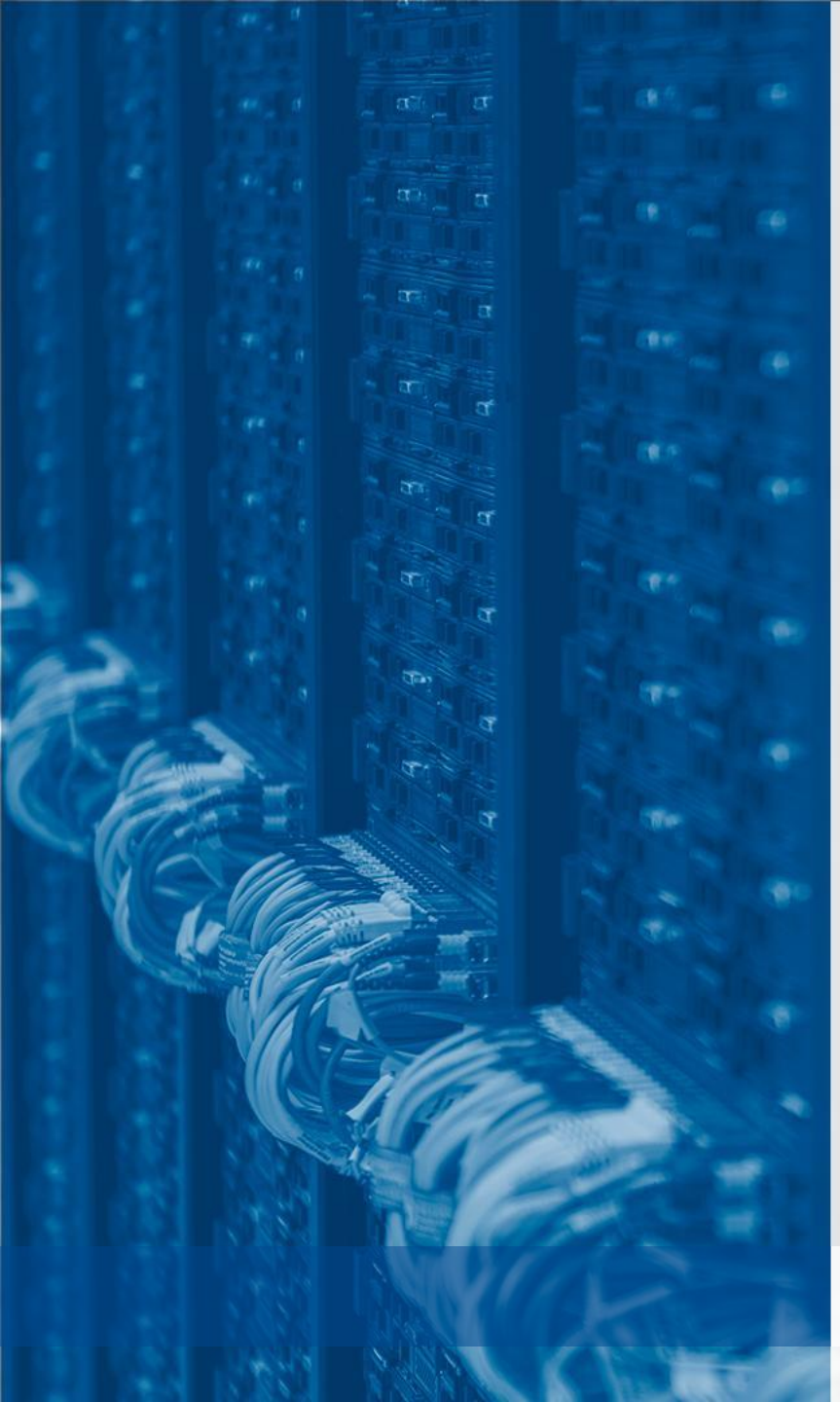


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Online model evaluation

Incorporating ESMValTool to BSC's EC-Earth workflow

Javier Vegas Regidor

The starting point

- EC-Earth is run at BSC using our own workflow manager: Autosubmit
- Everything, from deployment to data storage was automated…
- … but we lacked an automated tool to produce plots to monitor the runs
- We used Barakuda for a while, but is was never incorporated into the workflow

Our idea for monitoring

- We started experimenting with a live app that generated maps, timeseries ... on the fly.
- At the end, it was too slow for monitoring, although we will keep it as a solution for quick analysis.
- The solution is to precompute the outputs at regular intervals during the workflow and use the web app just to display them.
- We save the images in a specific folder in our storage following a well defined convention so we can completely decouple both parts.

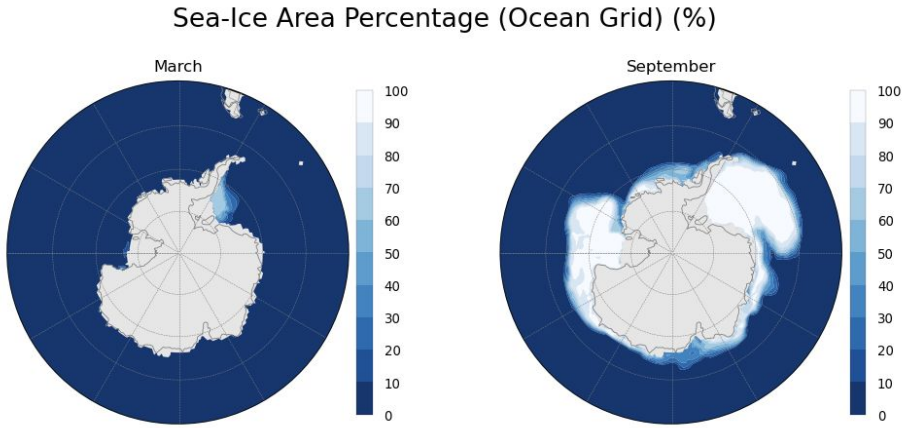
Why we use ESMValTool?

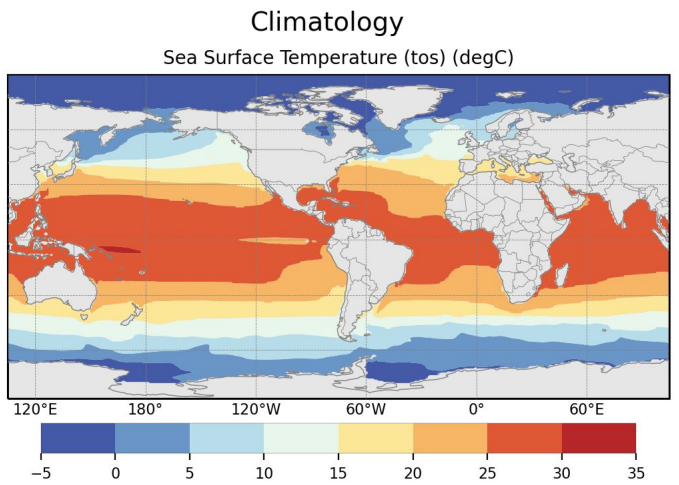
- In the past, we have struggled to support all the different tools used by the scientists, so we are trying to reduce their number as much as possible
- There is an effort to support this functionality directly in ESMValTool, so we expect to benefit from it in the future
- It allows us to easily plug in all our in-house tools if needed
- Even if by now we are only displaying the model data, it will make easier for us to add references to the plots if needed (i.e observations, renalysis, control runs...)
- We already decided years ago that all our model data will be stored in cmor-like format, so that is not a burden for us

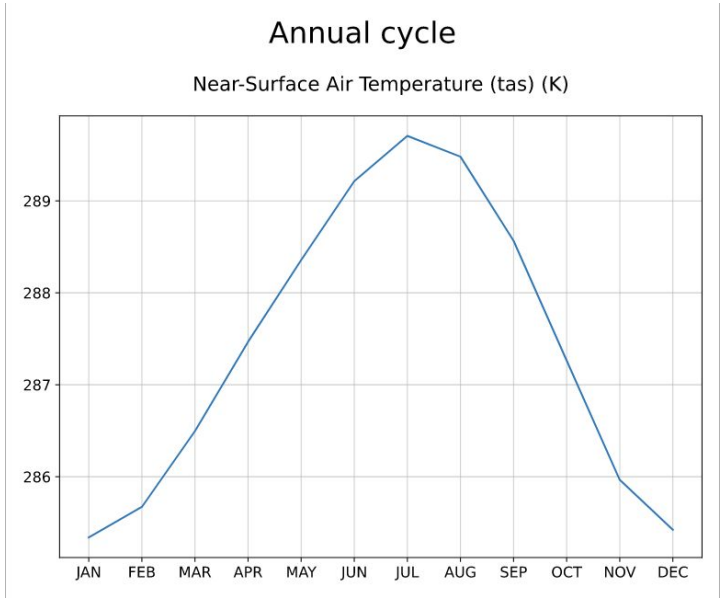
How are we (starting) to do it?

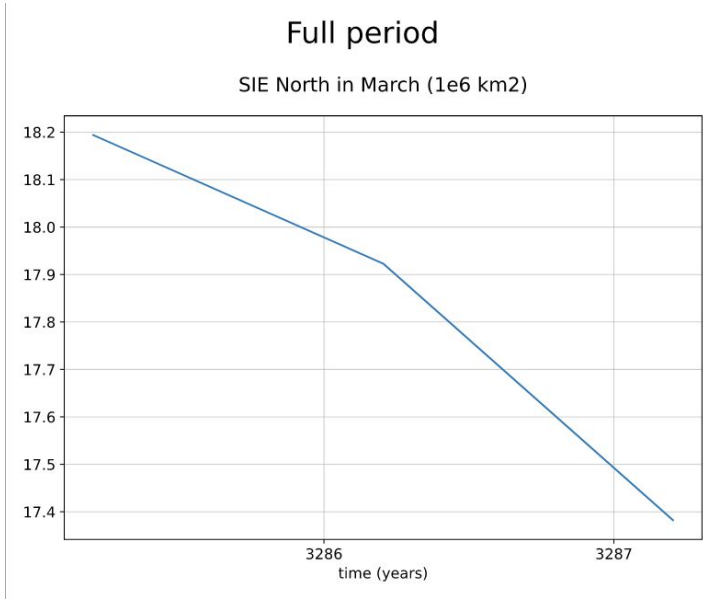
- We have deployed ESMValTool using Singularity containers in our analysis machine.
- We have created a diagnostic script that just plots the data received from ESMValTool's preprocessor.
- This diagnostic can generate different kinds of plots (climatologies, timeseries, annual cycles ...) and we are able to tweak each of them easily without touching the diagnostic code.
- Results are saved directly on the path that will be readed from the app.
- ESMValTool can use output from other diagnostic tools if they write the output in CMOR-like format (as ours do).
- We are developing in parallel a Shiny app for easy visualization.

The results









Barcelona Supercomputing Center
Centro Nacional de Supercomputación

How we implemented it?

- We created a ESMValTool diagnostic that will plot the output directly from the preprocessor and store the files in a given location following our convention: `${PATH_TO_EXP_DATA}/plots/${REALM}/${VARIABLE}`
- Filename also follows a convention: `${PLOT_TYPE}_${VARIABLE}_${DATASET}_${MIP}_${EXPERIMENT}_${ENSEMBLE}.${FILE_TYPE}`
- A shiny app scan those folders and allow the users to select the plots they want to look at

The diagnostic script

- It’s only work is to get preprocessed data and plot it accordingly to the option it receives
- It uses mapgenerator, a BSC in house tool based on Cartopy. It was developed for our operational services as a command line tool to easily generate plots from NetCDF files
- By using it, we can tweak all the relevant options for us just modifying function parameters.
- We have a separated yaml file that contains the styles for the different kind of plots and tweaks based on the variables. We can even tweak the options for each variable and plot
- The aim is that users can add new plots of an already existing type without

Our yaml config file

```
maps:
  global:
    projection: PlateCarree
    projection_kwargs:
    central_longitude: 285
    smooth: True
    lon: [-120, -60, 0, 60, 120, 180]
    lat: [-90, -60, -30, 0, 30, 60, 90]
    colorbar_location: bottom
    extent: null
    suptitle_pos: 0.87
  arctic:
    projection: NorthPolarStereo
    projection_kwargs:
    central_longitude: 270
    lon: [-180, -150, -120, -90, -60, 0,
30, 60, 90, 120, 150, 180]
    lat: [50, 60, 70, 80, 90]
    smooth: True
    draw_labels: True
    suptitle_pos: 1.
```

```
variables:
  default: &default
    colors: RdYlBu_r
    N: 20
    bad: [0.9, 0.9, 0.9]
  pr:
    <<: *default
    colors: gist_earth_r
    bounds: 0-10.5,0.5
    extend: max
  heatc0-300m:
    <<: *default
    extend: both
    bounds: 3.e11-3.75e11,0.05e11
  sos:
    default:
      <<: *default
      bounds: 25-41,1
      extend: both
    arctic:
      bounds: 25-40,1
    antarctic:
      bounds: 30-40,0.5
```

Incorporating it to the workflow

- The last step was to automate the generation of the figures so they are created and updated during the model executions
- To do this, we created a new job in our workflow that run in our analysis machine after the data arrives to our storage, which it does in a chunk by chunk fashion
- To make users life easier, the job itself is able to remove variables from the recipe that are not part of the outclass of the model. This means that we do not need to keep special recipes for runs with reduced output

Incorporating it to the workflow