Climate Modelling Diagnostics Software

User Guide

Produced by: Adanna Akwataghibe

August 24, 2019

If you come across any problems, see section 8 for possible solutions or contact me at aa14415@ic.ac.uk

Table of Contents

1

SECTION

Getting started with the Climate Modelling Software

The Climate Modelling (Diagnostics) Software is written using Python 3 and enables users to compute climate model diagnostics. This software makes use of many python modules and libraries. These libraries must be installed in order to successfully run the program. Please follow the steps below to download the package and install the required dependencies.

1.1 Quick setup

These steps give a quick setup using just Anaconda. If you get stuck on any of the steps, please start from section 1.2.

- Download / clone the repository unto your computer from here

  (https://github.com/msc-acse/acse-9-independent-research-project-AdannaAkwats)

- You must first download and install conda, for example from `http://conda.pydata.org/miniconda.html` in order to install some dependencies used in the package.

- Once conda is installed, then you can install Iris using conda on any platform with the following command:

```
$ conda install -c conda-forge iris
```

- Install basemap for plotting

```
$ conda install basemap
```

- Go into directory:

```
$ cd acse-9-independent-research-project-AdannaAkwats
```

- Install the rest of the required python package dependencies with the following command:

```
$ pip install -r INSTALL.txt
```

  If everything works then some example code is available in section 3 to run. If not, go to the step-by-step approach from section 1.2.

1.2 Downloading package

The software package is stored in GitHub, a repository hosting service. You can either download the folder **manually** or through the Linux **terminal**.

1.2.1 Manual download

1. Go to the the repository linked here. (https://github.com/msc-acse/acse-9-independent-research-project-AdannaAkwats)

2. Go the upper right hand corner and click on the **Clone or download** button. Then click on **Download Zip**.

3. Extract the folder from the zipped file that has downloaded.

4. Go to a command line terminal and go to the directory where the folder is stored. Then:

```
$ cd
    acse-9-independent-research-project-AdannaAkwats-master
```

5. To check the contents in the folder, do:

```
$ ls
```

   If you do not see the folder **ClimateModelling** in the list, then you may need to **cd** again.

   If you can see the folders **ClimateModelling** and **ClimateModelling_Parallel**, then the package has successfully downloaded.

1.2.2 Terminal download (Suitable for Linux and Mac)

1. Go to the terminal and check that since we are installing through *Git*, we have to ensure that it is installed. Type on the command line:

```
$ git --version
```

If this gives an output like `git version 2.17.1` (the numbers can be different), then git is already installed.

If git is not installed, then to install type on the command line:

```
    $ sudo apt install git-all
```

Note: If **sudo** does not work, remove it and just put `apt install git-all`. After it has finished installing, check for the version (as in the previous step) to make sure it has installed properly.

2. To download from git, you have to clone the repository (the link has no spaces):

```
$ git clone https://github.
   com/msc-acse/acse-9-independent-research-project-AdannaAkwats
   .git
```

3. Go into the folder to make sure it is cloned properly

```
    $ cd acse-9-independent-research-project-AdannaAkwats
```

and

```
    $ ls
```

If you can see the folders **ClimateModelling** and **ClimateModelling_Parallel**, then the package has successfully downloaded.

1.3 Installing all requirements

The following instructions are suitable for Linux, macOS or Windows (with an Ubuntu subsystem or Anaconda).

1. First ensure that Python is installed on the machine.

```
    $ python --version
```

If you can see a version number, then Python is installed.

If Python is not installed, then install it by:

```
    $ sudo apt-get update
    $ sudo apt-get install python3.6
```

2. You need to install Miniconda in order to get some of the libraries. (The link has no spaces).

```
    $ wget https://repo.continuum.
        io/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

```
    $ chmod +x Miniconda3-latest-Linux-x86_64.sh
```

```
    $ ./Miniconda3-latest-Linux-x86_64.sh
```

After this you should see a "Welcome to Miniconda", and pressing Enter, you will see a license to accept.

Press enter until you get to the end of the licence and type *yes* to accept.

Press enter to confirm the location. Type *yes* for the installer to initialise Miniconda by running conda init.

After packages are installed, you should see a message saying "Thank you for installing Miniconda3".

Restart the terminal so that all the changes can take effect.

After restarting, check conda has installed by checking the version:

```
$ conda --version
```

3. We can now install the rest:

```
$ conda install -c conda-forge iris
```

And for the plotting (used to overlay the continents on the map plots):

```
$ conda install basemap
```

4. To install the rest of the dependencies, make sure you are in the git repo folder `acse-9-independent-research-project-AdannaAkwats` folder and see the **ClimateModelling** and **ClimateModelling_Parallel** folders. Then do:

```
$ pip install -r INSTALL.txt
```

At the end, it should have: "Successfully installed Cython-0.29.13 Shapely-1.6.4.post2 attrs-19.1.0 cdo-1.5.3 more-itertools-7.2.0 netCDF4-1.4.2 numpy-1.16.3 pluggy-0.6.0 py-1.8.0 pytest-3.5.1 python-dateutil-2.7.3 scipy-1.2.1" seaborn-0.9.0 xarray-0.12.3" or something similar.

5. If pip is not installed, then install pip with:

```
$ sudo apt-get python-pip
```

and then go back to step 4. As before, if using `sudo` does not work, just use `apt-get python-pip` instead.

We now have all packages installed!
Some example code is available in section 3 to run.

1.4 Installing all requirements (for Windows without Anaconda or Ubuntu subsystem)

In Windows, there are 2 ways to run Python programs - either by downloading **Anaconda** (a Python Data Science Platform) or setting up and downloading the **Windows Subsystem for Linux**.

1.4.1 Using Anaconda

Follow these steps on the official Anaconda site to download Anaconda for Windows: `https://docs.anaconda.com/anaconda/install/windows/`.Download the **Python 3.7 version**.
After set up is complete, open the Anaconda command prompt, go back to step 2 in section 1.3.

1.4.2 Using the Windows Ubuntu subsystem

Follow the steps given in the official Microsoft web page `https://docs.microsoft.com/en-us/windows/wsl/install-win10`. I recommend downloading one of the **Ubuntu** terminals.
After set up is complete, open the Ubuntu terminal, go back to step 1 in section 1.3.

SECTION 2

Package structure

The package is structured as follows.

```
acse-9-independent-research-project-AdannaAkwats/
|-- ClimateModelling_Parallel/
|    |-- ... # Parallelised version of ClimateModelling/
|
|-- ClimateModelling/
|    |-- DATA/
|    |-- INPUT/
|         |-- user_function/
|         |-- input.txt
|         |-- input_example.txt
|         |-- ...
|
|    |-- RESULTS/
|         |-- ensemble_averages/
|
|    |-- directories.py
|    |-- output.log
|    |-- main.py
|
|-- INSTALL.txt
|-- README
|-- UserGuide.pdf
```

- `DATA/` : The default folder where NetCDF model output files are stored. You can redirect this directory in `directories.py` if the data is stored somewhere else.
- `INPUT/` : Folder where all input files (e.g `input.text`, mask files etc) are stored.
- `RESULTS/` : Folder where all results and files from software run is stored. In `ensemble_averages`, the NetCDF output files from program are stored.
- `output.log` : File where all progress and error messages are stored.

SECTION 3

Running the software

Enter the **ClimateModelling** folder:

```
$ cd ClimateModelling
```

There are 2 ways to run the software, either using an input file or using command line arguments.

1. To run the `input_example.txt` file which calculates the mean of air temperature in North America and plots the map and histogram:

   ```
   $ python main.py -ex
   ```

   After running, it should print a SUCCESS message on the command line and show the plots.

   NOTE: If you run this on a terminal does not allow plotting, an error will be thrown. So you have to set plot to "False" in the input file.

2. After filling the arguments in `input.txt`, the code can be run with this input by:

   ```
   $ python main.py
   ```

3. You can also use the command line, e.g. to run with the `input_example.txt` arguments:

   ```
   $ python main.py 1970 -pf E1_north_america -v
       air_temperature -e 1 -a mean -p 1 -o
   ```

   For help on what each argument means:

```
$ python main.py --help
```

4. After each run, you can check the progress or potential error messages by:

```
$ less output.log
```

3.0.1 Debugging

When the program is run successfully, the following message prints out to the terminal:

```
PROGRAM FINSISHED.
>> Progress and potential errors are logged in output.log file.
>> To open output.log, type in cmd line: less output.log
```

If no message prints out, it is because an error has been seen in the code, this may be caused by invalid inputs being seen. Open the `output.log` file created and this will give you a summary of what the code has been doing, and any errors that may have been thrown.

SECTION 4

Software input

This section details the input required by the package.

4.1 File format

The names of the files in the default DATA folder should be in the general format:

```
"{START OF FILENAME}_{YEAR}.nc"
```

where

- `{START OF FILENAME}` is the prefix of the file
- `{YEAR}` is the year

Note that the separation underscore '_' before the year does not need to be an underscore. E.g. it can be separated by a dot.

You can also add the *ensemble number* in the file name:

```
"{START OF FILENAME}_ens{NUM}_{YEAR}.nc"
```

where

- `{START OF FILENAME}` is the prefix of the file
- `{YEAR}` is the year
- `{NUM}` is the ensemble number

Value of `{NUM}`: If the number of ensembles are less than 100, then the values of `{NUM}` should be 101, 102, ..., 199. This represent the 1st, 2nd ...99th ensemble member. The '1' given at the start is given for readability. Similarly if the number of ensembles is less than 1000, then values of `{NUM}` are 1001, 1002, ...1100, 1102, ...1999, for the 1st all the way to the 999th member.
If the file name does not have the `ens{NUM}` then it is assumed that the model output is the first (and only) ensemble member.
For example, if I had 3 ensembles of the model output for air temperature at 1970. I could have the file names:

```
 - air_temp_ens101_1970.nc
 - air_temp_ens102_1970.nc
 - air_temp_ens103_1970.nc
```

If you have multiple years stored in one file then the format is:

```
"{START OF FILENAME}_{YEAR1}_{YEAR 2}.nc"
                     or
"{START OF FILENAME}_ens{NUM}_{YEAR1}-{YEAR 2}.nc"
```

where

- `{START OF FILENAME}` is the prefix of the file
- `{YEAR 1}` and `{YEAR 2}` are the start and end year of the data in the file.

Example file names:

Some valid example names are:

```
  -  dic_deltap_ens102_1d_1953.nc
  -  cCwd_ens101_2015_2065.nc
  -  pcmdi.ipcc4.mri_cgcm2_3_2a.picntrl.run1.annual.thetao_O1.1990s.nc
(Files from CMIP6):
  -  thetaoga_Omon_EC-Earth3-Veg_historical_r1i1p1f1_gn_185301-185312.nc
  -  ccb_Amon_CNRM-CM6-1_highresSST-present_r1i1p1f2_gr_200001-200912.nc
```

Invalid example names:

```
  -  dic_deltap.nc
```

A year **must** be added to the file name.

4.2 File data assumptions

- Multiple files do not have overlapped data. (i.e. 2 files should not have the same data)

- Daily or monthly increments of data, higher resolution of data input e.g. hourly is not supported.
- The time increments of data must be consistent. E.g. if you are passing 2 NetCDF files as input, **both** must have either monthly or daily data.
- Only rectangular grids are supported. If (for example) the model uses a curvillinear grid, this will have to be converted to a rectangular grid before passing into the program.

4.3 Input arguments

The following are the required arguments needed in the program input.

4.3.1 Start date of analysis

start_date : This can be in the following formats:

YYYY-MM-DD : e.g. 2020-04-12

YYYY-MM : e.g. 2020-04

YYYY : e.g. 2020

- If day is not given, the 1st of the given month will be used i.e. 2020-04 => 2020-04-01
- If day and month is not given, 1st Jan will be used as the start date i.e. 2020 => 2020-01-01

start_date1, start_date2 : 2 start dates can be put for control and future runs respectively. This is only used if Calculate index argument is set or multi-model analysis between future and historical data is performed .

In command line:

start_date or *start_date1 start_date2*

4.3.2 Prefix

prefix_of_file : This is the prefix of the file name.
prefix_of_file1, prefix_of_file2 : 2 prefixes can be put for control and future runs respectively. This is only used if Calculate index argument is set or multi-model analysis between future and historical data is performed.
If 2 prefixes are given, 2 start dates and 2 end dates **must** be set.

In command line:

`-pf` *prefix_of_file* or `-pf` *prefix_of_file1 prefix_of_file2*

For example, for file : `air_temp_1970.nc`, the prefix chosen can be `air` or `air_temp` or even `ai`. As long as the file names stored start with the prefix, it will be selected. You should ensure that if multiple model outputs are stored in `DATA/`, then the (unique) prefix of the files wanted should be used.

4.3.3 Variables

var1, var2, ... : Variables of data to analyse

In command line:

`-v` *var1 var2 ...*

The variables given must match the variable names in the files, otherwise an error will be thrown.

4.3.4 Number of ensembles

num_of_ens : The number of ensembles to analyse

In command line:

`-e` *num_of_ens*

The following are optional arguments that can be set as pleased. If any of these arguments are left blank, then it is automatically seen as false / not set.

4.3.5 End date of analysis

.

end_date : This is in the same format as the start date
If end date is not given:

- If only start year is given, the end date is automatically set to the 31 Dec of start year
- If start year and month is given, then end date is set to the end of the start month

If end date given:

- If day is not given, the end of the given month will be used i.e. 2020-04 => 2020-04-30
- If day and month is not given, 31 Dec will be used as the end date i.e. 2020 => 2020-12-31

end_date1, end_date2: **Must** be set if 2 start dates are given.

In command line:

`-end` *end_date* or `-end` *end_date1, end_date2*

4.3.6 Analysis

a1, a2, ... : Analysis performed on data
The types of analysis that can be selected:

- mean
- std (Standard deviation)
- rms (Root mean squared error)
- median

  All the analysis calculated with respect to time, unless the Spatial argument is set.

In command line:

`-a` *a1 a2 ...*

Multi-model analysis between historical and future runs: If 2 start dates and 2 end dates are given, then the difference between the analysis between the time periods are computed. **Important:** The time steps of historical and future **must** be equal in order to successfully compute the difference, if analysis is set (without spatial). E.g. if you have start dates : 1850, 2010 and end dates : 1900, 2015, the program **will** fail with a Value error since the time period of the first date is 50, which is not the same as the time period of the next date, 5. However, start dates : 1850, 2010 and end dates : 1855, 2015 will not fail as the time period is 5 years for both date sets.

4.3.7 Spatial

True/False : Analysis will be calculated spatially instead of temporally.

In command line:

```
-sp
```

4.3.8 Total ensemble stats

This setting is only applicable if multiple ensembles are given.

True/False : All ensembles are averaged (using mean) into one ensemble.

In command line:

```
-t
```

4.3.9 Plot

ens_num : Plot map, histogram and timeseries (and boxplots) graphs given specific ensemble to plot. The ensemble number must be > 0. The plots show in pop up figures when program is run.

In command line:

```
-p ens_num
```

If only 1 ensemble given, then *ens_num* = 1.
If Total ensemble stats is set, then *ens_num* = 1 plots for the averaged ensemble.
When setting plot, the data given to plotting function will be the data passed through the Analysis argument. If the analysis argument is not set, then it will be the data with no analysis.
So if the data given to the plotting function is 2D (lat, lon), then the map is plotted. If data is 1D (time) then the time series and boxplots are plotted. The histogram data will be plotted regardless of dimension of data (except some error is thrown).

4.3.10 Monthly

True/False : Set to True if data in file is stored in monthly increments, otherwise daily increments is assumed.

In command line:

```
-m
```

If there is only 1 time step in the data file, then setting Monthly has no effect.

4.3.11 Grid

latitude, longitude : Extracts the grid point that latitude and longitude lies in.
sample_points.txt : If a list of points are defined in a .txt file, then extracts the list of points. An example of what the file should contain is stored in `INPUT/sample_points.txt`.
example_file.nc : Regrids to the rectangular grid in NetCDF file given. This file should be stored in the `DATA/` folder.
The interpolation scheme used is nearest neighbour.

In command line:

`-g` *latitude, longitude* or `-g` *sample_points.txt* or `-g` *example_file.nc*

Regridding only works between rectangular grids (this holds for Sample points as well). This setting **cannot** be used in conjunction with the Sample setting.

4.3.12 Sample

latitude, longitude : Extracts the sample point that latitude and longitude lies in.
sample_points.txt : If a list of points are defined in a .txt file, then extracts the list of points. An example of what the file should contain is stored in `INPUT/sample_points.txt`.
example_file.nc : Regrids to the rectangular grid in NetCDF file given. This file should be stored in the `DATA/` folder.
The interpolation scheme used is linear.

In command line:

`-s` *latitude, longitude* or `-s` *sample_points.txt* or `-s` *example_file.nc*

Regridding only works between rectangular grids (this holds for Grid points as well). This setting **cannot** be used in conjunction with the Grid setting.

4.3.13 Mask file

mask_example.txt :Extracts region from data using a file that contains the description of multiple polygons. An example file is given in `INPUT/mask_example.txt`.

In command line:

```
-mk mask_example.txt
```

Example file in `INPUT/mask_example.txt`:

```
# Mask file expected to have only one list of polygons
# Comments in file can have '#' at the start of the line
# Each polygon is a list of tuples of integers (longitude, latitude)
# level is the depth level of mask. If this is not specified,
# then all levels of data are used.
# Note: level >= 1 and <= depth in NetCDF file
# You can also put a range of levels to choose separated by '-'
[(-25, -40), (75, -40), (-25, 40), (75,40)]
level: 2
```

If data is 3D (time, lat, lon), level would select within the time dimensions. If the data is 4D (depth, time, lat, lon), then level selects within the depth dimension.

4.3.14 Save output

True/False : Saves data output in NetCDF files and saves tables of histogram and timeseries as .txt files. If Plot set, then saves the figures as .png as well.

In command line:

```
-o
```

4.3.15 Covary

True/False : Histogram co-variance. There must be only 2 variables in Variables argument. If Plot or Save output is set, then a 2D histogram plot is created.

In command line:

```
-cv
```

4.3.16 Histogram bin selection

hist_choice : Choose the number of bins used in the histogram. The default = fd, the Freedman Diaconis Estimator. To see a list of other estimators, go to this the NumPy page.
num_of_bins: You can also put the number of bins e.g. 10, as the input.
num_of_bins1, num_of_bins2: If 2D histogram is being plotted, then the number of bins for each variable **must** be given.
histogram_bins.txt: If you want to specify the bin edges used in the histogram, you can input the file - an example file is given in `INPUT/histogram_bins.txt`

In command line:

`-ht` *hist_choice* or `-ht` *num_of_bins* or `-ht` *num_of_bins1 num_of_bins2*

4.3.17 Longitude centre

centre : Longitude to centre data on

In command line:

`-lc` *centre*

4.3.18 User function

file_name, function_name : Use function written by the user and stored in `INPUT/user_function` folder for analysis. The *file_name* is the name of file that contains function in `user_function` folder. The *function_name* is the name of function to call. There is an example file with functions in `INPUT/user_function`.

In command line:

`-u` *file_name function_name*

When the user function setting is turned on, then the analysis argument is automatically not used. If Plot argument is set, the graphs plotted are using only the original data (without the user function applied).
The user function argument does not consider the setting of the Total ensemble stats argument. This is because, the user function is applied to all ensembles.

Example file in `INPUT/user_function/example_function.py`

```
import iris

"""
This is a script that given an example of a user-given function
in the command line or input file :
file_name = example_function
func_name = simple_equation
"""

def simple_equation(cube):
    """
    Perform simple arithmetic with variables
    :param data: a dictionary of variables and their data (iris.cube)
    :return: result of equation, name of results, unit
    """

    Variables are stored in the cube dictionary
    The names of variables used have to match ones
    in the Variables argument provided in the input file or command line.

    temperature = cube['temp'].data
    salinity =  cube['sal'].data

    # Perform equation with temperature and salinity
    result = 2 * temperature + 4 * salinity


    Set the attributes of data
    name = 'result'
    long_name = 'result calculated using simple equation'
    unit = 'K'

    return result, name, long_name, unit
```

4.3.19 Calculate areas

True/False : Calculate areas of grid boxes of latitude and longitude and saves to NetCDF file `RESULTS/areas.nc`.

In command line:

```
-ca
```

4.3.20 Calculate index

index : Calculate the index given The control run is the FIRST file prefix set and the corresponding start/end date. The future run is the SECOND file prefix set and the corresponding second start/end date.
Types of inidices that can be calculated:

- enso : The Oceanic Niño Index (ONI)
- nino12 : Niño 1+2 Index
- nino4 : Niño 4 Index
- tni : The Trans-Niño Index (TNI)
- iod : Indian Ocean Dipole (IOD) Mode Index
- amo : Atlantic Multidecadal Oscillation (AMO) Index
- pdo : Pacific Decadal Oscillation (PDO) Index
- ao : Arctic Oscillation (AO; Northern Annular Mode) Index
- aao : Antarctic Oscillation (AAO; Southern Annular Mode) Index
- nao : North Atlantic Oscillation (NAO) Index

In command line:

```
-i index
```

Important: One file for control and one file for future run is expected.
If multiple files are used for the control run, then combine them into one first, this application gives the input arguments necessary.
In order to use the Calculate index functionality, CDO needs to be installed, since the indices are calculated using this package.

1. Go to `https://code.mpimet.mpg.de/projects/cdo/files`
2. Download the most recent **cdo-1.*.*.*.tar.gz** file

3. On the command line, go to the place where it is downloaded.
4. Extract data from tar zip file:

   ```
   $ tar xvzf cdo-1.*.*.*.tar.gz
   ```

5. cd into newly extracted directory

   ```
   $ cd cdo-1.*.*.*/
   ```

6. do:

   ```
   $ make
   ```

7. do:

   ```
   $ make install
   ```

8. To check if installation has happened successfully, do

   ```
   $ cdo -V
   ```

5

SECTION

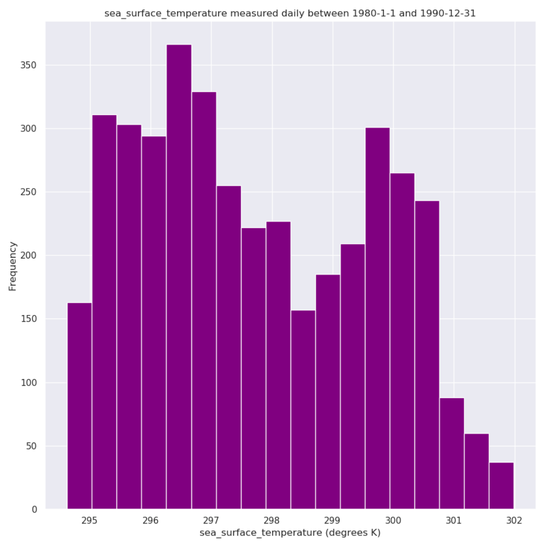
Some applications

The following are some scenarios.

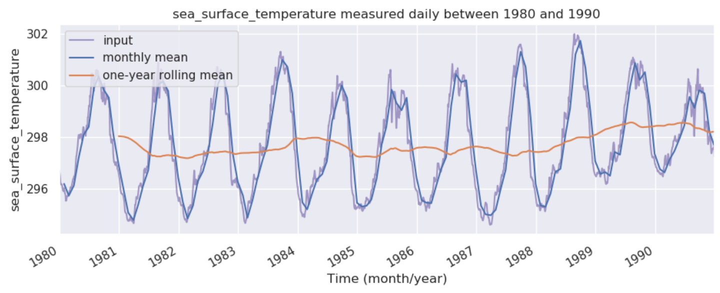
5.0.1 Extract sea surface temperature between 1980 and 1990 at grid point (14.5, -27.5) and output a time series, histogram and NetCDF file with mean data

```
# REQUIRED ARGUMENTS
# ----------------------------------------------------
Prefix: sst
Start date of analysis: 1980
Variables: sst
Number of ensembles: 1
#
# ----------------------------------------------------
# OPTIONAL ARGUMENTS
# ----------------------------------------------------
End date of analysis: 1990
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid: 14.5, -27.5
Sample:
Mask file:
```

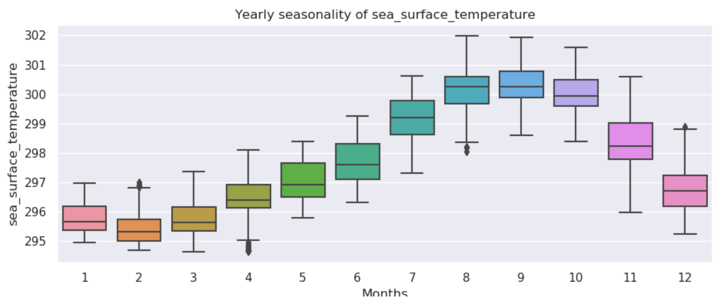
```
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```



(a) Histogram



(b) Time series

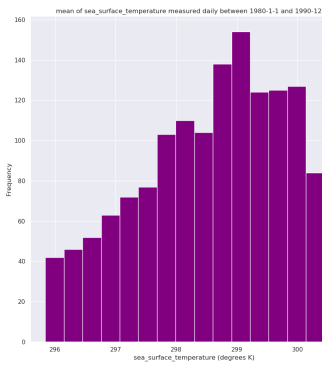


(c) Box plots

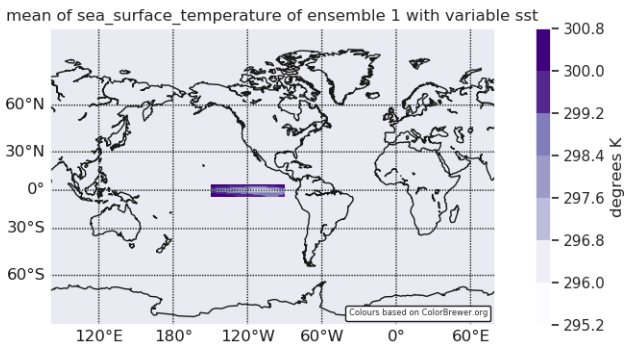
Figure 1: Plots generated for scenario 5.0.1

5.0.2 Calculate mean sea surface temperature between 1980 and 1990 using region of NINO-3 and output histogram, map and NetCDF file with mean data

```
# REQUIRED ARGUMENTS
# ----------------------------------------------------
Prefix: sst
Start date of analysis: 1980
Variables: sst
Number of ensembles: 1
#
# ----------------------------------------------------
# OPTIONAL ARGUMENTS
# ----------------------------------------------------
End date of analysis: 1990
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample:
Mask file: nino3_mask.txt
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```



(a) Histogram

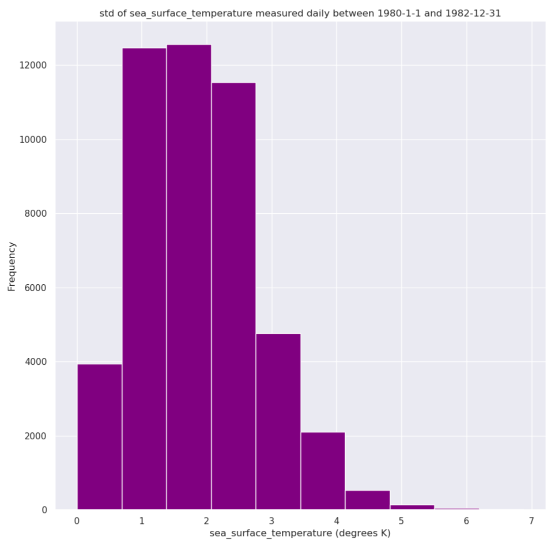

mean of sea_surface_temperature of ensemble 1 with variable sst
60°N
30°N
0°
30°S
60°S
120°E
180°
120°W
60°W
0°
60°E
Colours based on ColorBrewer.org
300.8
300.0
299.2
298.4
297.6
296.8
296.0
295.2
degrees K

(b) Map with NINO-3

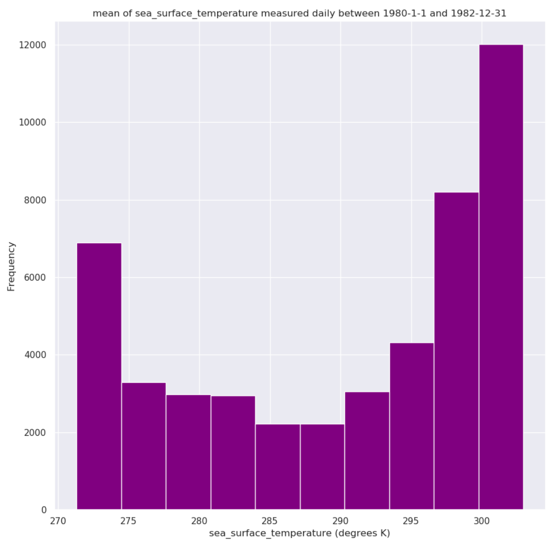
Figure 2: Plots generated for scenario 5.0.2

5.0.3 Calculate mean and std of sea surface temperature between 1980 and 1982, with longitude centred at -30 degrees east. Histogram bins = 10 and grid boxes areas calculated. Outputs are NetCDF files containing areas and mean and std.

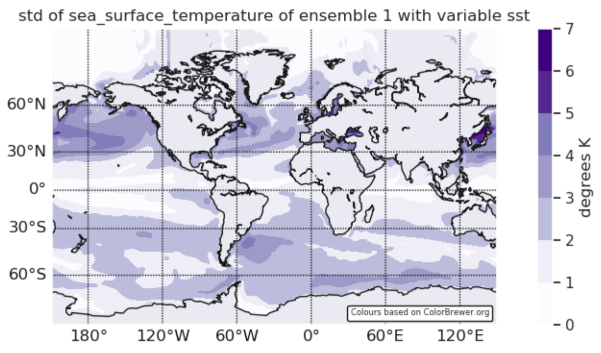
```
# REQUIRED ARGUMENTS
# ----------------------------------------------------
Prefix: sst
Start date of analysis: 1980
Variables: sst
Number of ensembles: 1
#
# ----------------------------------------------------
# OPTIONAL ARGUMENTS
# ----------------------------------------------------
End date of analysis: 1982
Analysis: mean, std
Spatial:
Total ensemble stats:
Plot:
Monthly:
Grid:
Sample:
Mask file:
Save Output:
Covary:
Histogram bin selection: 10
Longitude centre: -30
User function:
Calculate areas: True
Calculate index:
```



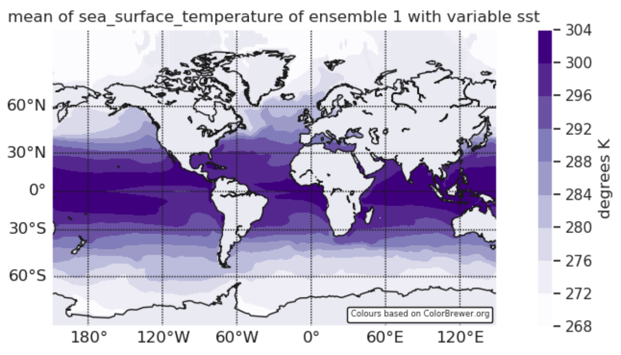
(a) Histogram of Standard deviation



(b) Histogram of mean


std of sea_surface_temperature of ensemble 1 with variable sst
degrees K
Colours based on ColorBrewer.org

(c) Map of Standard deviation


mean of sea_surface_temperature of ensemble 1 with variable sst
degrees K
Colours based on ColorBrewer.org

(d) Map of mean

Figure 3: Plots generated

5.0.4 Combine consecutive files of sea surface temperature into one

Given data files

```
    - sst_1950_1959.nc
    - sst_1960_1969.nc
    - sst_1970_1979.nc

# REQUIRED ARGUMENTS
# ----------------------------------------------------
Prefix: sst
Start date of analysis: 1950
Variables: sst
Number of ensembles: 1
#
# ----------------------------------------------------
# OPTIONAL ARGUMENTS
# ----------------------------------------------------
End date of analysis: 1979
Analysis:
Spatial:
Total ensemble stats:
Plot:
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

Output file generated: `sst_1950_1979.nc`.

For more applications/scenarios, go to the integration tests section.

SECTION 6

Tests

6.1 Unit tests

The testing framework Pytest was used to construct the unit tests written. In order to run the unit tests, Pytest has to be installed with:

```
$ pip install –U pytest
```

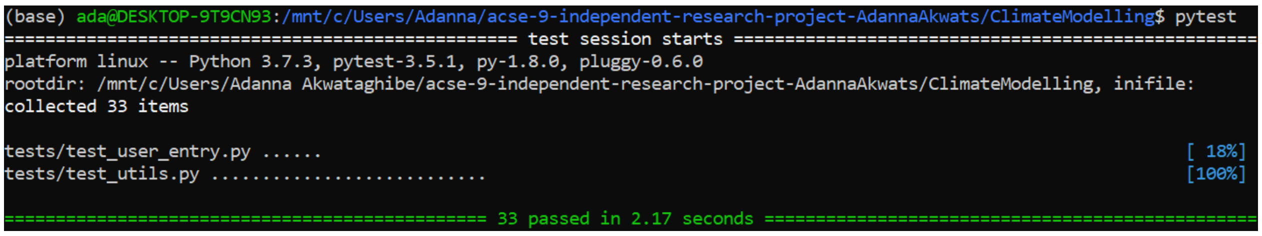
After installation,

```
$ cd ClimateModelling/
```

and finally run:

```
$ pytest
```

You should get output similar to:



```
(base) ada@DESKTOP-9T9CN93:/mnt/c/Users/Adanna/acse-9-independent-research-project-AdannaAkwats/ClimateModelling$ pytest
================================================== test session starts ===================================================
platform linux -- Python 3.7.3, pytest-3.5.1, py-1.8.0, pluggy-0.6.0
rootdir: /mnt/c/Users/Adanna Akwataghibe/acse-9-independent-research-project-AdannaAkwats/ClimateModelling, inifile:
collected 33 items

tests/test_user_entry.py ......                                                                                   [ 18%]
tests/test_utils.py ...........................                                                                   [100%]

=============================================== 33 passed in 2.17 seconds ================================================
```

Figure 4: Pytest command line output

6.2 Integration tests

We did some *scenario* testing, where we constructed test cases, ran them and generated an expected output. During the development of the code, these scenarios were run frequently to ensure that the actual output matched the expected output.
The tests are stored in `ClimateModelling/integration_tests`.

Integration tests folder structure

```
ClimateModelling/integration_tests/
|-- Test1_mean_spatial_monthly/
|-- Test2_std_monthly/
|    | -- data/
|    | -- results/
|    | -- plots/
|    | -- input.txt
|    | -- other input files ...
| ...
```

- `data/` contains the NetCDF data files used when running the program
- `results/` contains the programs NetCDF outputs
- `plots/` contains the .png and .txt files for maps/histogram/timeseries generated.
- `the input files` consists of the `input.txt` file as well as any `mask.txt`, `sample.txt`, `histogram.txt` files used.

You can use the data and the input given to run some applications. And then check that the output generated matches the plots and results.
The test cases

1. `Test_mean_spatial_monthly` : Calculate the spatial mean of air pressure at convective cloud base between 1979 and 2014 measured monthly, with histogram bin size of 10.
2. `Test_std_monthly` : Calculate the standard deviation of air pressure at convective cloud base between 1979 and 2014 measured monthly.
3. `Test_multimodel_median_monthly` : Calculate the multi model median difference for air pressure at convective cloud base measured monthly, between 1950-1959 (control) and 2010-2014.
4. `Test_multimodel_mean_spatial_monthly` : Calculate the multi model spatial mean difference of air pressure at convective cloud base measured monthly, between 1950-1959 (control) and 2010-2014.

5. `Test_enso_index_monthly` : Calculate the ENSO index for air pressure at convective cloud base measured monthly, between 1950-1959 (control) and 2010-2014.

6. `Test_gridpoint_monthly_areas`: Extract the grid point (20, 20) for air pressure at convective cloud base between 1979 and 2014 measured monthly.

7. `Test_samplefile`: Regrid air pressure at convective cloud base between 1979 and 2014 measured monthly to grid of model output for North America.

8. `Test_mask_mean_monthly_centred_0.0` : Calculate the mean of air pressure at convective cloud base between 1979 and 2014 measured monthly within region given by the mask and with the longitude centred at 0.0.

9. `Test_mean_total_spatial` Calculate the spatial mean of 2 ensembles of air temperature at 1970, giving one average over all ensembles.

10. `Test_mean_4D_mask_2vars` : Calculate the mean of temperature and relative humidity at 2193 within the region given by mask. (The grid used is 4D (depth, time, lat, lon).

11. `Test_4D_user_function` : Perform equation between temperature and relative humidity given by user function of temperature and relative humidity at 2193. (The grid used is 4D (depth, time, lat, lon).

12. `Test_4D_2DHist` : Calculate the mean of temperature and relative humidity and create 2D histogram between the variables.

13. `Test_4D_2DHist_file` : Create 2D histogram between temperature and relative humidity using user defined histogram bins.

14. `Test_total` : Combine 2 ensembles of air temperature at 1970, giving one average over all ensembles.

7

SECTION

Running on different setups

The program was set up and run on different types of machines:

1. PC
2. Workstation (ssh-ed into Linux workstation)
3. HPC (High Power Computing) cluster

If you are running the program on a PC or Workstation, then the program should run without any additional setup.

Running in a HPC requires a little more care. The parallelised version of the code is stored in the `ClimateModelling_Parallel/` folder. There are some requirements in order to run on HPC:

7.0.1 Storage

When using multiple ensembles (especially for already large model outputs), the output given by the program can get very big - depending on the analysis being done. In some HPC systems, the storage used for the cluster job needs to be specified ahead of time. The size of the input files + the output needs to be taken into account when specifying the space allocated for the job.

7.0.2 Number of Processors

The number of processors used in the code, can be set in the file
`ClimateModelling_Parallel/parallel_setting.py`.

7.0.3 Plotting

On a NPC cluster, the graphical display for plots is not available. Therefore, the Plotting argument **must not** be set, in order for the program to run successfully.
For example, this application would need to be changed to the input file below to work on a HPC.

```
# REQUIRED ARGUMENTS
# ----------------------------------------------------
Prefix: sst
Start date of analysis: 1980
Variables: sst
Number of ensembles: 1
#
# ----------------------------------------------------
# OPTIONAL ARGUMENTS
# ----------------------------------------------------
End date of analysis: 1990
Analysis: mean
Spatial:
Total ensemble stats:
Plot:                # The plot is not set
Monthly:
Grid:
Sample:
Mask file: nino3_mask.txt
Save Output: True
Covary:
Histogram bin selection: fd
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

8

SECTION

Troubleshooting

Below are some bugs found in the code that you may encounter.

8.0.1 (Seemingly) Non-unique prefix

Say you have 2 files `air_temp_1970.nc` and `air_temp2_1970.nc` and only wanted to analyse the first one. Setting the Prefix argument to `air_temp` selects both files instead of just the files. Instead just use the prefix: `air_temp_`. You just need to choose a unique prefix to the first file. The entire file name can be chosen as a prefix if you are in a special case where the file names are too similar.

8.0.2 Number of histogram bins not equal to what selected

When choosing the number of bins as an integer, the number may not be exactly the same as what you set as the argument, the NumPy function used to plot histogram seems to do some sort of auto bin calculation.
Alternatively, if one of the estimators is chosen, sometimes the number of bins generated causes a memory error. If this is the case, the program reverts to plotting the histogram with about 10 bins.

8.0.3 Changes to dimension names

When saving the output file, the Python package xarray sometimes saves dimensions as dim_0 and dim_1 , not by the original dims names longitude/latitude.

8.0.4 Plot not displayed on Windows 10

In order to display plots in Windows (from the Ubuntu subsystem), you need to download Xming for Windows from `https://sourceforge.net/projects/xming/`. Then

1. Start XLaunch
2. Select **Multiple Windows**
3. Set **Display number** to **0**
4. Click Next
5. Select **Start no client** option
6. Click Next
7. Do not change anything in **Specify Parameter settings** , Click Next again.
8. Save configuration - Click on this saved file to start set up XLaunch if you log out of your system.

Now that the connection is made, go to the linux cmd and type

```
$ export DISPLAY=:0
```

Now running the code with plotting on should give a display.
If you are running on a Linux terminal using ssh, then this web-page linked gives the steps to connect to Xming when ssh-ing using putty.

8.0.5 Permission error given when writing/finding file

If a permission error is thrown when writing/finding file, make sure there are no spaces within your file path.

8.0.6 ValueError: Variable '..' has multiple fill values

This is a problem with the Python package xarray. You will need to remove the 'missing value' attribute since xarray fails when both fill and missing value are set.

8.0.7 Processes when running in parallel unable to transfer data

The code is parallelised between ensemble data. So each processor does work for a single ensemble. This means that if the data is too big, then transferring data between processes can become an issue. This can lead to the code throwing a `multiprocess.pool` memory error, or the code may stay in a hanged state - and take a *very* long time to load.

8.0.8 AttributeError: attributes of masked are not writeable

If this error is thrown: *AttributeError: attributes of masked are not writeable*, then the sampe/grid point or mask has values of all NaNs. A new point or mask regions that is not encompassed just of NaNs needs to be chosen.

8.0.9 ValueError: operands could not be broadcast together with shapes (**,) (**)

If you are trying to perform multi-model analysis, the error is most likely due to the different time period between the control and future run. Go to the Multi model section for more help.