The Journal of Open Source Software

# PyTupli: Enabling Collaboration in Offline Reinforcement Learning

**Hannah Markgraf** [1¶], **Michael Eichelbeck** [1], **Daria Cappey**[1], **Selin Demirtürk**[1], **Yara Schattschneider**[1], **and Matthias Althoff** [1]

**1** Technical University of Munich, Germany ¶ Corresponding author

**DOI:** 10.21105/joss.10144

**Software**

- Review
- Repository
- Archive

**Editor:** Owen Lockwood

**Reviewers:**

- @ab93
- @rahuldevikar

**Submitted:** 10 December 2025
**Published:** 29 June 2026

**License**
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

## Summary

Offline reinforcement learning (RL) offers a powerful way to derive effective decision-making policies for control problems using pre-collected data. However, managing and sharing such datasets in collaborative research settings can be challenging, as proper versioning, filtering, and access control are required. PyTupli is a free, Python-based toolkit designed to support this workflow by providing an easy-to-use yet specialized framework that remains independent of external service providers. Unlike centralized platforms, PyTupli is designed for self-hosted deployment, giving research teams full control over their data infrastructure and access policies. Its client library allows users to serialize and store control problems, upload new data, and retrieve precisely the subsets they need through flexible and expressive filters. Built-in metrics help evaluate dataset coverage and utility, informing both dataset selection and algorithm design for offline RL. To ensure secure deployment, a container-based server component offers authentication, role-based access control, and automated certificate provisioning. Together, these capabilities enable researchers to create, manage, exchange, and analyze datasets for offline RL in a robust and accessible manner.

## Statement of need

Reinforcement learning (RL) provides powerful methods for decision-making under uncertainty, but training RL agents typically requires extensive interaction with the underlying system or a computationally expensive simulator. Offline RL alleviates this requirement by training agents solely on previously collected data (Lange et al., 2012). Such datasets contain tuples of state, action, next state, and reward, obtained from real systems or simulators, which we refer to as tuple datasets.

Effectively managing, sharing, and curating these datasets is essential for collaborative offline RL research, yet existing tools provide only partial support. Platforms like Zenodo or the free version of Hugging Face allow users to share finalized datasets but are not suitable for ongoing or private collaborations. Furthermore, they lack mechanisms for tracking dataset structure or performing efficient queries. Traditional databases are better suited but require expertise to design and maintain workflows.

PyTupli addresses this gap by providing a Python toolkit for creating, storing, and sharing tuple datasets for custom environments. We provide a Docker container that can be deployed independently to maintain control over data. Each dataset is associated with a benchmark, stored as JSON and linked to artifacts such as time-series data, hyperparameters, or trained policies. PyTupli includes integrated user and access management.

Since offline RL performance depends on dataset quality (Asadulaev et al., 2025; Schweighofer et al., 2022; Suttle et al., 2025), PyTupli offers extensive filtering capabilities (Liu et al., 2023;

Younis et al., 2024). PyTupli enables tuple-level filtering, allowing rebalancing datasets or selecting transitions from specific regions of the state space. In addition, PyTupli provides metrics for assessing dataset coverage and reward characteristics, which can predict performance (Asadulaev et al., 2025; Schweighofer et al., 2022) and guide algorithm selection.

## State of the Field

Publicly available tuple datasets have been essential for advancing offline RL algorithms (Kostrikov et al., 2021; Kumar et al., 2020). These collections span domains such as robotics, games (Formanek et al., 2023; Fu et al., 2020; Gulcehre et al., 2020; Younis et al., 2024), power systems (Qin et al., 2022), and autonomous driving (Lee et al., 2024; Liu et al., 2023). As offline RL matures, it is applied to more diverse, task-specific problems. However, no toolbox supports collaborative dataset creation and sharing for custom environments.

Minari (Younis et al., 2024) is the closest related tool, offering standardized datasets and functionality for filtering and recording interactions. However, it focuses on distributing datasets for established benchmarks. Although Minari permits users to request publication of custom datasets on its official platform, this approach is often unsuitable for ongoing or proprietary work or for datasets that evolve over time. PyTupli instead targets collaborative workflows for custom control tasks, enabling teams to share datasets without relying on a central server. Additionally, it provides tools for assessing dataset quality or coverage.

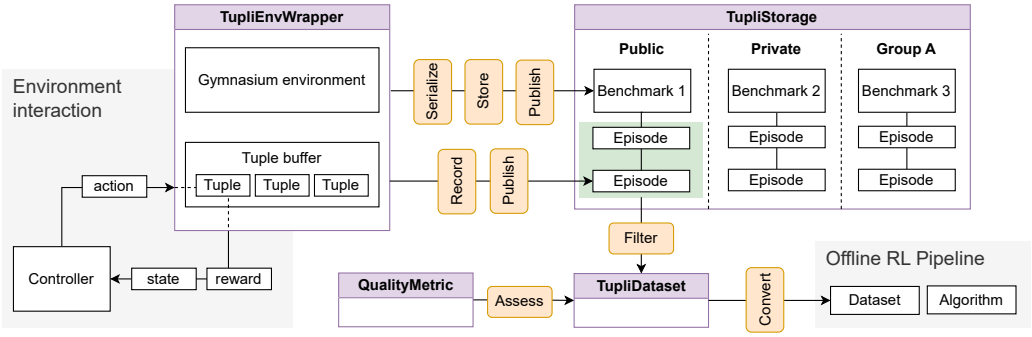


**Figure 1:** Overview of the core functionalities of PyTupli.

## Core Functionalities

We briefly describe the core functionalities of PyTupli which are illustrated in Figure 1.

**Benchmark and Artifact Management**: PyTupli enables users to store any control task defined as a Gymnasium environment. Environments may include parameterizable configurations that produce variations in task dynamics. A fully specified environment is stored as a benchmark, providing a unique reference for reproducible evaluation of controllers. Benchmarks can reference additional data, such as exogenous inputs, time-series data, or pre-trained models. These external units, referred to as artifacts, are stored independently and linked to multiple benchmarks to avoid duplication.

**Data Management**: PyTupli supports ingesting, storing, and querying structured datasets (RL tuples), including their relation to existing benchmark problems and any relevant metadata. MongoDB serves as the backend, providing scalable storage and retrieval. Table 1 shows how ingestion and retrieval times scale with dataset size. Since benchmarking was performed against a locally deployed server, network latency is excluded from all reported times and will add to these figures in practice.

**Multi-User Collaboration and Access Control:** PyTupli facilitates collaborative workflows through private, group, and public scopes. Based on their assigned role, users can store, retrieve, delete, and publish objects. A server-side backend with FastAPI provides a REST interface for secure, programmatic access, while token-based authentication ensures secure sharing across teams or organizations.

**Integration with Existing Offline RL Infrastructure:** An interface to the Gymnasium framework enables users to record interactions with Gymnasium environments as RL tuples. Furthermore, retrieved tuple datasets are made available in a form that can easily be converted into the dataset formats used by existing offline RL libraries such as d3rlpy (Seno & Imai, 2022).

**Assessment of Dataset Quality:** PyTupli exposes dataset quality metrics through its client API, allowing users to assess dataset suitability without leaving the workflow. These metrics are implemented from the offline RL literature and require no additional setup beyond a retrieved dataset.

**Table 1:** Upload and download times for established datasets averaged over 10 runs, measured on a consumer laptop with a local server deployment. We chose two examples with low, medium, and high dataset size from the Minari collection, where $M$ denotes the number of episodes and $N$ the total number of tuples. However, not only the size, but also the nature of observations has a strong influence on processing times.

| Dataset | Size | $M$ | $N$ | Upload (s) | Download (s) |
|---|---|---|---|---|---|
| **D4RL** | | | | | |
| door/human-v2 | 3.5MB | 25 | 7K | 0.83 | 0.24 |
| hammer/human-v2 | 6.2MB | 25 | 11K | 1.11 | 0.40 |
| antmaze/medium-play-v1 | 605.2MB | 1K | 1M | 173.26 | 126.62 |
| **Atari** | | | | | |
| pitfall/expert-v0 | 351.7MB | 10 | 65K | 18.56 | 16.51 |
| **Mujoco** | | | | | |
| ant/expert-v0 | 1.92GB | 2K | 2M | 64.17 | 29.65 |
| humanoid/expert-v0 | 2.95GB | 1K | 999K | 96.61 | 55.32 |

## Quality Metrics

### Return-Based Metrics

PyTupli computes trajectory quality (TQ) (Schweighofer et al., 2022) and average Q-value (Asadulaev et al., 2025), which inform algorithm choice for a given dataset. High TQ favors behavioral cloning, while low TQ favors value-based methods. Estimated return improvement (Swazinna et al., 2021) relates maximum and average returns. Average Q-value operates at the tuple level and is a strong predictor of performance, requiring a Q-function fitted via Bellman updates.

### Coverage-Based Metrics

PyTupli also computes state-action coverage metrics, since low coverage is known to reduce offline RL performance (Schweighofer et al., 2022). Supported metrics include an entropy approximation via unique state-action pairs (Schweighofer et al., 2022) and behavioral entropy (Suttle et al., 2025), which uses density-based weighting of state-action space regions.

## Software Design

PyTupli is designed around a clear separation between a lightweight client library and a centralized server backend. On the client side, PyTupli integrates via a Gymnasium environment

wrapper, which is a well-established abstraction in the RL ecosystem. The server component exposes a REST API that implements functionality specific to offline RL datasets, such as structured storage of benchmarks, episodes, and artifacts, as well as flexible filtering and access control. A centralized service was favored over object storage or Git-based workflows because offline RL datasets require domain-specific querying and metadata handling that these approaches do not natively support. Deployment is simplified via a Docker Compose setup, providing a production-ready stack that can be launched with minimal configuration. Here, the API server is hidden behind an Nginx web server acting as a reverse proxy for increased scalability. We chose MongoDB as the backend due to its ability to store heterogeneous, evolving data without rigid schemas while supporting efficient indexing for nested fields. GridFS enables integrated storage of large artifacts, avoiding external dependencies.

Existing tools such as Minari focus on distributing pre-existing datasets for offline RL. PyTupli instead targets research groups that need to create, host, and curate custom benchmarks collaboratively. This fundamental difference in scope and architecture made contributing to existing projects impractical, motivating the development of new software tailored to this use case.

## Research Impact Statement

Since its release on PyPI in May 2025, PyTupli has been downloaded approximately 65 times per month, suggesting early uptake. The project is actively maintained by two contributors with automated testing and CI/CD pipelines. Documentation is available via Read the Docs, and tutorials provide practical guidance. PyTupli has been integrated into the CommonPower framework, demonstrating applicability in real research settings.

## AI Usage Disclosure

While all core software design decisions within PyTupli were made by the authors, Visual Studio Code with Claude Sonnet 4.5 was used for scaffolding tests. GPT 5.2 was used for refining the manuscript. All AI-assisted outputs were reviewed, edited, and validated by the authors.

## Acknowledgements

This work was partially supported by the German Research Foundation (AL 1185/9-1).

## References

Asadulaev, A., Karray, F., & Takac, M. (2025). Expert or not? Assessing data quality in offline reinforcement learning. *arXiv Preprint arXiv:2510.12638*. https://doi.org/10.48550/arXiv.2510.12638

Formanek, C., Jeewa, A., Shock, J., & Pretorius, A. (2023). *Off-the-grid MARL: Datasets and baselines for offline multi-agent reinforcement learning*. 2442–2444. https://doi.org/10.65109/vmil8420

Fu, J., Kumar, A., Nachum, O., Tucker, G., & Levine, S. (2020). D4RL: Datasets for deep data-driven reinforcement learning. *arXiv Preprint arXiv:2004.07219*. https://doi.org/10.48550/arXiv.2004.07219

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., & others. (2020). RL unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 7248–7259. https://dl.acm.org/doi/abs/10.5555/3495724.3496332

Kostrikov, I., Nair, A., & Levine, S. (2021). Offline reinforcement learning with implicit Q-learning. *arXiv Preprint arXiv:2110.06169*. https://doi.org/10.48550/arXiv.2110.06169

Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *33*, 1179–1191. https://dl.acm.org/doi/abs/10.5555/3495724.3495824

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning: State-of-the-art* (pp. 45–73). Springer. https://doi.org/10.1007/978-3-642-27645-3_2

Lee, D., Eom, C., & Kwon, M. (2024). AD4RL: Autonomous driving benchmarks for offline reinforcement learning with value-based dataset. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 8239–8245. https://doi.org/10.1109/ICRA57147.2024.10610308

Liu, Z., Guo, Z., Lin, H., Yao, Y., Zhu, J., Cen, Z., Hu, H., Yu, W., Zhang, T., Tan, J., & others. (2023). Datasets and benchmarks for offline safe reinforcement learning. *arXiv Preprint arXiv:2306.09303*. https://doi.org/10.48550/arXiv.2306.09303

Qin, R.-J., Zhang, X., Gao, S., Chen, X.-H., Li, Z., Zhang, W., & Yu, Y. (2022). NeoRL: A near real-world benchmark for offline reinforcement learning. *Advances in Neural Information Processing Systems*, *35*, 24753–24765. https://doi.org/10.52202/068431-1795

Schweighofer, K., Dinu, M., Radler, A., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., Eghbal-Zadeh, H., & Hochreiter, S. (2022). A dataset perspective on offline reinforcement learning. *Conference on Lifelong Learning Agents*, 470–517. https://doi.org/10.48550/arXiv.2111.04714

Seno, T., & Imai, M. (2022). d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, *23*(315), 1–20. https://dl.acm.org/doi/abs/10.5555/3586589.3586904

Suttle, W. A., Suresh, A., & Nieto-Granda, C. (2025). Behavioral entropy-guided dataset generation for offline reinforcement learning. *arXiv Preprint arXiv:2502.04141*. https://doi.org/10.48550/arXiv.2502.04141

Swazinna, P., Udluft, S., & Runkler, T. (2021). Measuring data quality for dataset selection in offline reinforcement learning. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–8. https://doi.org/10.1109/SSCI50451.2021.9660006

Younis, O. G., Perez-Vicente, R., Balis, J. U., Dudley, W., Davey, A., & Terry, J. K. (2024). *Minari* (Version 0.5.0). Zenodo. https://doi.org/10.5281/zenodo.13767625