

# 1   epyc: Computational experiment management in Python

2   **Simon Dobson**<sup>1</sup>

3   <sup>1</sup> School of Computer Science, University of St Andrews, Scotland UK

DOI: [10.21105/joss.03764](https://doi.org/10.21105/joss.03764)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Andrew Stewart](#) ↗

## Reviewers:

- [@zbeekman](#)
- [@lorenzo-rovigatti](#)

Submitted: 06 September 2021

Published: 05 November 2021 <sup>11</sup>

## License

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

## 4   Summary

5   epyc is a Python module for designing, executing, storing, and analysing the results of large  
6   sets of possibly long-running computational experiments, as are often found when writing sim-  
7   ulations of complex networks and other domains. It allows the same experimental code to be  
8   run on single machines, multicore machines, and computational clusters without modification,  
9   and automatically manages the execution of an experiment for different parameter values and  
10   for multiple repetitions.

## Statement of need

12   Computation and simulation have joined theory and experiment as a third component of many  
13   scientific fields. Computational experiments share many features with traditional experiments:  
14   they need to be structured, repeated, repeatable, reproducible, and interpretable ([Software  
15   Sustainability Institute, 2020](#)). There is often a need to perform a computation at a number  
16   of points across a multi-dimensional parameter space, aggregating multiple repetitions and  
17   wrangling results for analysis and presentation. Often the experiments being performed are on  
18   such a scale as to require the use of multicore machines and/or computing clusters to perform  
19   multiple experiments simultaneously. The logistics of managing this process can become quite  
20   complicated, and benefits from automation.

21   epyc defines a conceptual framework for performing series of computational experiments, and  
22   implements this in pure Python. Computational “laboratories” perform a collection of “ex-  
23   periments” whose parameters and results are recorded in a “lab notebook” for later retrieval.  
24   Laboratories take a parameter space consisting of any number of individual parameters having  
25   either single values or ranges, and use a computational experimental “design” to determine  
26   which combinations are used in running experiments.

## 27   Main features

28   Modern research software has to operate in a range of computational environments and at a  
29   range of scales. epyc is fully portable to any Python environment, and can operate across the  
30   scales. Laboratories can be sequential (a single core on a machine), parallel (multiple cores  
31   on a machine), or clustered (multiple cores on multiple machines). A key design goal of epyc  
32   is to allow the same experimental code to run under all these configurations, to maximise the  
33   re-use of experiments. Cluster-based laboratories can be accessed asynchronously, allowing a  
34   set of experiments to be started and their results retrieved later without remaining connected  
35   to the cluster: useful for laptop users. epyc also integrates fully into the [Jupyter](#) ecosystem,  
36   allowing notebook-based experimental protocols to still use compute clusters.

Experimental design is a neglected feature of computational science, despite its strong presence in traditional settings (Bailey, 2009). The choice of design translates an experiment and a space of possible parameters into a collection of experimental runs at selected points in that parameter space. Typically choices are to use every combination of parameters (a factorial design) or to take corresponding values from each parameter range (a pointwise design): other designs can easily be added.

epyc defines a lifecycle for experiments through which they are configured to their parameters, set-up, run, and torn-down, before returning the experimental results alongside the experimental parameters and some metadata on the execution of the run, to be inserted into a notebook. A small set of experiment combinators separate the logic of a single experiment from structuring and other tasks. This means that any experiment can be repeated and statistically summarised as a single unit, and also encourages the development of re-usable computational analytics rather than manual handling that can be harder to reproduce.

Lab notebooks can be persistent to store the results of experiments in a non-destructive form, structured as individual result sets with notes and metadata. Notebooks store all the data and metadata in a portable format to improve the reproducibility of computational experiments. Results can be accessed through `pandas` DataFrames for easy wrangling and visualisation. Lab notebooks store their data in `HDF5`, a standard format for large-scale archival storage that augments raw data with annotations and metadata, as well as making results obtained in `epyc` easily portable to other tools.

## Differences with other tools

`epyc` is designed to manage experiments rather than provide sophisticated workflows, since for many application such workflows are largely unnecessary. Instead, `epyc` defines a conceptual model of computational experiments (laboratories, designs, experiments, and notebooks) and supports each component as flexibly and orthogonally as possible.

Unlike `Dask`, `epyc` works independently of libraries such as `numpy` or `scikit-learn` (although it can run experiments using these tools). It also adopts a control- rather than data-driven approach (also unlike `Dask`, and unlike more traditional workflow engines such as `Taverna` which leads to more predictable computational requirements and fits better with the approaches of certain disciplines.

## Main applications

`epyc` arose as part of an on-going research programme in network science and complex systems, as a way to manage the thousands of repeated simulations needing to be performed across large parameter spaces. It provides experiment management for the `epydemic` simulation framework for complex networks, from which it has contributed to a series of papers exploring networks and their processes (for example Mann et al. (2021a), and a book on network-based disease modelling (Dobson, 2020). While originally closely integrated into this system `epyc` is now a stand-alone domain-independent experimental management framework.

## Compatibility and availability

`epyc` works with versions of Python from Python 3.6, and can be installed directly from `PyPy` using `pip`. API and tutorial documentation can be found on `ReadTheDocs`. Source code is available on `GitHub`, where issues can also be reported.

## References

- 79
- 80 Bailey, R. A. (2009). *Design of comparative experiments*. Cambridge University Press.  
81 ISBN: [978-051161-148-3](#)
- 82 Dobson, S. (2020). *Epidemic modelling – some notes, maths, and code*. Independent Pub-  
83 lishing Network. ISBN: [978-183853-565-0](#)
- 84 Mann, P., Smith, V. A., Mitchell, J., & Dobson, S. (2021a). Symbiotic and antagonistic  
85 disease dynamics on clustered networks using bond percolation. *Physical Review E*, 104(2).  
86 <https://doi.org/10.1103/PhysRevE.104.024303>
- 87 Mann, P., Smith, V. A., Mitchell, J., & Dobson, S. (2021b). Two-pathogen model with  
88 competition on clustered networks. *Physical Review E*, 103(6). [https://doi.org/10.1103/](https://doi.org/10.1103/PhysRevE.103.062308)  
89 [PhysRevE.103.062308](https://doi.org/10.1103/PhysRevE.103.062308)
- 90 Pitcher, M., Bowness, R., Dobson, S., Eftimie, R., & Gillespie, S. (2020). Modelling the effects  
91 of environmental heterogeneity within the lung on the tuberculosis lifecycle. *Journal of*  
92 *Theoretical Biology*, 506. <https://dx.doi.org/10.1101/2019.12.12.871269>
- 93 Software Sustainability Institute. (2020). *Guides for researchers*. [https://www.software.ac.](https://www.software.ac.uk/resources/guides/guides-researchers)  
94 [uk/resources/guides/guides-researchers](https://www.software.ac.uk/resources/guides/guides-researchers)