Analyzing and Visualizing Open Precipitation Data

Authors: Kostas Leptokaropoulos and Shubo Chakrabarti

Table of Contents

Introduction

Public Data: Many public databases have been created for the purposes of making data freely accessible to the scientific community. A best practice is to assign a unique identifier to a dataset, so that it is discoverable. A common form of a unique identifier is a Digital Object Identifier or DOI® which points to the data.

Access Public Data: To access and process public data, you can use several routes.

- Download data files to your local machine and work with them in MATLAB®.
- Access data directly via an API. MATLAB's function that import data from NetCDF files from RESTful API used by many portals.

- If the portal offers only Python® bindings, call Python from MATLAB.

Data formats: MATLAB supports a broad range of data formats

- There is a wide range of scientific data formats that can be natively read in MATLAB. They include NetCDF, HDF5 and GRIB as well as more specialized data formats.
- In addition, the Mapping Toolbox™ contains built-in functions to read data from many online data repositories in standard geo-data formats.
- Sometimes data import functions may be written by the Geoscience community, and published on the MATLAB File Exchange - a portal for community contributions in MATLAB. All community contributions are covered by open source licenses, which means they can be re-used, modified or added to. Exact terms and conditions depend on the licenses used by the author. ## Data and Resources

In this example, we use data covered by permissive license, (i.e., the Creative Commons Attribution 4.0 International License - CC BY 4.0). It is important for Open Science to have such data since under CC BY 4.0 license you are free to share, copy, redistribute and adapt the material as long as you give appropriate credit, provide a link to the license, and indicate if any changes were made.

We access global climate data from the WCRP CMIP6 (World Climate Research Programme Coupled Intercomparison Project - Phase 6), which are located at: **https://esgf-data.dkrz.de/search/cmip6-dkrz/**.

The Data Space is hosted by the German Climate Computer Center (DKRZ) and the Infrastructure for the European Network for Earth System Modelling (IS-ENES), as a part of the global Earth System Grid Federation (ESGF).

The user can apply multiple filters and preferences for a large variety of climate parameters, such as see ice thickness, air pressure, phytoplankton mass concentration, etc, provided by different sources. In our example we used the following filtering:

- CF Standard name: precipitation flux
- Frequency: day
- Nominal Resolution: 10km, 25km

Make sure to check the box *"Show all Replicas*".

Example links for the data:

- data Link (1950 - 2014); e.g., data 19500101-19501231
- data Link (2015 - 2050); e.g., data 20450101-20451231

Data Acknowledgment Statement (see *CMIP6 terms of use*):

"We acknowledge the World Climate Research Programme, which, through its Working Group on Coupled Modelling, coordinated and promoted CMIP6. We thank the climate modeling groups for producing and making available their model output, the Earth System Grid Federation (ESGF) for archiving the data and providing access, and the multiple funding agencies who support CMIP6 and ESGF. We used the CMIP6 model data produced by the Research Center for Environmental Changes, Academia Sinica, Taiwan (AS-RCEC), which is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0; https://creativecommons.org/licenses/).

Further information about this data can be found via the further_info_url (recorded as a "Global Attributes: License" in these .nc files). The data producers and data providers make no warranty, either express or implied, including, but not limited to, warranties of merchantability and fitness for a particular purpose. All liabilities arising from the supply of the information (including any liability arising in negligence) are excluded to the fullest extent permitted by law."

Import Data from the web

In [1]:
```
clearvars; clc; close all
```

You can read any 2 datasets from the example links indicated above, one imported as "URLpast" and a subsequent one as "URLproj".

There are a bunch of functions that you can use to interact with NetCDF files.

In [2]:
```
% read NETCDF data from the web
  URLpast = "https://esgf-data04.diasjp.net/thredds/dodsC/esg_dataroot/CMIP6/HighResMIP/
  URLproj = "https://esgf-data04.diasjp.net/thredds/dodsC/esg_dataroot/CMIP6/HighResMIP/
```

Read the precipitation flux as "Variable"

In [3]:
```
ncdisp(URLpast)
ncdisp(URLproj)
Variable = "pr";
```

Read and filter Geospatial Data

read longitude and latitude from the .nc file

In [4]:
```
lon = double(ncread(URLpast,'lon'));
lat = double(ncread(URLpast,'lat'));
```

Filter Data in Time and Space

Select boundaries of a rectangular area

In [5]:
```
%minimum and maximum longitude
minlon = 25;maxlon = 40; % set values from 0 to 360
%minimum and maximum latitude
minlat = 35;maxlat = 50; % set values from -90 to 90
```

Select first day and number of days after the first *(their sum must not exceed 365)*

In [6]:
```
starttime = 100;counttime = 190;% set values from 1 to 365
```

Find the indexes of longitude and latitude that correspond to the selected geographical boundaries

In [7]:
```
indlon = find(lon>=minlon & lon<=maxlon);
indlat = find(lat>=minlat & lat<=maxlat);
```

```
startlon=indlon(1);countlon=numel(indlon);
startlat=indlat(1);countlat=numel(indlat);
```

Import Filtered Data

Import longitude, latitude, time and precipitation values for past and future data, as constrained in the previous section.

In [8]:
```
lat = double(ncread(URLpast,'lat',startlat,countlat));
lon = double(ncread(URLpast,'lon',startlon,countlon));
tpast = double(ncread(URLpast,'time',starttime,counttime));
tproj = double(ncread(URLproj,'time',starttime,counttime));
Varpast = double(ncread(URLpast,Variable,[startlon startlat starttime],[countlon countla
Varproj = double(ncread(URLproj,Variable,[startlon startlat starttime],[countlon countla
```

read the precipitation unit from the .nc file

In [9]:
```
unit = ncreadatt(URLpast,'pr','original_units');
```

Convert units from sec-1 to day-1

In [10]:
```
Varpast = Varpast*86400;
 Varproj = Varproj*86400;
 newunit = "kg m^-^2 day^-^1"
```

Calculate dates

calculate date vectors, as days passed since the reference date in the .nc file

In [11]:
```
Tpast = datetime('1948-1-1')+tpast,Tproj = datetime('1948-1-1')+tproj
Tpast.Format = 'dd-MMM-uuuu';Tproj.Format = 'dd-MMM-uuuu';
Tpast_str = string(Tpast);
Tproj_str = string(Tproj);
```

Select a day to visualize precipitation data

In [12]:
```
close all
```

Select a day

In [13]:
```
t1 = Tpast_str(6);inx = Tpast_str == t1;
```

Create a grid with precipitation values

In [14]:
```
[lat1,lon1] = meshgrid(lat,lon);
Var1past = Varpast(:,:,inx);
Var1proj = Varproj(:,:,inx);
mV = min([min(Var1past(:)),min(Var1proj(:))])
MV = max([max(Var1past(:)),max(Var1proj(:))])
```

Customize colormap limits to the selected data

In [15]:
```
CBmin = -9.00419266706534e-16;
  CBmax = 51.43537279218435;
```

Visualize and compare past and projected data

Create a 2 frame figure. Plot the past data in the left frame and the projected data in the right frame. Add the coastlines in both figures. The text displayed in the figure (title, colorbar) is also directly imported from the .nc file metadata and the parameter values we have set.

In [16]:
```
tiledlayout(1,2)
nexttile % past
worldmap([minlat maxlat],[minlon maxlon])
geoshow(lat1,lon1,Var1past,'DisplayType','texturemap','FaceAlpha',0.5);
geoshow('landareas.shp','EdgeColor','black','FaceColor','none')
title([ncreadatt(URLpast,Variable,'long_name'),'on',t1])
  clim([CBmin CBmax])

nexttile %projection
worldmap([minlat maxlat],[minlon maxlon])
geoshow(lat1,lon1,Var1proj,'DisplayType','texturemap','FaceAlpha',0.5);
geoshow('landareas.shp','EdgeColor','black','FaceColor','none')
 cb = colorbar; % display colorbar
 cb.Location='eastoutside';
 cb.Label.String = (newunit);
title([ncreadatt(URLproj,Variable,'long_name'),'on',Tproj_str(inx)])
  clim([CBmin CBmax])

 colormap turbo
```

Some Stats!

In this section a comparison is performed between the past and projected data.

- First, the precipitation time series (daily, solid lines; cumulative, dashed lines) are plotted in the same plot for direct comparison.
- Then, box charts are generated to show the daily sum average precipitation values distribution (including the outliers)
- Finally, 3 Hypothesis Tests are performed to evaluate the statistical significance of the results from the comparison of the past and projected precipitation:

  a) The Wilcoxon rank sum test: Tests the null hypothesis that past and projected data are samples from continuous distributions with equal medians, against the alternative that they are not.

b) The 2 sample t-test: Tests the null hypothesis that past and projected data come from independent random samples from normal distributions with equal means and equal but unknown variances. The alternative hypothesis is that the past and projected data comes from populations with unequal means.

c) The 2 sample Kolmogorov - Smirnov test: Tests the null hypothesis that that past and projected data are from the same continuous distribution against the alternative hypothesis that they are from different continuous distributions.

Compare incremental and cumulative precipitation

In [17]:
```
N = numel(tpast)
```

Select number of days (between 1 and N)

In [18]:
```
T1 = Tpast_str(1);inx1 = Tpast_str == T1;n1 = find(inx1);
T2 = Tpast_str(190);inx2 = Tpast_str == T2;n2 = find(inx2);
n = n2 - n1 +1;
if n2<=n1;error('End date (n2) should be greater than start date (n1)');end
```

Generate Daily/Cumulative Plots and Box Charts

In [19]:
```
Boxp1 = zeros(size(n,1));Boxp2 = zeros(size(n,1));cou=1;
for i=n1:n2
Boxp1(cou) = sum(Varpast(:,:,i),"all");
Boxp2(cou) = sum(Varproj(:,:,i),"all");
cou=cou+1;
end
```

Daily and Cumulative plots

In [20]:
```
figure
plot(Tpast(n1:n2),Boxp1,'r-',Tpast(n1:n2),Boxp2,'k-',LineWidth=1);datetick('x',6);ylabel
hold on
yyaxis right;plot(Tpast(n1:n2),cumsum(Boxp1),'r--',Tpast(n1:n2),cumsum(Boxp2),'k--');dat
legend(num2str(year(Tpast(1))),num2str(year(Tproj(1))),num2str(year(Tpast(1))),num2str(y
```

Box charts

In [21]:
```
figure
group = categorical([repmat("past", size(Boxp1));repmat("projection", size(Boxp2))]);
boxchart([Boxp1',Boxp2']);xticklabels({'past','projection'});
title(["Daily Sum Average","past/projection ratio = "+num2str(sum(Boxp1)/sum(Boxp2),'%7.
```

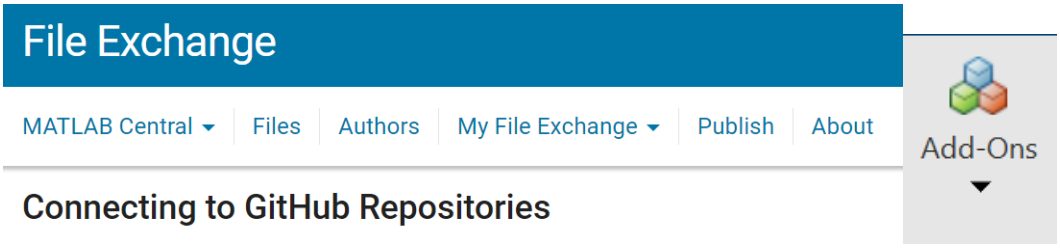
Perform Statistical Tests

In [22]:
```
[p.Wilcoxon,h.Wilcoxon] = ranksum(Boxp1,Boxp2); % null hypothesis, h0, equal medians
[h.ttest2,p.ttest2] = ttest2(Boxp1,Boxp2); % null hypothesis, h0, equal means
[h.kstest2,p.kstest2] = kstest2(Boxp1,Boxp2); % null hypothesis, h0, samples come from t
h.ttest2 = logical(h.ttest2);

if h.Wilcoxon==0;disp(['>> The <strong>Wilcoxon</strong> rank sum test FAILS to reject t
else; disp(['>> The <strong>Wilcoxon</strong> rank sum test REJECTS the hypothesis of eq
end

if h.ttest2==0;disp(['>> The 2-sample <strong>t-test</strong> FAILS to reject the hypoth
else; disp(['>> The 2-sample <strong>t-test</strong> REJECTS the hypothesis of equal mea
end

if h.kstest2==0;disp(['>> The 2-sample <strong>Kolmogorov-Smirnov</strong> test FAILS to
else; disp(['>> The 2-sample <strong>Kolmogorov-Smirnov</strong> test REJECTS the hypoth
end
```

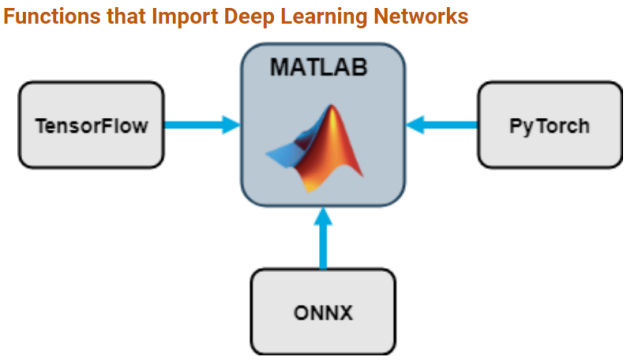
Publish reusable MATLAB code for reproducible results

To enable collaboration partners, reviewers and the community reuse your MATLAB code and reproduce your results.

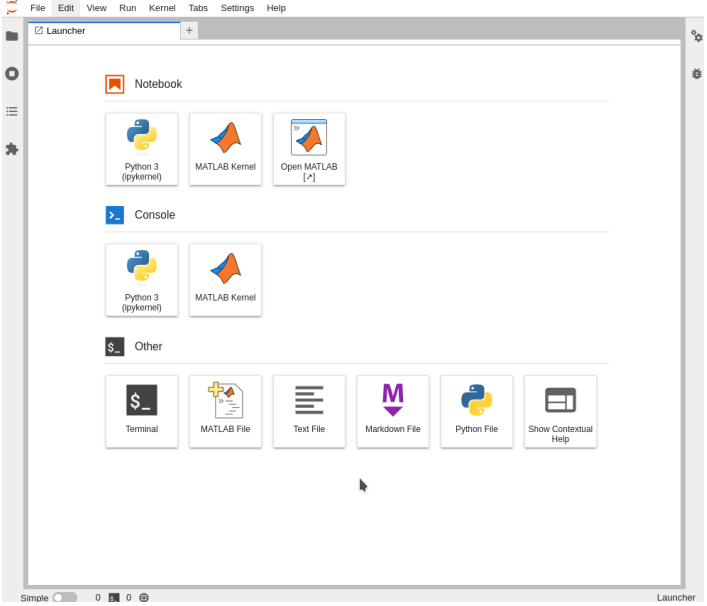
- Publish your MATLAB code (eg: on GitHub) and generate a DOI (digital object identifier) by linking it to a DOI generating portal (e.g. Figshare®, Zenodo®). Make your research output findable by including as much information as needed in the metadata. Document your code well explaining steps required to reproduce clearly and explicitly.
- Make sure you include a license for your code that specifies reuse and re-distribution rights for the code. Various open source licenses are available. BSD, MIT and Apache licenses are commonly used for open research software.
- Link your GitHub repository to File Exchange to make your MATLAB code available to MATLAB users via the Add-Ons button.



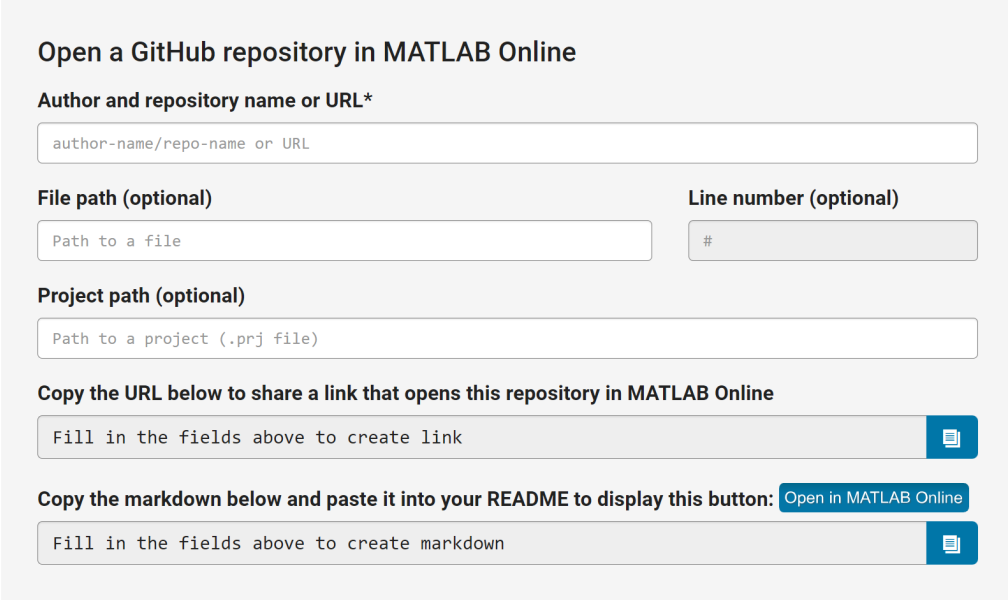
- Make your MATLAB code **interoperable**. MATLAB is interoperable with several other languages including C, Fortran and Python. MATLAB can be directly called from Python using the MATLAB Engine for Python® which is available as a PyPI® package and can be installed using the command `pip install matlab.engine` from Python. MATLAB code can also be packaged as a Python library and called from Python. Deep Learning models from other frameworks are interoperable with MATLAB either using the ONNX™ interface or via direct interfaces that exist, for example, for Pytorch® and Tensorflow™ models.



- MATLAB is interoperable with cloud architectures such as JupyterHub® and MATLAB code can also be used within Jupyter Notebooks. Here is a link to a Jupyter® notebook of the same example used here. There is an official MATLAB kernel for Jupyter Notebooks - read about it here. **To easily convert a Live Script into a Jupyter® notebook use the `export` function** .



- Run your MATLAB code on the browser directly from GitHub. Copy and paste the GitHub repo address into this app. That will generate a command, which when pasted into your README, will create a "Open in MATLAB Online™" button on your GitHub repository. By clicking on this button, users will be able to run your code in the browser on MATLAB Online.

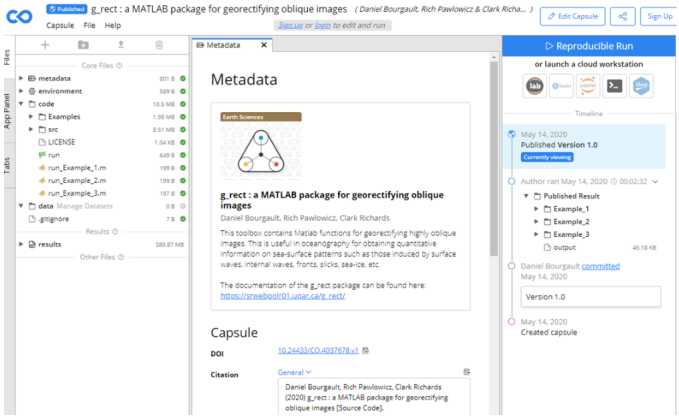


- Make your MATLAB code reproducible by using a reproducibility portals that supports MATLAB. One example is Code Ocean®. On Code Ocean, you can upload your MATLAB code including dependencies. Once uploaded, your code is tested and published as a Code Ocean "capsule" which can be run online or downloaded and run locally by users. Code Ocean also generates a DOI for your code capsule. For Live Scripts, convert the `.mlx` file into e.g., a `.m` file, a `.html` or a.ipynb file using the `export` function for best results. Here is the DOI for the Code Ocean capsule of the this code. Read more about MATLAB on Code Ocean here.

Code Reproducibility and Reuse Sites that Host MATLAB

A number of sites that focus on code reproducibility and reuse host MATLAB on the cloud. Researchers can view and download the MATLAB code for their own use, while publishers, such as Nature, use the sites to conduct peer reviews including running and verifying the code for computation-based papers.

For example, we shared information about Code Ocean a few years ago, showcasing how researchers can upload their code and run code posted by others on Code Ocean without needing to access their own MATLAB license.



- **Warning**: Before making your code available on the cloud, make sure all dependencies including any data that is needed for your code to run is uploaded along with the code. Also make sure any path and/or filenames that refer to local directories are appropriately renamed.
- FAIR standards: FAIR is an acronym that stands for **F**indable, **A**ccessible, **I**nteroperable and **R**eproducible. It is an accepted standard for research output (code, data) and is often required for your research results to be in compliance with "Open Science" standards. Adhering to the above pointers helps in making your MATLAB code FAIR

In [ ]: