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

Sandy H.S. Herho\(^\text{a,1}\), Dasapta E. Irawan\(^\text{b,2,*}\)

\(^\text{a}\) Independent Researcher Desa Krimun, Losarang Indramayu, West Java, 45253, Indonesia
\(^\text{b}\) 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

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].
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).

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).
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.

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

We built this docker image using the following command:

docker build -t py-meteo-num

Fig. 4 The setup of py-meteo-num Dockerfile
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]:
1. import xarray as xr
2. import matplotlib.pyplot as plt
3. import cartopy.crs as ccrs
4. plt.style.use('ggplot')
5. %matplotlib inline
```

Then, we open the NetCDF file:

```
In [2]:
1. ds = xr.open_dataset('http://dapds00.nci.org.au/thredds/dodsC/rr3/CMIP5/output1/CSIRO-BOM/ACCESS1-3/historical/atmos/Amon/r1i1p1/mon/atmos/Amon kidneys/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]:
1. 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]:
1. 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]:
1. 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]:
1. pr_bm_klim = pr_bm.sel(time=slice('1961-01','1990-12')).mean(dim='time')
2. 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]:
  1. plt.figure(figsize=(40,30));
  2. proj = ccrs.PlateCarree();
  3. pr_anom_bul = pr_bm_anom_mod.groupby('time.month').mean(dim='time')
  4. p = pr_anom_bul.plot(col='month',col_wrap=4,
                         subplot_kws=dict(projection=proj),
                         transform=ccrs.PlateCarree(),
                         cmap = 'bwr_r');
  5. 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


[41] “Gnu general public license.”