IS-ENES3 Deliverable D9.5

Final IS-ENES3 ESMValTool version

Reporting period: 01/01/2022 – 31/03/2023

Authors: Rémi Kazeroni (DLR), Bouwe Andela (NLeSC), Bill Little (MetOffice), Saskia Loosveldt Tomas (BSC), Valeriu Predoi (UREAD), and the ESMValTool, ESMValCore and Iris development teams

Reviewers: Mario Acosta (BSC), Carsten Ehbrecht (DKRZ), Angelika Heil (DKRZ)

Release date: 30/03/2023

ABSTRACT

This report describes the work achieved within IS-ENES3 on improving and extending the capabilities and performance of the ESMValTool backend as well as on coupling externally developed diagnostics and metrics to ESMValTool. This work has been driven by user requirements and community standards defined within IS-ENES3, leading to a major rewriting of ESMValTool. All changes were first released with version 2.0. This software package has been specifically designed for performance intensive tasks with a technical backend based on the Iris library. The improved modularity of ESMValTool facilitates the coupling of diagnostic packages and recipes as, for instance, extensively used by scientists for the IPCC AR6 report. Continuous technical developments on ESMValTool applying modern software engineering techniques led to regular releases with further improvements. With version 2.8 (March 2023), all developments described in this report have been released.

Revision table			
Version	**Date**	**Name**	**Comments**
Release for review	24/02/2023	Rémi Kazeroni	First version sent to internal reviewers
Final version	30/03/2023	Rémi Kazeroni	Revised after reviews

Dissemination Level	
PU	Public

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824084

Table of contents

Executive Summary

This report describes the work achieved within IS-ENES3 on improving and extending the capabilities and performance of the ESMValTool backend as well as on coupling externally developed diagnostics and metrics to ESMValTool. This work has been driven by user requirements and community standards defined within IS-ENES3, leading to a major rewriting of ESMValTool. All changes were first released with version 2.0. This software package has been specifically designed for performance intensive tasks with a technical backend based on the Iris library. The improved modularity of ESMValTool facilitates the coupling of diagnostic packages and recipes (evaluation workflows) as, for instance, extensively used by scientists for the IPCC AR6 report. Continuous technical developments on ESMValTool applying modern software engineering techniques led to regular releases with further improvements. With version 2.8 (March 2023), all developments described in this report have been released.

The work presented in this document significantly improved ESMValTool, which is now a robust, scalable and easy-to-use infrastructure for model evaluation. The wider climate community is encouraged to continue using ESMValTool and contributing to its developments as needed for their scientific applications.

Future improvements for ESMValTool include in particular further widening of the range of scientific applications, developing recipe testing workflows to automate the detection of changes in recipe output, and further improving the memory efficiency (lazy capabilities) to prepare the evaluation tool for processing high-resolution data.

The development team of ESMValTool will continue to provide guidance and support. This will help advancing ESMValTool and ensure its long-term sustainability.

1. Objectives

The Earth System Model Evaluation Tool (ESMValTool) [1] is an open-source, community-developed, climate model diagnostics and evaluation software package. ESMValTool has been developed with the aim of taking model evaluation to the next level by facilitating analyses of many different model components, providing well-documented source code and scientific background of implemented diagnostics and metrics and allowing for traceability and reproducibility of results (provenance). ESMValTool was originally designed and optimized to handle the large data volume of the output from CMIP6 [2] and has been used to support about 50 analyses used in the IPCC WGI AR6 [3]. It consists of a backend (ESMValCore) that performs common pre-processing operations and a diagnostic part which includes diagnostics and performance metrics for specific scientific applications.

This report describes the work done since the start of IS-ENES3 on improving and extending the capabilities and performance of ESMValTool. The main developments targeted in this deliverable are the following ones:

- Improving ESMValTool to efficiently handle the large amount of data from CMIP6, future phases of CMIP and CORDEX data together with observational data required for model evaluation;
- Ensuring efficiency, provenance tracking, automated testing, and documentation following community standards and modern software engineering practices;
- Improving user-friendliness including in particular easier installation and portability to multiple platforms and compute infrastructures;
- Facilitating the integration of externally developed diagnostic packages and metrics as well as implementation of AR6 recipes.

This deliverable includes a final IS-ENES3 ESMValTool release, namely v2.8 (March 2023), which comprises all developments and enhancements described in this report. This final IS-ENES3 release is based on ESMValTool version 2.0, a major update from the first release of the evaluation tool. ESMValTool v2.x has been documented scientifically and technically in a series of papers [1,3,4,5,6].

2. Technical improvements of ESMValTool

2.1. Coding workshops and coordination of ESMValTool activities

During the project, a series of 8 coding workshops, funded by IS-ENES3, have been organized (in-person, hybrid and virtual meetings) to work on technical improvements of ESMValTool and discuss strategic issues. These events have been beneficial to structure the ESMValTool community, shape its governance and coordinate continuous developments of the software. The summaries of these meetings are made available on the ESMValTool website. ESMValTool activities have also been coordinated during regular WP9 video calls, meetings of the ESMValTool Technical Lead Develop Team and meetings between ESMValTool and Iris developers. The outcome of such meetings is made publicly available on the ESMValGroup GitHub where the software is developed.

2.2. Modularity and workflow

In the design phase of ESMValTool v2.0, it was decided to fully separate the preprocessor part from the diagnostic one in order to provide more flexibility and improve the maintainability of the evaluation tool. The preprocessor part consists of common operations, such as extraction, regridding, masking, time subsetting, and reformatting to CMOR standards, which are applied on input data before these are used in diagnostic scripts for scientific analyses. In the predecessor version, ESMValTool v1, these two parts were not clearly separated which could result in inefficient data preprocessing and slow performance when performing analyses of large data volumes [1].

In order to address the performance bottleneck and allow for efficient evaluation of large amounts of input data with the execution of parts of the processing in parallel tasks (see Section 2.6.1), the structure of ESMValTool has been fully revised and now comprises two main components: ESMValCore, which is a Python Package providing preprocessing capabilities, and ESMValTool, which is a suite of diagnostic scripts and performance metrics routines. ESMValCore is written in Python 3 and takes the advantage of the Iris library (Met Office, 2010-2023) to efficiently process large amounts of input data (see Section 2.7). A preprocessor module of ESMValCore is a Python function that takes an Iris cube and a set of arguments as input and returns a cube. The Iris library is used to load the data as cubes and pass these between preprocessor functions used in the evaluation process. The parameters controlling preprocessor functions can be specified in an ESMValTool recipe (see below). The ESMValCore module comes with an API that makes its functionality accessible by external software or usable interactively, such as in a Jupyter Notebook (see examples in the ESMValCore package).

The diagnostic part consists of multi-language scientific diagnostics currently supporting scripts and packages written in Python 3, NCL, R, and Julia to enable contributions from a wider community of scientists. Support for other freely available languages could be added upon request. The evaluation workflow is controlled by a set of parameters specified in a configuration file and an ESMValTool recipe, both using the YAML format. A recipe contains four main sections: the documentation section which provides a short description of the analysis and corresponding references, the dataset section that lists input data used in the evaluation, the preprocessor section in which preprocessing functions, their settings and order are specified, and a diagnostics section that defines the diagnostic scripts used, their settings and order. The diagnostic package contains many example recipes on which more complex analyses can be built.

ESMValCore also contains a task manager that governs the evaluation workflow. This includes a data finder, the preprocessing operations with output written to netCDF files and further analyses with tailored diagnostic scripts; see Fig. 1 for a schematic representation of ESMValTool v2. The revisited design of ESMValTool allows the software to determine which tasks of the evaluation workflow can be run in parallel (see Section 2.6.1 on performance improvements). The enhanced modularity of ESMValTool v2 eases the maintenance of the packages and the addition of new features, not only for core functionalities but also regarding the implementation of new scientific analyses using diagnostic scripts and external packages.

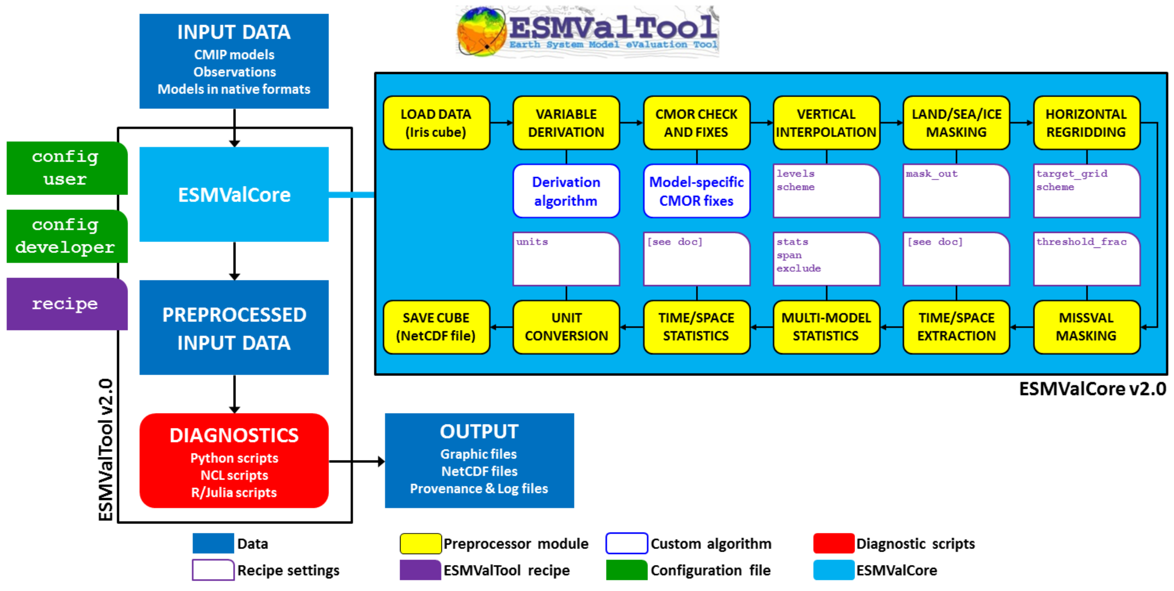


Figure 1: Schematic of the ESMValTool architecture (from [1]).

2.3. Continuous development

The developments of ESMValTool and ESMValCore are done as open-source projects on GitHub, where the source code and discussions about ongoing developments and new features are publicly available. The development of both packages relies on modern software engineering techniques, such as automated testing (see Section 2.4), code reviews, software quality monitoring and extensive documentation. While maintaining very high scientific standards for the diagnostics, coding standards have been relaxed compared to ESMValCore contributions to make it easier for scientists to contribute their code to ESMValTool. Development and maintenance of core functionalities affecting the whole software package is usually done by software engineers whereas contributions to diagnostics and analyses are mainly done by scientists.

2.3.1. Documentation

The documentation of ESMValTool and ESMValCore is available at https://docs.esmvaltool.org and hosted by ReadTheDocs. It is built automatically with Sphinx to allow for continuous improvements of the documentation as the packages are being developed. The documentation has been extensively revised during the release of ESMValTool v2.0 and continuously improved in following releases. Among others, it includes instructions on how to get started with the software, a gallery showcasing results produced with ESMValTool, extensive documentation about available recipes and diagnostics, a description of all preprocessor modules, and the documentation of the public API of ESMValCore. The source code of the documentation is available on GitHub and consists of reStructuredText files (.rst) which can be edited via GitHub pull requests.

2.3.2. Contribution guidelines

Extensive developer documentation has been written to explain how to make use of the continuous integration services employed for the development of the packages (ESMValTool contribution guidelines, ESMValCore contribution guidelines). These guidelines are based on requirements on coding standards defined in WP5[1] and on years of experience developing ESMValTool. Contributors can find instructions on how to use static analysis code tools to check the quality of their code for the different languages supported. These guidelines also include recommendations on how to write clear and concise code, write and preview documentation, handle automated tests and add new dependencies to the packages. Detailed recommendations are available for contributors of recipes, diagnostics and reformatting scripts for observational and reanalysis datasets.

When developing core functionalities or the API, contributors are asked to avoid making backward incompatible changes. As such changes can sometimes become unavoidable to enhance

[1] available on https://iseneseval.github.io/diagstandards/best_practices.html

ESMValTool or improve its user-friendliness, guidelines have been added on how to discuss such changes with the Technical Lead Development Team and handle feature deprecations over the course of two consecutive releases[2].

Guidelines have also been added regarding recipe maintenance[3] which is necessary to ensure that public ESMValTool recipes can still be run with newer versions of the software and that the outputs are not affected by changes to core functionalities. When contributing a new recipe to ESMValTool, it is usually expected that, at least, one contributor would act as "recipe maintainer" and be the first contact point for the Technical Lead Development Team. The ESMValTool development community strives to release new versions in which a maximum number of recipes works correctly. If the maintenance of a recipe is no longer possible, that recipe could be retired as described in the contribution guidelines[4].

2.3.3. Code reviews

Code reviews have been introduced and embraced by the community to ensure reliability of new contributions and transfer of knowledge within the community. Guidelines for code reviews have been added both for ESMValTool and ESMValCore. Checklists are automatically generated when opening a pull request to ease the work for contributors and reviewers. For ESMValTool, two separate code reviews are usually done: a technical review to check the code quality and a scientific review to check the relevance and accuracy of the changes. The code reviews for ESMValCore consist of more thorough technical assessments of the code quality and design to ensure that the changes adhere to the coding standards without reducing the performance of the package.

2.3.4. Releases

During the IS-ENES3 project, a total of **15 releases**[5] of ESMValTool has been made. This includes the major release of ESMValTool v2.0, preceded by 4 beta releases, and followed by 9 regular releases and a bugfix one. For ESMValCore, a total of **40 releases**[6] has been made, including the major release of ESMValCore v2.0, preceded by 12 alpha and beta releases and followed by 8 regular releases, 2 bugfix releases, and 17 release candidates. These releases were all coordinated by core developers involved in IS-ENES3 and supported by the manager of the previous release.

[2] https://docs.esmvaltool.org/projects/esmvalcore/en/latest/contributing.html#backward-compatibility
[3] https://docs.esmvaltool.org/en/latest/community/maintainer.html
[4] https://docs.esmvaltool.org/en/latest/community/broken_recipe_policy.html#broken-recipe-policy
[5] https://github.com/ESMValGroup/ESMValTool/releases
[6] https://github.com/ESMValGroup/ESMValCore/releases

Releases follow a regular 4-months schedule that is publicly available in the documentation[7]. Frequent releases enable the development community to make new features and bug fixes widely available in a timely manner. The release of both packages is tied together, starting with ESMValCore. Since version 2.4, ESMValCore is first made available as a release candidate that is used to test all recipes of ESMValTool (see Section 2.4.4). If bugs are found during this testing phase, a new release candidate is made available to fix those issues and recipes are tested again. This process minimizes the risks of having to release bugfix versions of ESMValCore if bugs are found while testing recipes. The release of the final ESMValCore version is followed by the ESMValTool release of the same version number within a few days. Detailed steps on how to make releases are available in the documentation of ESMValTool[8].

2.4. Automated testing

2.4.1. Testing strategy and continuous integration services

Automated testing is an important and integral part of any modern distributed software package. It is meant to continuously examine the correct and desired behaviour of the code, both atomically (unit tests that test the behaviour of specific, individual functions), and globally (integration or regression tests that examine the smooth running of entire sub-assemblies of the code, isolated or with regard to (parts of) the workflow). Testing is directly motivated by the code's nature of being under constant development. Using a number of common-use settings (e.g. running in different modes - development vs stock builds, documentation builds, running over different supported platforms, etc.), we can identify various breakage points by performing testing. As such, we have deployed continuous integration test suites over two main testing platforms: CircleCI and GitHub Actions. The reason why we use two different platforms (vendors, but only their free plans are used) is that each offers a number of benefits. CircleCI offers easy use of already deployed containerized builds (so that the user does not have to wait for a complete rebuild of the environment, and an installation of ESMValTool happens before actually running the test suite). GitHub Actions offers free testing over multiple supported operating systems, and using different versions of Python. CircleCI tests are usually run with every pull request, since they offer a fast, yet fairly comprehensive alternative to testing, and Github Actions are run nightly, being more comprehensive, but also slower.

We have a large number of unit, integration, and regression tests. The latter tests are used to compare numerical output versus desired results. This is a type of integration test which is more oriented towards correct results. The test run times are usually short since we use a number of modern optimization tools. We run tests on multiple cores (multiple processes), we optimize

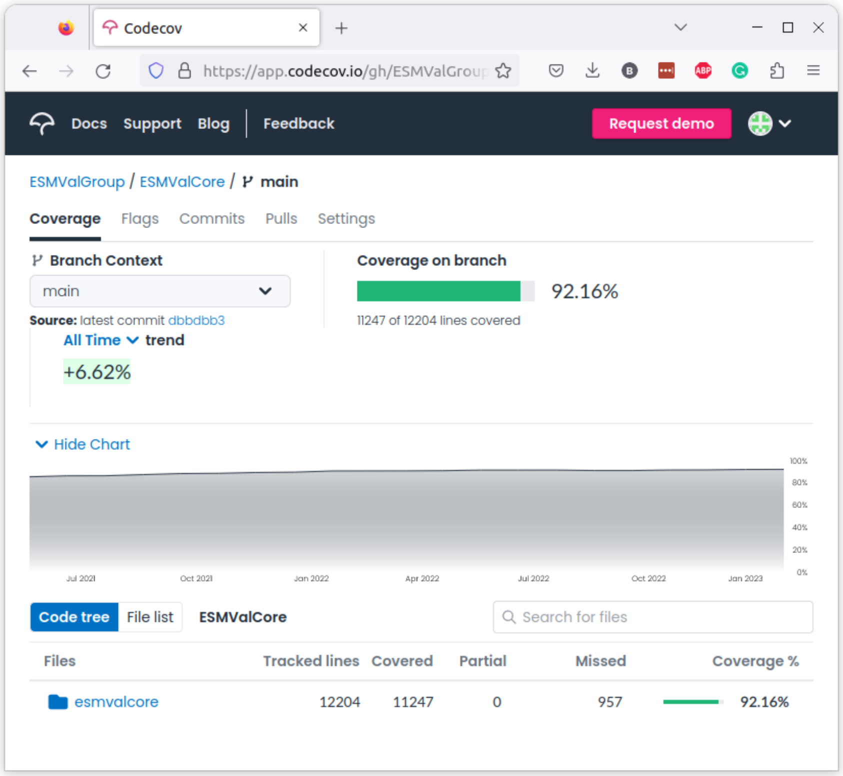
[7] https://docs.esmvaltool.org/en/latest/community/release_strategy.html#release-schedule
[8] https://docs.esmvaltool.org/en/latest/community/release_strategy.html#detailed-timeline-steps

memory bandwidth/consumption by utilizing Dask numerical arrays for loading testing data, and we employ test monitoring: a dedicated test that monitors the performance of all our tests, allowing us to make adjustments/optimization every time we notice tests that are computationally inefficient. Testing usually covers all functional aspects of the package: from testing the building of the environment (thus finding any possible conflicts between two or more of the package's dependencies, obsolete locations of our dependencies, etc.) to actually running "mock"[9] end-to-end analyses. Another aspect of testing is to create the "last stable environment" - in a world where software packages evolve rapidly, we may encounter situations where the environment becomes unsolvable for a period of time. To eliminate this possibility, we run periodic tests that build stable environments, and record them as hard copies in conda lock files[10], that can later be used to recreate the last stable environment.

2.4.2. Test coverage

In summer 2021 we started using the code coverage service Codecov to make code coverage more visible on pull requests to the ESMValCore and for each pull request, we require that changed code is covered by unit tests. This has led to a team culture where everyone writes unit tests for their contributions, thus improving reliability and maintainability of the code. Figure 2 shows how the code coverage increased since we started using Codecov.

[9] mock here denotes that sample or testing data is used, rather than real-life data, which is very large and difficult to be transferred to various testing sites; these tests aim to find any broken links in the analysis workflow.
[10] https://docs.esmvaltool.org/en/latest/quickstart/installation.html#installation-from-the-conda-lock-file



Source: latest commit dbbdbb3

Coverage on branch 92.16%

11247 of 12204 lines covered

+6.62%

Files	Tracked lines	Covered	Partial	Missed	Coverage %
esmvalcore	12204	11247	0	957	92.16%

Figure 2: Code coverage evolution over time.

As ESMValCore is the library that underpins the ESMValTool, we set higher standards for the reliability and code quality of contributions to ESMValCore than to ESMValTool, because if something breaks in ESMValCore, many users are affected. ESMValCore contributors are typically technically skilled enough to implement unit tests and the Technical Lead Development Team helps those who are not yet skilled enough. For ESMValTool, we chose not to make unit tests compulsory because fewer users are affected if a single diagnostic or recipe breaks and contributors typically do not have the skills and/or time to implement unit tests. To ensure a reliable user experience with ESMValTool, we instead rely on peer review to produce known good outputs and test by running the recipes regularly and comparing the results to those known good outputs. This is described in more detail in section 2.4.4 Automatic regression testing. For these reasons, we only use a code coverage service for ESMValCore.

2.4.3. Testing service for recipe

In collaboration with WP7, a testing service for scientific contributions to ESMValTool has been set up. This service consists of a bot, named ESMValBot, that is deployed and run on a virtual machine hosted at DKRZ. It allows developers to test their recipes by requesting a run simply by writing a comment in their ESMValTool pull request: “`@esmvalbot Please run` `recipe_xyz.yml`". Such a comment triggers a new installation of ESMValTool, followed by the recipe run. The usage is limited to recipes not requiring more than 64 GB of RAM. The service has access to data available at DKRZ. This addition enhances the user-friendliness for developers and reviewers of recipes who do not need to install ESMValTool on their local machine and obtain the input data beforehand. The output of these recipe runs are automatically uploaded to a web portal and can be browsed via an html page generated with each run.

2.4.4. Automatic regression testing

During the release process of the packages, all recipes are tested against a release candidate of ESMValCore. This is necessary to detect whether changes in core functionalities of ESMValTool could break existing recipes or modify their output. If a bug is discovered and needs to be fixed before the release, a new release candidate of ESMValCore is made available and the set of recipes is tested again. In order to run all recipes at once on HPC machines and monitor the jobs, a cylc suite has been added to ESMValTool. An overview webpage has been developed to display the output of all recipes produced with these tests (see e.g. test runs for the release of ESMValTool v2.8[11]). This webpage is used by the release manager to ask maintainers to check whether their recipes ran fine and to detect possible problems.

Furthermore, a utility script[12] has been added to compare one or more run(s) against previous runs, for example produced during the previous release. This script compares netCDF and PNG files between two runs and outputs a warning when a given run requires human inspection of the results.

In the next phases of the development of ESMValTool, more efforts are planned towards automatizing the detection of changes in the output due to modifications in core functionalities and performing such checks more often than only for releases. The development of such test workflows would ease the maintenance of the packages by detecting issues early on in the development cycle and streamline the release procedure of ESMValTool.

2.5. Containerization and packages

The installation procedure of ESMValTool has been simplified to improve the user-friendliness for new users. The packages have been deployed on the conda-forge channel and made available as containers to facilitate their installation and usage on various compute platforms. ESMValTool supports installation on Linux and MacOS. It can be used on systems ranging from personal laptops to major HPC systems. Since many of the users work on HPC systems that offer direct access to large data pools, ESMValTool is pre-configured to find data on such systems, including: JASMIN

[11] https://esmvaltool.dkrz.de/shared/esmvaltool/v2.8.0/debug.html
[12] https://docs.esmvaltool.org/en/latest/utils.html#comparing-recipe-runs

at CEDA, Levante at DKRZ, Nord3v2 at BSC, Ciclad at IPSL, and the ETHZ and MetOffice servers.

2.5.1. Conda packages

ESMValCore and ESMValTool packages use a fairly large number (20-40) of direct dependencies: Python/NCL/R/Julia software packages that are imported and used in various runtime tasks. The names of these dependencies are stored in YAML environment files[13] (oftentimes together with various constraints on specific dependency versions). These are used by the employed dependency solver to build a virtual environment, where ESMValCore and ESMValTool are finally installed. We source about 99% of our dependencies from Anaconda's conda-forge channel and the rest from PyPi. We use mamba as a dependency solver, since it is fast. Mamba has indeed a C++ integrated solver which makes it much faster and as accurate as conda's Python solver. Because conda-forge is probably the most popular and diverse software package ecosystem for Python, our packages are also built and deployed on conda-forge (package building involves a process similar to installing it, including the creation of a virtual environment holding all the package's dependencies, installing, and testing it). Conda-forge package building, deployment, and maintenance for both ESMValCore and ESMValTool is done by a few members of the Technical Lead Development Team, and it is usually triggered by a new release. Our packages are also deployed to the Python Packages Index (PyPI) to diversify deployment options, and to allow users that do not use conda to install ESMValCore and ESMValTool.
For all of these platforms, installation instructions of ESMValCore[14] and ESMValTool[15] have been added to the documentation.

2.5.2. Containers

Both ESMValCore and ESMValTool are available to be installed as containers either using Docker or Singularity. The choice to support both platforms was motivated by the fact that not all HPC infrastructures may allow the use of Docker due to permission rights. The images are hosted on DockerHub and their installation procedure has been reported in the documentation[16]. The installation and building of the images is periodically and automatically tested as part of the CircleCI workflow (see Section 2.4.1).

[13] https://github.com/ESMValGroup/ESMValTool/blob/main/environment.yml
[14] https://docs.esmvaltool.org/projects/esmvalcore/en/latest/quickstart/install.html
[15] https://docs.esmvaltool.org/en/latest/quickstart/installation.html
[16] https://docs.esmvaltool.org/en/latest/quickstart/installation.html#docker-installation

2.6. Improved performance

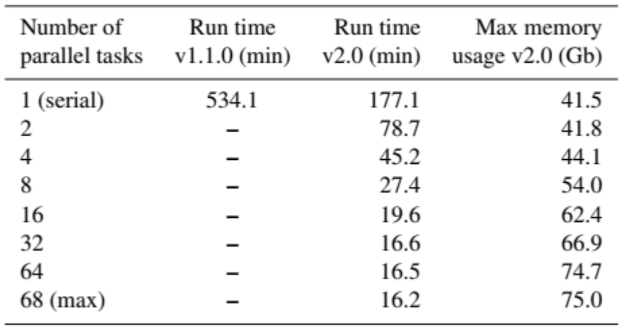
2.6.1. Task parallelism

The development of ESMValTool version 2 was motivated by the increased data volume available in CMIP6 and the improved spatial and temporal resolutions of the models. The new design of ESMValTool includes a fully revised preprocessor part that has significantly improved the performance of the evaluation tool. To assess the gain in performance between ESMValTool versions 1 and 2, a benchmark test was conducted in [1] to compare runtimes for the performance metrics recipe called `recipe_perfmetrics_CMIP5.yml`[17] (v2.0) and its analogous namelist (v1.1.0) on the DKRZ supercomputer Mistral (see Fig. 3). The comparison restricted to a single task shows a runtime reduced by a factor 3 in the serial mode of version 2.0, as a result of code refactoring, switching from NCL (v1) to Python/Iris (v2), and improved preprocessing capabilities. Note that the execution of parallel tasks was not supported in ESMValTool v1.

The performance has been further improved thanks to the task-based parallelization capability introduced in v2.0. The workflow manager of ESMValTool determines the number of independent tasks, preprocessor and diagnostic tasks, to be executed for a recipe run as well as their dependency to ancestor tasks. The workflow manager allows for the parallel execution of independent tasks. This is achieved with the Python package `multiprocessing` supporting the execution of parallel tasks within a compute node. The default number of tasks is set to the number of CPUs available. This number can be changed in the main configuration file where the user can set the maximum number of tasks that can be run in parallel. To assess the additional performance gain, the benchmark test was extended using an increasing number of parallel tasks in ESMValTool v2.0. This shows that the runtime can be reduced by a factor of about 30 compared to version 1. For this specific test case, the benchmark test also shows that performance gain is marginal above 32 tasks, showing that the runtime of recipes is limited by the longest running tasks.

A larger number of parallel tasks can significantly speed-up the execution of a recipe. Nevertheless, since data from simultaneously running tasks must be stored in memory at the same time, a too large number of tasks may lead to memory errors. In practice, the ideal number of parallel tasks for an optimal runtime depends on the total number of tasks of a recipe, their duration and the memory imprint for each of these. Recipes with high memory demands may require to lower the number of maximum tasks used, increasing the runtime. For public ESMValTool recipes, an estimate of the runtime and memory requirements can be viewed on the recipe portal (see Section 2.4.4) that is updated after each release[18].

[17] https://github.com/ESMValGroup/ESMValTool/blob/main/esmvaltool/recipes/recipe_perfmetrics_CMIP5.yml
[18] https://esmvaltool.dkrz.de/shared/esmvaltool/v2.8.0/debug.html



Number of parallel tasks	Run time v1.1.0 (min)	Run time v2.0 (min)	Max memory usage v2.0 (Gb)
1 (serial)	534.1	177.1	41.5
2	–	78.7	41.8
4	–	45.2	44.1
8	–	27.4	54.0
16	–	19.6	62.4
32	–	16.6	66.9
64	–	16.5	74.7
68 (max)	–	16.2	75.0

Figure 3: Times required for running recipe_perfmetrics_CMIP5.yml with ESMValTool v1.1.0 and v2.0 using different numbers of maximum parallel tasks (from [1]). The corresponding maximum memory usage as diagnosed in v2.0 is also shown. Each number in this table corresponds to the median of 10 ESMValTool runs, to account for the variability in the performance across different nodes. The nodes used for this analysis feature 24 physical cores.

2.6.2. Support for lazy data evaluation

Additionally, the performance improvements of ESMValTool largely benefitted from the use of the Iris library and its data abstraction feature. Iris cubes allow users to perform a wide variety of cube operations in a computationally efficient way, such as subsetting, extraction, merge, concatenation, regridding, and interpolation. Iris takes advantage of the Dask library that provides lazy evaluation. The data, provided as Dask Arrays, are loaded into memory only when absolutely necessary, enabling out-of-core processing and allowing Iris to scale efficiently from single machines to multi-core systems.

The adoption of Iris and Dask greatly improved the memory usage of preprocessor functions of ESMValTool. Further improvements of the memory imprint management are expected with the adoption of Dask distributed schedulers. These schedulers execute the task graphs created with Dask Arrays. These need to be optimized for the specific problems that they are implemented for in order to take full advantage of distributed schedulers. Such enhancements would open up the possibility to run ESMValTool on multiple nodes, facilitating the evaluation of high-resolution data. This would also be beneficial for users working on small to moderate sized clusters by allowing them to run memory-intensive recipes, such as IPCC AR6 recipes (see Section 2.9), which are currently only runnable on large HPC systems.

2.7. Iris developments to improve the performance of the ESMValTool backend

Since 1 Jan 2019 there have been **14 releases** of SciTools/iris, including the **v3.x** major release milestone. Several themes of work have been addressed over the duration of the project to improve specific performance issues, and to provide a more robust, stable, and feature rich capability to the ESMValTool backend developers and community. Our coverage of the CF Metadata Conventions has been extended, which also includes adoption of the UGRID Conventions for unstructured support. Likewise the SciTools/iris data model has been matured and extended, with efforts focused on offering a wider range of lazy, out-of-core processing capable operations; from loading through to analysis and saving. In particular, regridding was a major bottleneck for ESMValTool, however lazy regridding within SciTools/iris is now available along with the new SciTools-incubator/iris-esmf-regrid package, which offers lazy regridding using the Earth System Modeling Framework (ESMF) and Dask for both rectilinear and curvilinear grids, and unstructured meshes. Such improvements will be particularly beneficial for the evaluation of CORDEX data for which support to ESMValCore was recently added (see D9.4).

A summary of previous notable outcomes include:

1. Extending the Iris data model to support CF Metadata Ancillary Data and Status/Quality Flags
2. Support for lazy Linear, Nearest and AreaWeighted regridding schemes
3. Introduction of the Iris Common Metadata API, with lenient/strict behaviors
4. Overhaul of cube arithmetic with extended capability
5. Addressing Dask bottlenecks with NetCDF loading
6. Documentation refresh and rebanding to encourage better community engagement and adoption
7. Support for UGRID and unstructured meshes
8. Adoption of Airspeed Velocity for continuous, automated benchmarking, allowing easy identification of performance gains and regressions across the SciTools/iris code base
9. Welcoming several ESMValTool developers as valued SciTools/iris core developers

More recent outcomes include:

1. SciTools/iris v3.4.0 release (1 Dec 2022) including the following notable changes:
    a. Significantly improved Pandas interoperability for n-dimensional cubes
    b. Improved metadata handling for coordinates of unstructured meshes
    c. Support for regridding derived coordinates for the PointInCell regridding scheme
    d. Performance optimisations:
        i. SUM, COUNT and PROPORTION aggregation operations on real data,
        ii. cube intersect, subset and extract operations,

        iii. PointInCell regridding scheme

2. SciTools-incubator/iris-esmf-regrid v0.5.0 release (14 Oct 2022) included the following notable changes:
    a. Added support for bilinear regridding for unstructured meshes
    b. Added curvilinear support for unstructured regridders
    c. Support pre-computed weights as either arrays or matrices
3. Release of SciTools-incubator/python-stratify v0.3.0 is *pending*, which includes support for lazy vertical regridding

Note that, SciTools/iris v3.5.0 release is due to be released on 27 Apr 2023.

Going forward, we are developing a more performant, zero-copy interoperability bridge between xarray and Iris. This will allow direct access to xarray features more easily from Iris cubes, and vice versa. We are aiming to target SciTools/iris v3.6.0 release (Aug 2023) for this new capability.

2.8. Implementation of standard interfaces and provenance

A set of recommendations and standards were described in D5.2 in order to facilitate the coupling of externally developed diagnostic scripts and packages into ESMValTool. The guidelines take into account the structure of an ESMValTool recipe, which is to be used as the driver in order to load the data, check it against the CMOR standards and, if needed, perform preprocessing functions to prepare the data for the external diagnostics. After the data has been loaded, an ESMValTool diagnostic script can be used as a wrapper in order to call the routines from the external diagnostics, save the output following the ESMValTool output structure, and record the provenance of the execution.

Therefore, as long as external diagnostic packages are compatible to be installed as ESMValTool dependencies, it should be possible to run them within the ESMValTool's framework by setting up a recipe to define the needed data and a diagnostic script to act as a wrapper. Examples of externally developed diagnostics that have been integrated into ESMValTool are described in Section 2.10.

However, this standard interface sets some limitations in the sense that the coupling between externally developed diagnostics and ESMValTool requires the development of the wrappers. As having to develop further code in order to couple diagnostics may discourage some contributors, the standard interface could be elaborated further in order to make the coupling more straightforward, without the need to write custom wrappers for each diagnostic package.

The recording of provenance uses the W3C PROV format and is done for each diagnostic called in a recipe. This includes scientific provenance information as well as names and global netCDF attributes of all input files. The provenance can be checked in .xml provenance files generated with the output files. Possible extensions of the provenance mechanism were discussed in D3.3, including the ability to rely on externally developed templates instead of handling the provenance information in the ESMValTool code [7]. Nevertheless, the dependency to such external templates may lead to confidentiality issues when the provenance of evaluation runs is stored on remote systems.

2.9. Towards the integration of IPCC AR6 recipes

2.9.1. Development of IPCC AR6 recipes

During this project, ESMValTool has been used extensively by 38 scientists to support the analyses used for about 50 figures of the IPCC WGI AR6. This work greatly benefitted from the technical improvements made to ESMValTool, facilitating the analysis of the large amount of data in CMIP6. After the publication of the report, contributing scientists were asked to archive their code into a repository of the ESMValGroup organization in order to prepare the integration of the code into the public ESMValTool. This preparation included the documentation of the recipes and diagnostics used, recording of the compute environments, custom changes to ESMValTool or ESMValCore. In April 2022, this repository was turned into a public archive which has been used by the IPCC Technical Support Unit as a basis for the code citation of the figures produced with ESMValTool.

While continuous developments of ESMValTool are done publicly on GitHub, the guidelines for the creation of AR6 figures differ and require to keep the code private until the publication of the report. The timeline to develop recipes is often longer than the release cycles of ESMValTool. Scientists may not always wish to update their code and installation of ESMValTool after each new release due to the extra work this could imply. Consequently, code developed for the production of AR6 figures often cannot be readily merged into the public ESMValTool as is. It requires that the contributors and the ESMValTool development community work together on updating and adapting the code to account for the changes that were made to ESMValTool so that AR6 recipes could work with the latest ESMValTool. During this project, efforts have been devoted to integrate recipes and diagnostics used in the AR6 into the public ESMValTool. This work mostly focused on the Chapter 3: Human Influence on the Climate System [8] which contains most of the figures produced with ESMValTool. The AR6 recipes[19] added to ESMValTool can be used to reproduce up to 9 figures of this chapter and are documented online[20]. To showcase the integration of such

[19] https://github.com/ESMValGroup/ESMValTool/tree/main/esmvaltool/recipes/ipccwg1ar6ch3

[20] https://docs.esmvaltool.org/en/latest/recipes/recipe_ipccwg1ar6ch3.html

recipes, we reproduce the IPCC AR6 Figure 3.9 which is generated using the recipe `recipe_ipccwg1ar6ch3_fig_3_9.yml` (see Fig. 4).
The integration of AR6 recipes also included the publication of several reformatting scripts (known as CMORizer scripts) for observational data used in these analyses. The ESMValTool development community aims at pursuing the support for the integration of AR6 recipes into ESMValTool upon request and availability of the contributing authors.

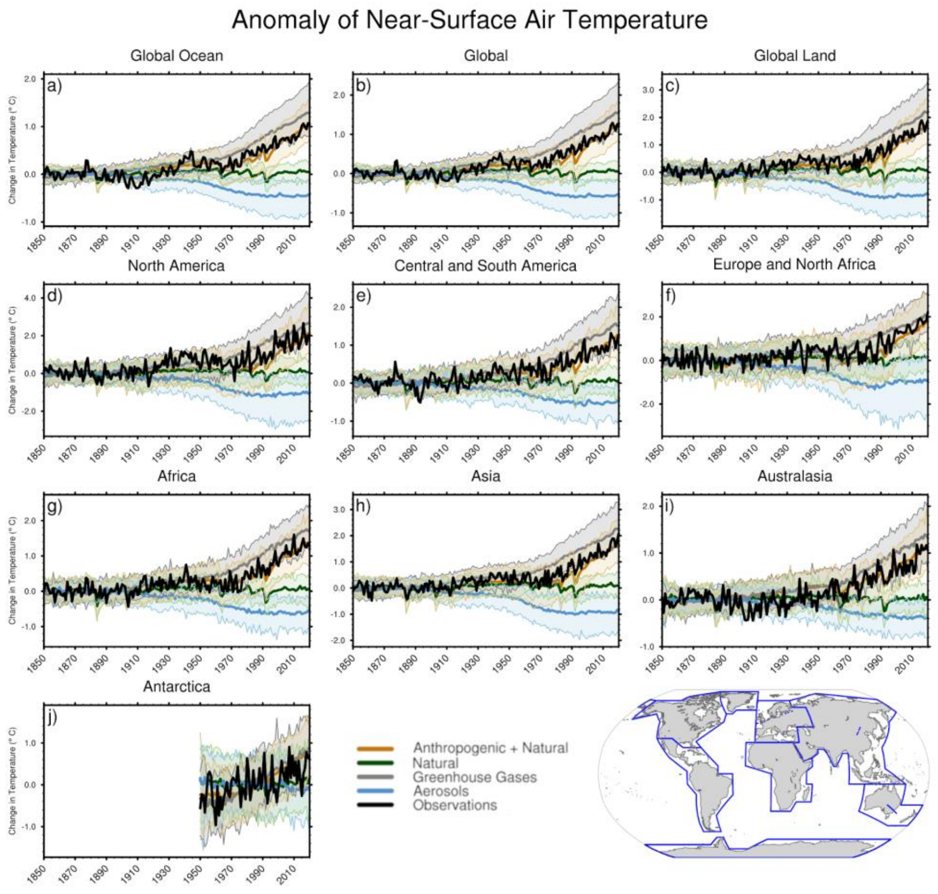


Figure 4: Figure 3.9 in [8] - Global, land, ocean and continental annual mean near-surface air temperatures anomalies in CMIP6 models and observations. Further details can be found in the AR6 online documentation.

2.9.2. Dataset versioning of input data used in IPCC AR6 recipes

Recipes developed for the AR6 can use more than a thousand input datasets, covering the last three phases of CMIP. Handling such large amounts of datasets in recipes may be challenging for users and developers, particularly because data availability can be affected by various issues. For example, versions of datasets may vary between different ESGF nodes attached to compute clusters where ESMValTool is run, making it difficult to compare the evaluation output produced on different clusters. Also, datasets may be retracted or updated on ESGF and this could lead to version mismatches with local ESGF data pools. Until version v2.8, ESMValTool was configured to use by default the latest local version of a CMIP dataset and if no local copy was available, the latest version available on the ESGF servers. Moreover, the version numbers of input datasets could not be recorded in recipes. These are only available in the log files and provenance records generated for each run [7] but such files are usually not available to other users. These issues motivated further ESMValTool developments on improving the support for dataset versioning in order to ease the integration of AR6 recipes, facilitating future work based on these recipes and enhancing the user-friendliness of ESMValTool. The recent changes include:

- ability to easily populate recipes with all datasets available both locally and on ESGF;
- for every recipe run, a copy of the recipe is saved with all dataset versions fixed for easy reproducibility;
- ability to correctly specify ancillary variables (such as cell area and land area fraction) in the recipe and automatically find them. These variables are required for accurate computations, especially for models on irregular grids.

2.10. Coupling of externally developed diagnostics and metrics to ESMValTool

ESMValTool has initially been designed to process and analyze output from CMIP models (e.g. [9]), but has been extended over the project to other applications, for example: to compare different observational and reanalysis datasets (e.g. [10]), to provide preprocessing of forcing data for hydrological models [11] and work with observational uncertainties (e.g. [12]), and also to include machine learning analysis methods (e.g. [13]). Technical improvements made to ESMValCore have facilitated the integration of new diagnostics, covering a wider range of applications. This is reflected by the steady growth of the number of recipes included in each release of ESMValTool, from **94 in v2.0** until **150 in v2.8**.

While many of the diagnostics used in recipes are scripts developed by individual scientists, others rely on specialized evaluation packages for specific applications. Their integration in ESMValTool opens up the adoption of these specialized evaluation packages by the whole scientific community. Increasing the adoption of the package by other research groups represents a major challenge for the ESMValTool since it can result in additional work for the core development team to maintain these externally developed analyses. Several external packages have been coupled to ESMValTool during this project, such as: the Climate Variability Diagnostics Package[21] (CVDP) developed by NCAR's Climate Analysis Section as an analysis tool that documents the major modes of climate variability in models and observations, the AutoAssess metrics package developed by the

[21] https://docs.esmvaltool.org/en/latest/recipes/recipe_cvdp.html

MetOffice which is applied for different realms like the stratosphere[22] or the land-surface permafrost[23], and the extreme events indices library[24] from the joint CCl/CLIVAR/JCOMM Expert Team on Climate Change Detection and Indices (ETCCDI). These three packages, programmed respectively in NCL, Python, and R, highlight the diversity of open languages supported at the diagnostic level. The usage of such packages in ESMValTool recipes is showcased on the recipe portal developed in WP7[25]. In order to couple external packages to ESMValTool, developers need to ensure that coding standards and guidelines are met and that provenance records and scientific documentation are available (see Section 2.8). Furthermore, it is also necessary to check whether the licenses of the added dependencies are compatible with the Apache 2.0 license of ESMValTool in order to guarantee the sustainability of the software.

3. Conclusions and Recommendations

Significant effort has been devoted to technical improvements of ESMValTool in IS-ENES3. This work led to the second major release of ESMValTool, version v2.0, and marked a significant step forward to ensure the long-term viability of the package. Continued development of ESMValTool resulted in regular releases of the software, extending the capabilities of the evaluation tool, improving the performance, the maintainability, and the user-friendliness. All of these developments described in this document have been included in the final IS-ENES3 version, ESMValTool v2.8.

The restructured and completely rewritten code significantly improved the performance of ESMValTool by enabling running preprocessing and diagnostic tasks in parallel, reducing the run time (real time) by up to a factor 30. The performance improvements largely benefitted from the adoption of the Iris library in the preprocessor modules of ESMValTool and by implementing lazy capabilities for improved memory efficiency. Joint developments of ESMValCore and Iris led to regular improvements through IS-ENES3, such as an extension of regridding options, performance optimisations, improved APIs, and better support for unstructured data. These developments enabled, among others, the evaluation of CORDEX with ESMValTool. Moreover, the new design of ESMValTool facilitates the coupling of external packages and the integration of contributions to the code from scientists. This is highlighted by a steady growth of the number of recipes in released versions, targeting a wide range of applications.

[22] https://docs.esmvaltool.org/en/latest/recipes/recipe_autoassess_stratosphere.html
[23] https://docs.esmvaltool.org/en/latest/recipes/recipe_autoassess_landsurface_permafrost.html
[24] https://docs.esmvaltool.org/en/latest/recipes/recipe_extreme_events.html
[25] https://esmvaltool.dkrz.de/shared/esmvaltool/v2.8.0/

A major motivation for this work was the challenges posed by the large data volumes generated by model intercomparison projects like CMIP6. The technical developments made during IS-ENES3 enabled scientists to employ ESMValTool extensively in the IPCC WGI AR6 report. The first AR6 recipes have been merged into the official ESMValTool release and can now be used to reproduce 9 AR6 figures. The ESMValTool development community will strive to maintain these recipes and integrate additional ones upon request and availability of contributing scientists. Additionally, ESMValTool v2.x has been used in a number of scientific publications on evaluation of ESMs, benefitting from its improved performance and capabilities [9,12,13,14,15].

All of these developments relied on the adoption of modern software engineering practices and coding standards that improved the quality and maintainability of the packages. In order to make it easier for scientists to contribute while ensuring very high standards for the performance-oriented preprocessor modules, different levels of strictness have been implemented for the ESMValTool and ESMValCore packages. Automated testing services and various strategies for testing recipes have been implemented to ensure that the code performs correctly and that recipe output is not affected by changes to the code. Moreover, the adoption of community standards, such as provenance tracking and standard interfaces to couple external packages, helped to raise the scientific quality of the evaluation analyses and made easier the use of specialized analysis packages (AutoAssess, CVDP, ETCCDI). The work within IS-ENES3 also significantly contributed to make installation of the packages easier and facilitate application of ESMValTool on multiple platforms from HPC systems to local compute resources.

Building on the achievements in IS-ENES3, future developments of ESMValTool will include the widening of the range of scientific applications covered by the diagnostics, the implementation of recipe testing workflows to automate the detection of changes in recipe output in a continuous manner, and the application of ESMValTool to high-resolution data. The latter requires making the whole set of preprocessor functions lazy in order to optimize the memory requirements and to further improve the performance. Resources for starting this work after the end of IS-ENES3 have already been allocated within the German national modeling strategy project (natESM).

As an extension of the work carried out in the framework of IS-ENES3, first steps have been made towards the inclusion of ESMValTool in the ENES-Research Infrastructure (ENES-RI) as a service tool to evaluate and intercompare climate model output. This will help to further align the ESMValTool strategy with the wider European research infrastructure. Fruitful collaborations with HPC centers established within IS-ENES3, in particular with DKRZ, will continue beyond the end of this project to provide, for instance, a testing platform for future ESMValTool releases. Several European and national projects are planning on using ESMValTool as main evaluation tool. If funded, these projects will provide resources for many future developments such as enhancing the diagnostic part and related technical aspects. IS-ENES3 as a project focusing on general infrastructural and technical developments has contributed significantly to enhance the capability of the software and such projects will remain very valuable to maintain and further improve ESMValTool.

Acknowledgments

The authors of this deliverable gratefully acknowledge contributions to ESMValTool, ESMValCore, and Iris over the past 4 years which led to this final IS-ENES3 ESMValTool version (v2.8). These contributions were made by the co-authors of technical overview publication on ESMValTool v2.0 [1], the members of the ESMValTool Core Development Team, and the Iris Core Developers: Will Benfold, Lisa Bock, Anna Booton, Björn Brötz, Ruth Comer, Lee de Mora, Faruk Diblen, Laura Dreyer, Niels Drost, Paul Earnshaw, Philip Elson, Veronika Eyring, Bettina Gier, Birgit Hassler, Tremain Knight, Nikolay Koldunov, Axel Lauer, Ruth Lorenz, Benjamin Mueller, Patrick Peglar, James Penn, Mattia Righi, Elias Sadek, Manuel Schlund, Jon Seddon, Alistair Sellar, Kim Serradell, Breixo Soliño Fernández, Javier Vegas-Regidor, Stephen Worsley, Martin Yeo, and Klaus Zimmermann. Additional contributions from the ESMValTool and Iris developer communities are also acknowledged.

References

[1] Righi, M., Andela, B., Eyring, V., Lauer, A., Predoi, V., Schlund, M., Vegas-Regidor, J., Bock, L., Brötz, B., de Mora, L., Diblen, F., Dreyer, L., Drost, N., Earnshaw, P., Hassler, B., Koldunov, N., Little, B., Loosveldt Tomas, S., and Zimmermann, K.: Earth System Model Evaluation Tool (ESMValTool) v2.0 - technical overview, Geosci. Model Dev., 13, 1179-1199, doi:10.5194/gmd-13-1179-2020, 2020.

[2] Eyring, V., Bony, S., Meehl, G. A., Senior, C. A., Stevens, B., Stouffer, R. J., and Taylor, K. E.: Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization, Geosci. Model Dev., 9, 1937-1958, doi:10.5194/gmd-9-1937-2016, 2016.

[3] Eyring, V., Bock, L., Lauer, A., Righi, M., Schlund, M., Andela, B., Arnone, E., Bellprat, O., Brötz, B., Caron, L.-P., Carvalhais, N., Cionni, I., Cortesi, N., Crezee, B., Davin, E., Davini, P., Debeire, K., de Mora, L., Deser, C., Docquier, D., Earnshaw, P., Ehbrecht, C., Gier, B. K., Gonzalez-Reviriego, N., Goodman, P., Hagemann, S., Hardiman, S., Hassler, B., Hunter, A., Kadow, C., Kindermann, S., Koirala, S., Koldunov, N., Lejeune, Q., Lembo, V., Lovato, T., Lucarini, V., Massonnet, F., Müller, B., Pandde, A., Pérez-Zanón, N., Phillips, A., Predoi, V., Russell, J., Sellar, A., Serva, F., Stacke, T., Swaminathan, R., Torralba, V., Vegas-Regidor, J., von

Hardenberg, J., Weigel, K., and Zimmermann, K.: Earth System Model Evaluation Tool (ESMValTool) v2.0 - an extended set of large-scale diagnostics for quasi-operational and comprehensive evaluation of Earth system models in CMIP, Geosci. Model Dev., 13, 3383-3438, doi:10.5194/gmd-13-3383-2020, 2020.

[4] Lauer, A., Eyring, V., Bellprat, O., Bock, L., Gier, B. K., Hunter, A., Lorenz, R., Pérez-Zanón, N., Righi, M., Schlund, M., Senftleben, D., Weigel, K., and Zechlau, S.: Earth System Model Evaluation Tool (ESMValTool) v2.0 - diagnostics for emergent constraints and future projections from Earth system models in CMIP, Geosci. Model. Dev., 13, 4205-4228, doi:10.5194/gmd-13-4205-2020, 2020.

[5] Weigel, K., Bock, L., Gier, B. K., Lauer, A., Righi, M., Schlund, M., Adeniyi, K., Andela, B., Arnone, E., Berg, P., Caron, L.-P., Cionni, I., Corti, S., Drost, N., Hunter, A., Lledó, L., Mohr, W. C., Paçal, A., Pérez-Zanón, N., Predoi, V., Sandstad, M., Sillmann, J., Sterl, A., Vegas-Regidor, J., von Hardenberg, J., and Eyring, V.: Earth System Model Evaluation Tool (ESMValTool) v2.0 - diagnostics for extreme events, regional and impact evaluation, and analysis of Earth system models in CMIP, Geosci. Model Dev., 14, 3159-3184, doi:10.5194/gmd-14-3159-2021, 2021.

[6] Schlund, M., Hassler, B., Lauer, A., Andela, B., Jöckel, P., Kazeroni, R., Loosveldt Tomas, S., Medeiros, B., Predoi, V., Sénési, S., Servonnat, J., Stacke, T., Vegas-Regidor, J., Zimmermann, K., and Eyring, V.: Evaluation of Native Earth System Model Output with ESMValTool v2.6.0, Geosci. Model Dev., 16, 315-333, doi:10.5194/gmd-16-315-2023, 2023.

[7] Andela, B., Bedia, J., Cofiño, A., Kazeroni, R., Pagé, C., San Martín, D., Sénési, S., Servonnat, J., Spinuso, A., Veldhuizen, M., and Zimmermann, K.: IS-ENES3 White Paper on provenance handling in the model evaluation process, doi:10.5281/zenodo.5759571, 2021.

[8] Eyring, V., N.P. Gillett, K.M. Achuta Rao, R. Barimalala, M. Barreiro Parrillo, N. Bellouin, C. Cassou, P.J. Durack, Y. Kosaka, S. McGregor, S. Min, O. Morgenstern, and Y. Sun, 2021: Human Influence on the Climate System. In Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change [Masson-Delmotte, V., P. Zhai, A. Pirani, S.L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M.I. Gomis , M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T.K. Maycock, T. Waterfield, O. Yelekçi, R. Yu, and B. Zhou (eds.)]. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, pp. 423-552, doi:10.1017/9781009157896.005, 2021.

[9] Bock, L., Lauer, A., Schlund, M., Barreiro, M., Bellouin, N., Jones, C., Meehl, G. A., Predoi, V., Roberts, M. J., and Eyring, V.: Quantifying progress across different CMIP phases with the ESMValTool, J. Geophys. Res., 125, e2019JD032321, doi:10.1029/2019JD032321, 2020.

[10] Hassler, B., and Lauer, A.: Comparison of Reanalysis and Observational Precipitation Datasets Including ERA5 and WFDE5, Atmosphere, 12(11), 1462, doi:10.3390/atmos12111462, 2021.

[11] Hut, R., Drost, N., van de Giesen, N., van Werkhoven, B., Abdollahi, B., Aerts, J., Albers, T., Alidoost, F., Andela, B., Camphuijsen, J., Dzigan, Y., van Haren, R., Hutton, E., Kalverla, P., van Meersbergen, M., van den Oord, G., Pelupessy, I., Smeets, S., Verhoeven, S., de Vos, M., and Weel, B.: The eWaterCycle platform for open and FAIR hydrological collaboration, Geosci. Model Dev., 15, 5371–5390, doi:10.5194/gmd-15-5371-2022, 2022.

[12] Lauer, A., Bock, L., Hassler, B., Schröder, M., Stengel, M.: Cloud climatologies from global climate models - a comparison of CMIP5 and CMIP6 models with satellite data, Journal of Climate, 36(2), 281-311, doi:10.1175/JCLI-D-22-0181.1, 2023.

[13] Schlund, M., Lauer, A., Gentine, P., Sherwood, S. C., and Eyring, V.: Emergent constraints on Equilibrium Climate Sensitivity in CMIP5: do they hold for CMIP6?, Earth Syst. Dynam., 11, 1233-1258, doi:10.5194/esd-11-1233-2020, 2020.

[14] Gier, B. K., Buchwitz, M., Reuter, M., Cox, P. M., Friedlingstein, P., and Eyring, V.: Spatially resolved evaluation of Earth system models with satellite column-averaged CO2, Biogeosciences, 17, 61156144, doi: 10.5194/bg-17-6115-2020, 2020.

[15] Tebaldi, C., Debeire, K., Eyring, V., Fischer, E., Fyfe, J., Friedlingstein, P., Knutti, R., Lowe, J., O'Neill, B., Sanderson, B., van Vuuren, D., Riahi, K., Meinshausen, M., Nicholls, Z., Tokarska, K.B., Hurtt, G., Kriegler, E., Lamarque, J., Meehl, G., Moss, R., Bauer, S.E., Boucher, O., Brovkin, V., Byun, Y., Dix, M., Gualdi, S., Guo, H., John, J.G., Kharin, S., Kim, Y., Koshiro, T., Ma, L., Olivié, D., Panickal, S., Qiao, F., Rong, X., Rosenbloom, N., Schupfner, M., Séférian, R., Sellar, A., Semmler, T., Shi, X., Song, Z., Steger, C., Stouffer, R., Swart, N., Tachiiri, K., Tang, Q., Tatebe, Q., Voldoire, A., Volodin, E., Wyser, K., Xin, X., Yang, S., Yu, Y., and Ziehn, T.: Climate model projections from the Scenario Model Intercomparison Project (ScenarioMIP) of CMIP6, Earth Syst. Dynam., 12, 253-293, doi: 10.5194/esd-12-253-2021, 2021.

Abbreviations

API: Application Programming Interface
CMIP: Climate Model Intercomparison Project
CMOR: Climate Model Output Rewriter
CPU: Central Processing Unit
ESGF: Earth System Grid Federation
ESMValTool: Earth System Model eValuation Tool
HPC: High Performance Computing
IPCC AR6: The Sixth Assessment Report of the Intergovernmental Panel on Climate Change
netCDF: Network Common Data Form