Open Source Guidelines

of the Potsdam Institute for Climate Impact Research

Authors: Torsten Albrecht, Lavina Baumstark, Jan Philipp Dietrich, Robert Gieseke

The Potsdam-Institute for Climate Impact Research (PIK) supports contributing to and releasing of Open Source software, for example by

- improving and contributing back to the Open Source software tools that make the computational research at the Potsdam-Institute possible
- releasing software tools that are developed as part of research projects under an Open Source license
- making modeling code available as Open Source software

As a member of the Leibniz association PIK is a co-signer of the "Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities" (https://openaccess.mpg.de/Berlin-Declaration) which was written to promote the Internet as a functional instrument for a global scientific knowledge base and a vision for disseminating knowledge and making it widely and readily available to society:

> "In order to realize the vision of a global and accessible representation of knowledge, the future Web has to be sustainable, interactiveand transparent. Content and software tools must be openly accessible and compatible."

These guidelines comply with and complement the PIK Guidelines for Ensuring Good Scientific Modelling Practice.

Summary of recommendations

- Code development:
  - use PIK's GitLab instance or GitHub for version management and publications
- License:
  - Tools: BSD-2-Clause (see list https://spdx.org/licenses)
  - Models: AGPL-3.0 (Affero-GPL) with contributors agreement

As scientific models are different from typical Open Source software more details and considerations are specified below. Most of it is also applicable to other research software.

Policy for Simulation Models at PIK

Models have become an integral part of science and represent as such an scientific achievement. However, models are not yet properly integrated into the scientific discourse. There are various reasons to go Open Source with a model, in particular regarding the **reproducibility and transparency** of the findings presented in publications. But it may also induce incentives to improve model code and to pass on **software development** skills to following generations of scientist and to foster **collaborations within the scientific community**. On the other hand, funding agencies (e.g. Horizon 2020) often require an evaluation of a potential for commercial use of a model project, which in principle does not contradict an Open Source publication, but these considerations may be relevant for the possible license scheme and should be preferably discussed in an early stage of the model project strategy.

This guideline should serve as assistance for any modeling group at PIK, to **demonstrate necessary steps and recommended choices** to take and to make aware for certain aspects that should be considered and discussed within the model developers team at some point.

Steps to take

1. Check formal requirements → checklist for models, proprietary products
2. Model development strategy → version management: GitHub, GitLab
3. Choose License → copyleft vs. permissive compatibility
4. Documentation → model implementation, user manual
5. Final check by administration → application form, contributors agreement

1. Check formal requirements and compatibility

Most models rely on products provided by external partners such as data or code modules. For these parts it has to be checked whether they can be published together with the model and if so under which conditions. The recommended workflow can be found as checklist and application form here.

1. The easiest case is that these products are already published themselves Open Source, but even in this case it has to be checked clearly under which conditions the product can be reused. The **requirements of Open Source licenses** are very different in this context and not all open source licenses are compatible to each other (more information about the different licenses in the license section).

2. In the case that the model uses a **proprietary product** we recommend to address the supplier of the product directly and to negotiate under which circumstances the product can or cannot be part of the final Open Source publication. This also holds true for a product under an Open

Source license which is not compatible to the Open Source license which is planned to be applied for the model. In these cases it might be possible that the supplier can provide you the product under a different license. Alternatively it should be checked if the proprietary component can be replaced by an Open Source component. Otherwise it can be published without this component and the user needs to acquire a separate copy.

In order to avoid causing resentment and under consideration of **good scientific practice** it is always recommendable to address the supplier of a product if there are any concerns about the right usage of the provided product.

2. Define a model development and publication strategy

Code publication requires a platform on which the code is provided to others. Technically, a simple upload of the model code (e.g. as a compressed folder) to your web page, on a blog or within supplemental websites of journals, fulfills all requirements for an Open Source publication. That is, the model code can be freely accessed, used, changed, and shared by anyone.

Yet, depending on the complexity of the model projects and the number of contributors, it is reasonable to provide code using a **version management system** such as Git or Mercurial on a publicly available server or even better on a web-based repository hosting services such as GitHub. This allows to easier follow, maintain and contribute to the model development and it is a crucial element if the community aspect of the Open Source project should play any role.

1. Our **recommendation**, especially suitable also for external collaborations is **GitHub** (https://github.com) since it is a well established Open Source distribution platform. The model repository can just be made publicly available, meaning that everybody can see all versions of the model and will instantaneously receive the newest changes to the code. This approach is highly recommended if the emphasis of the Open Source project is on the **interaction with the community** as this allows externals to work as effectively on the code as people directly in the group at PIK and to discuss new features with externals already in the development phase.
2. However, there are also **more restrictive solutions** possible in which model code is developed first internally. Bitbucket and **GitLab** (PIK has its own GitLab server, https://gitlab.pik-potsdam.de) provide free private repositories which can later be made public or moved to GitHub. Also GitHub has free private plans available for educational use (students). If the model is ready for publication just push the stable code version from the private repository to a public repository. This delay, however, may induce intransparency and hamper the interaction with externals.

3. Choose an appropriate license for the model

"Open Source" means more than just free access to the source code (transparency), the distribution of Open Source software must comply with certain defined criteria, see http://opensource.org/osd.html. **Open Source licenses provide a legal framework for the subsequent use and distribution of the software**. There are very different approved open source licenses available with quite different restrictions. A general overview on different types of licences can be found at http://www.ifross.org/lizenz-center/ and http://choosealicense.com/licenses/.

Regarding the demands of European Union funding agencies for scientific project proposals a proper management of intellectual property takes an essential role, i.e. examining the possibility of commercial or industrial exploitation and its legal protection. All Open Source Licenses permit a commercial use, but some (GPL) prohibit proprietary derivative works (license fees etc.).

1. An important difference among Open Source licences is the inclusion or exclusion of the so called "copyleft" (in contrast to copyright as it "protects freedom"). **Copyleft** means that a derivative of a product which was published under a copyleft license has to be published under the same conditions. This basically makes sure that code published under such an Open Source license will also remain open in the future in any case. This gives free software developers an advantage over proprietary developers: a software that they can use, while **proprietary developers cannot use it**. (The Linux kernel and Eclipse are licensed with copyleft).

   If you want to go for a license with copyleft (after compatibility check with third-party components) we recommend to use the GNU Affero GPL license (version 3 or later http://www.gnu.org/licenses/licenses.html). It is fully compatible to the commonly used standard GNU GPL license (version 3 or later), but in addition covers the use case of "Software-as-a-Service" (SaaS) by requiring also in these cases the publication of the corresponding source code. This becomes for instance relevant if someone provides access to a model as an online service by running it on a web server.

2. Using a so called **permissive license**, which is a license that does not contain copyleft, gives the user more freedom in the use of the code so that **derivatives of the code can become proprietary**, but it can still be free software and hence useful to the free software community. (Apache Server uses a permissive license)

   As a non-copyleft free software license we recommend to use the **2-clause BSD** (https://opensource.org/licenses/BSD-2-Clause). Both, the (A)GPL and the BSD licenses are well-established, used frequently and the BSD licensed code can be included in GPL-ed software (which is not necessarily the case for different Open Source licenses).

If you have chosen a license you must download a full text version of the license and save it to the root directory of the project with the name LICENSE or COPYING (see examples in the Appendix). Add a **copyright notice** that includes PIK as the copyright owner and the year(s) of the release. Authors of the project (who want to be named) can be added to the copyright notice or (if many) named in an AUTHORS file. It is recommended to add the copyright notice as a header near the beginning of every source file of your project (or at least a pointer to where the full notice is found):

> # (C) 2014-2016 Potsdam Institute for Climate Impact Research (PIK),
> # authors, and contributors <see AUTHORS file>
> # Licensed under GNU AGPL Version 3 <see LICENSE file>

Add a **copying permission statement** declaring under which license terms the software is distributed. See the following instructions for information how this procedure should look in detail for the recommended licenses:

> GPL: http://www.gnu.org/licenses/gpl-howto.html
> BSD: http://opensource.org/licenses/BSD-2-Clause

> This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.
>
> You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Copying permission statement should come right after the copyright notices, just replace the program name. If model code has been modified you should mention from where the original code has been included and who has modified the code at which date.

You may add individual copying requests as part of good scientific practise (not part of the license text and hence not legally binding), e.g. that scientific publications based on modified model code should imply a publication of the code modifications. Under certain circumstances you may consider trademark protection.

An example for a GPL licensed model at PIK is MAgPIE (see also the Appendix):

```
# (C) 2008-2016 Potsdam Institute for Climate Impact Research (PIK),
# authors, and contributors see AUTHORS file
# This file is part of MAgPIE and licensed under GNU AGPL Version 3
# or later. See LICENSE file or go to http://www.gnu.org/licenses/
# Contact: magpie@pik-potsdam.de
```

An example for code under a 2-clause BSD license developed at PIK is Makeinstall for Matlab: http://tocsy.pik-potsdam.de/makeinstall.php

Contributors Agreement

While the licenses under which the source code and documentations are licensed describe the rules under which the product can be used (outbound licensing) the contributors license agreement (CLA) describes the conditions under which contributions to the project will be handled (inbound licensing).

It is usually used to make sure that contributors actually own all required rights on their contribution and to create a common understanding between project owner and contributor what can be done later on with the contribution.

Especially in cases with external collaborations on a model/software under a copyleft license (e.g. GPL) it is recommended to have a contributor's agreement in place to avoid possible conflicts in the future. In the case of models that did not intend an open strategy in the beginning of their development, all contributors (eventually of different research projects) must be identified and asked for permission.

We recommend to use http://contributoragreements.org to create a contributor agreement for a project. In Github this agreement can be applied using CLA assistant (https://cla-assistant.io) which automatically checks for all push requests that the corresponding contributors have signed the given contributors agreement (if not, they are asked to do so). Signing itself happens within Github.

On http://contributoragreements.org you have different choices to build your contributors agreement. We recommend the following settings:

- Non-exclusive license: Giving the project owner the right to relicense it but does not retract any rights of the contributor on their contribution
- Permission to relicense the code under any Open Source license which is either approved by the Free Software Foundation or Open Source Initiative (Option 3)
- Permission to license provided documentations under cc-by (http://creativecommons.org/licenses/)
- Traditional patent licensing agreement

4. Prepare public documentation of the model

Model documentation is an integral part of the Open Source strategy providing the necessary transparency. Even though code can be published Open Source without any documentation this is probably not a very useful approach. For the documentation a major issue is to keep synchronisation between developing code and corresponding documentation. However documentation does not need to be boring homework after the implementation. Model documentation can be already part of the model design process with a clear plan of requirements, the algorithm and testing. There are different levels of model documentations.

1. **User manuals** should provide installation instructions and some simple working examples with sample data sets, in particular to simplify getting started for new users. They should be compact and updated regularly, e.g. for major releases of the model.
2. Mathematical model descriptions are usually published in the scientific literature.
3. **Model implementation documentation** should bridge the gap between the program language which is optimized for efficiency and the human language. It should provide an interface to understand the used discretization scheme and in the intention of individual functions. For some programming languages (i.e. in particular object-oriented) external software is available that **automatically generates** documentation of the model implementation (as html or pdf) using inline marker comments in the code providing an overview of the complex model hierarchy (see for instance:

   Doxygen (C++, Python, Fortran) http://www.stack.nl/~dimitri/doxygen/,

   Roxygen2 (R) http://cran.r-project.org/web/packages/roxygen2/, or

   Sphinx (Python and C/C++) http://www.sphinx-doc.org/en/stable/.)

If the model documentation is not already part of the model code, but rather a separate document, it should be provided together with the model with an appropriate license. For documents creative commons (http://creativecommons.org/licenses/) is a good basis for licensing. From the offered licenses we recommend to use the **Attribution International license cc-by 4.0** (https://creativecommons.org/licenses/by/4.0/) which requires to give credit to the originator of the file. Besides that no further restriction exist for the CC BY license meaning that users can reuse and or modify the documentation in any way they like (e.g. also using parts of the documentation for their own work). If there is a need for a more restricted documentation you can check the other licenses of the Creative Commons license portfolio. Just be aware that a more restricted license might complicate the handling and further development of the documents.

5. Receive official permission by PIK administration

The decision of a modeling group to release as Open Source needs to be confirmed by a co-chair of the research domain, analogous to the work-flow for the publication of research articles. To clarify code ownership and publication rights the model has to undergo a formal application process. This helps the

administration to evaluate risks and to double-check whether all preconditions for Open Source publication have been fulfilled. The corresponding checklist and application form can be found here.

No matter what platform you choose, in all cases a **digital object identifier (DOI)** (http://www.doi.org/) should be assigned to it to have a clear and standardized reference to the code. To receive a DOI you have to **send a request to the Telegrafenberg Campus Library** (via Marcel Meistring) together with the URL the DOI should currently refer to (if the location of your code changes in the future you must inform the library about the new address to make sure that the DOI is still working). More information: http://bib.telegrafenberg.de/en/publishing/library-as-publisher/identifier/.

6. Considerations and criteria for smaller models and tools

Under discussion and development ...