Foreman-Mackey around 2015, inspired by an earlier tool called triangle.py. It is widely accepted in the scientific Python community and often used alongside emcee (MCMC sampler), PyMC3, and other probabilistic programming libraries. The version used in this project is accessed via `corner.__version__`.

4. Maintenance and Contributions:
The package is actively maintained on GitHub (https://github.com/dfm/corner.py). Contributions are welcome, and the project follows best practices in open-source development, including version control, issue tracking, and pull request reviews.

5–6. Installation and Complexity:
corner.py installs easily using pip: `pip install corner`. It has minimal dependencies and integrates seamlessly into any Python data science environment.

7. Source Code Access:
The full source code is publicly available on GitHub at https://github.com/dfm/corner.py. The repository includes the source code, example usage, issue tracker, and documentation.

8. Usage in Other Projects:
corner.py is often used in academic papers and data analysis pipelines. For example, it is used in conjunction with emcee to visualize posterior samples from astrophysical models.

9. How the Code is Used:
The package is intended for use in Jupyter notebooks or Python scripts. It requires passing a NumPy array of sample data and returns a matplotlib Figure object.

10. Example Use:
The Jupyter notebook included in this project demonstrates generating a synthetic dataset using a multivariate normal distribution and plotting the results using corner.py.

11–12. Figures and Plotting:
corner.py leverages matplotlib to produce high-quality figures. In this project, we created two plots:
- A full 6D corner plot to observe correlations among all parameters.
- A 3D corner plot focusing on a subset of parameters for more detail.

Each plot provides a clear visual representation of both marginal and joint parameter distributions, including the true means as reference lines.

13. Language and Dependencies:
corner.py is implemented entirely in Python. Its primary dependencies are NumPy and matplotlib, which are standard in scientific computing.

14–15. Inputs and Outputs:
Inputs to corner.py include a 2D NumPy array representing sample data. Outputs are visual plots of the data, optionally returning the matplotlib axes for further customization.

16–17. Testing and Reliability:
Although formal unit testing is limited, the reliability of corner.py is supported by its simple functionality and widespread usage. It is a visualization tool, so errors are minimal and typically cosmetic or aesthetic.

18. Dependencies:
The package depends on standard Python libraries like NumPy and matplotlib. These are used for numerical computation and visualization, respectively.

19. Documentation:
Documentation is available on GitHub, including installation instructions, example plots, and parameter explanations. The usage examples are clear and sufficient for most use cases.

20. Citation:
Users are encouraged to cite the original paper:
Foreman-Mackey, D. (2016). corner.py: Scatterplot matrices in Python. The Journal of Open Source Software, 1(2), 24.

21–22. References and Citations:

- The official GitHub repository: https://github.com/dfm/corner.py
- Foreman-Mackey et al. (2013), emcee: The MCMC Hammer
- Berta et al. (2021), Bayesian inference in exoplanet detection

23. Learning Experience:
Through this project, I became familiar with concepts of Bayesian inference, multivariate distributions, and visualization of high-dimensional parameter spaces. I learned to simulate data and assess correlation structures visually.

24. Disclaimer:
This project was done independently and represents my first hands-on use of the corner.py library and related techniques. I had no prior experience with this specific tool.

Summary:
The project provided a comprehensive overview of how corner.py can be used to gain insight into complex datasets. The figures generated offer a compact and interpretable visualization of multidimensional distributions, aiding in the analysis and presentation of probabilistic results. This report, the accompanying Jupyter notebook, and the self-contained nature of the code demonstrate a complete application of the package in a typical scientific workflow.

How corner.py Works

corner.py is a specialized visualization tool designed to handle multivariate distributions, often arising in Bayesian analysis or MCMC-based inference problems. It creates what is known as a "corner plot" or "scatterplot matrix," a concise visual summary of marginal and joint parameter distributions.

Key Features:
- **Diagonal plots** display one-dimensional histograms for each parameter.
- **Lower triangle plots** show two-dimensional scatterplots between every pair of parameters.
- **Truth lines** (optional) help compare estimated values with known or expected values.
- **Title annotations** provide statistical summaries like mean and standard deviation.

Internally, corner.py uses matplotlib to construct a grid of subplots. Users supply a 2D NumPy array representing sample values. The array has shape `(n_samples, n_parameters)`, and corner.py automatically generates plots for each parameter and parameter-pair combination. The result is a single figure that gives a complete view of correlations and distributions in a high-dimensional dataset.

corner.py also supports customization, such as:
- Parameter labels
- Titles with statistical summaries
- Overlaying true values

- Color maps and contour levels

This simplicity combined with flexibility makes it ideal for scientific reporting.

Exploring the GitHub Repository

📌 **Repository URL**: https://github.com/dfm/corner.py

This GitHub repository is the official home of corner.py. It provides everything needed for installation, usage, and contributing to the project.

Contents of the Repository:
- **Source Code (`corner.py`)**: Contains the core logic of the plotting tool.
- **Documentation**: Usage examples and API descriptions are clearly outlined in the README.
- **Examples Folder**: Includes ready-to-run code snippets demonstrating typical use cases.
- **Issues Tracker**: Allows users to report bugs or request features.
- **Setup Scripts**: Enables easy pip installation and dependency resolution.

The repository is maintained by Dan Foreman-Mackey, a scientist with extensive contributions to open-source astrophysical tools. Community contributions are welcomed via pull requests, and the repository includes version control and GitHub Actions for continuous integration.

By referencing the GitHub repo, users gain access not only to the package itself but also to its broader scientific context, ongoing development, and community support. The open-source nature and high citation count further validate its reliability in the academic world.

This deeper understanding of how corner.py operates, and how it is supported through its well-maintained GitHub repository, highlights its value as a tool for both research and practical data analysis.