Journal of
open research software

Eerland, W et al 2017 Teetool – A Probabilistic Trajectory Analysis Tool. *Journal of Open Research Software,* 5: 14, DOI: https://doi.org/10.5334/jors.163

SOFTWARE METAPAPER

Teetool – A Probabilistic Trajectory Analysis Tool

Willem Eerland[1], Simon Box[1], Hans Fangohr[1,2] and András Sóbester[1]

[1] University of Southampton, GB

[2] European XFEL GmbH, Holzkoppel 4, 22869 Schenefeld, DE

Corresponding author: Willem Eerland (w.j.eerland@soton.ac.uk)

Teetool is a Python package which models and visualises motion patterns found in two- and three-dimensional trajectory data. It models the trajectories as a Gaussian process and uses the mean and covariance of the trajectory data to produce a confidence region, an area (or volume) through which a given percentage of trajectories travel. The confidence region is useful in obtaining an understanding of, or quantifying, dispersion in trajectory data. Furthermore, by modelling the trajectories as a Gaussian process, missing data can be recovered and noisy measurements can be corrected. Teetool is available as a Python package on GitHub, and includes Jupyter Notebooks, showing examples for two- and three-dimensional trajectory data.

Keywords: Python; Gaussian process; motion patterns; trajectory patterns; confidence region
Funding statement: The authors gratefully acknowledge the funding provided under research grant EP/L505067/1 from the Engineering and Physical Sciences Research Council and Cunning Running Software Ltd. The research data and code generated as part of this study are openly available at https://doi.org/10.5281/zenodo.251481.

(1) Overview

Introduction

In recent years, researchers have been focusing on the analysis of large scale trajectory data sets obtained from surveillance devices, which contains motion patterns of objects [1]. A trajectory here is defined as the path of an object through space. In most applications of interest to scientists and engineers, objects do not move around randomly. Examples are cars on the road, which follow traffic rules, and aircraft near an airport, which follow instructions from air traffic control. Modelling (i.e. create a statistical model of where they may be located in space) and visualising motion patterns is an important step in understanding the behaviour of the trajectory data.

When analysing road traffic or pedestrian environments, trajectory data are generally visualised in two dimensions (usually a planar projection). Makris and Ellis [2] apply a spline-like model to represent the motion patterns found in the data. This model contains a mean function, the average of all trajectories, described by a discrete number of equidistant nodes. Here, the dispersion of the trajectories at each node is calculated by examining the cross section perpendicular to the local node direction, and fitting a univariate Gaussian to the points where the trajectories cut this cross section. The result, seen in **Figure 1**, is an envelope with a spline-like representation, capturing a given percentage of trajectories in a two-dimensional space. Another approach is to replace the mean function with a sequence of discrete Gaussian nodes [3]. The result is similar, as both these methods produce models that have an average trajectory and a surrounding region representing the dispersion (a confidence region related to a percentage of the data).

Motion patterns found in three-dimensional aircraft trajectory data have been visualised as flight corridors, where the data were clustered according to the flightpath [4]. The properties of the flight corridors are such that there is an average trajectory, and a dispersion in vertical and lateral direction, perpendicular to the mean trajectory. Flight corridors have also been generated via Gaussian processes, with the added benefits of being able to handle missing data and noisy measurements [5].

In terms of software tools for trajectory data analysis, there is a Python package named Flowtracks [6] available that handles the storage of trajectory data, but includes no specific algorithm for modelling motion patterns. This package is a database management solution providing a bridge between different formats of trajectory data. Furthermore, there is the Integrated Noise Model [7], a computer model that was used to evaluate aircraft noise impacts in the vicinity of airports, which has a functionality to use weighted tracks to approximate the dispersion found in the trajectory data. The use of weighted tracks is required to keep the computational load of the subsequent noise calculations in check, as otherwise the number of flight trajectories will increase the number of required calculations. However, placing these tracks is left to the user and is not learned from the trajectory data.

Teetool is a Python package that learns the motion patterns found in trajectory data by modelling the motion patterns found in trajectory data via Gaussian processes. This approach enables the recovery of missing data and the handling of noisy measurements. The package takes as input both two- and three-dimensional trajectory data as a function of another value (e.g. time or absolute distance covered). However, as an output the statistical modelling only concerns itself with the spatial characteristics and by default considers the complete range of this value. Furthermore, the package assists in visualising the patterns in two and three dimensions. The modelling methods are *similar* to the one applied to aircraft trajectories in Eerland, Box and Sóbester [5] to identify flight corridors. We have made modifications to their method, notably to reduce the computational cost when the available trajectory data is not noisy, or when there are no parts missing. The calculations in the Teetool package are performed using NumPy [8] and Scipy [9]. The data are visualised in two and three dimensions using Matplotlib [10] and Mayavi [11] respectively. Data are handled via Pandas [12], and examples are included using Jupyter Notebooks [13]. Teetool has already been utilised to model and visualise rocket trajectory data as produced by a stochastic rocket simulator [14, 15].

The remainder of this paper covers the implementation and architecture, quality control, the availability, and reuse potential of the software.

Implementation and architecture

The package is written in Python and consists of seven classes. An overview how these classes communicate with the user is available in **Figure 3**. The user starts by initialising a **World** class and adding trajectory data – this class handles multiple sets of trajectory data. From each set, a model is created with parameters as specified by the user. The specifics of these parameters are described in the documentation and include a selection of the modelling method. Based on these parameters the **GaussianProcess** and **Basis** classes are initialised and called. Note that from a user's perspective the classes **Model**, **GaussianProcess**, and **Basis** are not visible, as these are called via the **World** class, hence there is no direct line going towards these classes starting from the user. The visualisations are presented via the **Visual_2d** and **Visual_3d** classes, for two- and three-dimensional representations respectively. A sequence diagram of the initialization of a **World** object and the modelling via resampling is displayed in **Figure 2**.

An example using two-dimensional trajectory is visible in **Figure 4**. It shows the trajectory data (**Figure 4a**) and the

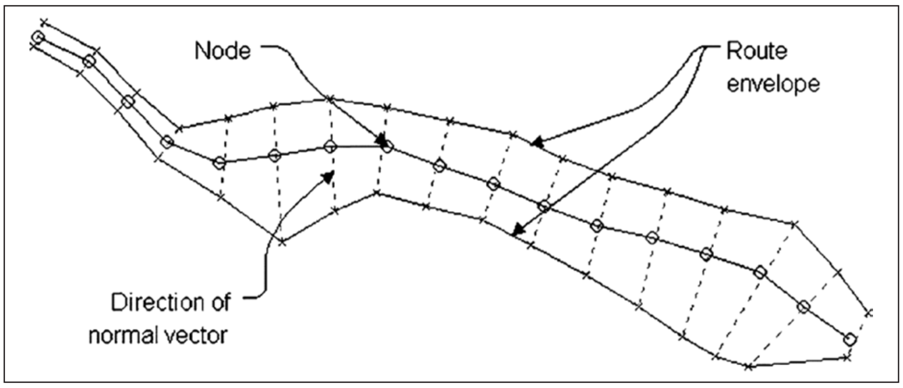

Figure 1: An example of a route model, capturing the main axis and borders of a specific route [2].

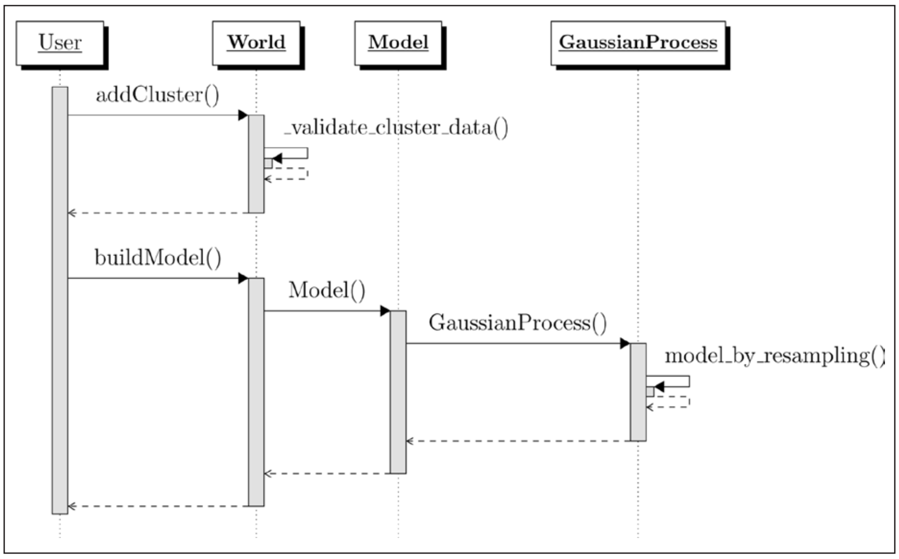

Figure 2: A sequence diagram visualising the chain of events following the initialization of a **World** object and the modelling via resampling.

confidence region corresponding to two standard deviations (**Figure 4b**). The trajectory data in this example is artificially generated using random sampling from a Gaussian distribution, and aims to demonstrate the confidence region in two dimensions. An example using three-dimensional rocket flight trajectories is shown in **Figure 5**. These flight trajectories are generated using a stochastic rocket simulator, and emulates uncertainties in launch and atmospheric conditions [15]. The settings are such that a rocket is launched from position (0, 0, 0) towards the North (positive Northing direction), at a 10 degree declination angle. It shows only 50 trajectories of the 500 used to create the model, and also includes the confidence region corresponding to one standard deviation. The final example shows five flightpaths of aircraft approaching an airport. The three-dimensional view is seen in **Figure 6**, while a two-dimensional slice at a constant altitude of the confidence region (of two standard deviations), is visible in **Figure 7**. All figures shown here are generated using the Jupyter notebooks included in the repository.

Quality control

Unit tests with an overall coverage of 87% are available and executed using `pytest`. Instructions on how to run these tests are included in the README.MD file. Continuous integration has been implemented using Travis CI, and the latest test results are available online at https://travis-ci.org/WillemEerland/teetool. The Jupyter notebooks used to produce **Figures 4** to **7** can be found in the EXAMPLE folder, and provide a demonstration of basic functionality to the user. A code snippet of EXAMPLE/EXAMPLE_TOY_2D.IPYNB is included here:

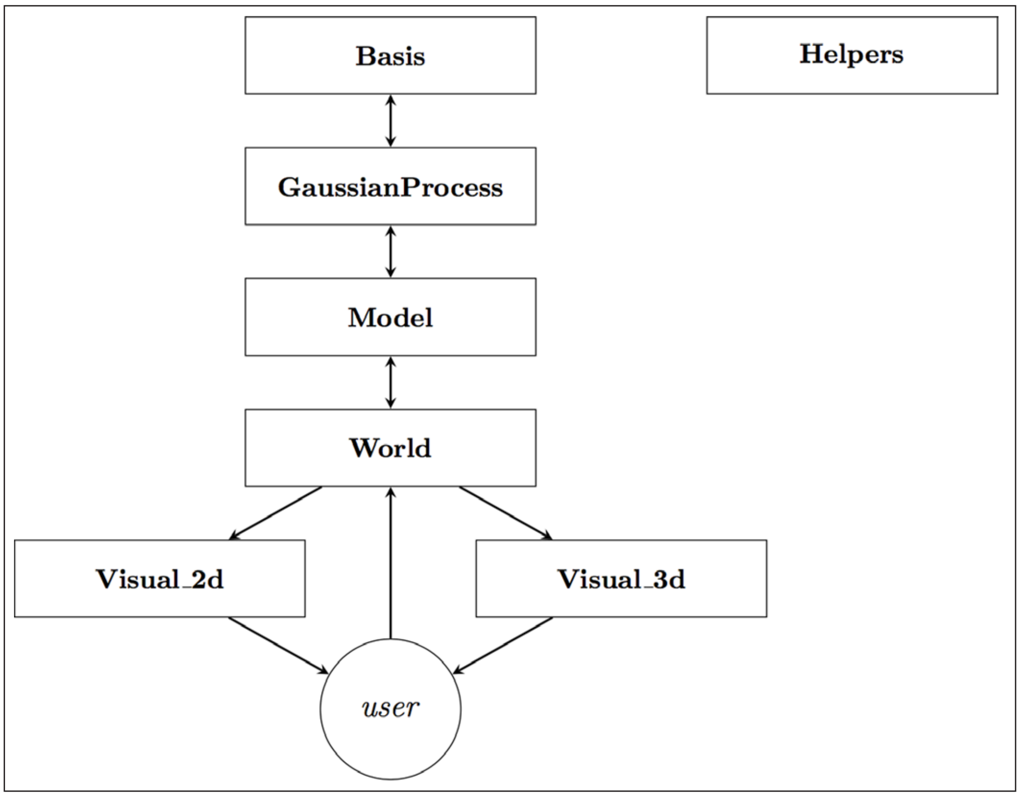

Figure 3: A schematic representation of the seven classes found within Teetool, and the interaction with the user.

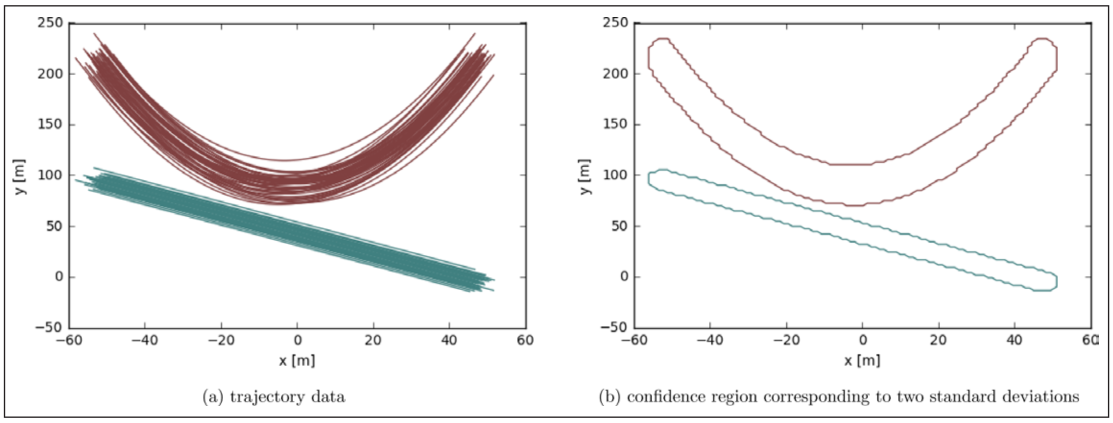

Figure 4: Gaussian distributed artificially generated trajectory data **(a)**, aimed at demonstrating the functionality of the confidence region **(b)**.

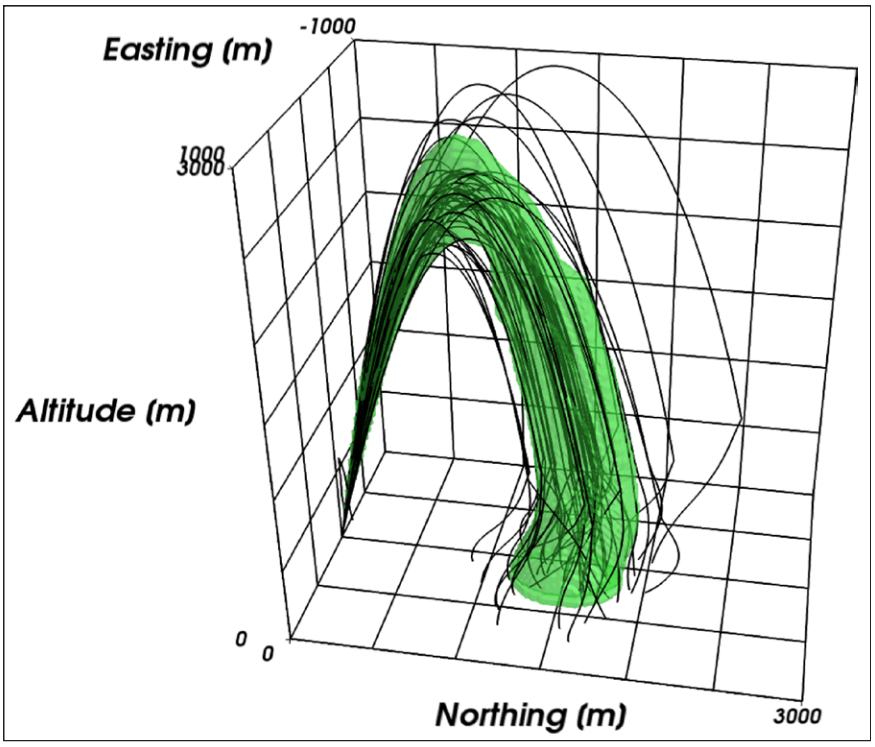

Figure 5: Rocket flight trajectory data (lines) as produced by a stochastic rocket simulator in uncertain launch- and atmospheric conditions, including the corresponding one standard deviation confidence region (shaded volume).

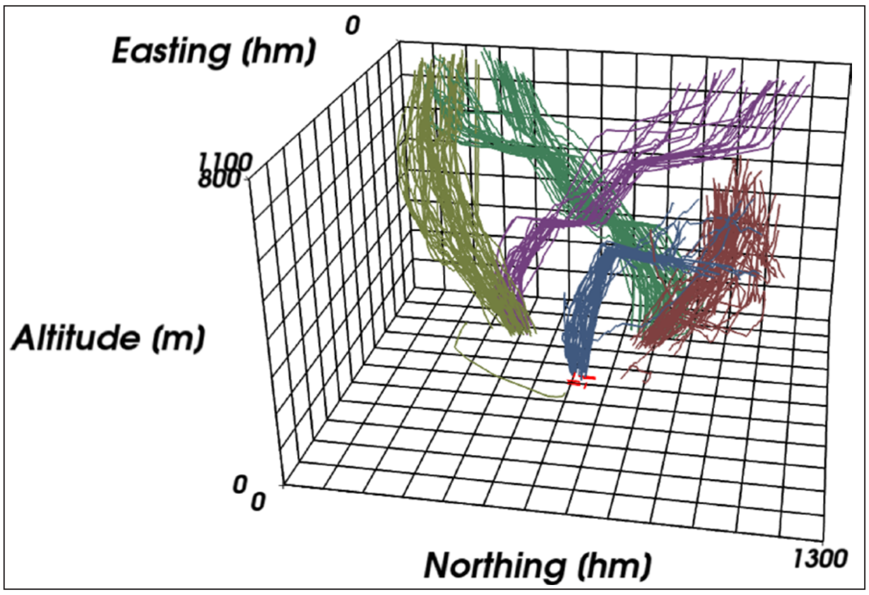

Figure 6: Aircraft trajectory data of approaching aircraft as measured by a ground-based radar at the Dallas/Fort Worth (DFW) airport, bundled in five common flightpaths.

```
# import package
import teetool as tt
# create a world
world = tt.World(name='toy', ndim=2,
resolution=[100, 100])
# add data
world.addCluster(cluster_data=cluster_data_1,
cluster_name='one')
```

where **cluster_data** is a list of **(x, Y)**. Here **Y** is a $[N \times D]$ matrix, representing the multi-dimensional coordinates as a function of the $[N \times 1]$ vector **x**. These are NumPy arrays where N is the total number of data-points and D is the dimensionality of the trajectory data.

(2) Availability

Operating system

Teetool is able to function on any operating system that supports a standard Python installation, which includes Linux, Windows, and macOS.

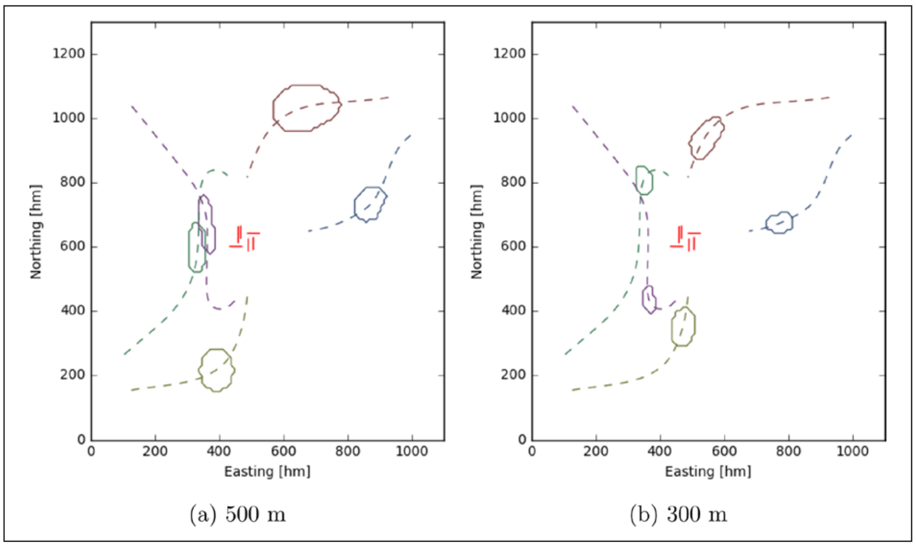

Figure 7: Confidence regions (i.e. flight corridors) at a constant altitude (i.e. a horizontal plane intersection of the flight corridors), and mean trajectories (dashed lines), as generated from the trajectory data seen in Figure 6.

Programming language
Python == 2.7

Additional system requirements
No special requirements.

Dependencies
The following Python libraries are a required dependency:
NumPy ≥ 1.11
SciPy ≥ 0.18.1
The following Python library is a requirement for **Visual_2d class**:
Matplotlib ≥ 1.5.1
The following Python library is a requirement for **Visual_3d class**:
Mayavi ≥ 4.5.0
The following Python library is a requirement for handling the data used in the Jupyter Notebook examples:
Pandas ≥ 0.19.2

List of contributors
1. Willem Eerland (developer)

Software location
Archive
Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.251481
Licence: MIT
Publisher: Zenodo
Version published: 1.0
Date published: 19/01/2017

Code repository
Name: GitHub
Persistent identifier: https://github.com/WillemEerland/teetool/releases/tag/v1.0
Licence: MIT
Version published: 1.0
Date published: 11/01/2017

Language
English.

(3) Reuse potential
The software is capable of modelling (i.e. create a statistical model of where they may be located in space) and visualising the motion trend of any kind of trajectory data. It has already been applied to model the motion patterns of three-dimensional trajectory data in two applications. The first application is to visualise the flight corridors as found in trajectory data as measured by a ground-based radar. In the second application the software has been used to visualise the behaviour of the output trajectory data from a stochastic rocket simulator. Furthermore, examples have been included to analyse two-dimensional trajectories, which are often seen in the field of road traffic analysis.

The modelling methods are accessible via the **World** class, and when desired it is possible to access this class directly and request the raw output of the model. Documentation on how to address each of the classes is available at https://willemeerland.github.io/teetool/, and is automatically generated via doxygen. The software is open-source and available on GitHub (https://github.com/WillemEerland/teetool), which is the preferred channel for any potential contributors to contact the developers. Support is available and provided via GitHub Issues.

Competing Interests
The authors declare that Hans Fanghor is an Associate Editor of the Journal of Open Research Software.

References
1. **Zheng, Y** 2015 'Trajectory Data Mining: An Overview'. In: *ACM Transactions on Intelligent Systems and Technology*, 6(3), pp. 1–41, ISSN: 2157–6904. DOI: https://doi.org/10.1145/2743025

2. **Makris, D** and **Ellis, T** 2005 'Learning semantic scene models from observing activity in visual surveillance', In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(3), pp. 397–408. DOI: https://doi.org/10.1109/TSMCB.2005.846652
3. **Hu, W, Xiao, X, Fu, Z, Xie, D, Tan, T** and **Maybank, S** 2006 'A system for learning statistical motion patterns', In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9), pp. 1450–1464. ISSN: 0162–8828. DOI: https://doi.org/10.1109/TPAMI.2006.176
4. **Salaun, E, Gariel, M, Vela, A E** and **Feron, E** 2012 'Aircraft proximity maps based on data-driven flow modeling', *Journal of Guidance, Control, and Dynamics*, 35(2), pp. 563–577, DOI: https://doi.org/10.2514/1.53859
5. **Eerland, W J, Box, S,** and **Sóbester, A** 2016 'Modeling the Dispersion of Aircraft Trajectories Using Gaussian Processes', In: *Journal of Guidance, Control, and Dynamics*, 39(12), pp. 2661–2672. DOI: https://doi.org/10.2514/1.G000537. eprint: https://eprints.soton.ac.uk/399818/
6. **Meller, Y** and **Liberzon, A** 2016 'Particle Data Management Software for 3D Particle Tracking Velocimetry and Related Applications – The Flowtracks Package', *Journal of Open Research Software*, 4(1), DOI: https://doi.org/10.5334/jors.101
7. **He, B, Dinges, E, Hemann, J, Rickel, D, Mirsky, L, Roof, C J, Boeker, E R, Gerbi, P J** and **Senzig, D** 2007 Tech. rep. Federal Aviation Administration (FAA), *Integrated Noise Model (INM) Version 7.0 User's Guide*, pp. 116–117. URL: https://www.faa.gov/.
8. **Walt, S v d, Colbert, S C** and **Varoquaux, G** 2011 'The NumPy Array: A Structure for Efficient Numerical Computation', In: *Computing in Science & Engineering*, 13(2), pp. 22–30, DOI: https://doi.org/10.1109/MCSE.2011.37
9. **Jones, E, Oliphant, T, Peterson, P,** et al. 2001 *SciPy: open source scientific tools for Python*, URL: https://www.scipy.org (visited on 20/04/2017).
10. **Hunter, J D** 2007 'Matplotlib: A 2D graphics environment', *Computing in Science & Engineering*, 9(3), pp. 90–95, DOI: https://doi.org/10.1109/MCSE.2007.55
11. **Ramachandran, P** and **Varoquaux, G** 2011 'Mayavi: 3D Visualization of Scientific Data', In: *Computing in Science & Engineering*, 13(2), pp. 40–51, ISSN: 1521–9615, DOI: https://doi.org/10.1109/MCSE.2011.35
12. **McKinney, W** 2010 'Data Structures for Statistical Computing in Python'. In: van der Walt, S and Millman, J. *Proceedings of the 9th Python in Science Conference*, pp. 51–56, URL: http://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf.
13. **Kluyver, T, Ragan-Kelley, B, Pérez, F, Granger, B, Bussonnier, M, Frederic, J, Kelley, K, Hamrick, J, Grout, J, Corlay, S, Ivanov, P, Avila, D, Abdalla, S, Willing, C,** et al. 2016 'Jupyter Notebooks — a publishing format for reproducible computational workflows', In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas: Proceedings of the 20th International Conference on Electronic Publishing*. IOS Press. pp. 87–90, DOI: https://doi.org/10.3233/978-1-61499-649-1-87
14. **Eerland, W J, Box, S, Fangohr, H** and **Sóbester, A** 2017 'An open-source, stochastic, six-degrees-of-freedom rocket flight simulator, with a probabilistic trajectory analysis approach', In: *AIAA Modeling and Simulation Technologies Conference*, Texas, USA: American Institute of Aeronautics and Astronautics, 5(1), eprint: https://eprints.soton.ac.uk/403364/. DOI: https://doi.org/10.2514/6.2017-1556
15. **Eerland, W J, Box, S** and **Sóbester, A** 2017 'Cambridge Rocketry Simulator – A Stochastic Six-Degrees-of-Freedom Rocket Flight Simulator', In: *Journal of Open Research Software*, 5(1), eprint: https://eprints.soton.ac.uk/id/eprint/405278. DOI: https://doi.org/10.5334/jors.137

How to cite this article: Eerland, W, Box, S, Fangohr, H and Sóbester, A 2017 Teetool – a Probabilistic Trajectory Analysis Tool. *Journal of Open Research Software,* 5: 14, DOI: https://doi.org/10.5334/jors.163

Submitted: 31 January 2017 **Accepted:** 04 May 2017 **Published:** 17 May 2017