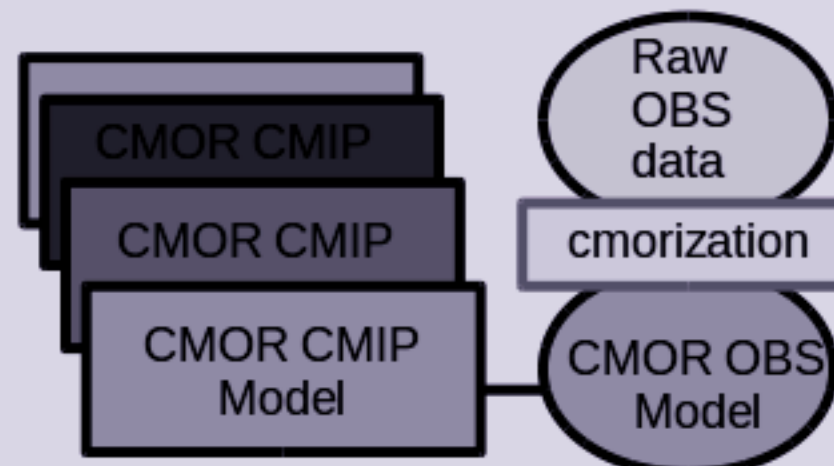# ESMValTool
Earth System Model eValuation Tool
Community evaluation and diagnostics tool

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

- Raw OBS data
- cmorization
- CMOR CMIP
- CMOR CMIP
- CMOR CMIP Model
- CMOR OBS Model

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);
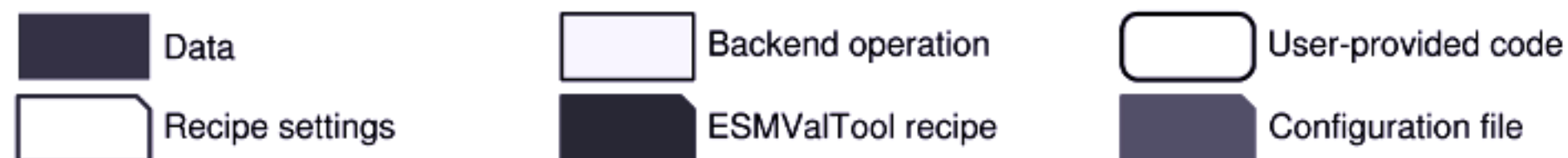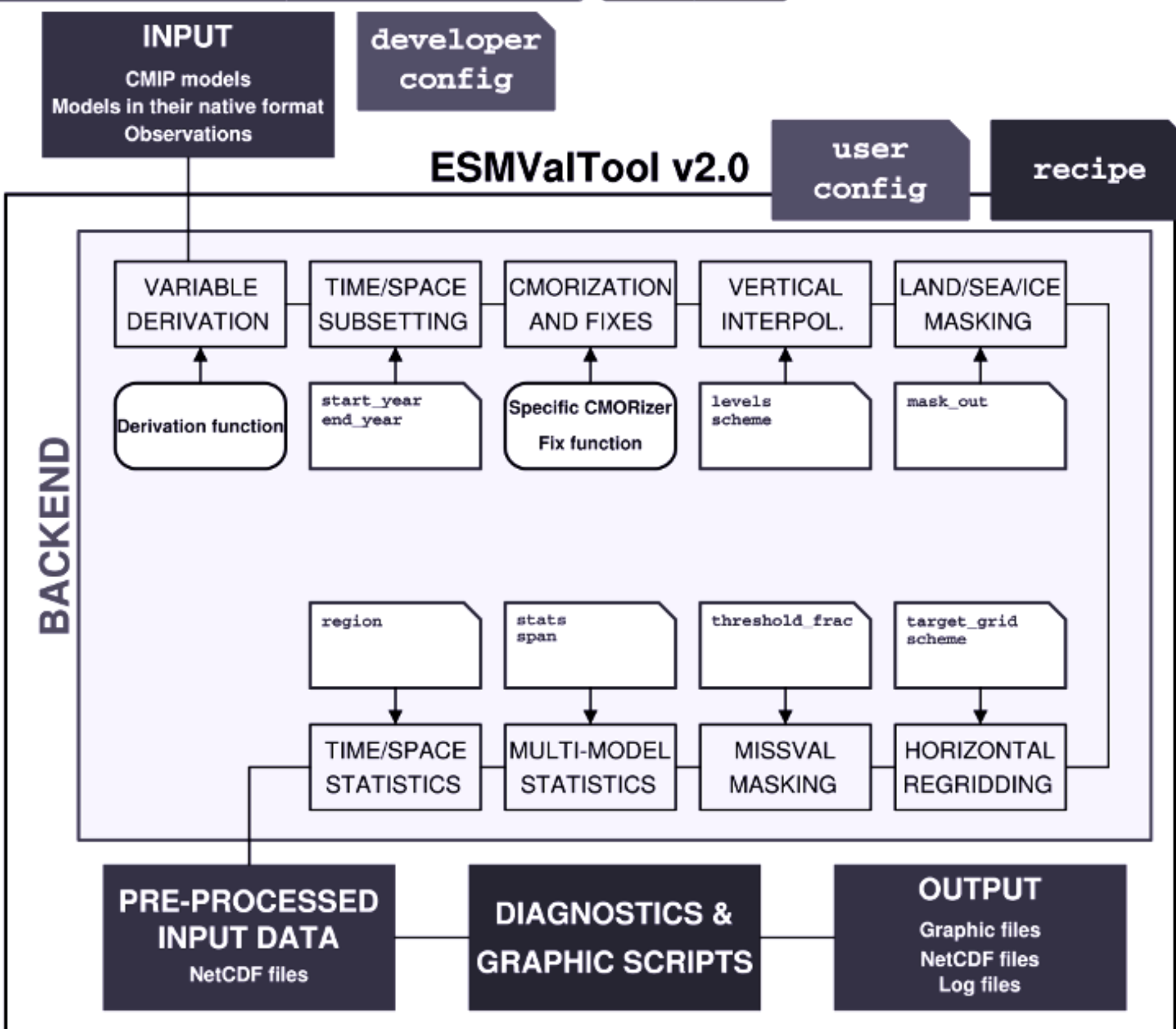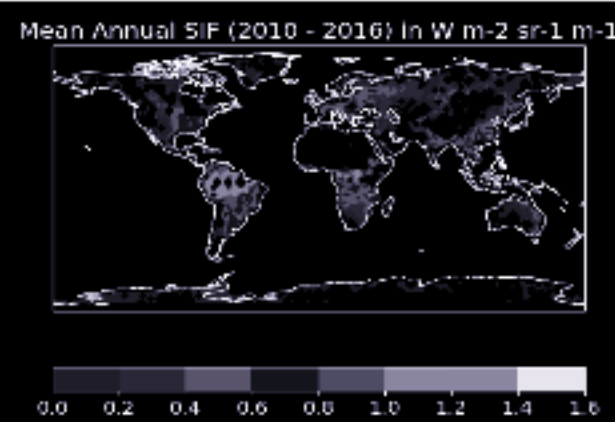- outputs netCDF files, plots and metrics;

Diagnostics library: constantly **growing** and evolving based on the **needs** and **contributions** of climate scientists = ecosystem.
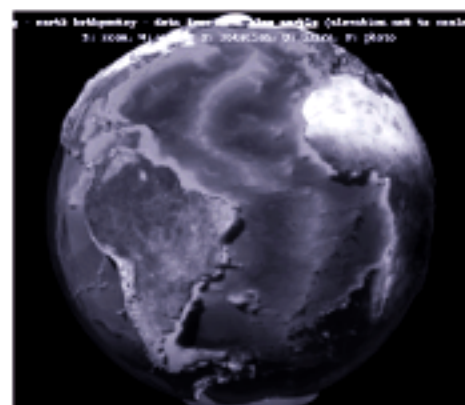
Publication:
- data provenance information is provided by the tool for easy tracking and analysis replication;
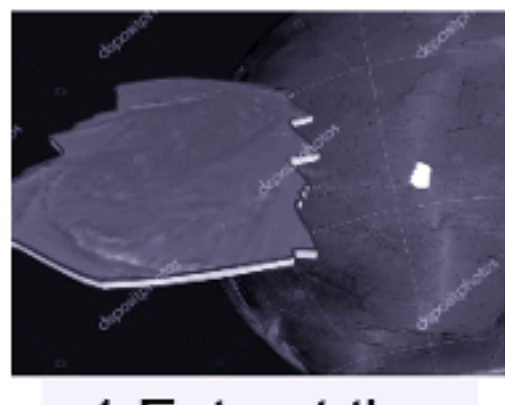- preprocessor output can be reused for diagnostic re-runs, if needed, to save computational resources.

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;

Final output:
- data files
- provenance
- plots
- happiness

Mean Annual SIF (2010 - 2016) in W m-2 sr-1 m-1

ESMValTool v2.0

INPUT
CMIP models
Models in their native format
Observations

developer config

user config

recipe

BACKEND

VARIABLE DERIVATION	TIME/SPACE SUBSETTING	CMORIZATION AND FIXES	VERTICAL INTERPOL.	LAND/SEA/ICE MASKING
Derivation function	start_year end_year	Specific CMORizer Fix function	levels scheme	mask_out

region	stats span	threshold_frac	target_grid scheme
TIME/SPACE STATISTICS	MULTI-MODEL STATISTICS	MISSVAL MASKING	HORIZONTAL REGRIDDING

PRE-PROCESSED INPUT DATA
NetCDF files

DIAGNOSTICS & GRAPHIC SCRIPTS

OUTPUT
Graphic files
NetCDF files
Log files

Legend:
- Data
- Backend operation
- User-provided code
- Recipe settings
- ESMValTool recipe
- Configuration file

Preprocessor example

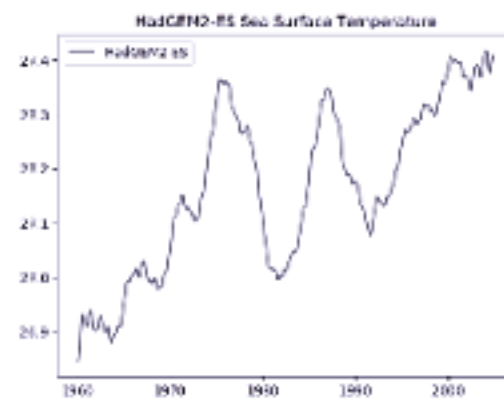Start with 3D + time data (Temperature)

1. Extract the surface layer

2. Extract equatorial region

$$Weighted\ Avg_x = w_1x_1 + w_2x_2...w_nx_n$$
$$w = relative\ weight(\%)$$
$$x = value$$

3. Take the area-weighted average of the data.

HadGEM2-ES Sea Surface Temperature

Product is time dimensional (1D) cube.

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]$ git clone https://github.com/ESMValGroup/ESMValTool.git
[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.

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]$ 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
```