International Journal of Data Science
Vol. 2, No. 1, June 2021, pp. 38-46
ISSN 2722-2039

PY-METEO-NUM: Dockerized Python Notebook Environment For Portable Data Analysis Workflows In Indonesian Atmospheric Science Communities

Sandy H.S. Herho[a,1], Dasapta E. Irawan[b,2,*]

[a] Independent Researcher Desa Krimun, Losarang Indramayu, West Java, 45253, Indonesia
[b] Applied Geology Research Group, Bandung Institute of Technology, Bandung, West Java, 40132, Indonesia
[1] sandyherho@protonmail.ch; [2] r-win@office.itb.ac.id
* corresponding author

ARTICLE INFO

Article history
Received June 16, 2020
Revised June 30, 2020
Accepted December 11, 2020

Keywords:
reproducible
computation
data analysis
docker
Python
Atmospheric Science

ABSTRACT

Reproducibility and replicability in analyzing data is one of the main requirements for the advance-ment of scientific fields that rely heavily on computational data analysis, such as atmospheric science. However, there are very few research activities that field in Indonesia that emphasize the principle of transparency of codes and data in the dissemination of the results. This issue is a major challenge for the Indonesian scientific community to verify the output of research activities from their peers. One common obstacle to the reproducibility of data-driven research is the portability issue of the computing environment used to reproduce the results. Therefore, in this article, we would like to offer a solution through Debian-based dockerized Jupyter Notebook that have been installed with several Python libraries that are often used in atmospheric science research. Through this containerized computing environment, we expect to overcome the portability and dependency constraints that often faced by atmospheric scientists and also to encourage the growth of research ecosystem in Indonesia through an open and replicable environment.

1. Introduction

Computation and numerical modelling have been an integral part of the development of atmospheric science since the beginning of its development in the 1950s [1]. This tendency is further strengthened by the general scientific trend towards an era that utilizes big data as a playground for the implementation of statistical learning concepts. Indonesian atmospheric science communities, as part of the worldwide scientific communities, which already have a background in computing history that is far longer than the current big-data trend certainly would not want to be left behind to implement statistical learning algorithms that are currently popular in analyzing weather and climate data (these are several case studies of the deep learning algorithm implementations to weather and climate data in Indonesia has already been conducted: [2, 3, 4, 5]). This increasing number of quantitative research cultures should be appreciated, given that the statistical learning methods could certainly sharpen the analysis of meteorological disaster-prone areas such as Indonesia.

To support the need for statistical learning research in the area of atmospheric science, openness of data and source code that can be reproduced for subsequent research is needed, to create a sustainable science system. Reproducibility meant here is the openness in the process and to make the whole components (datasets, codes, analysis) publicly available which is one of the four principles on The Open Science Project website as follows [6]:

Transparency: openness in methods, observation, and data collection; Public availability and reusability of data: publicly available data, so it can be reused for various purposes; Public accessibility and transparency of scientific communication: study results are open and transparent to the public; Use web-based / open source tools to facilitate scientific collaboration: in its implementation, research uses software and computing infrastructure that is open-source and portable;

Unfortunately, none of the atmospheric science journals in Indonesia applies the four principles above as a whole. We believe this situation is due to the diverse educational backgrounds, skills, and infrastructure in the atmospheric science ecosystem. However, given such situation, reproducibility and portability of computational-based processing would be beneficial to the ecosystem. In this article, we want to offer a solution to this problem through the use of the Python computational language, which is the current most popular scripting language for data processing in the atmospheric science communities [7], which is run through the Jupyter Notebook environment that is run on Linux Container (LXC) virtualization using Docker which has a cross-platform functionality (operating system-agnostic) and has been widely applied as a computational container in the various scientific domains as reproducible research tools [8, 9, 10, 11]. We have documented the results of this initial trial in the free and open-source docker image we call, py-meteo-num.

2. Method

2.1. Materials

2.1.1. Scripting skill

As we use mainly command line scripting language to develop the platform, a basic to intermediate Python scripting skill is needed. We understand that it involves a steep learning curve. In time, the steep learning curve would bring a nice trade off to the user, as the resulting work will be more sustainable and reusable by interesting party with the growing size of Python community in the atmospheric science fields. Many online and free tutorials are available on the internet offering free to reuse codes for weather and climate data processing using Python [12, 13]. Potential users of this docker platform should spend a short time self-driven Python training to get familiar with the scripting environment and workflow.

2.1.2. Jupyter Notebook

Jupyter Notebook (Figure 1) is a web-based interactive computing environment that allows us to create, execute, and disseminate code, graphics, and also human-readable texts(in Markdown and LATEX formats). On the py-num-meteo itself, only the Python 3 kernel installation is performed using the Anaconda distribution. In addition to the default Python libraries from Anaconda, py-meteo-num also provides several additional libraries used for atmospheric science data processing as suggested by the Python for Atmospheric and Ocean Science (PyAOS) community [14].

International Journal of Data Science
Vol. 2, No. 1, June 2021, pp. 38-46
ISSN 2722-2039

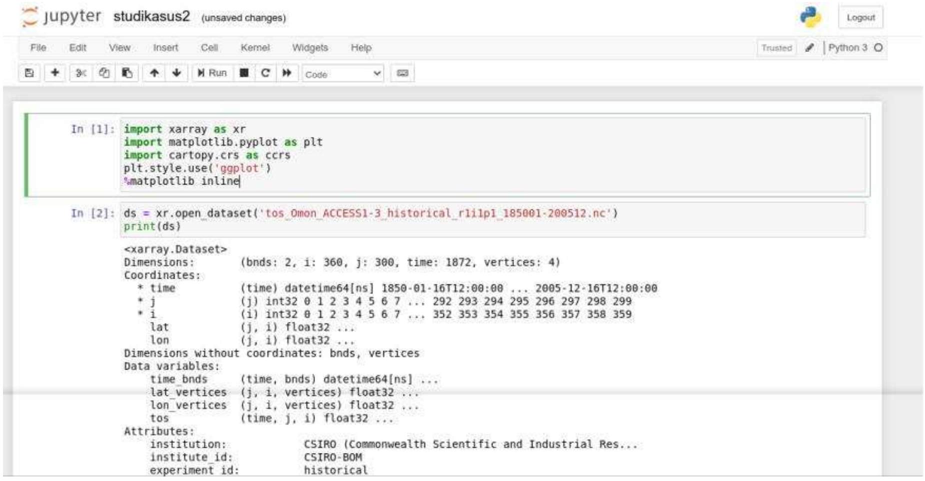

Fig. 1 Display of Jupyter Notebook.

2.1.3. Docker

Figure 2 shows the components of Docker. The main component is the docker host (docker engine) that manages other components of the system. Users can pull pre-built docker images (in our case we used Debian Buster) from public repositories (e.g., DockerHub and Github) via the docker engine. An image is a series of Linux commands together with the required binary and data files. The docker engine caches the downloaded images in its local repository. To run an image, the docker-engine allocates an isolated container of the Linux kernel to the image. An instance of a running image is called docker container (or container).

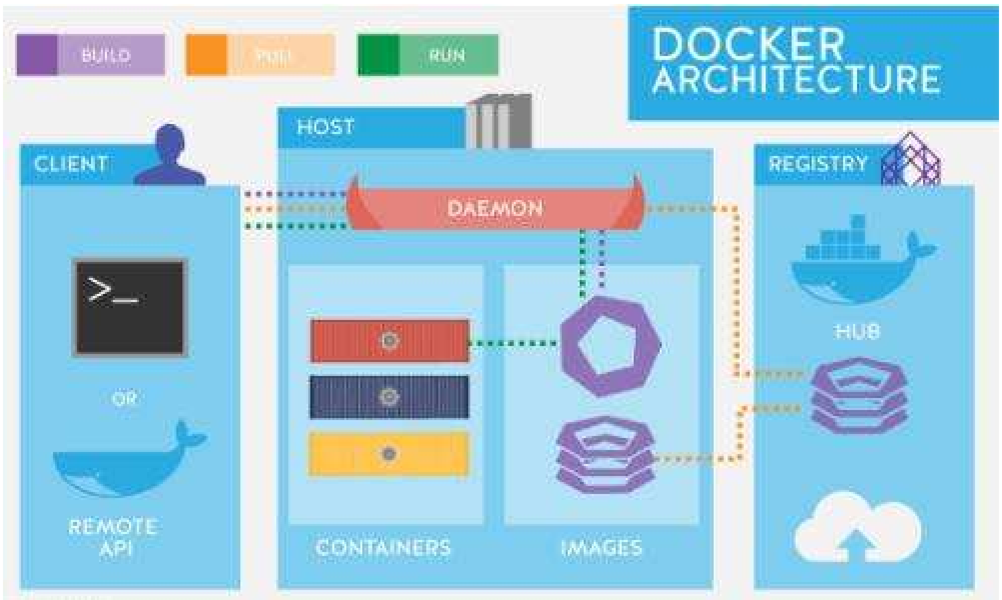

Fig. 2 The Docker architecture.

Figure 3 shows the lifecycle of a container. It starts when a container is created from an image, until the container is killed. A container can also be paused/unpaused or stopped/restarted. We can manage a container lifecycle via the Docker command line interface (CLI).

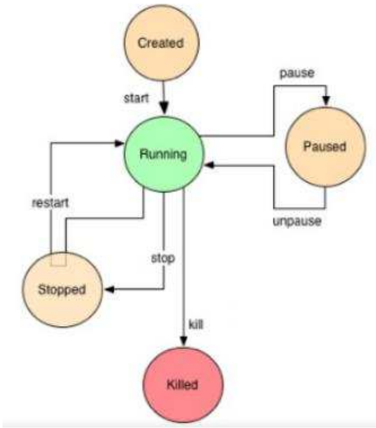

Fig. 3 The lifecycle of a docker container.

Although a docker container is launched from an image, images and containers are different entities inside Docker. An image is an artifact related to the development phase, whereas a docker container is an object related to the run-time. Commands such as pull, push, and commit are image-specific commands, while exec , run , and pause are container-specific commands.

2.2. Method

The py-meteo-num was built on top of a Debian-based docker image, then we added several layers of basic applications that users need, such as wget and bzip (which were needed to install Anaconda distribution), sudo (used to access root), and distribution Anaconda.In addition to Anaconda's built-in libraries, we were also installing other Python libraries that are most likely needed for weather and climate data processing. These libraries are shown in Table 1.

Table 1. Extended Python libraries in the container.

Library	Description
basemap	Plotting 2D data on maps in Python [15].
basemap-data-hires	Plotting on map projections (with coastlines and political boundaries) using matplotlib [15].
cartopy	Geospatial data processing in order to produce maps and other geospatial data analyses [16].
cbsyst	Calculating seawater carbon and boron chemistry [17].
cdo	CLI tools to manipulate and analyse climate and Numerical Weather Prediction (NWP) model data [18].
cdsapi	Python API to access the Copernicus Climate Data Store (CDS) [19].
climlab	Process-oriented climate modeling [20].
cmocean	Colormaps for oceanography [21].
ctd	Tools to load hydrographic data formats as pandas DataFrames [22].
fbprophet	Automatic time-series forecasting procedure [23].
gsw	Gibbs SeaWater Oceanographic Package of TEOS-10 [24].
iris	Analyse and visualise meteorological and oceanographic data sets [25].
metpy	A collection of tools in Python for reading, visualizing and performing calculations with weather data. [26]
mpld3	D3.js viewer for matplotlib [15].
netcdf4	Provides an object-oriented python interface to the netCDF version 4 library [27].
opencv	Computer vision and machine learning software library [28].
owslib	Open Geospatial Consortium (OGC) web service utility library [29].

paegan	A high level Common Data Model (CDM) library for array based met/ocean data sets [30].
pydap	Pure Python Opendap/DODS client and server [31].
pygrib	Python GRIB (editions 1 and 2) reader.
pymc3	Bayesian probabilistic programming in Python [32].
siphon	A collection of Python utilities for accessing remote geoscience data. [33]
tensorflow	an open source machine learning framework [34].
windspharm	Spherical harmonic computations on vector winds [35].
wrf-python	Diagnostic and interpolation routines for WRF-ARW data [36].
xarray	Processing N-D labeled arrays and datasets in Python [37].
xclim	Library of derived climate variables, i.e. climate indicators, based on xarray [38].

As the user's default working directory, we provide the /home/debian/ directory. And as a display of computing interface, Jupyter Notebook runs on port 8888. The Dockerfile that we made is visible in Figure 4.

```
# py-meteo-num : Docker image for computational atmospheric sciences

# Anti-Copyright (a-c) Sandy Herho and Dasapta Erwin Irawan (2020).
# Distributed under the terms of the GNU GPLv3.

# We will use Debian 10 (Buster) for our
image FROM debian:buster-slim

LABEL maintainer="Sandy Hardian Susanto Herho <sandyherho@meteo.itb.ac.id>"

# Updating Debian packages
RUN apt update && yes|apt upgrade

# Adding wget and bzip2
RUN apt install -y wget bzip2

# Add sudo
RUN apt -y install sudo

# Add user Debian with no password, add to sudo group
RUN adduser --disabled-password --gecos '' debian && \
    adduser debian sudo && \
    echo '%sudo ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers
USER debian
WORKDIR /home/debian/
RUN chmod a+rwx /home/debian/

# Anaconda installing
RUN wget https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh && \
    bash Anaconda3-2020.02-Linux-x86_64.sh -b && \
    rm Anaconda3-2020.02-Linux-x86_64.sh

# Set path to conda
ENV PATH /home/debian/anaconda3/bin:$PATH

# Updating Anaconda packages
RUN conda update conda && \
    conda update anaconda && \
    conda update --all

# Installing needed packages for atmospheric science research
RUN conda install -c conda-forge basemap cmocean basemap-data-hires cartopy \
    pydap metpy wrf-python siphon opencv fbprophet ctd pymc3 pygrib windspharm \
    paegan iris mpld3 owslib gsw cbsyst climlab xclim cdsapi cdo && \
    conda install -c anaconda netcdf4 xarray tensorflow

# Configuring access to Jupyter
RUN mkdir /home/debian/notebooks && \
    jupyter notebook --generate-config --allow-root && \
    echo "c.NotebookApp.password = u'sha1:6a3f528eec40:6e896b6e4828f525a6e20e5411cd1c8075d68619'" >>
/home/debian/.jupyter/jupyter_notebook_config. py

# Jupyter listens port: 8888
EXPOSE 8888

# Run Jupyter notebook as Docker main process
CMD ["jupyter", "notebook", "--allow-root", "--notebook-dir=/home/debian/notebooks", "--ip='*'", "--port=8888", "--no-browser"]
```

Fig. 4 The setup of py-meteo-num Dockerfile

We built this docker image using the following command:

```
docker build -t py-meteo-num
```

3. Results and Discussions

Here we use a case Study from the Maritime Continent precipitation anomaly visualization from CSIRO ACCESS 1.3 output. In this section, we present one case study in which we illustrate the use of Python for visualizing precipitation anomaly from CMIP5 CSIRO ACCESS 1.3 historical simulation output [39] within Jupyter notebook using py-meteo-num as a docker container. The corresponding notebook is available from our Github repository.

The first step that users must do is make a pull request from the DockerHub repository via the CLI with the following command:

```
docker pull herholabs/py-meteo-num
```

To start Jupyter Notebook session, run the following command:

```
docker run --name herholabs/py-meteo-num -p 8888:8888 --env="DISPLAY" -v "${PWD/notebooks:/home/debian/notebooks}" -d meteo-num
```

Then open your browser and enter port 8888: http://localhost:8888/. User will be asked to enter a password. For the password requested, enter: root.

We use Jupyter Notebook on the docker container to display the average monthly rainfall anomalies from historical ACCESS-1.3 climate models over the last 16 years of the data period (January 1990 to December 2005) over the Maritime Continent using the xarray library [37].

We begin by importing several libraries in the Python scientific computing environment:

In [1]:
```
import xarray as xr
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
plt.style.use('ggplot')
%matplotlib inline
```

Then, we open the NetCDF file:

In [2]:
```
ds = xr.open_dataset('http://dapds00.nci.org.au/thredds/dodsC/rr3/CMIP5/output1/CSIRO-BOM/ACCESS1-3/historical/mon/atmos/Amon/r1i1p1/latest/pr/pr_Amon_ACCESS1-3_historical_r1i1p1_185001-200512.nc')
```

Then we have to extract the precipitation Data Array from the dataset:

In [3]:
```
pr = ds['pr']
```

Because the precipitation Data Array unit is still in kg.m 2/s (mm/s), we need to change it to mm/month: Next, we extract the Data Array at the Maritime Continent coordinates [40]:

In [4]:
```
pr_bm = pr.sel(lat=slice(-20,20),lon=slice(90,160))
```

Then we extract anomalous data only in the period of January 1990 to December 2005.

In [5]:
```
pr_bm_anom_mod = pr_bm_anom.sel(time=slice('1990-01', None))
```

We make the precipitation data from January 1961 to December 1990 as a benchmark for measuring the modern rainfall anomalies over the Maritime Continent:

In [6]:
```
pr_bm_klim = pr_bm.sel(time=slice('1961-01','1990-12')).mean(dim='time')
pr_bm_anom = pr_bm - pr_bm_klim
```

Finally, using the groupby() method, we classify the data based on the average monthly precipitation anomalies and we show them in the Figure 4:

```
In [7]:
plt.figure(figsize=(40,30));
proj = ccrs.PlateCarree();
pr_anom_bul = pr_bm_anom_mod.groupby('time.month').mean(dim='time')
p = pr_anom_bul.plot(col='month',col_wrap=4,
                     subplot_kws=dict(projection=proj),
                     transform=ccrs.PlateCarree(),
                     cmap = 'bwr_r');

for ax in p.axes.flat:
    ax.coastlines();
```

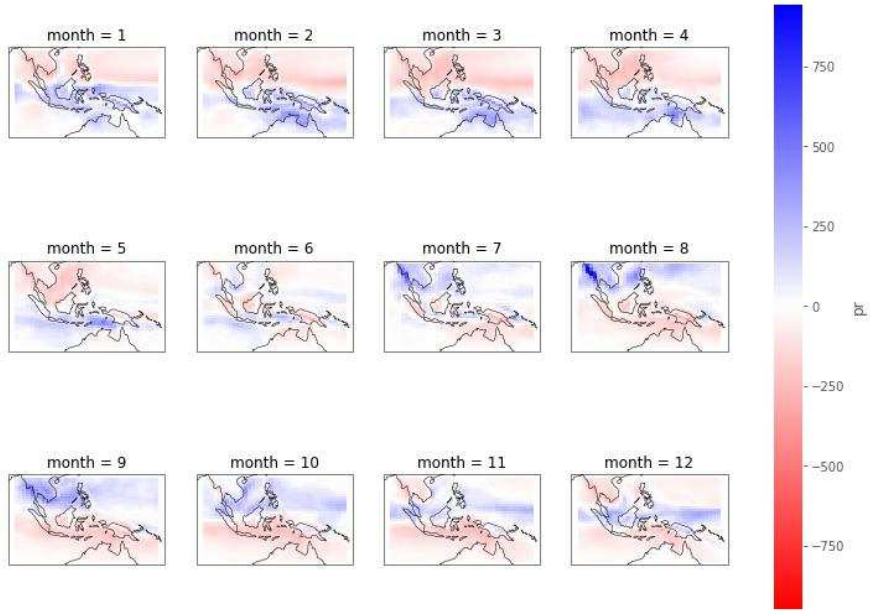

Fig. 5 Monthly precipitation anomalies over the Maritime Continent from January 1961 to December 1990.

Based on this case study, we can confirm that the Jupyter Notebook that we run on this docker container, can be used as a computing resource for analyzing weather and climate data in the form of multidimensional arrays.

4. Conclusion

PY-METEO-NUM is a prototype of a containerized computing environment to analyze weather and climate data that still needs to be further developed. Here we want to demonstrate the importance of the portability of the computing environment in the atmospheric science fields. To support the development of open and reproducible atmospheric science research in Indonesia, we encourage users to make pull requests on the GitHub to make changes and improvements to our Dockerfile under the terms of the GNU GPLv3 License [41].

Acknowledgements

We thank Pradipto (Kyoto University) for his feedback on early drafts of this manuscript and P3MI ITB for funding the publication and dissemination stage.

Author's contributions

SHSH: formulating ideas, write the codes and drafting the manuscript DEI: formulating ideas and drafting the manuscript

Conflict of Interest

The authors have no competing interests to declare.

Supporting materials

All codes and data is accessible on our Github project site.

References

[1] J. M. Wallace and P. P. Hobbs, Atmospheric Science An Introductory Survey, vol. 92 of International Geophysics Series. Academic Press, 2 ed., 2006.

[2] S. Nurcahyo, F. Nhita, and Adiwijaya, "Rainfall prediction in kemayoran jakarta using hybrid genetic algorithm (ga) and partially connected feedforward neural network (pcfnn)," in 2014 2nd International Conference on Information and Communication Technology (ICoICT), pp. 166–171, 2014.

[3] F. Nhita, D. Saepudin, Adiwijaya, and U. N. Wisesty, "Comparative study of moving average on rainfall time series data for rainfall forecasting based on evolving neural network classifier," in 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI), pp. 112–116, 2015.

[4] Gunawansyah, T. H. Liong, and Adiwijaya, "Prediction and anomaly detection of rainfall using evolving neural network to support planting calender in soreang (bandung)," in 2017 5th International Conference on Information and Communication Technology (ICoIC7), pp. 1–6, 2017.

[5] J. A. Suyatno, F. Nhita, and A. A. Rohmawati, "Rainfall forecasting in bandung regency using c4.5 algorithm," in 2018 6th International Conference on Information and Communication Technology (ICoICT), pp. 324–328, 2018.

[6] D. E. Irawan, C. N. Rachmi, H. Irawan, J. Abraham, K. Kusno, M. T. Multazam, K. K. Rosada, S. H. Nugroho, G. Kusumah, D. Holidin, and N. A. Aziz, "Penerapan Open Science di Indonesia agar riset lebih terbuka, mudah Diakses, dan Meningkatkan Dampak Saintifik. (Indonesia) [The application of Open Science in Indonesia so that research is more open, easily accessible, and increases the scientific impact]," Berkala Ilmu Perpustakaan dan Informasi, vol. 13, no. 1, pp. 25–36, 2017.

[7] J. W.-B. Lin, "Why python is the next wave in earth sciences computing," Bulletin of the American Meteorological Society, vol. 93, no. 12, pp. 1823–1824, 2012.

[8] R. Almugbel, L.-H. Hung, J. Hu, A. Almutairy, N. Ortogero, Y. Tamta, and K. Y. Yeung, "Reproducible bioconductor workflows using browser-based interactive notebooks and containers," Journal of American Medical Informatics Association, vol. 25, no. 1, pp. 4–12, 2017.

[9] C. Boettiger, "An introduction to docker for reproducible research," SIGOPS Oper. Syst. Rev., vol. 49, p. 71–79, Jan. 2015.

[10] J. P. Hacker, J. Exby, D. Gill, I. Jimenez, C. Maltzahn, T. See, G. Mullendore, and K. Fossell, "A containerized mesoscale model and analysis toolkit to accelerate classroom learning, collaborative research, and uncertainty quantification," Bulletin of the American Meteorological Society, vol. 98, no. 6, pp. 1129–1138, 2017.

[11] N. H. D. Morris, S. Voutsinas and R. Mann, "Use of docker for deployment and testing of astronomy software," arXiv preprint arXiv:1707.03341, 2017.

[12] U. Team, Unidata Python Training, accessed June 9, 2020. https://unidata.github.io/ python-training/.

[13] J. W.-B. Lin, A Hands-On Introduction to Using Python in the Atmospheric and Oceanic Sciences. 2012.

[14] D. Irving, "Python for atmosphere and ocean scientists," Journal of Open Source Education, vol. 2, no. 16, p. 37, 2019.

[15] J. D. Hunter, "Matplotlib: A 2d graphics environment," Computing in Science Engineering, vol. 9, no. 3, pp. 90–95, 2007.

[16] Met Office, Cartopy: a cartographic python library with a matplotlib interface. Exeter, Devon, 2010 - 2015.

[17] Oscar, "oscarbranson/cbsyst: beta," Aug. 2018.

[18] C. S. Zender, "Analysis of self-describing gridded geoscience data with netcdf operators (nco)," Environmental Modelling Software, vol. 23, no. 10, pp. 1338 – 1342, 2008.

[19] P. Petrelli, "coecms/era5: python base codes to interface the CDS api and automate ERA5 download: first release v0.1," Nov. 2019.

[20] B. E. j. Rose, "Climlab: a python toolkit for interactive, process-oriented climate modeling," Journal of Open Source Software, vol. 3, no. 24, p. 659, 2018.

[21] K. M. Thyng, C. A. Greene, R. D. Hetland, H. M. Zimmerle, and S. F. DiMarco, "True colors of oceanography: Guidelines for effective and accurate colormap selection," Oceanography, vol. 293, September 2016.

[22] Fernandes, "python-ctd v0.2.1," Aug. 2014.

[23] S. J. Taylor and B. Letham, "Forecasting at scale," PeerJ Preprints, vol. 5, p. e3190v2, Sept. 2017.

[24] Filipe, "python-gsw v3.0.3," Aug. 2014.

[25] Met Office, Iris: A Python library for analysing and visualising meteorological and oceanographic data sets. Exeter, Devon, v1.2 ed., 2010 - 2013.

[26] R. M. May, S. C. Arms, P. Marsh, E. Bruning, J. R. Leeman, K. Goebbert, J. E. Thielen, and Z. S. Bruick, "Metpy: A Python package for meteorological data," 2008 - 2020.

[27] J. Whitaker, C. Khrulev, D. Huard, C. Paulik, S. Hoyer, Filipe, L. Pastewka, A. Mohr, C. Marquardt, B. Couwen-berg, M. Taves, J. Whitaker, M. Cuntz, M. Bohnet, M. Brett, R. Hetland, M. Korenciak,ˇ barronh, K. Onu, J. J. Helmus, J. Hamman, A. Barna, fredrik 1, B. Koziol, T. Kluyver, R. May, J. Smrekar, C. Barker, C. Gohlke, and B.P. Kinoshita, "Unidata/netcdf4-python: Version 1.5.3 release," Oct. 2019.

[28] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[29] T. Kralidis, "geopython/owslib: v0.20.0," 2020.

[30] K. Wilcox, A. Crosby, and B. McKenna, "https://github.com/asascience-open/paegan," 2018.

[31] P. Kershaw, R. Ananthakrishnan, L. Cinquini, B. Lawrence, S. Pascoe, and F. Siebenlist, "A flexible component based access control architecture for opendap services," 05 2010.

[32] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," PeerJ Computer Science, vol. 2, p. e55, Apr. 2016.

[33] R. May, S. Arms, J. Leeman, and J. Chastang, "Siphon: A collection of Python utilities for accessing remote atmospheric and oceanic datasets," 2014 - 2017.

[34] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), (Savannah, GA), pp. 265–283, USENIX Association, Nov. 2016.

[35] A. Dawson, "Windspharm: A high-level library for global wind field computations using spherical harmonics," Journal of Open Research Software, vol. 4, no. 1, 2016.

[36] W. Ladwig, "Wrf-python (version 1.3.2)," 2020.

[37] S. Hoyer and J. J. Hamman, "xarray: N-d labeled arrays and datasets in python," Journal of Open Research Software, vol. 5, no. 1, p. 10, 2017.

[38] D. Huard, T. J. Smith, P. Bourgault, T. Logan, sbiner, P. Roy, D. Caron, jwenfai, RondeauG, C. Whelan, and A. Stephens, "Ouranosinc/xclim: v0.17.0," May 2020.

[39] M. Collier and P. Uhe, "CMIP5 datasets from the ACCESS1.0 and ACCESS1.3 coupled climate models," tech. rep., The Centre for Australian Weather and Climate Research, 12 2012.

[40] C. S. Ramage, "Role of a tropical "maritime continent" in the atmospheric circulation," Monthly Weather Review, vol. 96, no. 6, pp. 365–370, 1968.

[41] "Gnu general public license."