EPhysLab

CIGUS Network
Research Centres of the
Galician University System

Campus Auga. Ourense
Universidade Vigo

GNU

Code accessibility and code quality in climate sciences

The case of the Coupled Model Intercomparison Project (CMIP)

Michael García-Rodríguez[1,2], Javier Rodeiro-Iglesias[2] and Juan A. Añel[1,3]

[1]EPhysLab, CIM-UVigo, Universidade de Vigo, Ourense, Spain
[2]School of Computer Sciences, Universidade de Vigo, Ourense, Spain
[3]GNU Project

Santiago de Compostela, 27 October 2026

Reproducibility needs the code

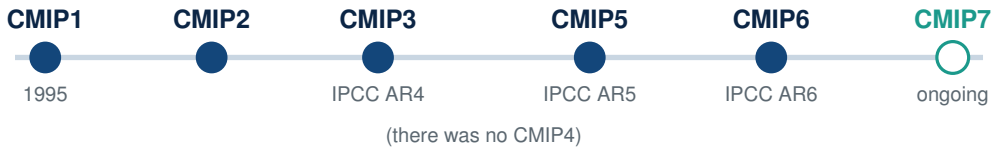
- A computational result can be checked only if the code, the inputs and the configuration are available.
- Climate models are the basis of the projections assessed by the IPCC. Most of them are coordinated through CMIP.
- Many journals now have code and data policies.
- Licences and intellectual property are part of the problem, and researchers often do not know the rules that apply to their own code.

Reproducibility
Same results under identical conditions.

Replicability
Consistent results under different conditions.

Definitions following ACM Artifact Review and Badging (2020).

CMIP: three decades of coordinated climate modelling



- Started in 1995 to share model results among groups.
- Models grew from a few processes to coupled atmosphere, ocean and land systems.
- Large legacy codebases, some parts written decades ago.
- Models share code with each other, and this inheritance is poorly documented.

Starting point: CMIP5 (Añel et al., 2021)

Procedure

1. First request sent from a student address, by someone outside the climate community.
2. The same request repeated two weeks later.
3. A third message identifying the team as climate scientists.
4. A short survey to centres that refused. Nobody answered it.

10/26
obtained (27/61 versions)

3/26
direct access (7/61 versions)

16/26
not obtained (34 versions)

More than half of the CMIP5 model versions were not available, even to research peers.

One centre answered only after the request was sent in French. Another gave access after a conversation at a conference.

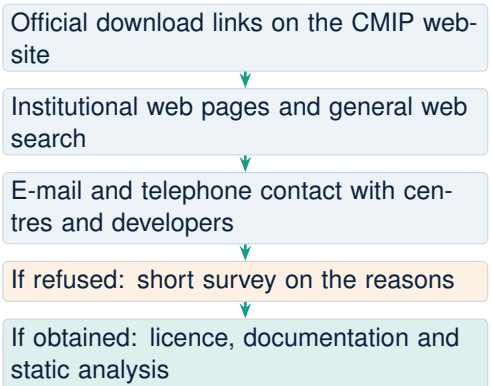
Añel, García-Rodríguez and Rodeiro (2021), GMD 14, 923–934.

This study: scope and method

Questions

- Can the source code be obtained?
- Under which licence and with which usage rights?
- Does it include instructions to compile, configure and run it?
- Which languages are used?
- How has static code quality changed across phases?

Scope: CMIP1 to CMIP6. 262 individual models, 121 of them coupled. CMIP7 is ongoing and excluded.



Scoring accessibility and reproducibility

★★★ **Maximum score**

Code available on the internet without restrictions, under a licence that allows full testing and evaluation.

One star less for each of these conditions

- Access requires contacting a researcher, centre or development group.
- A licence agreement must be signed.
- The requester must identify as a climate scientist.
- The licence restricts evaluation or use.

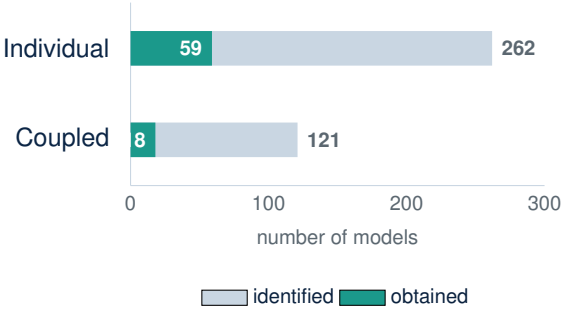
☆ An empty star means that the licence does not allow modification of the code.

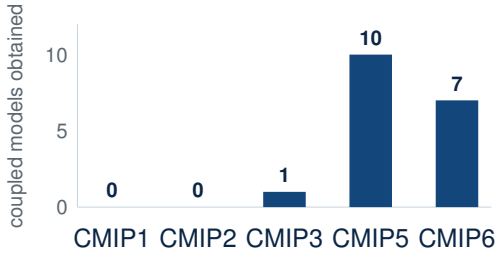
Also checked

- Is there a licence at all?
- Documentation, examples, dependencies and build files.

Reference case: a free licence such as GPLv3.

Access to the code across CMIP phases

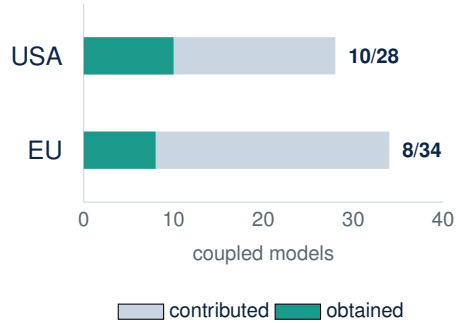




22% of individual models and 15% of coupled models were obtained. No code from CMIP1 or CMIP2 could be recovered because it was not preserved.

Geography and law

- USA, France, Germany and Norway gave access to all the models they had preserved.
- In the USA, works of federal employees are in the public domain (17 U.S.C. §105).
- NorESM is built on parts of CESM1 and may have inherited its licence conditions.
- Software can be patented in the USA and not in the EU. Even so, a larger share of US models was available.



EC-Earth and ECMWF models are developed by consortia of 12 to 23 European countries, depending on the phase.

Why is the code not shared?

What we observed

- Institutional or national legal restrictions.
- Concern about redistributing parts written by other groups.
- Old versions lost for lack of archiving.
- No explanation of licence choices. Some code came with no licence at all.
- No replies to the survey on reasons for refusal.

Answers to frequent objections

- *Hosting costs:* free long-term repositories with DOIs exist, for example Zenodo.
- *Proprietary dependencies:* they can be replaced by free software.
- *Loss of intellectual property:* distributing code does not usually transfer it.
- *Code not good enough:* sharing it is the way to improve it.

> There is no evidence that CMIP results are wrong. Lack of access to the code can still be used to question them.

Programming languages in CMIP models

- **Fortran** is the dominant language in every phase.
- **Python** is growing for pre- and post-processing, workflows and visualisation.
- **Perl** and **Bash** for job orchestration on HPC systems. **R** for statistical evaluation.
- **C/C++** for performance-critical modules, usually next to Fortran.

Language shares measured with GitLab repository analytics (GitHub Linguist).

Two exceptions

CFSv2-2011: 88.1% of the code is C.

NICAM.09: mostly Lisp, with Fortran second.

Developers extend existing codebases. This keeps Fortran in place across model generations.

FortranAnalyser: static analysis of Fortran code

- Evaluates code without compiling or running it.
- Works with any Fortran version, including legacy code.
- Developed at the EPhysLab because existing tools were of limited use for scientific Fortran.
- Gives a score from 0 to 10 for each file and a PDF report.
- Also reports lines, cyclomatic complexity, functions, variables and subroutine calls.

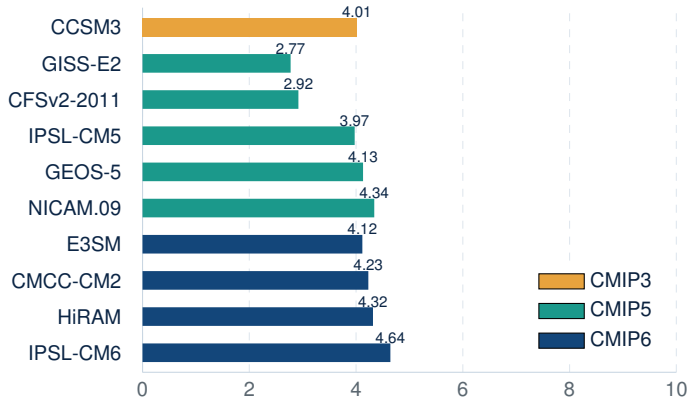
Metric	Range
Use of `implicit none`	0–2
Comments in files, variables, functions, subroutines, loops	0–2
Nested loops (complexity)	0–2
Comment-to-code ratio	0–2
Use of `exit`	0–1
Use of `cycle`	0–1

fortrananalyser.ephyslab.uvigo.es

doi:10.5281/zenodo.5942943

García-Rodríguez, Añel and Rodeiro-Iglesias (2024), Software Impacts 21, 100692.

Static code quality by CMIP phase



- Even the best models score below half of the maximum.
- Wide spread within CMIP5.
- Higher scores in CMIP6, which points to better workflows and coding standards.

FortranAnalyser scores (0–10) of the models discussed in the paper. Full list in Table 3 of García-Rodríguez et al. (2026).

The IPSL case: measuring helps to improve

3.97
IPSL-CM5
(CMIP5)



4.64
IPSL-CM6
(CMIP6)

Increase of 0.67 points, about 17%, between consecutive model generations.

- The IPSL developers applied the recommendations of a FortranAnalyser assessment.
- A static analysis tool inside the development cycle gives objective metrics and concrete changes to make.
- The same approach is open to any modelling centre.

Case study described in García-Rodríguez et al. (2024), Software Impacts 21, 100692.

The next phases: hybrid and machine learning models

- Machine learning components are not part of the CMIP phases analysed, but their arrival is foreseeable.
- They bring new languages and software ecosystems next to Fortran.
- Their behaviour depends on training data, calibration and configuration as well as on the source code.

Static analysis of the code

Provenance of training data

Documented training process

Interpretability of learned parameterisations

> Hybrid models will need to be assessed as systems of heterogeneous components, each with its own analysis tools.

Recommendations

For modelling centres

- Archive the frozen production versions used for international assessments, together with their outputs, in long-term repositories with DOIs.
- Publish code under free licences, with documentation and build instructions.
- Reduce dependencies on proprietary components.
- Add static code analysis to the development pipeline.

For CMIP, infrastructures and funders

- Include accessibility, licence and code quality checks in the evaluation workflow, for example next to ESMValTool.
- Use containers, cloud resources and workflow automation to make replication practical.
- Fund documentation, preservation and reproducibility work in addition to model development.

Summary

1. Access to CMIP model code remains limited: 59 of 262 individual models and 18 of 121 coupled models.
2. The code of CMIP1 and CMIP2 has been lost. Long-term preservation needs explicit planning.
3. Fortran dominates, and static quality scores stay far below the maximum.
4. Quality has improved from CMIP5 to CMIP6. The IPSL case shows that analysis tools lead to measurable changes.
5. New model types will require new ways to assess reproducibility.

> García-Rodríguez, M., Rodeiro-Iglesias, J. and Añel, J. A. (2026). Geosci. Model Dev. 19, 7687–7704. **doi:10.5194/gmd-19-7687-2026** (EGU highlight paper)

References

García-Rodríguez, M., Rodeiro-Iglesias, J. and Añel, J. A. (2026). Code accessibility and code quality across phases of the models of the Coupled Model Intercomparison Project. *Geosci. Model Dev.* 19, 7687–7704. doi:10.5194/gmd-19-7687-2026

García-Rodríguez, M., Añel, J. A. and Rodeiro-Iglesias, J. (2024). Assessing and improving the quality of Fortran code in scientific software: FortranAnalyser. *Software Impacts* 21, 100692. doi:10.1016/j.simpa.2024.100692

Añel, J. A., García-Rodríguez, M. and Rodeiro, J. (2021). Current status on the need for improved accessibility to climate models code. *Geosci. Model Dev.* 14, 923–934. doi:10.5194/gmd-14-923-2021

Añel, J. A. (2019). Reflections on the scientific method at the beginning of the twenty-first century. *Contemp. Phys.* 60, 60–62. doi:10.1080/00107514.2019.1579863

Añel, J. A. (2017). Comment on “Most computational hydrology is not reproducible, so is it really science?” by Hutton et al. *Water Resour. Res.* 53, 2572–2574. doi:10.1002/2016WR020190

Añel, J. A. (2011). The importance of reviewing the code. *Commun. ACM* 54, 40–41. doi:10.1145/1941487.1941502

Thank you

Juan A. Añel · j.anhel@uvigo.gal
EPhysLab, CIM-UVigo, Universidade de Vigo, Ourense

Funded by grants PID2021-124991OB-I00 and PID2024-158326NB-I00 (MICIU/AEI/10.13039/501100011033 and ERDF/EU) and by the Xunta de Galicia (GRC-ED431C 2025/37).