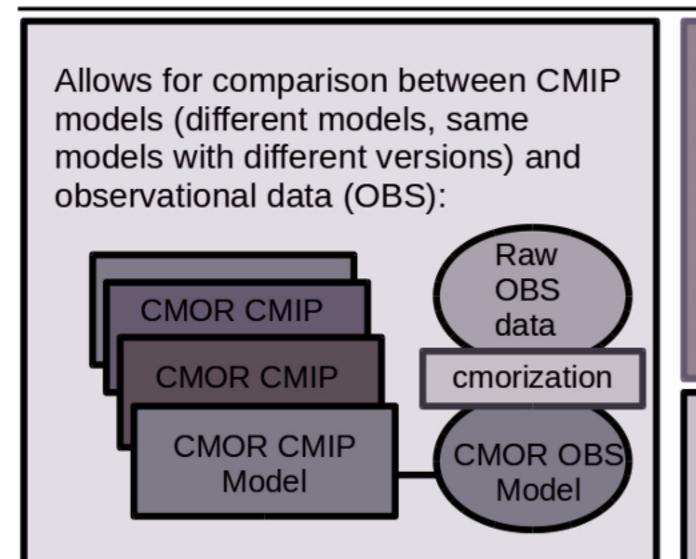
Community evaluation and diagnostics tool

## 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.



Preprocessing core:

derives custom variables;

start: 1997/12/01

end: 2002/12/01

- applies CMOR checks and fixes;

finds data (CMIP and OBS) in CMOR format;

data (time, area, volume extractions, level

runs standard data analysis steps common to all

selections, masking, regridding, multimodel stats

- outputs netCDF files and summary data files;

#### Diagnostics ecosystem:

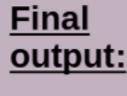
conversion to CMOR standards of OBS data;
use output from Preprocessor to run diagnostics;
large collection of standard metrics and peer-reviewed diagnostic routines in multiple languages (Python, NCL, R, Julia);

outputs netCDF files, plots and metrics;

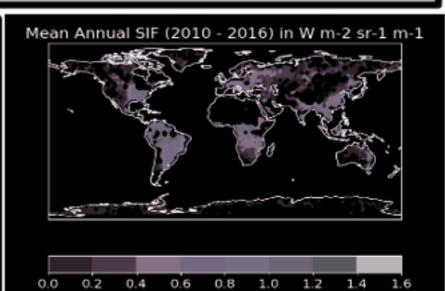
Diagnostics library: constantly **growing** and evolving based on the **needs** and **contributions** of climate scientists = ecosystem.

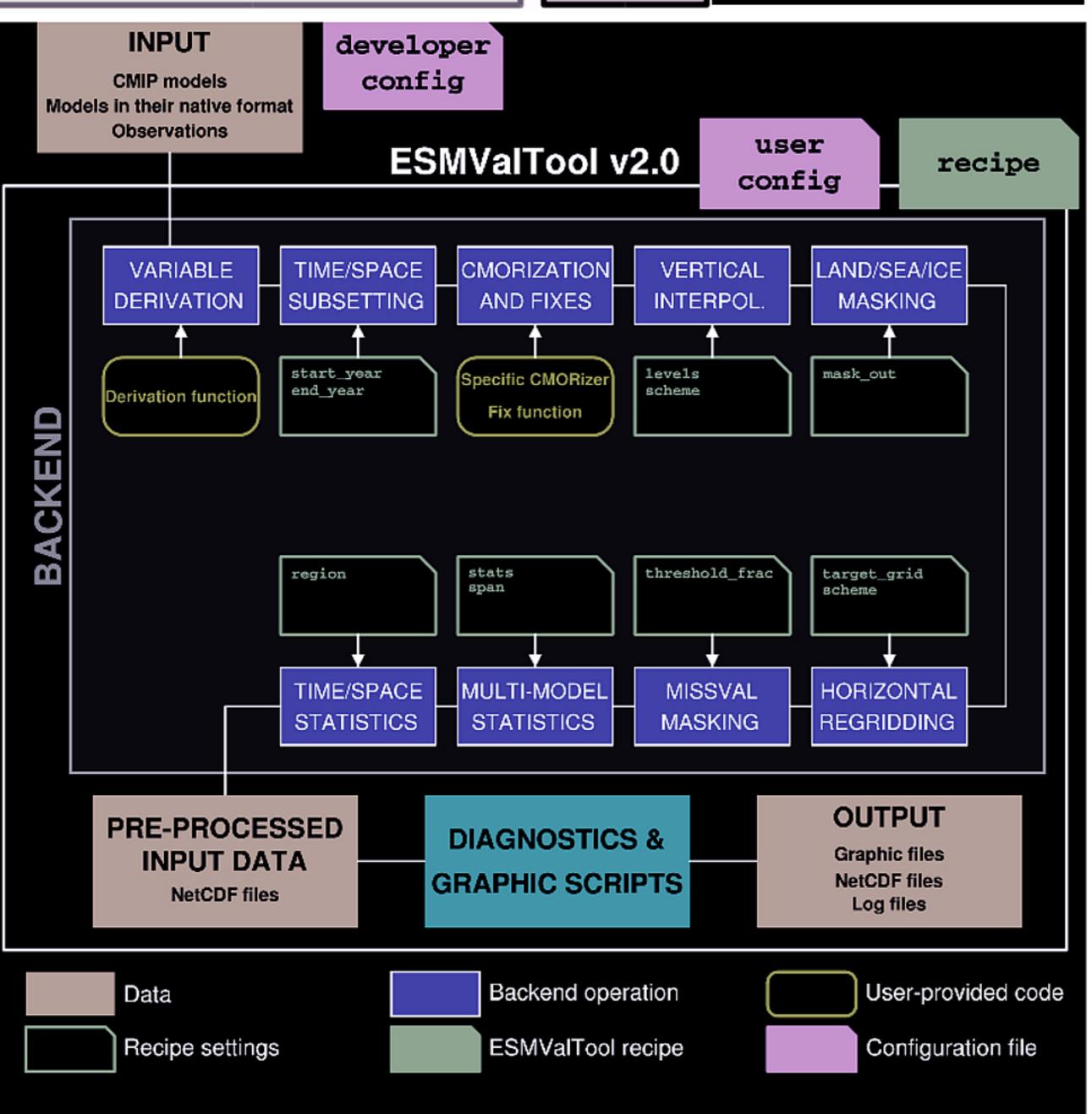
#### Publication:

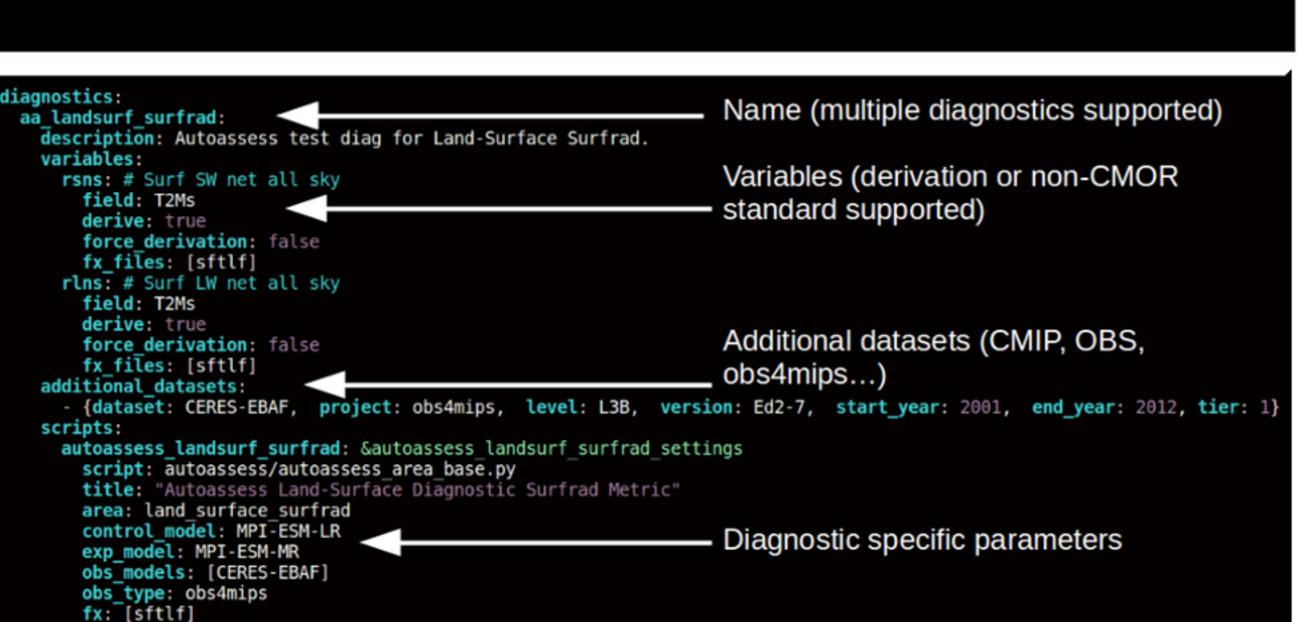
data provenance information is provided by the tool for easy tracking and analysis replication;
preprocessor output can be reused for diagnostic re-runs, if needed, to save computational resources.



-data files -provenance -plots -happiness







# 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 (esmval-core, 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 apriori, since the esmvaltool environment is a conda-based one.

The software package dependencies needed by esmvaltool are specified in the environment.yml file and this one is used to create the software environment.

After the environment has been created and activated, the tool is installed in the usual manner for a Python package.

## 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-definded 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.

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.

## Run command:

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





