ESMValTool

Earth System Model Evaluation Tool

Reliable and reproducible Earth System Model data analysis with ESMValTool

Valeriu Predoi (NCAS-CMS University of Reading, UK)
and Bouwe Andela (Netherlands eScience Centre, The Netherlands)
for the ESMValTool Technical Lead Team

Software ecosystem: ESMValTool and ESMValCore

ESMValTool: scientific analysis and diagnostics library (written in Python, NCL, R, and Julia) – contains reproducible recipes **with scientific output** (plots, data files etc) → **SCIENCE** is the main output, **LARGE and DIVERSE (coding skills, technical knowledge) COLLABORATIVE** group the developers

ESMValCore: Python package for working with CMIP(-like) data, responsible for running ESMValTool recipes. It finds and optionally downloads the input data, applies preprocessor functions (climate statistics, regridding, multi-model statistics etc) and passes the resulting NetCDF files on to the scientific analysis codes→ **COMPUTING** and **DATA REDUCTION** are the outputs, **SMALLER TECHNICAL TEAM (strong technical skills)** the developers

Software ecosystem: ESMValCore and ESMValTool

ESMValTool

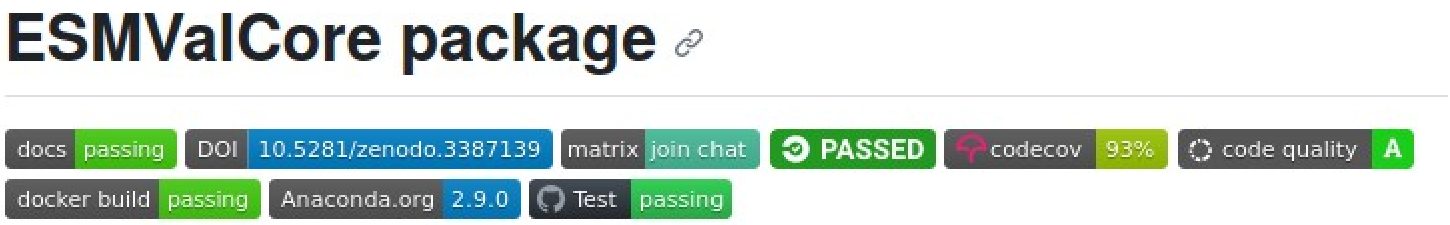
- lots of code (~200k lines)
- many dependencies (~100 direct dependencies, ~600 indirect dependencies), but should be easy to install
- provides ~100 recipes and diagnostics, which are fairly independent of each other

ESMValCore

- relatively compact codebase
- only a few dependencies
- reliability is key because it is used by every recipe

Testing is absolutely necessary to ensure correct functionality and portability, over long development cycles, with widely varied developers' skills and interests

Overall **testing strategy** - ESMValCore



ESMValCore package

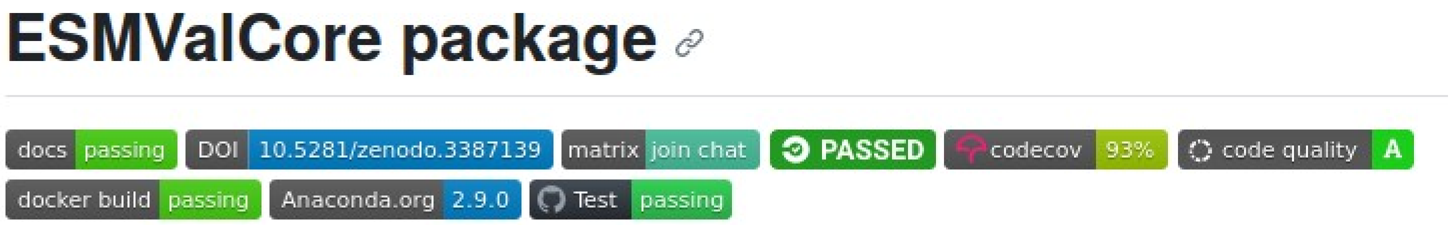
- Reliable Python package, responsible for **computationally-heavy pre-processing of climate data** (climate statistics, regridding, multi-model statistics etc) → **COMPUTING** and **DATA REDUCTION** are the outputs, **SMALL TECHNICAL TEAM** the developers

- Testing needs to be technically **diverse** and **comprehensive**

- Testing done for Linux and OSX and all recent Python versions

- Both **strict** and **in-depth** testing

Overall **testing strategy** - ESMValCore



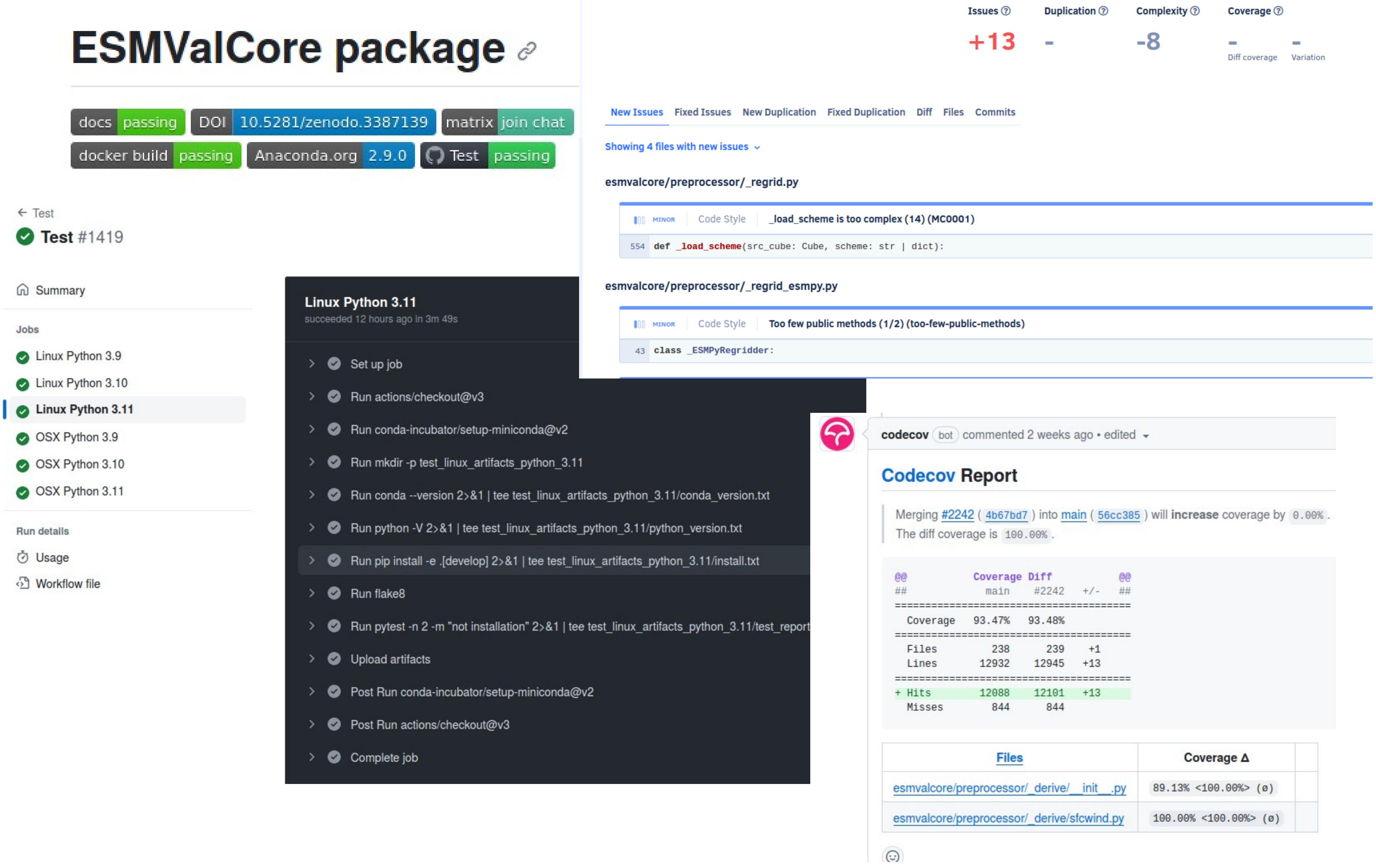
ESMValCore package

docs passing | DOI 10.5281/zenodo.3387139 | matrix join chat | PASSED | codecov 93% | code quality A

docker build passing | Anaconda.org 2.9.0 | Test passing

- Both **strict** and **in-depth** testing:
- **Core system tests:**
- - software environment fitness (building the environment, and installing the package in it, regularily)
- - backup environment recipe build and installation tests (`conda-lock`)
- - Python package build tests
- - Docker container(s) build and deploy tests
- **General purpose tests:**
- - unit/integration/regression (with sample data) tests
- - coding standards tests (mypy, `pylint`, and `flake8`)
- - code coverage check by Codecov, 100% coverage required for changes
- **Documentation:**
- - documentaton build and deploy tests

Overall **testing strategy** (for separate packages)



ESMValCore package

← Test

Test #1419

Linux Python 3.11

succeeded 12 hours ago in 3m 49s

```
Set up job
Run actions/checkout@v3
Run conda-incubator/setup-miniconda@v2
Run mkdir -p test_linux_artifacts_python_3.11
Run conda --version 2>&1 | tee test_linux_artifacts_python_3.11/conda_version.txt
Run python -V 2>&1 | tee test_linux_artifacts_python_3.11/python_version.txt
Run pip install -e .[develop] 2>&1 | tee test_linux_artifacts_python_3.11/install.txt
Run flake8
Run pytest -n 2 -m "not installation" 2>&1 | tee test_linux_artifacts_python_3.11/test_report
Upload artifacts
Post Run conda-incubator/setup-miniconda@v2
Post Run actions/checkout@v3
Complete job
```

Issues	Duplication	Complexity	Coverage	
+13	-	-8	- Diff coverage	- Variation

Showing 4 files with new issues

esmvalcore/preprocessor/_regrid.py

MINOR Code Style _load_scheme is too complex (14) (MC0001)

```
554 def _load_scheme(src_cube: Cube, scheme: str | dict):
```

esmvalcore/preprocessor/_regrid_esmpy.py

MINOR Code Style Too few public methods (1/2) (too-few-public-methods)

```
43 class _ESMPyRegridder:
```

codecov bot commented 2 weeks ago • edited

Codecov Report

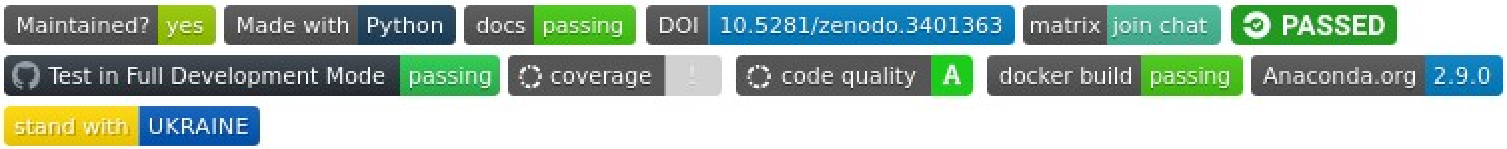
Merging #2242 (4b67bd7) into main (56cc385) will **increase** coverage by `0.00%`. The diff coverage is `100.00%`.

```
@@           Coverage Diff           @@
##             main    #2242   +/-   ##
=======================================
  Coverage   93.47%   93.48%           
=======================================
  Files         238      239    +1     
  Lines       12932    12945   +13     
=======================================
+ Hits        12088    12101   +13     
  Misses        844      844           
```

Files	Coverage Δ
esmvalcore/preprocessor/_derive/__init__.py	`89.13% <100.00%> (ø)`
esmvalcore/preprocessor/_derive/sfcwind.py	`100.00% <100.00%> (ø)`

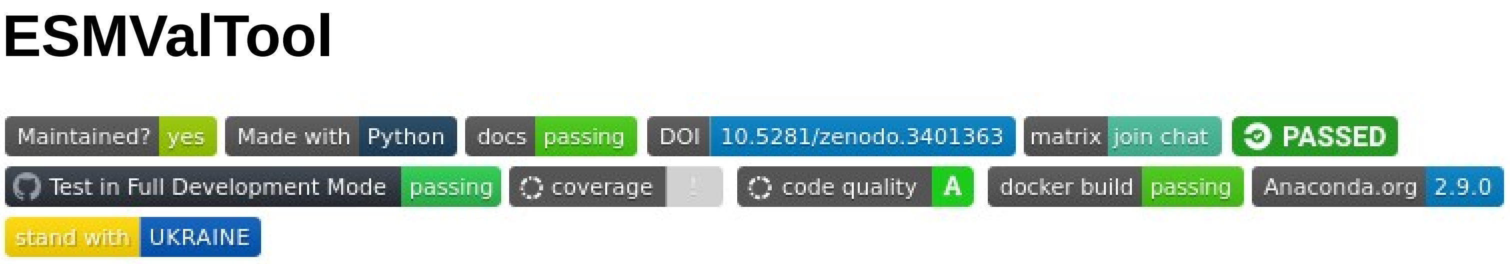
Overall **testing strategy** - ESMValTool

ESMValTool



- ESMValTool: scientific analysis and diagnostics library (written in Python, NCL, R, and Julia) – contains reproducible recipes **with scientific output** (plots, data files etc) → **SCIENCE** is the main output, **LARGE and DIVERSE (coding skills, technical knowledge) COLLABORATIVE** group the developers
- Testing needs to ensure **scientific correctness and allow for variability of developers' skills (ie not too restrictive, definitely not too lax, or "not great, not terrible")**
- Basic testing done for all supported OS and Python versions
- Scientific output-oriented tests
- Still include some technical testing (like for ESMValCore, but less strict)

Overall **testing strategy** - ESMValTool



ESMValTool

Maintained? yes | Made with Python | docs passing | DOI 10.5281/zenodo.3401363 | matrix join chat | PASSED

Test in Full Development Mode passing | coverage | code quality A | docker build passing | Anaconda.org 2.9.0

stand with UKRAINE

- Scientific output-oriented tests include:
  - Numerical and graphical output comparisons with previous, scientifically approved versions
  - Dedicated tool for recipe output comparison which is smart enough to handle small differences in numerical results in NetCDF files and small differences in plots through image hashing
  - Testing workflow is at the moment manual for every release, working on automation by setting up a “recipe test workflow”
- Mark I Eyeball testing (visualization of output) – comparison with figures in papers
- Input data specifications consistency tests
- Still with some technical testing like for ESMValCore, but less strict:
  - Limited unit/integration tests, only for a few shared components
  - More relaxed on coding standards (`pylint` and `flake8`)
  - No code coverage checks

FAIR research software

- Software releases are stored on Zenodo with a DOI
- Docker containers for reproducible software environments for every release
- A recipe with fixed input data versions is recorded for each recipe run
- ESMValCore records provenance, which includes the filenames and global NetCDF attributes of all input files used to create a figure.

For more information on FAIR research software, see:
Barker, M., Chue Hong, N.P., Katz, D.S. et al. Introducing the FAIR Principles for research software. Sci Data 9, 622 (2022).
https://doi.org/10.1038/s41597-022-01710-x

ESMValTool: take home message

- The tools have a modular design in which community members of varying skill level are able to contribute without compromising reliability and user experience for others
- Test and code quality requirements are adjusted to how many users and developers will be affected if a component breaks
- FAIR research software for doing open science