Imperial College London

MSc Applied Computational Science and Engineering

ACSE-9 Independent Research Project

A new Software Package for Climate Modelling Diagnostics (EZclim)

Author:
Adanna Akwataghibe
aa14415@ic.ac.uk
Github: AdannaAkwats

Supervisor:
Dr. Yves Plancherel

August 30, 2019

Abstract

Keywords:Climate models, outputs, diagnostics, EZclim, software

Climate model outputs are not only becoming more publicly available, they are also garnering popularity among non-climate modellers. With the increase in model output data, the rise in ensemble simulations and high frequency data in the model output, it has become more important to develop software that appeals to both climate and non-climate modellers, and to analyse the model outputs in a useful and efficient manner.

Thus, a new software package for Climate modelling diagnostics, *EZclim* was developed during this project. EZclim is a stand-alone package. It is developed using Python 3 and builds on existing climate model software: xarray and iris. EZclim takes the novel approach of combining the steps of reading, pre-processing and calculation of climate modelling diagnostics into a singular program, thus eliminating the need to use multiple software. EZclim provides a simple user interface that caters to climate modellers and those with limited experience in climate science. The interface is highly customisable for users that are more advanced in Python.

EZclim calculates climate modelling diagnostics like the mean, median, standard deviation and root mean squared error. The program output is saved in NetCDF files and can be represented visually using maps, histograms, time series and box plots. The software has the ability to regrid, extract regions, calculate climate indices, rotate the grid among other functionalities. Furthermore, members of the climate modelling community were engaged to ensure that the functions provided by the software were necessary and useful. Verification testing was carried out to ensure high quality product development.

---

1 Introduction

Climate models (Appendix A) are a core component of climate science. They help climate modellers understand how human influence is affecting the Earth's climate [1]. There is increasing interest in predicting future climate not just by climate modellers but also by insurance companies, energy companies, hedge funds, farming businesses, foreign aid - individuals that are not traditionally involved in climate modelling, but will either benefit or suffer from climate change. There has been a democratisation and internationalisation of the climate modelling effort. Climate model output is becoming more accessible, especially with cloud computing and more high-performance computing (HPC) systems becoming increasingly available in companies and universities.

Since the first climate model in 1994 [2], the number of climate models, their sizes and the number of numerical experiments to test climate scenarios, model performance and sensitivities have increased rapidly. As more and more models are being run, the spatial and temporal resolution and the grids types underlying the structure of these models have also been improving. The computational power used to run the models has also increased, enabling multiple runs of the same model, or, *ensembles*. An *ensemble* of a model involves different variations of the same model, run with either a change of initial conditions or parameter values. Ensembles are used to understand the uncertainty of the climate simulations. Each model output of an ensemble varies from the other ensemble members. Averages and analyses can be computed for the ensemble, giving an output that aids the understanding of the uncertainties within the models. For example, the average of an ensemble gives a good indication of the direction of change and the standard

deviation of the ensemble gives an indication of the robustness of the models [3]. In addition to the model outputs, there is an increase in the type of experiments available and therefore an explosion of available data to analyse. For instance, the Max Planck Institute Grand Ensemble (MPI-GE) [4] is currently the largest ensemble of a comprehensive climate model, with 100 members of simulations performed between 1850 and 2005. The large number of ensemble members is used to estimate the internal variability of the climate system. It is important to understand the climate variability of a system because the goal of climate modelling is to detect an attribute and extract the climate change signal in the noisy world, which help make dependable statistical predictions about the future. There are two types of ensembles: *Multi-model ensembles* and *Perturbed physics ensembles*. *Multi-model ensembles* or equivalently, *ensembles of opportunity* explore the *structural uncertainty* of models. The structural uncertainty is present when the uncertainty in the model output stems from the different numerical techniques or resolution of the model. These uncertainties are limited in how they are used because the model structure cannot be changed [2]. In a *Perturbed physics ensemble*, changes in the parameter values affect the model output [2]. This particular ensemble is used to show the *representational uncertainty* - the uncertainty when changing the parameter values.

The Coupled Model Intercomparison Project (CMIP) was developed as a means to standardise the model experiments so that the setup for each model would be the same.The intercomparison project created a coordinated global community-wide experiment. It allows each modelling centre to conduct experiments that transcend their own models' capabilities, by having other modelling groups perform experiments with the same setup on their own model. The most current model experiments are in CMIP 6. CMIP goes through a new iteration every five or six years. In each iteration, all the modelling centres agree to run specific sets of experiments. In earlier iterations, CMIP experiments would be to model the impact of a 1% annual increase in atmospheric CO2 emissions and analyse between models. More recent CMIP experiments have increased the emission scenarios, their details and the number of models being compared [5]. Data management and accessibility have been streamlined. For example, the output of CMIP simulations is now stored in distribution centres globally by the *Program for Climate Model Diagnostics and Intercomparison* (PCMDI). Figure 1 illustrate reasons that contribute to the increase of the models: resolution, model processes, variables, computing power and data as the CMIP experiments evolve.

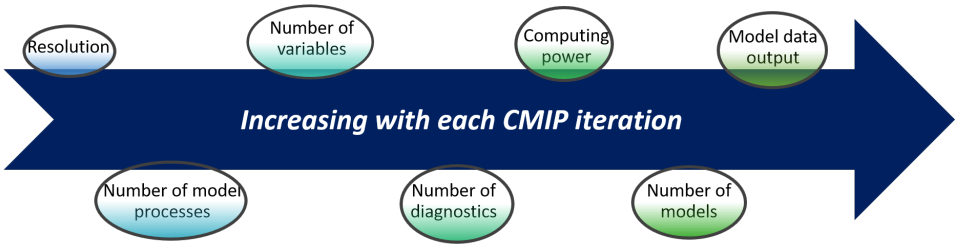


Figure 1 CMIP models evolution

In summary, model output has increased dramatically in combination with the amount of model simulations available and the improvements in computing power. There has also been an evolution and diversification of model users. The Climate Modelling Diagnostics software: *EZclim* developed here aims to handle and interpret model output in an efficient and reproducible manner. The software also aims to provide a user friendly environment to analyse climate models. The goal is the create a simple, yet customisable user interface to cater to a broader community of both climate modellers and those with limited experience in climate modelling.

1.1 Objectives

The objective is to develop a computationally efficient software package to facilitate climate model analysis. This involves:

- Calculating climate model diagnostics e.g. mean, standard deviation, etc. The software should also be highly customisable by enabling the integration of user given functions.
- Creating output for the user that effectively (and visually) summarises the analyses calculated e.g. histograms, time series, maps etc.
- Engaging with the climate modelling community to understand and isolate the common needs of the modellers.

2 Existing software

A few climate model analysis software packages already exist - Iris [6], xarray [7], ESMValTool [8] and the latest Matlab package by Greene et al. [9]. Of these, Iris and xarray are currently the most popular choices. Iris specifically caters to climate modellers or at least those with some level of experience in the field. Iris is a Python library used for working with Earth Science data. It contains functions to **aggregate**, and **interpolate** the data amongst other things [6]. Aggregating data (or combining data within the time or spatial axis) is useful when calculating the ensemble means. Interpolation is necessary when calculating a value of the variable at a sample or grid point in the latitude-longitude grid. It also builds on Numpy [10] and dask [11] to scale for both single CPU systems and for multiple processors [6].

xarray is a more recently developed Python package that focuses on working with labelled multi-dimensional arrays in a simple (and more straight-forward) manner. The package is tailored to work with NetCDF files and deals with reading and writing of data, interpolation, data arithmetic, time series and more. Inspired from pandas [12], it also uses dask [11] for scalability [7]. Since xarray is specialised in multi-dimensional arrays and NetCDF files, most of the functions in the package will be useful to incorporate in EZclim.

ESMValTool is a new initiative geared at integrating both Iris and xarray. In future developments, EZclim can be integrated into ESMValTool.

3 A new software package: *EZclim*

3.1 Philosophy/Rationale

Using Iris and xarray is quite involved, the user needs to know python and spend a significant amount of time learning how to call and use functions in python. The output of the climate models are usually saved as NetCDF files. NetCDF or *The Network Common Data form* is a library that deals with the

storage and retrieval of data in arrays. Data saved in NetCDF files are contained in a `dataset`. This dataset contains: dimensions, variables, global attributes (description, history, file name) etc. Currently users need to know how to read NetCDF files and pre-process their data. This means having to work with several software in each step [13] - reading the data files, pre-processing the file, computing the relevant diagnostics all require using different software and it can take a long time to determine what software is best for each step.

EZclim aims to alleviate all these problems by catering to users with/without experience in Python. The software also takes the novel approach of combining the steps of reading, pre-processing and calculation of climate modelling diagnostics into a singular program, thus eliminating the need to use multiple software.

Climate scientists are part of the cutting edge developments which consist of increase in climate model outputs and their resolutions (hourly, daily or monthly). However, model outputs are more publicly available and garnering popularity among non-climate scientists. These users with less experience in climate science may not have enough knowledge on how to obtain the model output analysis that they want. EZclim aims to shorten the gap between the climate scientists and users with limited knowledge in climate science.
When calling EZclim, the steps visible to the user are:

```
Input data -> Select options -> Get output
```

Notice how minimal these steps are. The user just needs to input the NetCDF model output data and set the options and relevant analysis to be performed. And finally, get the NetCDF data that the program outputs and any plots produced. The user does not need to know how to program in Python, although if they do have that experience, in the `Select options` phase, they can input their own python code for EZclim to run as a diagnostic. A more detailed chart will be shown in the Design, Implementation and User interface section.

3.2 Portability, installation and dependencies

EZclim is a stand-alone package. It is developed using Python 3 and builds on xarray and iris. The file `INSTALL.txt` (Appendix B.1) contains the dependencies required. An external software recommended for compilation is Anaconda. It is used to install `iris`. The installation instructions are in the Git repository containing the project [14] in the `README.md` and in more detail in the `UserGuide.pdf`.
GitHub was used as the version control software to keep track of the work and any code changes that were made. The code on GitHub was synced up with Travis CI [15] to build and test the software automatically each time the code was changed and pushed to Git. Using Travis helps verify whether any new additions to the code breaks.
EZclim was run and tested on multiple operating systems: Windows, Linux and Mac OS. It was also run on a PC, workstation and in a HPC system to ensure completeness and reproducible results. On HPC, the plotting option has to be turned off, since HPCs have no display servers. On Windows, if a Linux subsystem or a work station is ssh-ed into, and the plotting is turned on, the display may not be found. To remedy this, Xming [16], an X11 display server needs to be downloaded and installed on the PC. Then the X11 display server can be connected in order to display the plots. The set up of Xming is discussed in the `UserGuide` in the Git repo [14].

3.3 Design, Implementation and User interface

The agile software development approach was used throughout the project. Some key ways the approach was followed were:

- Meeting every week with either a new functionality or an improvement upon already developed routines. This was to ensure continuous delivery and improvement of the software.
- Focusing on users by having one-to-one meetings, collecting feedback from user testing and getting specific applications based on their scientific interest. This was used to ensure functionality of the software.
- Constant addition of new functionality, either from supervisor meetings or from user applications. Changes and adaptations of requirements were accepted.

The software was split into **four main components** (Figure 2) in order to break the code into smaller, modular chunks. The modularity of the code contributes to the object-oriented design of the software and is part of the agile software development. The four components used were:

1. **USER INPUT:**
   - Receive the NetCDF files given by the user.
   - Set the options and diagnostics selected by the user.
2. **EXTRACT:** Extract the relevant data from files using the options set by the user.
3. **CALCULATE DIAGNOSTICS:** Compute the analyses of the extracted data.
4. **SAVE OUTPUT:** Save the output of the analyses as NetCDF files and save plots (if the user selects the option).

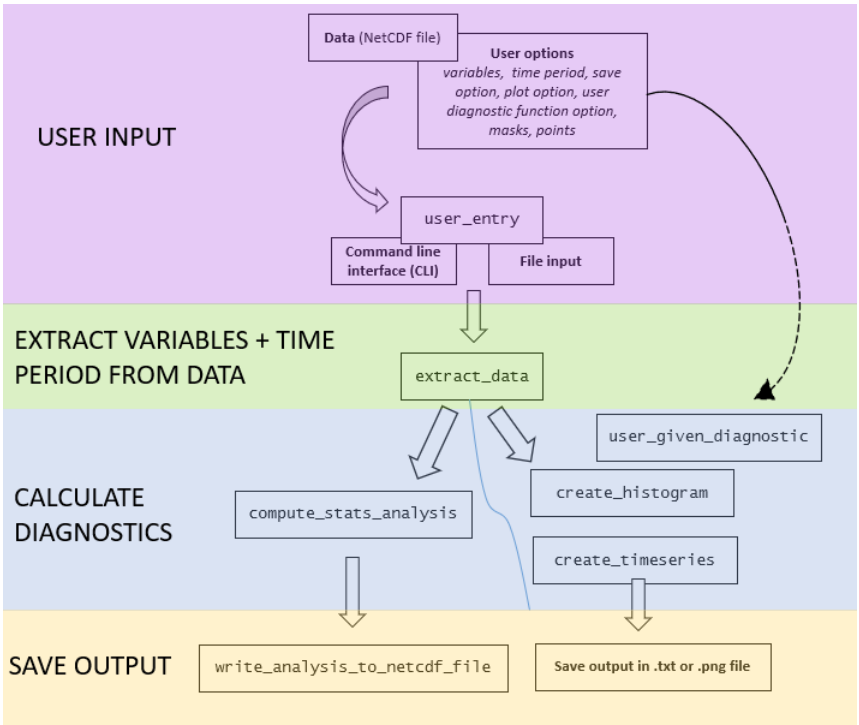


Figure 2 High level structure of code

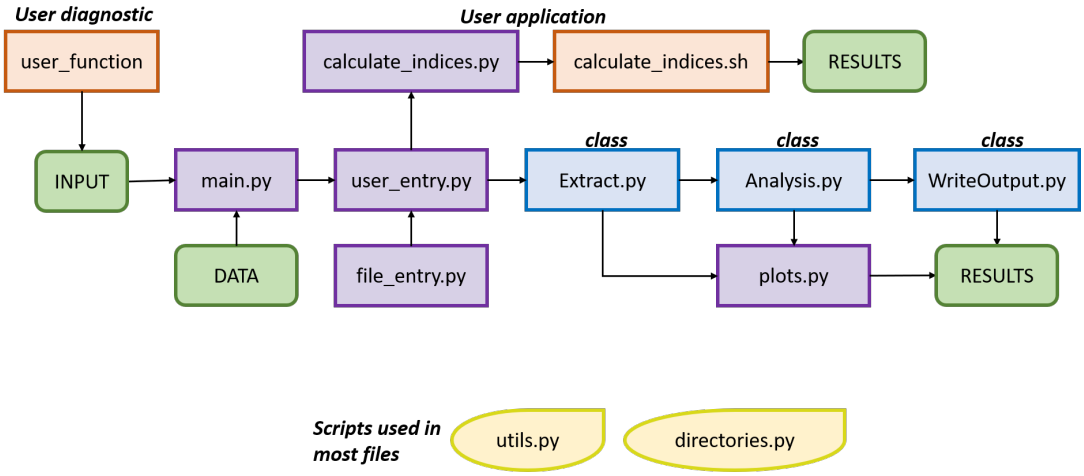


Figure 3 Low level structure of code

Each component is taken care of by separate python scripts or classes. Self-contained classes and scripts help effectively add new code without changing the overall structure of the main code, which enables efficient development. The file structure of the code is shown in Figure 3.
The 3 main folders used are: `DATA`, `INPUT` (where user input are stored and options are selected) and `RESULTS`. The path of these folders are defined in the `directories` script, and are customisable by the user.

The `DATA` folder holds all the original NetCDF files given by the user.

In contrast, the `INPUT` folder holds masks, list of sample/grid points, histogram bins defined by the user. Most importantly, `INPUT` contains the input file (and an example) which contains all the options that the user can select. The input file that the user fills in is in Figure 4.
In Figure 4, the arguments are split into required and optional. The arguments highlighted in orange are used in the `Extract` class, the options in green are used in `Analysis` class and the blue options are used in the `plot` script. Yellow signifies the use in `WriteOutput`. The variables, number of ensembles and total ensemble stats (all members of the ensemble are averaged) options are used across multiple classes. The `calculate_index.py` script is run only if the calculate index option is set.
An alternative to using the input file would be to use the command line with the right arguments set.
The python script `user_entry` handles the parsing of the command line arguments given by the user, as well as input file read in `file_entry` script. The pseudo code for this script is shown in Figure 5.

The python class `Extract` gets the right files from the `DATA` folder. Then it opens the files and extracts the relevant data according to the options set in the input file or command line. The class also deals with the extracting of regions or points in the data. Figure 6 shows the pseudo code of the function `extract_data` called in `user_entry`. The extracted data returned from `extract_data` is then passed to the `Analysis` class, if the user has set the analyse argument in input - the user may not always want to statistically analyse the data. For example, they may just want to extract a region and get the output. If the analysis option is not selected, the extracted data is passed to the `plot` script.
`Analysis` deals with computing statistical averages like the mean, median, root mean squared error (RMS) and standard deviation (STD). It also handles calling any user given functions. The class also deals with

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

Figure 4 input.txt

the calculation of multi-model averages, that is computing the difference between the analyses of future and historical model outputs. The routines contained deal with the computation of analyses *per ensemble*, where the analysis is performed for each ensemble separately and *across ensembles*, where all ensemble (analysis) results are averaged to get one final output. The output of these routines are then passed to the `plot` script. Figure 7 gives the pseudo code of the main function called in `Analysis`.

The functions for creating histograms, time series, maps and box plots are defined in `plot.py`. The plots are saved in the `RESULTS` folder as `.png` figures, and for histograms and time series, the tabular data is given as a `.txt` file.

The extracted data and the analysis are also passed to the class `WriteOutput`, where these data are converted from iris.cubes (a multidimensional labelled array) to NetCDF files, saved in `RESULTS`. NetCDF output could be made as input for analysis and plotting again if needed.

The `utils` script contains a collection of self-contained functions which are used to avoid duplication of code that will be called multiple times. It also handles functions that do not fit in ergonomically into any

```
def user_entry():
    ...
    From input.txt or input_example.txt files
    if input file is used:
        get arguments from input file
    else:  directly from command line
        parse the command line arguments
    ...
    Print the arguments set in the output.log file
    ...
    Extract data
    extract = Extract(...)
    data = extract.extract_data(...)
    ...
    Analyse data
    if analysis option is set:
        a = Analysis(...)
        if user diagnostic option is set:
            stats = a.compute_user_analysis(data, ...)
        elif multi model statistical analysis set:
            stats = a.calc_stats_difference(data, ...)
        else:
            stats = a.compute_stats_analysis(data, ...)
    ...
    Plot data
    if plot or output is set:
        create_histogram(data, stats, ...)
        create_timeseries(data, stats, ...)

        if plot:
        plot_map(data, stats, ...)
    ...
    Write data to output
    if output option is set:
        w = WriteOutput(...)
        w.write_analysis_to_netcdf(data, stats, ...)
    ...
```

Figure 5 Pseudo code of user_entry.py

```
def extract_data(prefix, start_date, end_date,
                            number of ensembles...):
    Get files in all ensembles using file prefix
    start and end date
    ...
    for files in each ensemble:
        get dataset from the files (using xarray)
        get time slice (using start and end date) in data set
        ...
        if longitude centre option is set:
            centre the data around this
            longitude (using iris)
        ...
        if mask option is set:
            self.get_mask(...)
        ...
        if sample or grid option is set:
            if sample set:
                interpolate using linear
                interpolation (using iris
            if grid set:
                interpolate using nearest neighbour
                scheme (using iris)
        ...
        return extracted data
```

Figure 6 Pseudo code of main function in Extract class

other class or script.
The added user application for calculating indices using both the bash script `calculate_indices.sh` and a python wrapper. See more details in Software capabilities below.

The scalability of the code determines the lifetime of the software. To improve scalability, EZclim contains classes that follow the *single responsibility principle* [17] - that is, each class is focused on a main goal.

Two features were added to EZclim to improve the user experience when running the program. The first feature is the input file. Setting arguments in the input file makes it much clearer and easier to run the program, as opposed to using command line arguments. Input files can also be saved and used in later runs. The other feature stemmed from the wait times when running the software for large ensembles. A progress bar was added, which updates after each major operation i.e. after extraction, analysis, plotting and saving. The progress bar provides the user with visual feedback needed to improve their experience running the software.

```
def compute_stats_analysis(analysis_list, total, spatial, ...):
    if total and analysis_list is not None:
        Get the average of all ensembles into one
        return self.calculate_avg_ensembles(...)
   elif not total and not analysis_list:
        return None, False, None
    ...
    Compute analysis for each variable
    for each variable in all ensembles:
        for each a in analysis_str:
            if a == 'mean':
                if spatial:

                else:
            ...
    if total:
        Combine all analysis from all ensembles
        into one ensemble output
        return self.compute_total_stats_analysis(...)
```

Figure 7 Pseudo code of main function in Analysis class

3.4 Testing

Unit and integration tests were used to verify the software. To ensure that the user specifications and requirements were met, the software was evaluated using user testing.

3.4.1 Unit tests

The unit tests stored in the `tests` folder are written using the Pytest framework. These tests are run on Travis each time the code is pushed on Git. The `utils.py` and `user_entry.py` scripts were tested in the files `test_utils.py` and `test_user_entry.py`.

3.4.2 Integration tests

Integration/system tests were written to check that the software components work correctly. These tests are stored in the `integration_tests` directory. Each test is arranged as follows:

```
Test_scenario/
|-- data/   contains input NetCDF files
|-- results/ contains output NetCDF files
|-- plots/  contains figures saved as .png and .txt files
|-- input files ...  input.txt, mask files, histogram bin files
```

The input files are included so that the output can easily be reproduced. After any substantial changes in the code occurs, the scenario presented in each test file is run and the expected output given in the `results/` and `plots/` folders is compared with the actual output of the software. New tests are written when a new functionality is added to EZclim.

3.4.3 User testing

The code was tested by engaging with some scientists in the climate modelling community, namely **Dr. Keith B. Rodgers** in the new modelling centre IBS Center for Climate Physics in Korea, He also provided some oceanographic climate model outputs that EZclim is tested with. **Dr. Emma Cavan** a research associate in Ecosystem Modelling, in the Department of Life Sciences at Imperial College London was included as one of the user testers, as a non-climate modeller.
One-on-one meetings were prepared with each user. The installation, set up and running of the program were walked-through with the users. User testing highlighted new needs for some applications such as the multi-model analysis between historical and future climate model outputs, this functionality was added. User tests also revealed different installation requirements based on OS used, since MacOS and Linux systems were used by the users, while EZclim was primarily developed on Windows. In the User Guide, how to install, set up and run on multiple OS is detailed and accounts for the various OS requirements.

3.5 Software capabilities

3.5.1 Leap years

Climate models are under different assumptions about leap years, which creates a problem when the program handles the time axis. EZclim was adapted to handle both "no leap" (years have 365 days regardless of the year) data and the normal Gregorian calendar.

3.5.2 File name convention

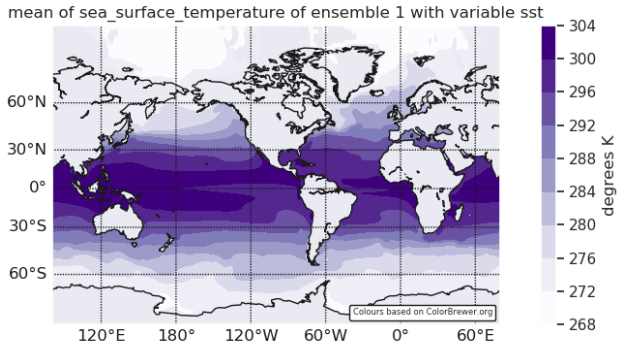
The CMIP6 model data output are usually stored around the world in different distribution centres, so the data is accessible from all over the globe [5]. The data centres have file name conventions which EZclim follows. The EZclim can also handle files with the format:

```
{START OF FILENAME}_ens{NUM}_{YEAR}
```

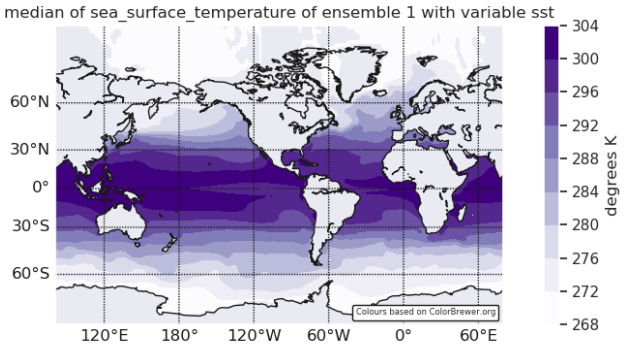
where `{START OF FILENAME}` is the prefix of the file, `{YEAR}` is the year and `{NUM}` is the ensemble number. In the case where the data given by the user does not follow any of the nomenclature above, the current workaround would be for the user to rename their files to follow the convention.

3.5.3 Analysis

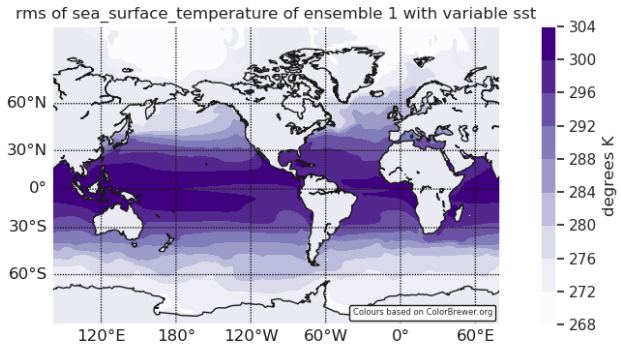
The **mean**, **median**, **root mean squared (RMS)** and **standard deviation (STD)** can be calculated for the model output in the input NetCDF files. By default, analysis are computed with respect to time (see `spatial` option below). As an example, a scenario is shown in Figure 8 below. Here, all averages are calculated (mean, median, STD and RMS) and the maps and histograms of the sea surface temperature between 1980 and 1990, from model ESM2M (Earth System Model, Appendix A) are plotted. The histogram tabular data is also saved as a text file. The input file used is in appendix B.2. The output data is saved in a NetCDF file. The analysed data and the merged original data files are stored in the output NetCDF file.
Spatial:
If the `spatial` option is set, the analysis is done with respect to space: latitude, longitude and depth (if included). In the scenario in 9, the spatial median and STD is calculated and the time series and box plots of the sea surface temperature between 1980 and 1990 are created. The time series tabular data is saved in a text file. The input file used is B.3. The time series plots in Figure 9 include both the monthly and yearly mean, to show the trend of the data. The box plots shown in Figure 10 gives the yearly average of the data analysed, and is a good indication of how spread out the values are.



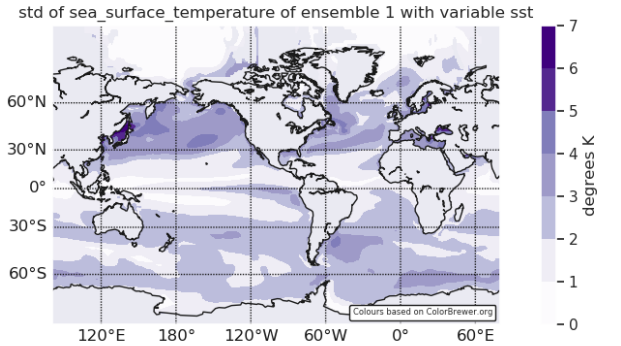
(a) Map plot of sea surface temperature mean from 1980 to 1990, using daily model output



(b) Map plot of sea surface temperature median from 1980 to 1990, using daily model output

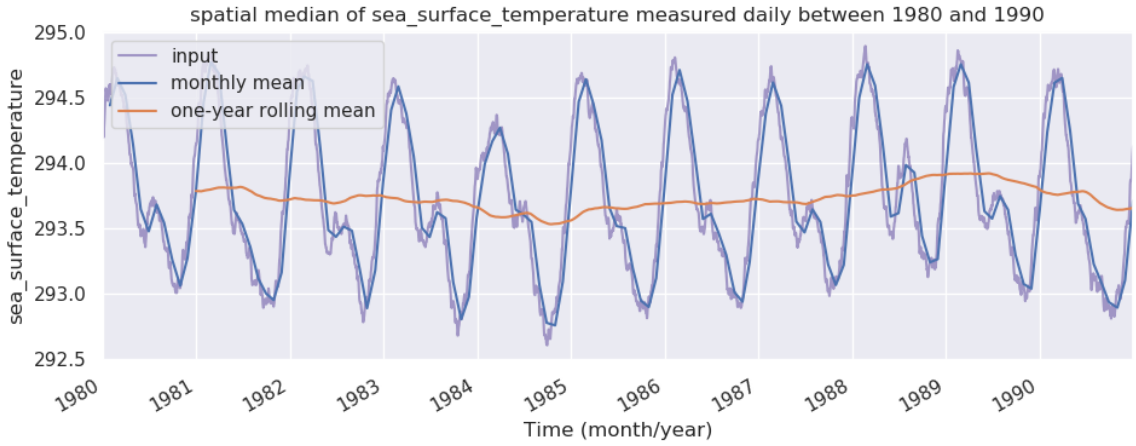


(c) Map plot of sea surface temperature RMS from 1980 to 1990, using daily model output

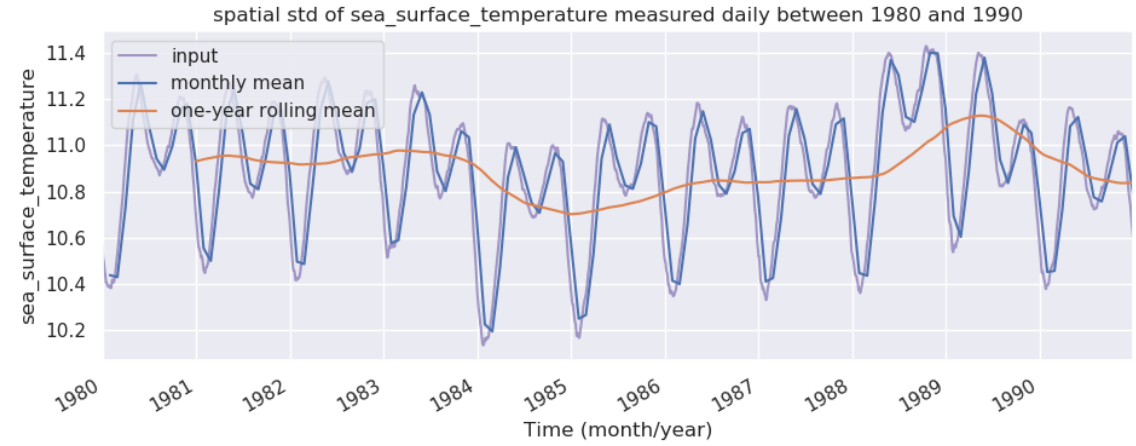


(d) Map plot of sea surface temperature STD from 1980 to 1990, using daily model output

Figure 8 Maps of all averages of sea surface temperature from 1980 to 1990, using daily model output computed

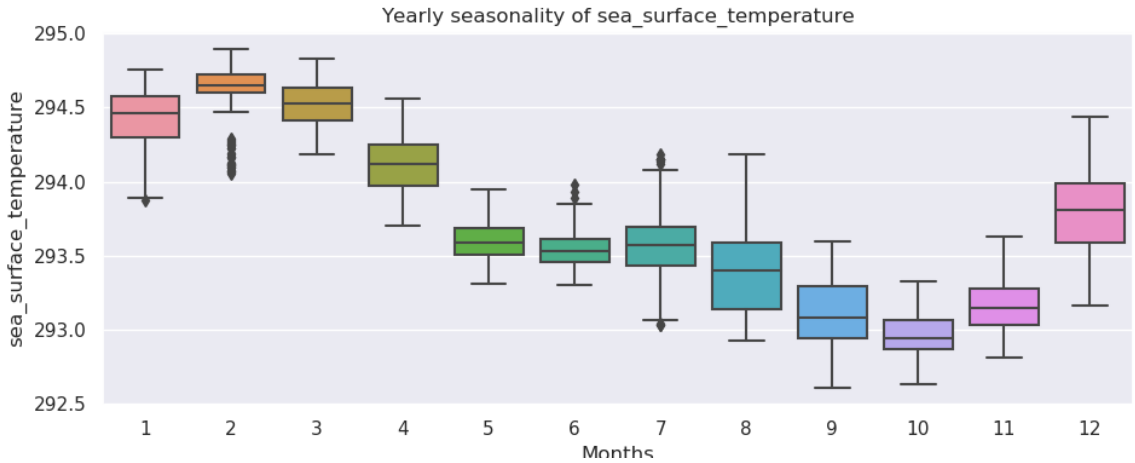


(a) Time series of sea surface temperature median, between 1980 and 1990 using daily model outputs

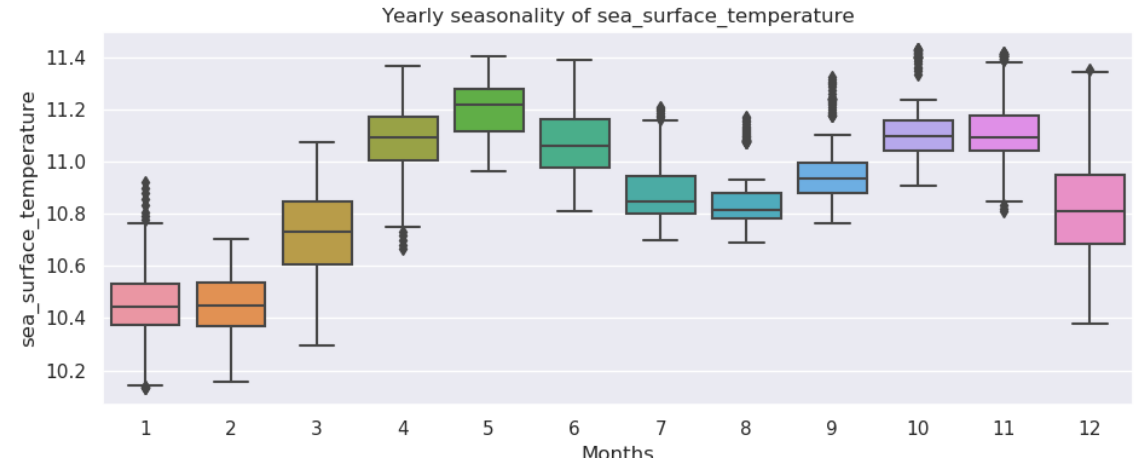


(b) Time series of sea surface temperature STD, between 1980 and 1990 using daily model outputs

Figure 9 Time series of median and STD of sea surface temperature



(a) Box plot of yearly averaged median of sea surface temperature between 1980 and 1990, using daily model outputs



(b) Box plot of yearly averaged STD of sea surface temperature between 1980 and 1990, using daily model outputs

Figure 10 Box plot of the yearly averages of the median and STD of sea surface temperature between 1980 and 1990, using daily model outputs

3.5.4 Total ensemble average

If multiple ensemble members (variations of the same model) are given, the average across all ensembles is calculated. One NetCDF file is given as output that contains the analysis across all ensembles. This option can work with or without any analysis option being set. If `total` is not set, and multiple ensemble members are given, then the analysis is computed for each ensemble. The number of output files is equivalent to the number of ensemble members, where each file contains the analysis performed for each ensemble. The input file for calculating the total median across 5 ensembles of sea surface temperature between 1980 and 1990, using daily model outputs is in B.4.

3.5.5 Masks

A region of interest can be specified by the user by giving a file containing the list of points of a polygon to extract the data from. The example is provided in the `INPUT` folder (`mask_example.txt`, see B.15). Each vertex of the polygon is defined with a tuple of longitudes and latitudes. If a three-dimensional region is to be extracted, then the depth level of the mask can be defined. In the current implementation, all the points within the masked region are selected. Any points outside the region boundary have their value set to NaN. The scenario in Figure 11 displays the analysis of the region defined by `mask_example.txt` in the sea surface temperature at 1980, from ESM2M model, using daily model output. The histogram in Figure 11b is constructed using only the data within the masked region. The input files used to run the simulation is B.5.

3.5.6 Grid operations

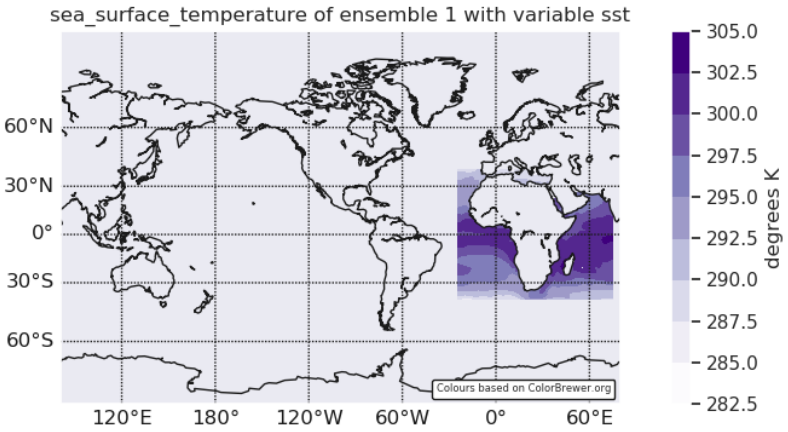
Sample points and Grid points

A singular sample or grid point can be defined or a list of points can be given in a text file. This functionality is added to isolate a point or series of points for analysis. The difference between sample and grid points is the type of interpolation scheme used. Grid points use nearest neighbour interpolation to select the nearest grid box in the data that the point lies in. The sample point uses a linear interpolation scheme which calculates the value at that point based on the surrounding points. Using input file B.6, a time series of the extracted grid point (20, 20) for air pressure at convective cloud base between 1979 and 2014 using monthly model outputs is created. The plots are shown in Figure 12.

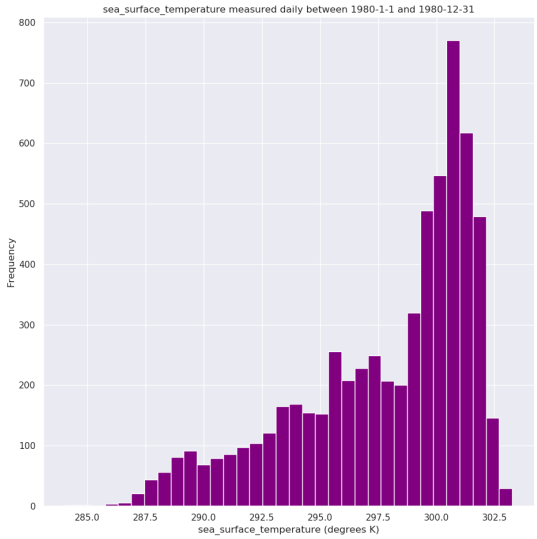
Regridding

The `sample` and `grid` option can also be used to regrid data from one rectangular grid to another, which is particularly useful for model intercomparison work or to compare model results to gridded data. If a NetCDF file is passed as an argument to sample or grid, the original data grid is regridded to the grid in the file given in the `sample`/`grid` argument. The scenario in Figure 13 is regridding the difference between ocean and atmosphere of dissolved inorganic carbon (DIC) at 1953 to a coarser (256x128) grid and to a grid centered on North America. The input files used are B.7 and B.8.

EZclim deals primarily with rectangular grids. However, some grids can be curvilinear (see Appendix A), and the software needs to be extended to have the ability to deal with these kinds of grids. Being able to rotate grids from/to curvilinear grid to/from rectangular grids would be an added advantage because many climate modellers do these conversions often.



(a) Map of sea surface temperature at 1980, within masked region defined in B.15, using daily model output

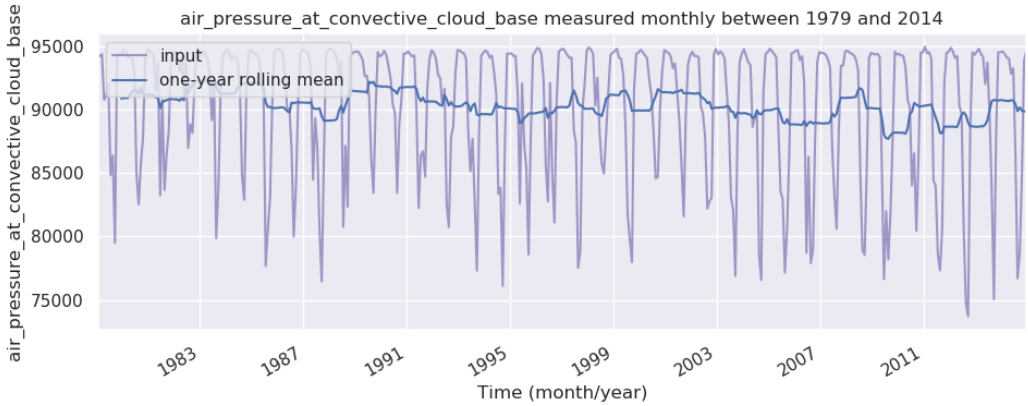


(b) Histogram of sea surface temperature at 1980, within masked region defined in B.15, using daily model output

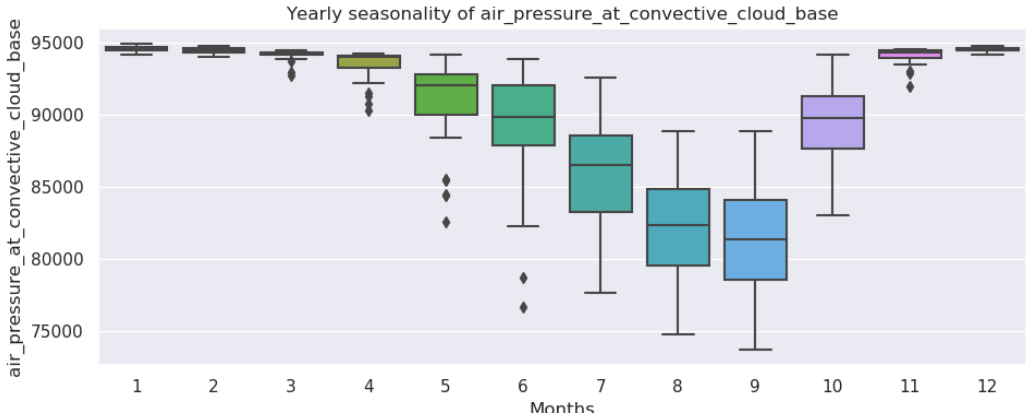
Figure 11 Map and histogram of sea surface temperature at 1980, within masked region defined in B.15, using daily model output

Rotating the grid

The longitude values of the Earth range from [-180,180], but since the Earth is spherical, -180 lies next to 180. Thus when rotating the grid to a new longitude centre, the software must ensure that the correct points lie next to each other. As an example, the mean difference of ocean and atmosphere of DIC at 1953, in Figure 13a is centred to a longitude of 10.0 shown in Figure 14. The input file used to create this is in B.9.



(a) Time series of air pressure at convective cloud base at grid point (20, 20), between 1979 and 2014, using monthly model outputs

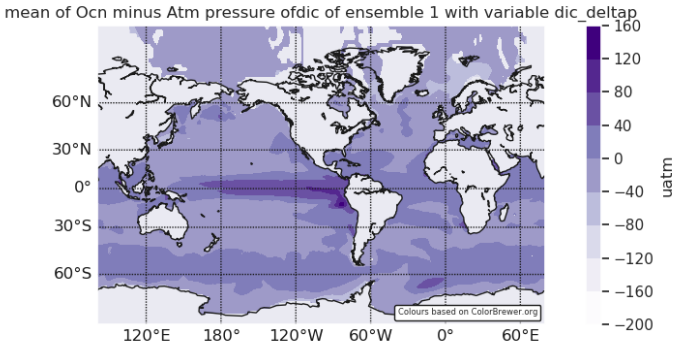


(b) Box plot of the yearly average of air pressure at convective cloud base at grid point (20, 20), between 1979 and 2014, using monthly model outputs

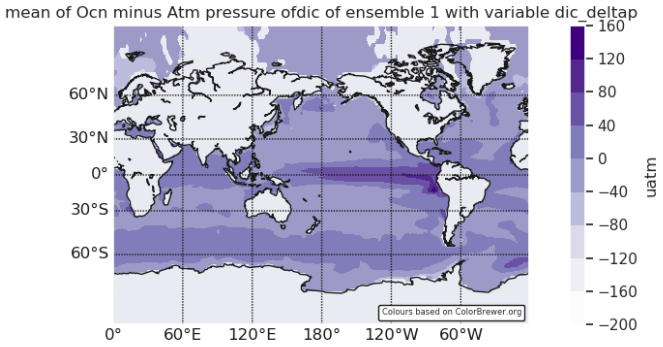
Figure 12 Time series and box plot of the air pressure at convective cloud base at grid point (20, 20), between 1979 and 2014, using monthly model outputs

3.5.7 Calculate areas

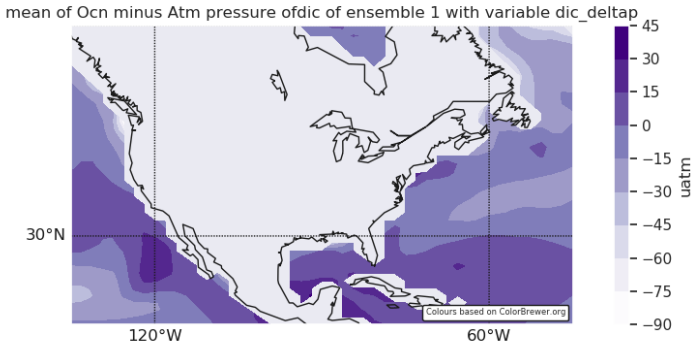
The area of each latitude/longitude grid box in the data can be computed if the `calculate areas` option is set. The area is only calculated once when the program is run, since the data grid does not change across variables or ensembles. This single calculation is a limitation for adaptive mesh models, but currently global climate models (Appendix A), do not have adaptive meshes. The areas are saved in a separate NetCDF file. The previous example was run and the areas of the grid boxes were calculated at the same time (Figure 14). The input file is the same as before in B.9 except now `Calculate areas = True`.
In EZclim, the averages are calculated without taking the area of the grid boxes into account. Also, EZclim currently does not have the functionality to compute integrals. The recently published article *Emergence of anthropogenic signals in the ocean carbon cycle* [18] is one example of why being able to calculate the area averages and integrated quantities are useful. These measures are often used by modellers to diagnose models behaviour.



(a) Mean difference between ocean and atmosphere of dissolved inorganic carbon (DIC) at 1953, using daily model outputs



(b) Regridded mean difference between ocean and atmosphere of dissolved inorganic carbon (DIC) at 1953, using daily model outputs on coarser grid of 256 x 128 longitude/latitude



(c) Regridded mean difference between ocean and atmosphere of dissolved inorganic carbon (DIC) at 1953, using daily model outputs, on North America centered grid

Figure 13 Regridding of mean difference between ocean and atmosphere of dissolved inorganic carbon (DIC) at 1953, using daily model outputs on different grids

3.5.8 Co-variance between two variables

If the `covary` option is set, the user can plot a 2D histogram of **2 variables** given in the `variables` argument. The program throws an error if the number of variables given is not 2. The number of histogram bins (for

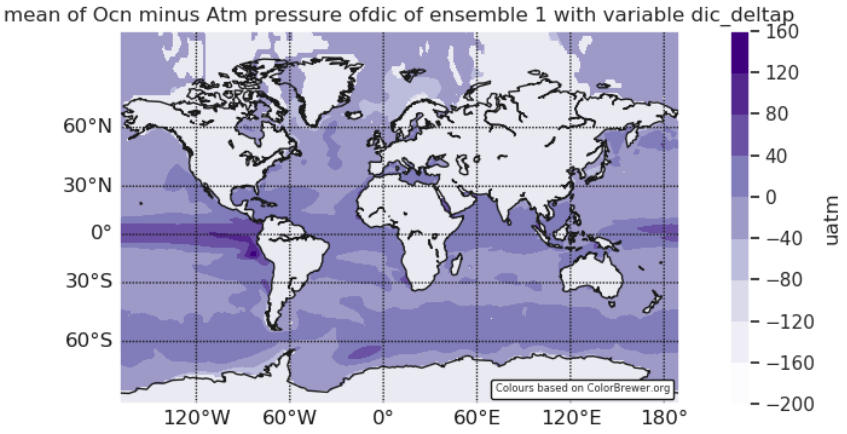


Figure 14 Mean difference of ocean and atmosphere of DIC centred at 10.0, using daily model outputs.

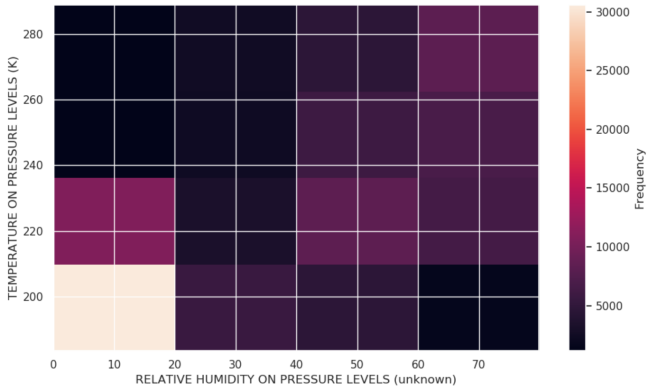
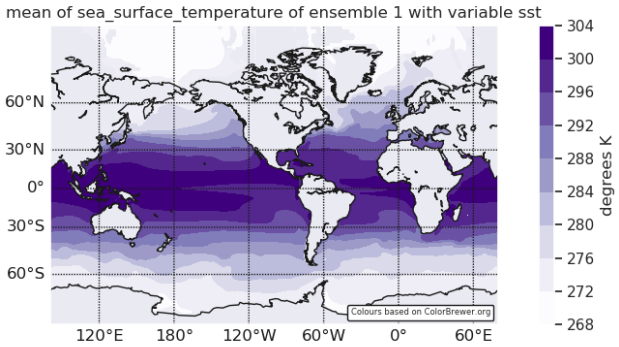


Figure 15 2D histogram using custom histogram bins, of mean temperature and relative humidity, using daily model outputs

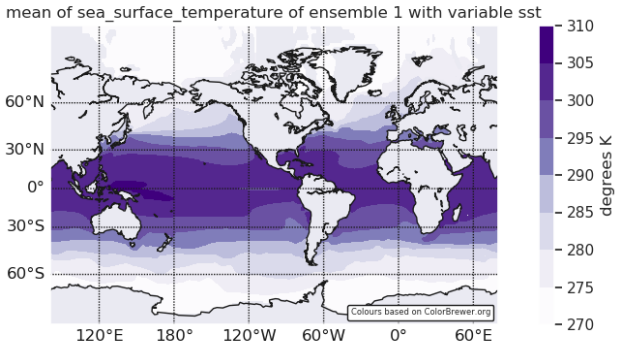
both variables) to use when plotting the 2D histogram should also be given. The users can also define the histogram bin edges themselves, by inputting a custom histogram bin file in the `histogram bin selection` argument. The file used is given in B.14. In the histogram file, each variable must be defined with the key `var`. The bin edges should then be specified on the line after the variable name and the variable that is on the x-axis and y-axis should be defined. Using the input file B.10, the mean temperature and relative humidity is calculated and the 2D histogram is created as shown in Figure 15, using the custom histogram bins given in `hist.txt`. A custom histogram file can also be used if 1D histograms are plotted.

3.5.9 Multi-model analysis between historic and future runs

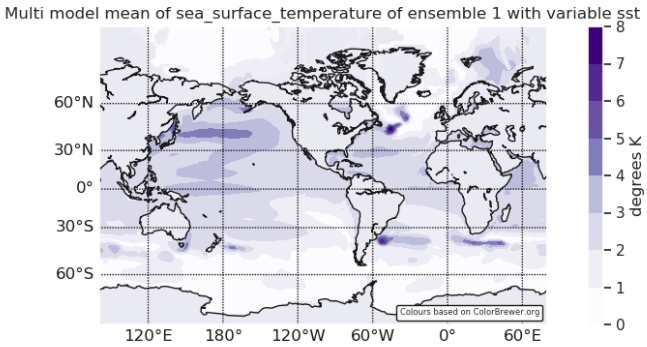
Multi-model refers to the comparison between 2 numerical experiments. For example, if the `analysis` option is the mean, then the output would be the absolute difference between the means of the 2 models. In input file B.12, we calculate the multi-model mean of sea surface temperature, using daily model outputs, between 1880-1890 and 2090-2100. The plots of the results is shown in Figure 16. The program with the spatial option being set produces results in Figure 17.



(a) Mean of the sea surface temperature between 1880 and 1890, using daily model outputs



(b) Mean of the sea surface temperature between 2090 and 2100, using daily model outputs

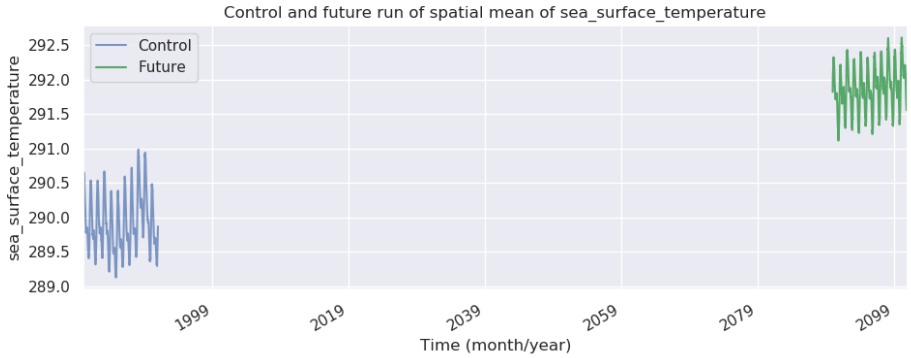


(c) Multi-model mean of sea surface temperature, between 1880-1890 and 2090-2100, using daily model outputs

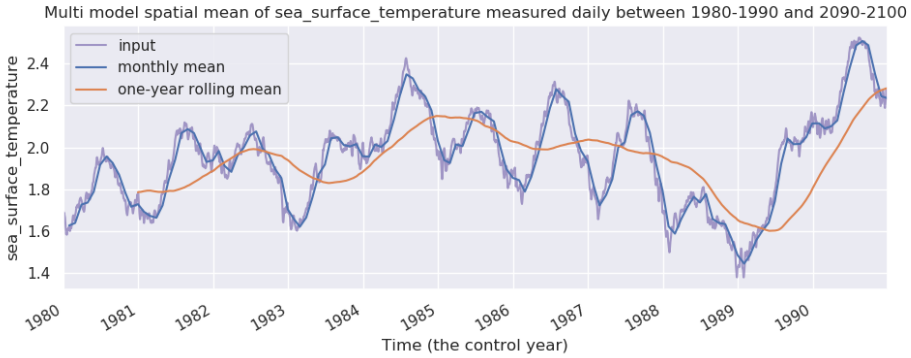
Figure 16 Maps of mean of sea surface temperature, using daily model outputs

3.5.10 Calculation of climate indices

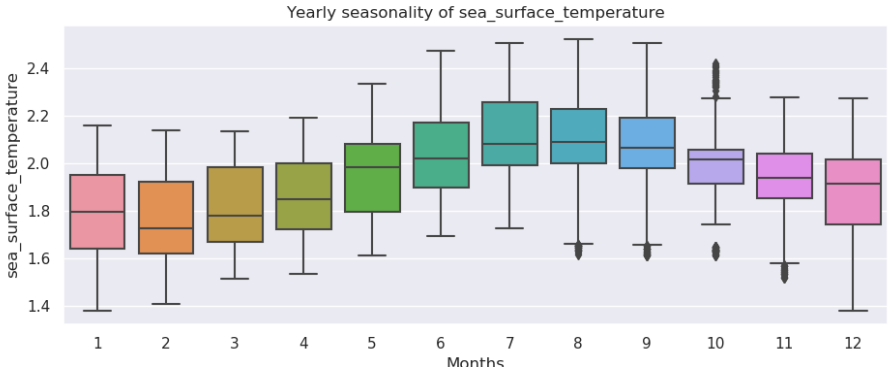
Climate indices can be calculated using EZclim. The bash script `calculate_indices.sh` is written using CDO [19].CDO is a library already optimised to perform operations in NetCDF files. It is, however, quite hard for non-experts to use. The python file `calculate_indices.py` is used as a wrap around the bash script and uses the arguments given by the input file of our program. Some of the indices that can be calculated are: The Oceanic Niño Index (ONI), The Trans-Niño Index (TNI), Indian Ocean Dipole (IOD) Mode Index, and many others. The climate indices are useful because their calculated values are used to



(a) Spatial mean of the sea surface temperature between 1880-1890 and 2090-2100, using daily model outputs



(b) Time series of spatial multi-model mean of the sea surface temperature between 1880-1890 and 2090-2100, using daily model outputs



(c) Box plots of spatial multi-model mean of sea surface temperature, between 1880-1890 and 2090-2100, using daily model outputs

Figure 17 Time series and box plots of the multi-model mean of sea surface temperature

describe the state and changes in the climate system. Some climate indices even seem to predict ecological processes better than the local weather [20]. If the `calculate indices` argument is set, then the python script is called (which in turn calls the bash script). EZclim also provides a template for users who are familiar with CDO to call other CDO operations using EZclim. An output NetCDF file is saved in `RESULTS`. The input file (B.13) is used to calculate the North Atlantic oscillation between air pressure at a convective cloud base between 1950-1959 and 2010-2014, using monthly model outputs.

3.5.11 User-given function

EZclim offers the basic analysis capabilities: mean, median, RMS etc. For users with more python experience, they may want to do more, so EZclim offers a highly customisable interface where the user can input any Python script (in the `INPUT/user_folder`) as a substitute to the preset analyses provided. The input file B.11 shows how this would be called in EZclim if a simple manipulation between temperature and relative humidity was to be performed. given by user function of temperature and relative humidity at 2193. The manipulation can be found in the example file below.
Example file in `INPUT/user_function/example_function.py`

```
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
```

When the user is creating their function to use, they need to have a function which takes a dictionary of iris cubes as a primary argument. The keys of the dictionaries are the variables. This enables the user to create manipulations with respect to any variables they need.
Having this added user functionality is useful in many applications. One example of an application is outlined in the article *Role of regression model selection and station distribution on the estimation of oceanic anthropogenic carbon change by eMLR* [21] about regression modelling being used to filter out natural variability in DIC. The user can add a function which performs the modelling and pass it in EZclim.

4 Discussions

4.1 Improving and measuring the efficiency of the code

The efficiency of the code was measured by running the software for a range of ensembles and measuring the time and memory usage of program.
The scenario tested was: *Calculate the mean of all ensembles for air pressure at convective cloud base between 1950-1959, measured monthly.* EZclim was run for 1 to 40 ensembles. Compute times of the program

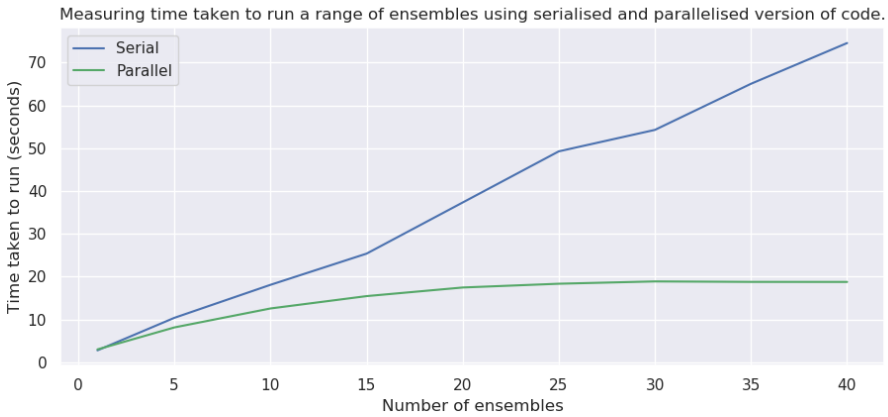


Figure 18 Measuring time taken to run a range of ensembles using serialised and parallelised version of code. The scenario used is the mean of all ensembles for air pressure at convective cloud base between 1950-1959, using monthly output.

runs are shown in Figure 18. As the number of ensembles run increases, the time till the program finishes also increases. The trend in Figure 18 indicates that the system will not scale well for a very large number of data files. There is a linear increase of time with respect to the work done (ensembles run), therefore the system is efficient even with an increased work load.

4.1.1 Parallelisation of the code

Parallelising the code takes advantage of the computing power of the multi-cores in the computer architecture. The python `multiprocessing` module was used to send and receive data from the processes. EZclim was parallelised in order to improve scalability and efficiency.
Two classes were parallelised: `Extract` and `Analysis`. Since each ensemble does not depend on the other, the data for each ensemble would be sent to a singular process. There was an attempt to parallelise `WriteOutput` but it was discovered during testing that when writing large files and passing arguments between processes, the data would get too large for the processes to send to each other. With both classes parallelised, the time taken to run a range of ensembles was measured in figure 18.
The time taken as the number of ensembles increase does not vary by a large degree (unlike in the serial case). Therefore, scalability of the code improves. The gradient between the work load and the time is lower than with serialised code, making for a more efficient code. However, with parallelisation, some care is needed because of the overhead due to the external libraries used (`multiprocessing`) and the cost to communicate between processes - this cost becomes important when using particularly large NetCDF data files. Below are a list of improvements that could be made to EZclim and issues that could be fixed if there was more time.

4.2 Further work

4.2.1 Conserving space when extracting a region

Currently, when extracting a specific region with a mask, all elements outside the masked region are set to NaN and the grid size of the original data does not change. In order to conserve space, instead of setting the values to NaN, the original grid size can be changed to that of the masked region. Interpolation around the edges of the mask for any non-regular polygons would have to be implemented.

4.2.2 Higher resolution: hourly data

More and more model outputs in CMIP 6 are going towards representative modelling and are using a higher resolution of time steps. Instead of monthly or daily data, hourly data outputs are used. EZclim currently deals with monthly and daily data, but extending to hourly data would be useful in the future. Writing our code using C/C++ (to speed up run time) or extending the parallelisation approach will be vital when dealing with these high frequency data outputs.

4.2.3 Possible memory limitations when writing files

When writing the final output of EZclim in a NetCDF file, the original data files are usually combined and are included in the output file. However for large data files, the writing speed and memory used is significantly increased. Judging when to save a combination of files or just the bare minimum needs to be investigated. An idea is to check the memory usage of the software while it is running and then make a decision based on the CPU usage and write-memory available.

5 Conclusion

EZclim, a self-contained software package for Climate modelling diagnostics was created with different functionalities. The output of the software are given as NetCDF and as visual plots. The addition of the testing capabilities enables us to have verified code. The user testing validates the code and ensures its usefulness. Scalability of the program was maintained, and the addition of parallelisation was an improvement to the software efficiency. Some members of the climate modelling community were also engaged to highlight the necessary user applications to add to the software.
EZclim adds value to existing climate modelling software through its simple, yet highly customisable user interface which caters to both non-climate and climate modellers. It also enhances efficiency by combining the steps of climate modelling diagnostics into a single program thus eliminating the need for multiple software.
Adding more advanced parallelisation techniques to run high frequency data outputs will further enhance the software.

Acknowlegdments

I really enjoyed this project. The main challenge was constantly ensuring that the code functionalities met the user demands, which is why making the code scalable to ease the constant extension of the code was required. This was my first exposure to climate modelling, and it was a great experience learning more about the subject.
First, I would like to thank my supervisor *Dr. Yves Plancherel*, for his help and guidance throughout the project. I really appreciate his support, availability and the patience with which he provided detailed explanations that helped me understand the topic better.
I would also like to thank *Dr. Keith B. Rodgers*, *Dr. Emma Cavan* and *Davor Dundovic* for their suggested applications to add to the software and their time spent as user testers.
Also, I would like to thank my *parents* and *brothers* for their support throughout my entire time at Imperial and for their continuous words of encouragement.
Finally, I would like to thank *GOD*, for everything that He has done for me. I would not be here today without His goodness.

References

[1] Carbon Brief. How do climate models work? https://www.carbonbrief.org/qa-how-do-climate-models-work, 2018.

[2] Ann Henderson-Sellers Kendal McGuffie. *The Climate Modelling Primer*. John Wiley & Sons, Inc., 4th edition, 2014.

[3] Sedláček Jan Knutti Reto. Robustness and uncertainties in the new CMIP5 climate model projections. *Nature Climate Change*, 3, 2012. https://doi.org/10.1038/nclimate1716.

[4] Maher, N., Milinski, S., Suarez-Gutierrez, L., Botzet, M., Dobrynin, M., Kornblueh, L., et al. The Max Planck Institute Grand Ensemble: Enabling the exploration of climate system variability. *Journal of Advances in Modeling Earth Systems*, 11, 2019. https://doi.org/10.1029/2019MS001639.

[5] Program for Climate Model Diagnosis and Intercomparison. https://pcmdi.llnl.gov/index.html, 2019.

[6] Met Office. *Iris: A Python library for analysing and visualising meteorological and oceanographic data sets*. Exeter, Devon, v1.2 edition, 2010 - 2013. http://scitools.org.uk/.

[7] S. Hoyer and J. Hamman. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 2017. http://doi.org/10.5334/jors.148.

[8] Eyring, V. and Righi, M. and Lauer, A. and Evaldsson, M. and Wenzel, S. and Jones, C. and Anav, A. et al. ESMValTool (v1.0) – a community diagnostic and performance metrics tool for routine evaluation of Earth system models in CMIP. *Geoscientific Model Development*, 9(5):1747–1802, 2016. https://doi.org/10.5194/gmd-9-1747-2016.

[9] Greene, Chad A. and Thirumalai, Kaustubh and Kearney, Kelly A. and Delgado, José Miguel and Schwanghart, Wolfgang and Wolfenbarger, Natalie S. and Thyng, Kristen M. and Gwyther, David E. and Gardner, Alex S. and Blankenship, Donald D. The Climate Data Toolbox for MATLAB. *Geochemistry, Geophysics, Geosystems*, 20(7):3774–3781, 2019. https://doi.org/10.1029/2019GC008392.

[10] Numpy. https://www.numpy.org/, 2019.

[11] dask. https://dask.org/, 2019.

[12] pandas. http://pandas.pydata.org/, 2018.

[13] Rebecca G. Asch, Darren J. Pilcher, Sara Rivero-Calle, and Johnna M. Holding. Demystifying Models: Answers to Ten Common Questions That Ecologists Have About Earth System Models. *Association for the Sciences of Limnology and Oceanography*, pages 241–269, August 2016. https://doi.org/10.1002/lob.10113.

[14] Github IRP repository. https://github.com/msc-acse/acse-9-independent-research-project-AdannaAkwats, 2019.

[15] Travis CI. https://travis-ci.org/, 2019.

[16] Xming. Xming X Server. http://www.straightrunning.com/XmingNotes/, 2019.

[17] Robert C Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[18] Schlunegger, S., Rodgers, K. B., Sarmiento, J. L., Frölicher, T. L., Dunne, J. P., Ishii, M., & Slater, R. Emergence of anthropogenic signals in the ocean carbon cycle. *Nature Climate Change*, 2019. https://doi.org/10.1038/s41558-019-0553-2.

[19] Schulzweida, Uwe. CDO User Guide. https://doi.org/10.5281/zenodo.2558193, February 2019.

[20] Hallett, T. B. and Coulson, T. and Pilkington, J. G. and Clutton-Brock, T. H. and Pemberton, J. M. and Grenfell, B. T. Why large-scale climate indices seem to predict ecological processes better than local weather. *Nature*, 430:71–75, 2004. https://doi.org/10.1038/nature02708.

[21] Plancherel, Y. and Rodgers, K. B. and Key, R. M. and Jacobson, A. R. and Sarmiento, J. L. Role of regression model selection and station distribution on the estimation of oceanic anthropogenic carbon change by emlr. *Biogeosciences*, 10(7):4801–4831, 2013. https://doi.org/10.5194/bg-10-4801-2013.

[22] Australian Academy of Science. What is Climate change? https://www.science.org.au/learning/general-audience/science-climate-change/1-what-is-climate-change, 2019.

[23] UK climate projections. https://www.metoffice.gov.uk/research/collaboration/ukcp, 2019.

[24] World Meteorological Organization. Frequently Asked Questions (FAQs). http://www.wmo.int/pages/prog/wcp/ccl/faqs.php, 2019.

[25] Flato, G., J. Marotzke, B. Abiodun, P. Braconnot, S.C. Chou, W. Collins, P. Cox, F. Driouech, S. Emori, V. Eyring, C. Forest, P. Gleckler, E. Guilyardi, C. Jakob, V. Kattsov, C. Reason and M. Rummukainen. Evaluation of Climate Models. In: Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change. 2013. https://www.ipcc.ch/site/assets/uploads/2018/02/WG1AR5_Chapter09_FINAL.pdf.

[26] Hegerl, G.C., F. W. Zwiers, P. Braconnot, N.P. Gillett, Y. Luo, J.A. Marengo Orsini, N. Nicholls, J.E. Penner and P.A. Stott. Understanding and Attributing Climate Change. In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. 2007. https://www.geos.ed.ac.uk/homes/ghegerl/AR4WG1_Pub_Ch09.pdf.

[27] Kerkhoff, Christian and Künsch, Hans R. and Schär, Christoph. Assessment of Bias Assumptions for Climate Models. *Journal of Climate*, 27(17):6799–6818, 2014. https://doi.org/10.1175/JCLI-D-13-00716.1.

A Appendix: A brief introduction to Climate models

Firstly, it is necessary for us to distinguish the difference between predicting day-to-day weather, or *Weather forecasting* and Climate Modelling. Weather forecasting tells us how the weather will be and the specific temperature within a *short* time period. On the other hand, Climate is a *statistic* description of the state and variability of a system, and gives us the *projections* of weather that occur in the far future (years or decades ahead) [22]. As an example, a climate model can predict that it will be warm in the summer of 2070 but it cannot tell you the exact temperature that it will be on a specific day. However, weather forecasting can tell you that tomorrow it will be $20°C$ or even more precisely, at 2pm tomorrow, it will be $20°C$.
Weather forecasting and climate modelling forecast predictions should theoretically be the same, but since we do not have infinite knowledge about the future, this is not practically possible. That being said, the UK Met Office have developed a climate analysis tool called the UK Climate Projections 18 (UKCP18), and it provides users with the most recent scientific evidence on climate change projections that they can use to plan for the future [23]. UKCP18 includes estimates of the range of probable outcomes of future climate in the UK. Therefore, the tool does not just use weather forecasting to predict daily or weekly temperature but it is using climate model outputs to predict and prepare for future climate change. The UKCP18 tool is a good indication of the push towards long-scale weather forecasting in order to help decision makers assess their risk exposure to climate [23].
Climate change refers to a change in the statistical description of the mean state of the climate or its variability. This change must last for an extended period of time [24].

A.1 What are climate models?

Climate models, or Coupled Global Circulation Models (CGCMs) are designed to represent the Earth's climate system. These are tools used to further our understanding of the Earth's past, present and future climate projections [25]. The *primary earth system components* that the coupled or Earth system model simulates include ocean, atmosphere, ice sheets and land. The models are run with many simulations under different scenarios to investigate the effect of each scenario in future climate conditions. One scenario could be running a model with only natural forcing (e.g. taking volcanic eruptions, solar output etc. into account) and another would be running the model with both natural forcing and human influences (also taking into account the emission of carbon and greenhouse gases or the effect of aerosols on the atmosphere). The model outputs are then measured against observed real-life data.

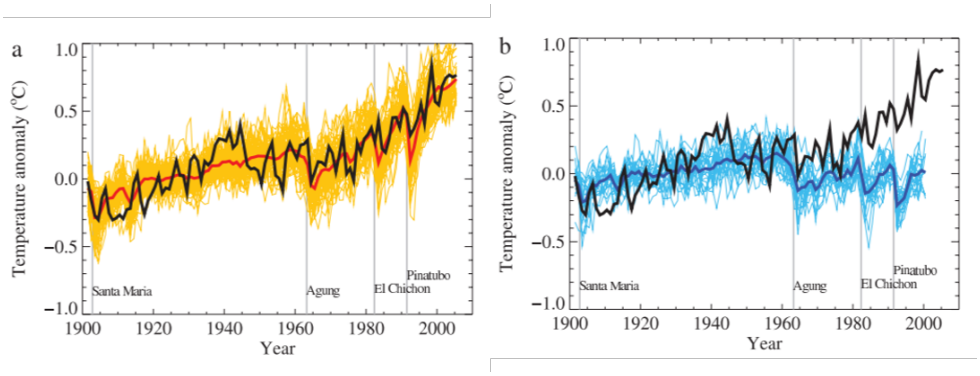


Figure 19 Comparison between global mean surface temperature anomalies ($°C$) from observations (black) and AOGCM simulations [26].

Figure 19 (a) shows that with natural forcing and human influence, the model fits closely to the observed data in the global mean temperature. Conversely in (b) without the inclusion of human forcing, the fit between the observed data and the model diverges towards the end of the 20th century.
In Figure 19, the models are compared with observed data, but this may not always be available, so the model needs to be compared to what is called a *control run* or *control simulation*. A control run is also needed to study how biased the run is compared to the actual data. Control runs are used to establish a baseline for the model. By comparing the climate model experiment with its control run, the climate change signal of the model can be detected. Similarly, the climate change signal of the observed data can be detected by comparing the observed data with the control run.
Since the model experiment may drift away from the observed data, the bias between the control run and observed data may dominate the climate change signal of the observed data. Since the control run is defined as a baseline of the model, the bias between the control and the model experiment should be 'cancelled', or at least minor compared to the climate change signal of the model [27].
The initial conditions (greenhouse gases, solar activity etc.) of the model are held in a reference state. Pre-industrial values of the initial conditions are used and the model is run for hundreds or thousands of years until it reaches a stable *dynamic* equilibrium, that is, the model output more or less has a specific pattern over hundreds of years. Control runs can be used to see and understand the *natural variability* of the climate - when multiple simulations are run with the same model with different initial conditions and the same (natural) forcing, the model output will vary for each simulation [26]. Natural variability is influenced by both natural variations (solar activity, volcanic eruptions) and also by internal variations, which are developed from coupled interactions between the earth system components, such as those happening in the tropical Pacific Ocean during an El Niño event (ENSO) [26]. These natural variations produces seasonal, yearly or even decades-long fluctuations in the climate.

A.2 How are CGCMs built?

CGCMs divide the primary earth system components into a 3-dimensional grid space and discretise and solve the equations governing the components using supercomputers, as these are extremely large systems to solve. Since the model is coupled, the equations are also solved using multiple processors in order to streamline and speed up the communications between components [2]. Each of the Earth system components are submodels which need to be connected together to form a coupled model. This interconnection is achieved using a *model coupler*. The submodels use differing grids and resolutions from each other, which means that their modelling approaches will be different and will have to be dealt with by the model coupler [2]. The model coupler controls the execution and time evolution of the climate model by coordinating and controlling the passage of information between the components: atmosphere, land, ice sheets and oceans [2]. Figure 20 shows the model coupler between the components, and illustrates the transfer of the fluxes from one component to another.
There are different kinds of grids used to discretise the system of equations.

A.2.1 Computational grids and discretisation

The system of equations are sometimes discretised using *Finite grid discretisation*, which is done by dividing the subsystems of the climate into rectangular/square grids. This is also known as the latitude-longitude grid. This was the earliest discretisation technique for both the ocean and the atmosphere [2]. The main disadvantage of this technique is that it suffers from *the polar problem*, where at the poles, the grid spacing becomes very small and converges to a point as shown in Figure 21a. The reduction of the spacing means that in order to maintain computational stability near the poles, smaller and smaller time steps have to be used. Unfortunately, this means that the computational efficiency would

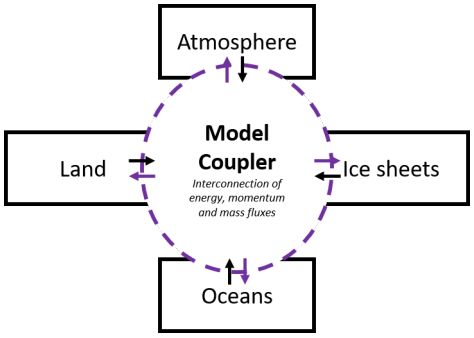


Figure 20 Model coupler

be reduced. Many models solve this stability and inefficiency problem by using *semi-Lagrangian timestepping* [2].

(a) Finite grid points (b) Geodesic grid

Figure 21 Different types of discretisation grids

A large number of *atmospheric* models use *spectral models* [2]. These models make use of the the spherical geometry of the earth when calculating the equations. In order to solve the equations, we need to transfer from grid space to spectral space and vice versa. This is actually the main issue with using this method as most of the computational time lies in the transformations between grid and spectral space at each timestep. However, unlike with finite difference grids, spectral models do not suffer from the *polar problem*. They are also extremely efficient, accurate and the stability in them allows the possibility of increasing and decreasing the resolution on regions of higher and lower interests respectively.

Geodesic grids aim to keep the advantages of finite difference grids and spectral models while reducing the effect of their disadvantages when solving the equations. The grid covers the sphere of the earth with triangular grids. Figure 21b shows a geodesic grid. There are two main advantages of using this kind of grid, the first being that since all the grid cells are nearly the same size, the uniformity allows for computational stability. The second advantage is its ability to be parallelizable, given that the grid can be split into neighbouring blocks each of which can be solved using a separate processor. Similarly to spectral models, the resolution on regions of higher and lower interests can be increased or decreased respectively.

A.3 Timeline of CGCMs

At the start of the production of GCM models not coupled and were either specifically atmosphere model or ocean models. Over time, GCMs became coupled atmosphere-ocean general circulation models (or

AOGCMs). Nowadays, more components of the earth system (e.g. Ice sheets, carbon cycle etc.) are added to the CGCMs [13]. *Earth System* Models (ESM) are CGCMs that include additional information on biogeochemical cycles [13]. The data obtained from ESMs will be used in our software.
The Intergovernmental Panel of Climate Change (IPCC) was established by World Meteorological Organisation (WMO) and the United Nations Environment Programme (UNEP). Their objective is to produce the most recent developments in climate science. Synthesis or assessment reports are published every five to seven years. The most recent is the AR5, but AR6 is now underway.
Over several years, several climate models have been developed. Their results are vital and included in the IPCC assessment reports. The most recent models were run for AR5 from the Coupled Model Intercomparison Project 5 (CMIP5). The model experiments from CMIP6 are currently being run to be injected into the new assessment report AR6 which is in development.
CMIP was developed as a means standardise the model experiments so that the setup for each model would be the same. This means that now if the model experiments are setup the same and their outputs differ, we know it is because of the model processes as opposed to the experimental design [1], i.e if the models were run with the same initial and boundary conditions, certain causes for the differing model output can be eliminated. The intercomparison project created a coordinated global community-wide experiment. And, it allows each modelling centre to conduct model experiments that transcends their own models capabilities, by having other modelling groups perform experiments with the same setup on their own model (that may have more capabilities).
CMIP goes through a new iteration every five or six years. In each iteration, all the modelling centres agree to run specific sets of experiments. In earlier iterations, CMIP experiments would be to model the impact of a 1% annual increase in atmospheric CO2 emissions and analyse between models. More recent CMIP experiments have increased the emission scenarios and their detail as well as increasing the number of models that are being compared [5]. The *Program for Climate Model Diagnostics and Intercomparison* (PCMDI) is an organisation that provides software and storage for the CMIP data in distribution centres globally. PCMDI deals with the diagnosis and intercomparison of CGCMs [5].

B Appendix: Installation and Input files

B.1 INSTALL.txt file

Listing 1 INSTALL.txt

```
matplotlib>=2,<3
netCDF4==1.4.2
numpy==1.16.3
pytest==3.5.1
python-dateutil==2.7.3
scipy==1.2.1
xarray
Shapely==1.6.4.post2
Cython
seaborn==0.9.0
cdo==1.5.3
cf-units>=2
cftime
dask[array]
six
progressbar==2.5
```

B.2 Input file for calculating averages of the sea surface temperature between 1980 and 1990, using daily outputs

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

B.3 Input file for calculating spatial median and STD of the sea surface temperature between 1980 and 1990, using daily outputs

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

B.4 Input file for calculating total median across 5 ensemble members of the sea surface temperature between 1980 and 1990, using daily outputs

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: sst
Start date of analysis: 1980
Variables: sst
Number of ensembles: 5
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis: 1990
Analysis: median
Spatial:
Total ensemble stats: True
Plot: 1
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.5 Input file for extracting region in sea surface temperature at 1980, using daily output

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

B.6 Input file for time series of the extracted grid point (20, 20) for air pressure at convective cloud base between 1979 and 2014 using monthly outputs

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: ccb
Start date of analysis: 1979
Variables: ccb
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis: 2014
Analysis:
Spatial:
Total ensemble stats:
Plot: 1
Monthly: True
Grid: 20, 20
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.7 Input file for regridding the mean of the difference of ocean and atmosphere of DIC at 1953, using daily output to a coarser 256x128 grid

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: dic_deltap
Start date of analysis: 1953
Variables: dic_deltap
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis:
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid: ccb_Amon_CNRM-CM6-1_highresSST-present_r1i1p1f2_gr
_195001-195912.nc
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.8 Input file for regridding the mean of difference of ocean and atmosphere of DIC at 1953, using daily outputs, on a North America centered grid

```
# REQUIRED ARGUMENTS
# ------------------------------ ----------------------
Prefix: dic_deltap
Start date of analysis: 1953
Variables: dic_deltap
Number of ensembles: 1
#
# -------------------------------------------------------
# OPTIONAL ARGUMENTS
# -------------------------------------------------------
End date of analysis:
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample: E1_north_america_ens101_1970.nc
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.9 Input file for the mean of difference of ocean and atmosphere of DIC at 1953, using daily outputs, to a longitude of 10.0

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: dic_deltap
Start date of analysis: 1953
Variables: dic_deltap
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis:
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre: 10.0
User function:
Calculate areas:
Calculate index:
```

B.10 Input file for 2D histogram using custom histogram bins, of mean temperature and relative humidity at 2193 using daily outputs

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: temp_rh
Start date of analysis: 2193
Variables: temp, rh
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis:
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary: True
Histogram bin selection: hist.txt
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.11 Input file to perform an equation between temperature and relative humidity given by user function of temperature and relative humidity at 2193, using daily outputs

```
# REQUIRED ARGUMENTS
# -------------------------------------------------------
Prefix: temp_rh
Start date of analysis: 2193
Variables: temp, rh
Number of ensembles: 1
#
# -------------------------------------------------------
# OPTIONAL ARGUMENTS
# -------------------------------------------------------
End date of analysis:
Analysis:
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function: example_function, temp_rh
Calculate areas:
Calculate index:
```

B.12 Input file to compute the multimodel mean of sea surface temperature, between 1880-1890 and 2090-2100, using daily outputs

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: sst
Start date of analysis: 1880, 2090
Variables: sst
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis: 1890, 2100
Analysis: mean
Spatial:
Total ensemble stats:
Plot: 1
Monthly:
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index:
```

B.13 Input file to calculate the North Atlantic oscillation between air pressure at a convective cloud base between 1950-1959 and 2010-2014, using monthly outputs

```
# REQUIRED ARGUMENTS
# -----------------------------------------------------
Prefix: ccb
Start date of analysis: 1950, 2010
Variables: ccb
Number of ensembles: 1
#
# -----------------------------------------------------
# OPTIONAL ARGUMENTS
# -----------------------------------------------------
End date of analysis: 1959, 2014
Analysis:
Spatial:
Total ensemble stats:
Plot: 1
Monthly: True
Grid:
Sample:
Mask file:
Save Output: True
Covary:
Histogram bin selection:
Longitude centre:
User function:
Calculate areas:
Calculate index: nao
```

B.14 Example file: hist.txt

```
names of variables and axis on plot
x: temp
y: rh

var: rh
0.0                min value of rh
19.95
39.9
59.849999999999994
79.8               max value of rh

var: temp
183.6640625     min value of temp
209.9662109375
236.268359375
262.5705078125
288.87265625    max value of temp
```

B.15 Example file: 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
```