Journal of
open research software

Dawson, A 2016 eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data. *Journal of Open Research Software,* 4: e14, DOI: http://dx.doi.org/10.5334/jors.122

SOFTWARE METAPAPER

eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data

Andrew Dawson[1]

[1] Atmospheric, Oceanic & Planetary Physics, Department of Physics, University of Oxford, Oxford, UK
andrew.dawson@physics.ox.ac.uk

The *eofs* library provides a high-level Python interface for computing empirical orthogonal functions (EOFs) and related quantities, with a focus on correctness and ease of use. The library is implemented in a modular hierarchical fashion, allowing computations using plain arrays, or the inclusion of metadata. The software provides a convenient package for users wanting to perform EOF analysis in Python, and integrates with popular libraries from atmospheric and climate science. The software is available on Github.

Keywords: EOF analysis; Meteorology; Oceanography; Climate; Python

(1) Overview

Introduction

Data sets in meteorology, oceanography, and climate are typically very large, containing data covering large spatial areas, observed or modelled over long periods of time. Studying variability in these data sets can be challenging, with coherent modes of large-scale spatial and temporal variability in the atmosphere-ocean system hidden amongst the noise of smaller scale physical processes. An often used technique for examining large-scale patterns of variability in such data sets is the analysis of empirical orthogonal functions (EOFs) [1]. Decomposing a complex data set varying in time and space into a set of EOFs and associated principal component time series (PCs) can allow insight into the most dominant modes of spatial variability, for example El Niño, one of the leading modes of climate variability, is often characterised by the first EOF and PC of sea surface temperature in the tropical Pacific [2].

The EOFs and PCs of a data set describe a new basis, where instead of a series of spatial observations varying in time, the data set is represented as a set of fixed spatial patterns or modes, which represent a given amount of the total variance in the data set, and a set of time series describing how each pattern changes with time. In typical applications the first few EOFs account for a large portion of the total variance, allowing the study of one or two modes to give insight into the variability present in the data set. The method of analysis is purely mathematical and does not depend on any physical properties of the quantity being analysed.

The process of computing and analysing EOFs and related structures is non-trivial, and highly error prone. For example, consider the computation of EOFs from a time-series of sea surface temperature on a latitude-longitude grid. First one must correctly weight the input data to account for spatial variability in the size of grid cells due to convergence of the meridians. The input data must then be reconfigured into a 2-dimensional form, and care taken to remove any missing values (e.g., values of an oceanographic field over land) so that the covariance matrix can be constructed, and the EOFs computed as the (possibly scaled) eigenvectors of the covariance matrix. In order to correctly interpret the EOFs it is necessary to undo the data preparation steps listed above: the eigenvectors must be reformed into 2-dimensional maps, inserting any missing values back into their correct locations, and weighting often needs to be removed. Typically one will not just be interested in the EOFs themselves but also in other derived quantities such as the PC time series associated with each EOF, or the projection of other fields onto the EOFs. Similar data preparation and reconfiguration procedures are required to construct these quantities and great care must be taken to ensure that the application of these procedures is consistent in the computation of each quantity.

There are existing software packages and libraries for computing EOFs and related quantities [3,4], but this type of data analysis is often done in an ad-hoc manner using un-published code. The publically available tools for EOF analysis are typically libraries that provide separate procedures to compute each required output, a design that cannot automatically ensure the self-consistency of the analysis outputs. Therefore the user is responsible for keeping track of the integrity of the analysis. One of the major motivations behind the development of *eofs* was to resolve this problem by taking advantage of object-oriented design. Using an object to encapsulate

the core information about how the input data set was transformed in order to do the EOF computation allows the construction of method calls to compute any required related quantity in a manner consistent with the original decomposition. This is not only convenient for the programmer as it removes a lot of tedious overheads, but also ensures correctness of the resulting quantities. The *eofs* library has been used to analyse data in a number of scientific studies [5,6].

Implementation and architecture

The *eofs* library is implemented in a hierarchical structure. The core of the library is an EOF solver object. The solver object is a numerical solver constructed by passing a data set to analyse in the form of a *NumPy* array [7], and optionally an array of weights that apply to that data. Method calls are then used to generate the required outputs, in the form of *NumPy* arrays (see **Table 1**). This design allows all methods of the solver object to know exactly what weighting, reconfiguration and scaling has taken place to produce the EOFs, and hence allows derived quantities to be computed in an internally consistent manner. This core solver object does not know (or care) about the meaning or structure of the input data set, and is thus generic.

On top of the core component there are interfaces that can apply the analysis to data structures that contain structured metadata as well as data values, specifically designed for meteorological and oceanographic data sets. These metadata-aware solvers are motivated the desire to improve data provenance and ensure the correctness of scientific results. These issues affect all scientific research, but have been strongly highlighted in the climate science community in recent years [8]. The metadata-aware interfaces provide a layer on top of the core solver that interprets metadata from the input and uses it to determine how the data set is structured. The metadata-aware solvers are able to automatically reconfigure input data sets and generate appropriate weights for them according to pre-defined weighting schemes, and crucially they are able to return objects with correct metadata that can be used to identify the returned field outside the context of the analysis program.

The metadata-aware solvers are implemented as wrapper classes around the core solver object. This allows them to interpret the metadata of their input, and reconfigure the data set and any weights appropriately ready to be passed to the core solver. The core solver is used to perform all computations, and the wrapper class applies appropriate metadata to the computed quantities before returning them to the user. This prevents users having to manually throw away metadata to apply a computation, then having to reconstruct the metadata for the output, a process which is time consuming and open to errors. The *eofs* library currently provides metadata-aware solvers that understand data structures from *iris* [9], *xarray* [10] and *cdms2* (part of UV-CDAT) [11]. The design of metadata-aware interfaces as wrapper classes around a numerical core makes extending the library to accommodate other data structures relatively straightforward.

The hierarchical design concept is also extended to variations on the EOF computation methodology. The *eofs* library provides extra interfaces for computing multivariate EOFs. These are similar to normal EOFs but they are computed from a covariance matrix formed from observations of different variables. A pertinent example of the use of this type of analysis is the computation of the real-time multivariate Madden-Julian Oscillation index [12]. The implementation of multivariate EOFs in *eofs* consists of a multivariate solver, which is wrapper class around the core solver, and whose job is to combine separate input data sets with their own weights into a single array with a single set of weights ready for input into the core solver, and to reverse this process for output quantities where necessary. There are metadata-aware interfaces layered on top of the multivariate solver that do the translation between metadata-carrying data structures and plain *NumPy* arrays. This design pattern could be followed in order to implement some of the numerous variations on EOF analysis [13].

Method name	Description
pcs	The (optionally scaled) principal component time series (PCs).
eofs	The (optionally scaled) empirical orthogonal functions (EOFs).
eofsAsCorrelation	The EOFs expressed as the correlation between each PC and the input data set at each grid point.
eofsAsCovariance	The EOFs expressed as the covariance between each PC and the input data set at each grid point.
eigenvalues	The eigenvalues (decreasing variances) associated with each EOF mode.
varianceFraction	The fraction of the total variance explained by each EOF mode.
totalAnomalyVariance	The total variance (sum of the eigenvalues).
northTest	The typical error associated with each eigenvalue using North's rule of thumb [16].
reconstructedField	Reconstructs the input data set using a specified number of EOFs.
projectField	Projects an arbitrary field onto the EOFs to produce a set of pseudo-PCs.
getWeights	The array of weights used for the analysis.

Table 1: The method calls available to all solver objects.

Quality control

The *eofs* library is provided with a suite of unit and integration tests to test the core functionality and correctness of the library. The end user can easily run these tests against the version of the library they have installed to verify it is working correctly before use.

The test suite is intrinsically part of the development process, and is expanded as the software is developed and new features are added. The tests are automatically run on the Travis CI continuous integration and delivery service [14] every time a pull request to the *eofs* repository is made, which helps prevent breakage of existing code and functionality by new contributions.

The *eofs* library also comes with some example code and data, which allow the end user to verify that the output of the library is as expected, as well as see an example of how the library can be used.

(2) Availability

Operating system

Linux, OSX, Windows.

Programming language

Python 2.7 or Python >= 3.3

Dependencies

setuptools >= 0.7.2

NumPy >= 1.6

iris >= 1.2 (optional; needed for iris metadata-aware solver)

cdms2 (optional; needed for cdms2 metadata-aware solver)

xarray (optional; needed for the xarray metadata-aware solver)

nose (optional; only needed for running the test suite)

pep8 (optional; only needed for running the test suite)

Some of the provided examples in the documentation require extra dependencies to run, which are not required for normal use of the software: netCDF4, matplotlib, cartopy

Software location

Archive (e.g. institutional repository, general repository) (required – please see instructions on journal website for depositing archive copy of software in a suitable repository)

Name: Zenodo

Persistent identifier: http://dx.doi.org/10.5281/zenodo.46871

Licence: GNU General Public License Version 3

Publisher: Andrew Dawson

Version published: 1.1.0

Date published: 03/03/2016

Code repository (e.g. SourceForge, GitHub etc.) (required)

Name: Github

Identifier: https://github.com/ajdawson/eofs

Licence: GNU General Public License Version 3

Date published: 03/03/2016

Language

English.

(3) Reuse potential

eofs is already used frequently by weather and climate researchers at institutions across the world. The library is distributed as part of the Ultrascale Visualization Climate Data Analysis Tools (UV-CDAT) project [11], and has been used in a number of publications that the author is aware of [e.g., 5, 6]. The potential for reuse is huge since *eofs* allows a complex and custom EOF analysis methodology to be implemented quickly and correctly in just a few object-oriented method calls. The library is flexible and well documented making it suitable for use in applications ranging from an interactive data exploration to integration within a complex data processing pipeline.

There is also much potential for reuse of *eofs* outside of the originally intended audience of meteorology, oceanography, and climate research. The term EOF analysis is used predominantly in the geophysical sciences, with the terms principal component analysis (PCA) and factor analysis commonly used to refer to the same procedure in other fields. The core library components implement this standard mathematical technique in a way that does not make assumptions about the form or meaning of the input data. Therefore *eofs* can be applied to any data set that it is believed can be understood in terms of an EOF decomposition, with the caveat that some of the terminology used in *eofs* originated in meteorology and may require some mental translation to transfer to other fields.

The software is documented on-line at http://ajdawson.github.io/eofs. The software is supported on a voluntary basis through the code repository's issue tracker. Contributions to the project are welcomed, and can be submitted by making a pull request to the *eofs* Github repository.

Competing Interests

The author declares that they have no competing interests.

Acknowledgements

A very early ancestor of eofs was the EOF solver from PyClimate [15], which provided a basis for the numerical solution of the EOF problem, although almost none of this code remains in *eofs*.

References

1. **Wilks, D S** 2006 Statistical Analysis in the Atmospheric Sciences. 2nd ed. London: Academic Press.
2. **Deser, C** and **Blackmon, M L** 1995 On the Relationship Between Tropical and North Pacific Sea Surface Temperature Variations. *Journal of Climate*, 8(6): 1677–2680. DOI: http://dx.doi.org/10.1175/1520-0442(1995)008<1677:OTRBTA>2.0.CO;2
3. **UCAR/NCAR/CISL/TDD,** 2016 The NCAR Command Language (Version 6.3.0). DOI: http://dx.doi.org/10.5065/D6WD3XH5
4. **Pedregosa, F, Varoquaux, G, Gramfort, A, Michel, V, Thirion, B, Grisel, O, Blondel, M, Prettenhofer, P,**

Weiss, R, Dubourg, V, Vanderplas, J, Passos, A, Cournapeau, D, Brucher, M, Perrot, M and **Duchesnay, E** 2011 Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

5. **Dawson, A** and **Palmer, T N** 2015 Simulating Weather Regimes: Impact of Model Resolution and Stochastic Parameterization. *Climate Dynamics*, 44(7): 2177–2193. DOI: http://dx.doi.org/10.1007/s00382-014-2238-x
6. **Irving, D** and **Simmonds, I** 2016 A New Method for Identifying the Pacific-South American Pattern and its Influence on Regional Climate Variability. *Journal of Climate*, submitted.
7. **van der Walt, S, Colbert, S C** and **Varoquaux, G** 2011 The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science and Engineering*, 13(2): 22–30. DOI: http://dx.doi.org/10.1109/MCSE.2011.37
8. **Maibach, E, Leiserowitz, A, Cobb, S, Shank, M, Cobb, K M** and **Gulledge, J** 2012 The Legacy of Climategate: Undermining or Revitalizing Climate Science and Policy? *WIREs Climate Change*, 3(3): 289–295. DOI: http://dx.doi.org/10.1002/wcc.168
9. **Met Office,** 2016 Iris: A Python Library for Meteorology and Climatology. Available at http://scitools.org.uk/iris.
10. **Hoyer, S, Kleeman, A** and **Brevdo, E** 2016 Xarray: n-d Labeled Arrays and Datasets in Python. Available at http://xarray.pydata.org.
11. **Williams, D N, Doutriaux, C, Chaudhary, A, Fries, S, Lipsa, D, Jhaveri, S, Durack, P J, Painter, J, Nadeau, D** and **Maxwell, T** 2016 uvcdat v2.4.0. DOI: http://dx.doi.org/10.5281/zenodo.45136
12. **Wheeler, M C** and **Hendon, H H** 2004 An All-Season Real-Time Multivariate MJO Index: Development of an Index for Monitoring and Prediction. *Monthly Weather Review*, 132: 1917–1932. DOI: http://dx.doi.org/10.1175/1520-0493(2004)132<1917:AARMMI>2.0.CO;2
13. **von Storch, H** and **Zwiers, F W** 1999 Statistical Analysis in Climate Research. Cambirdge: Cambridge University Press. DOI: http://dx.doi.org/10.1017/CBO9780511612336
14. **Travis CI:** https://travis-ci.org.
15. **Sáenz, J, Zubillaga, J** and **Fernández, J** 2002 Geophysical data analysis using Python. *Computers and Geosciences*, 28: 457–465. DOI: http://dx.doi.org/10.1016/s0098-3004(01)00086-3
16. **North, G R, Bell, T L, Cahalan, R F** and **Moeng, F J** 1982 Sampling Errors in the Estimation of Empirical Orthogonal Functions. *Monthly Weather Review*, 110(7): 699–706. DOI: http://dx.doi.org/10.1175/1520-0493(1982)110<0699:SEITEO>2.0.CO;2

How to cite this article: Dawson, A 2016 eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data. *Journal of Open Research Software,* 4: e14, DOI: http://dx.doi.org/10.5334/jors.122

Submitted: 07 March 2016 **Aceepted:** 18 April 2016 **Published:** 26 April 2016

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS