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 (eScience Centre, The Netherlands) for the ESMValTool Technical Lead Team

Software ecosystem: ESMValCore and ESMValTool

ESMValCore: highly optimized, pure-Python package, responsible for **computationally-heavy pre-processing of data** (climate statistics, regridding, multi-model statistics etc) → **COMPUTING** and **DATA REDUCTION** are the outputs, **SMALL TECHNICAL TEAM (strong technical skills)** the developers

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

Software ecosystem: ESMValCore and ESMValTool

Common to both:

- lots of code (~200k lines, minimal code rot)

- complex dependency environments (100 direct dependencies, 500-1000 dependencies in any given environment)

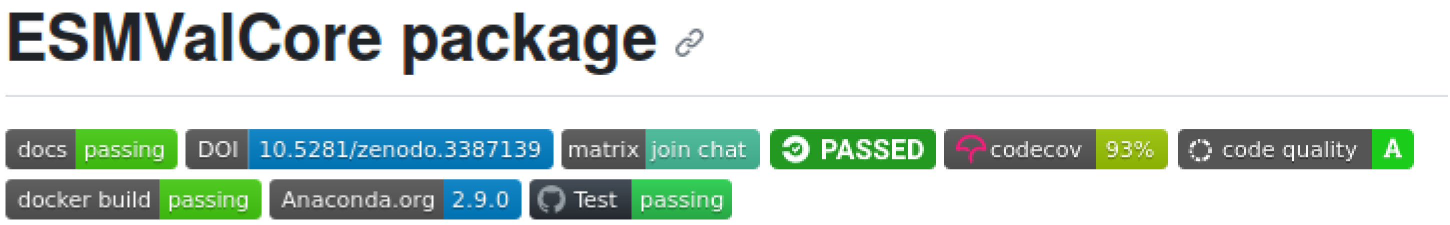
- need for wide supported platforms (operating systems, Python versions etc)

- need for ease of installation, and maintenance

- complex configurations, but made as simple as possible for users

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

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

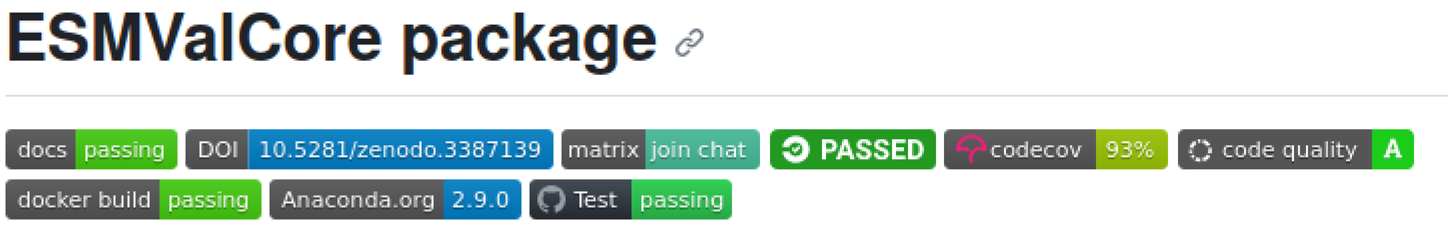
- Highly optimized, pure-Python package, responsible for **computationally-heavy pre-processing of data** (climate statistics, regridding, multi-model statistics etc) → **COMPUTING** and **DATA REDUCTION** are the outputs, **SMALL TECHNICAL TEAM** the developers ->

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

- Testing done for all supported operating systems and Python (as main language) 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

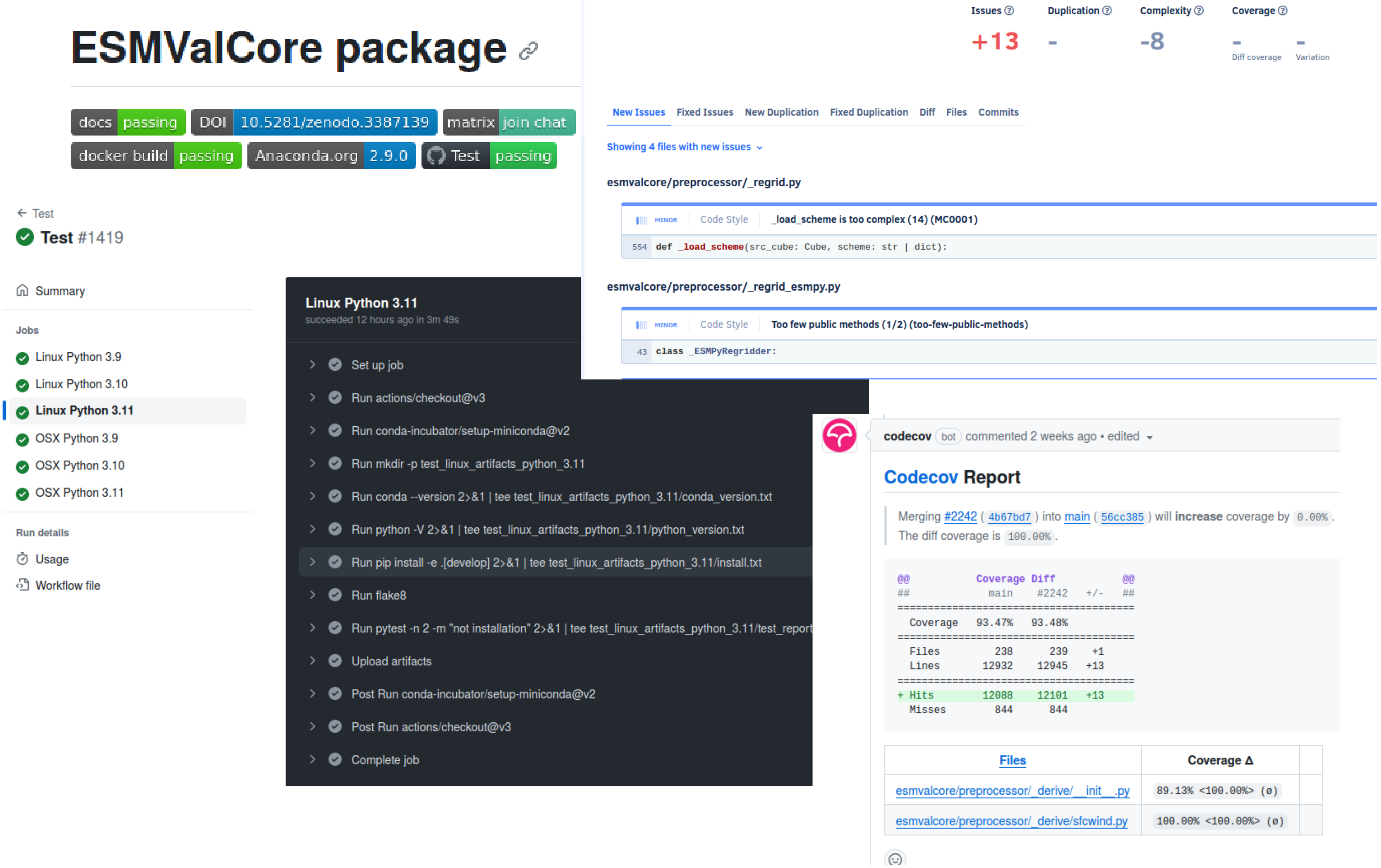
- Both **strict** and **in-depth** testing:

  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 (`pylint` and `flake8`)
  - test coverage tests (when adding new code)

  Documentation:
  - documentaton build and deploy tests

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



ESMValCore package

docs passing | DOI 10.5281/zenodo.3387139 | matrix join chat
docker build passing | Anaconda.org 2.9.0 | Test passing

← Test

✓ **Test** #1419

Summary

Jobs

- ✓ Linux Python 3.9
- ✓ Linux Python 3.10
- ✓ **Linux Python 3.11**
- ✓ OSX Python 3.9
- ✓ OSX Python 3.10
- ✓ OSX Python 3.11

Run details

- Usage
- Workflow file

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

New Issues | Fixed Issues | New Duplication | Fixed Duplication | Diff | Files | Commits

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

- Main scientific analysis and diagnostics library – responsible for **development of diagnostics and metrics (in recipes), with scientific output** (plots, files etc) → **SCIENCE** is the main output, **LARGE and DIVERSE 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")**

- 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 versions, and current version when changes occur; both manual and automated triggers) via a 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

  - EXIF and Mark I Eyeball testing (visualization of output)

  - input data specifications consistency tests

  - output provenance tests

- Still with some technical testing like for ESMValCore, but less strict:
  - FEWER unit/integration/regression (with sample data) tests
  - FEWER coding standards tests (`pylint` and `flake8`)
  - NO test coverage tests (when adding new code)

Provenance output and testing

all our software is stored on Zenodo for every release with a DOI
we have docker containers that offer pretty good software environment reproducibility,
we record the filenames and global NetCDF attributes of all input files.