

ESMValTool

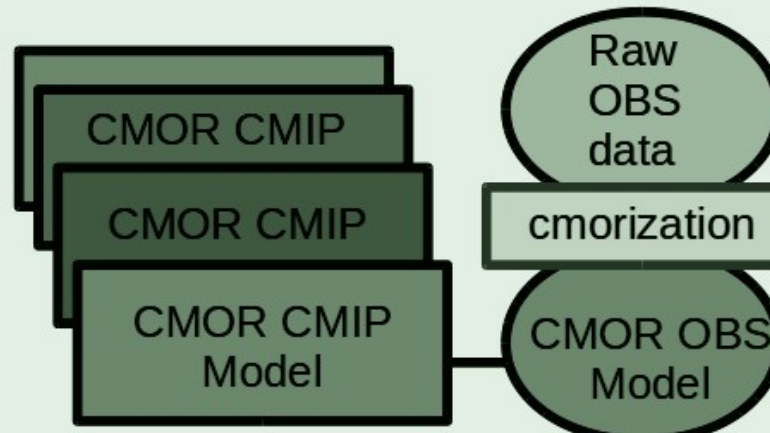
A powerful, community-driven diagnostics and evaluation tool for Earth System Models
Valeriu Predoi for the ESMValTool core development team



ESMValTool (Earth System Model eValuation Tool) in a nutshell

- A **community** diagnostics and performance metrics tool for the evaluation of Earth System Models (ESMs);
- allows for routine comparison of single or multiple **models**, either against predecessor versions or against **observations**;
- Framework: highly **modular** and flexible so that additional analyses can easily be added;
- Uses standardized **recipes** that represent specific diagnostics or performance metrics that have demonstrated their importance in ESM evaluation in the **peer-reviewed literature**.

Allows for comparison between CMIP models (different models, same models with different versions) and observational data (OBS):



Diagnostics ecosystem:

- conversion to CMOR standards of OBS data;
- use output from Preprocessor to run diagnostics;
- large collection of standard metrics and peer-reviewed diagnostic routines in multiple languages (Python, NCL, R, Julia);
- outputs netCDF files, plots and metrics;

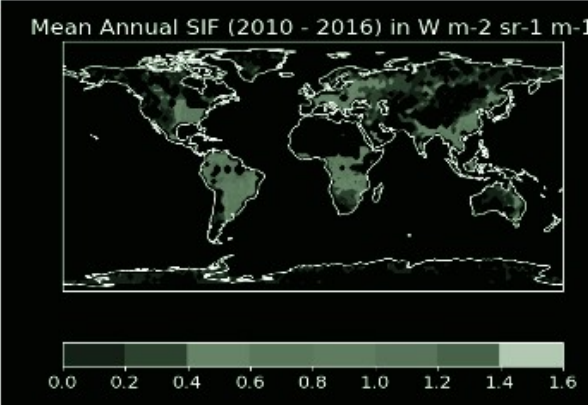
Diagnostics library: constantly **growing** and evolving based on the **needs** and **contributions** of climate scientists = ecosystem.

Publication:

- data provenance information is provided by the tool for easy tracking and analysis replication;
- preprocessor output can be reused for diagnostic re-runs, if needed, to save computational resources.

Final output:

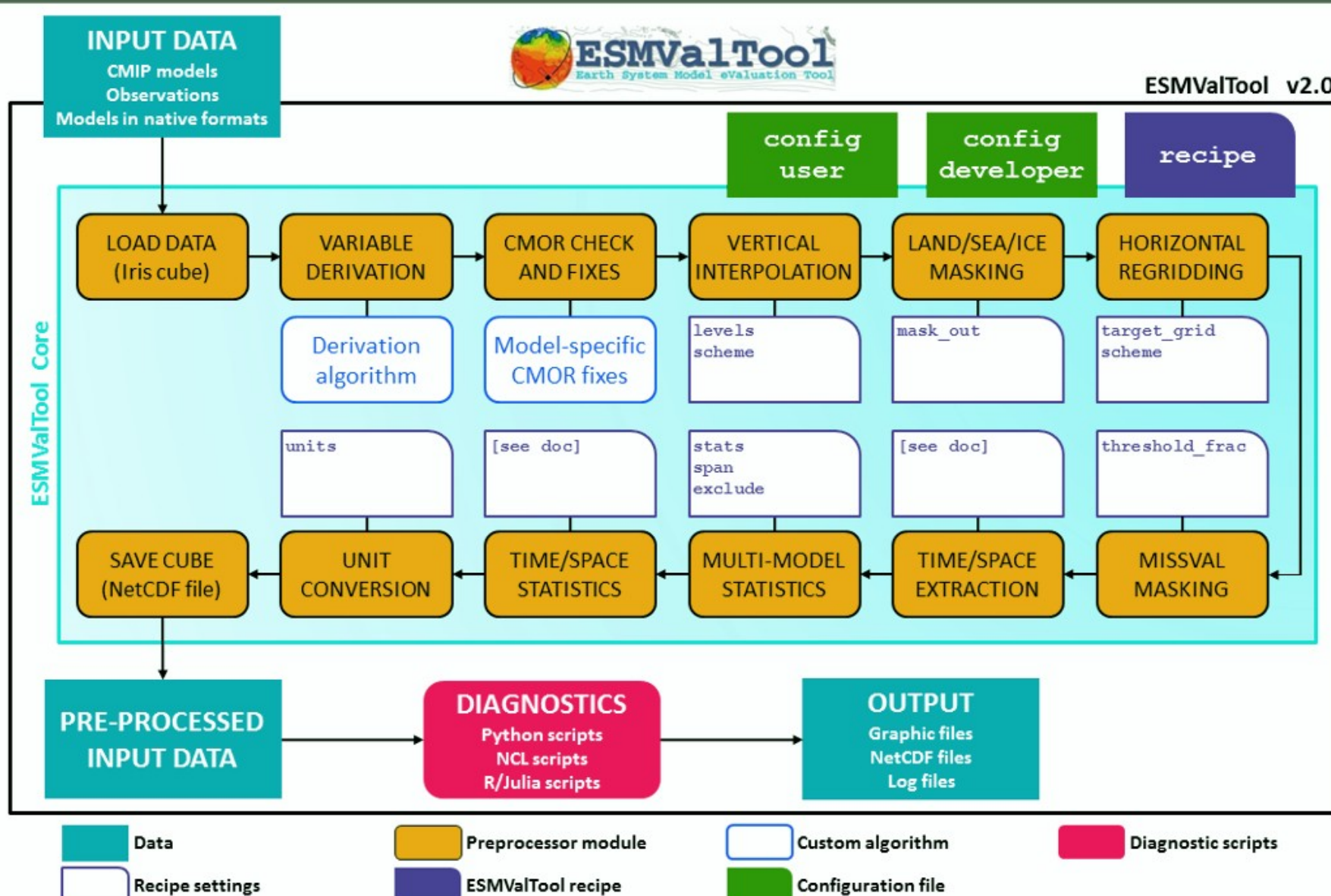
- data files
- provenance
- plots
- happiness



- Preprocessing core:**
- finds data (CMIP and OBS) in CMOR format;
 - applies CMOR checks and fixes;
 - derives custom variables;
 - runs standard data analysis steps common to all data (time, area, volume extractions, level selections, masking, regridding, multimodel stats etc.);
 - outputs netCDF files and summary data files;

Preprocessor schematic (esmvalcore)

Group of steps executing standardized functions in either standard or custom order



USR: CODEBASE and INSTALLATION

Package location:

<https://github.com/ESMValGroup/ESMValTool.git>

The most recent version of ESMValTool source code is hosted on GitHub and is constantly updated when new features are merged and with every new release. GitHub offers a flexible platform for continuous integration via developers' pull requests (PR), issue tracking and resolution and installation, unit and functional testing (via CircleCI in this case):
<https://github.com/ESMValGroup/ESMValTool/>

ESMValTool is a Python 3 package with cross-language dependencies meant to support diagnostics in various programming languages (Python, NCL, R etc.); it is comprised of the preprocessing core (esmvalcore, pure Python package) and of the diagnostics library (cross-language package; this uses esmval-core as a dependency). Installed versions are available directly on compute clusters like CEDA-Jasmin or DKRZ or the user can install it via `conda install`

Run command:

```
[user@machine]$ conda install -c esmvalgroup -c conda-forge esmvaltool
```

DEV: ENVIRONMENT SETUP and INSTALLATION

Environment file: `environment.yml`

For diagnostic (and core) developers it will be easier if they create the esmvaltool software environment and install the tool for development, testing and contributions to the code base via git.

A Python 3.x anaconda3 or miniconda3 installation is required *a-priori*, since the esmvaltool environment is a conda-based one.

Run command:

```
[user@machine]$ git clone https://github.com/ESMValGroup/ESMValTool.git
[user@machine]$ conda env create -n esmvaltool -f environment.yml
[user@machine]$ source activate esmvaltool
(esmvaltool) [user@machine]$ pip install -e ".[develop]"
```



USER INPUTS

User config file: `config-user.yml` Recipe: `recipe.yml`

User-defined parameters for the run e.g.: path to root directory for output; desired output graphical file formats (pdf, eps, png etc.); root paths for data and types of DRS.

A single configuration file can be used for a large number of diagnostic runs. It is also optimized for a minimum number of inputs.

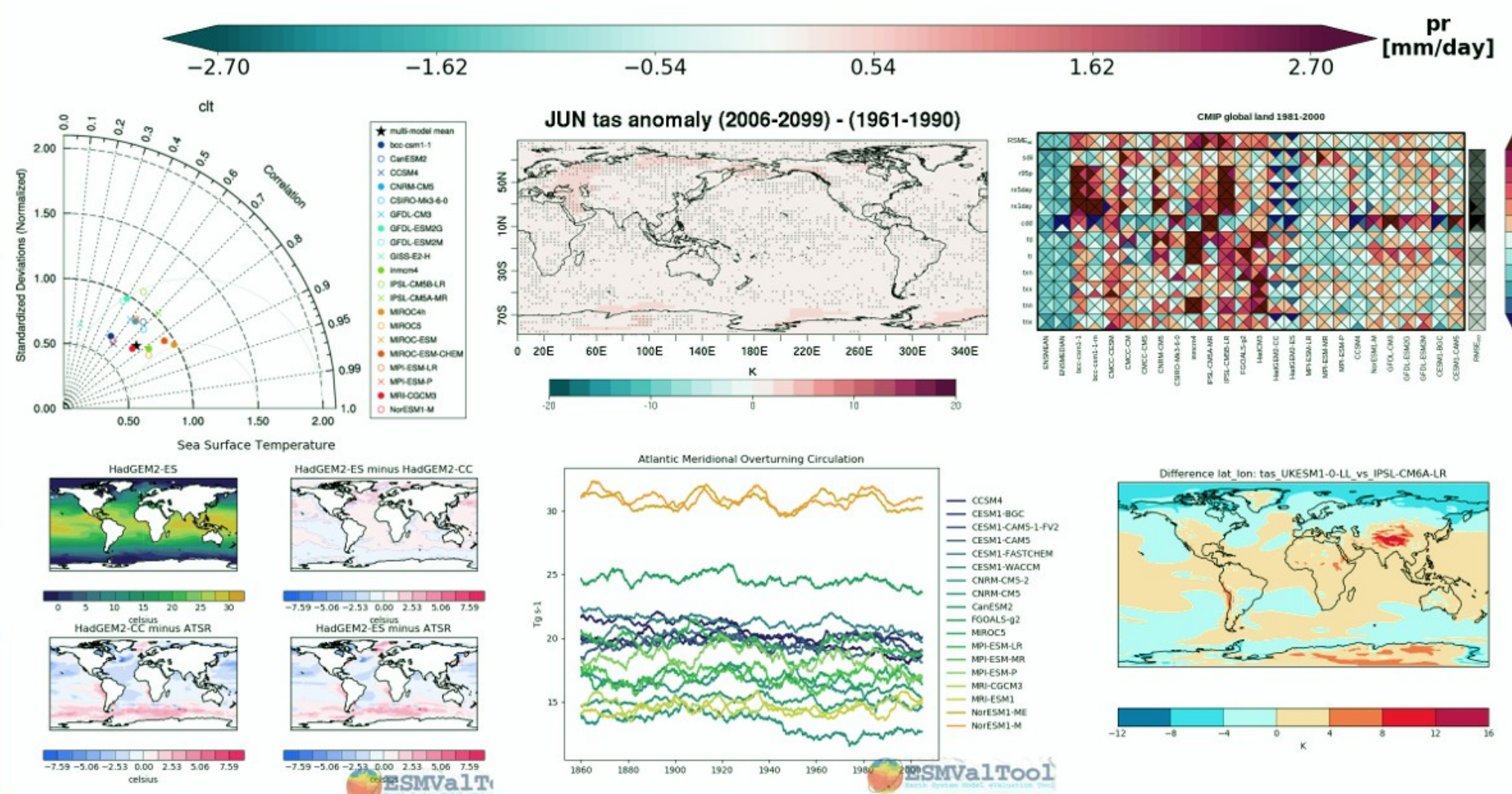
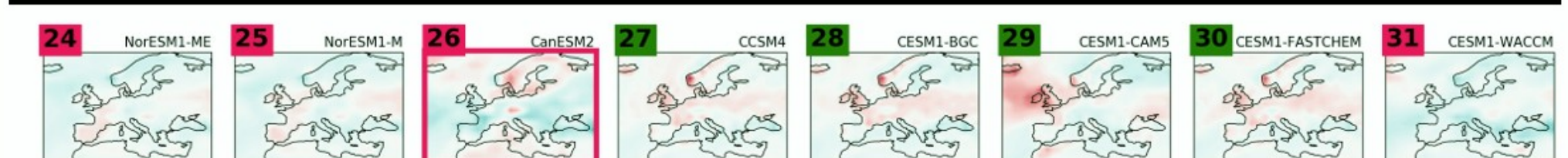
Run command:

```
(esmvaltool) [user@machine]$ esmvaltool -c config-user.yml recipe.yml
```

Parameters needed by the core preprocessor(s) and diagnostic(s): three groups: **dataset(s) parameters**, **preprocessor(s) settings** and **diagnostic(s) settings**.

Each diagnostic may use any number of datasets, any number of preprocessors and any combination of variables (standard or derived variables). A single recipe file can be used for a large number of preprocessors and diagnostics.

```
diagnostics:
aa: landsurf_surfrad:
  description: Autoassess test diag for Land-Surface Surfrad.
  variables:
    rns: # Surf SW net all sky
      field: T2Ms
      derive: true
      force_derivation: false
      fx_files: [sftlf]
    rlns: # Surf LW net all sky
      field: T2Ms
      derive: true
      force_derivation: false
      fx_files: [sftlf]
  additional_datasets:
    {dataset: CERES-EBAF, project: obs4mips, level: L3B, version: Ed2-7, start_year: 2001, end_year: 2012, tier: 1}
  scripts:
    autoassess_surfrad: $autoassess_surfrad_surfrad_settings
    script: autoassess/area_base.py
    title: "Autoassess Land-Surface Diagnostic Surfrad Metric"
    area: land_surface_surfrad
    control_model: MPI-ESM-LR
    exp_model: MPI-ESM-MR
    obs_models: [CERES-EBAF]
    obs_type: obs4mips
    fx: [sftlf]
    start: 1997/12/01
    end: 2002/12/01
```



Additional Notes