Opening the black box of energy modelling: strategies and lessons learned

Stefan Pfenninger <stefan.pfenninger@usys.ethz.ch>, ETH Zürich, Switzerland
Lion Hirth <hirth@neon-energie.de>, Neon Neue Energieökonomik GmbH *and* Hertie School of Governance, Germany *and* Mercator Research Institute on Global Commons and Climate Change (MCC), Germany
Ingmar Schlecht <ingmar.schlecht@unibas.ch>, University of Basel, Switzerland
Eva Schmid <schmid@germanwatch.org>, GERMANWATCH *and* PIK, Potsdam, Germany
Frauke Wiese <frwi@dtu.dk>, Danish Technical University, Denmark
Tom Brown <brown@fias.uni-frankfurt.de>, FIAS, University of Frankfurt, Germany
Chris Davis <c.b.davis@rug.nl>, University of Groningen, Netherlands
Birgit Fais <b.fais@ucl.ac.uk>, UCL, London, UK
Matthew Gidden <gidden@iiasa.ac.at>, IIASA, Laxenburg, Austria
Heidi Heinrichs <h.heinrichs@fz-juelich.de>, Institute of Energy and Climate Research, FZ Jülich, Germany
Clara Heuberger <c.heuberger14@imperial.ac.uk>, Imperial College London, UK
Simon Hilpert <simon.hilpert@uni-flensburg.de>, University of Flensburg, Germany
Uwe Krien <uwe.krien@rl-institut.de>, Rainer Lemoine Institute, Germany
Carsten Matke, <Carsten.Matke@dlr.de>, DLR Institute of Networked Energy Systems, Oldenburg, Germany
Arjuna Nebel <arjuna.nebel@wupperinst.org>, Wuppertal Institute, Germany
Robbie Morrison <robbie.morrison@posteo.de>,
Berit Müller <Berit.Mueller@rl-institut.de>, Rainer Lemoine Institute, Germany
Guido Pleßmann <Guido.Plessmann@rl-institut.de>, Rainer Lemoine Institute, Germany
Matthias Reeg <matthias.reeg@dlr.de>, DLR, Germany
Jörn C. Richstein <jrichstein@diw.de>, DIW, Berlin, Germany
Abhishek Shivakumar <ashi@kth.se>, KTH Royal Institute of Technology, Stockholm, Sweden
Iain Staffell <i.staffell@imperial.ac.uk>, Imperial College London, UK
Tim Tröndle <tt397@cam.ac.uk>, Cambridge University, UK
Clemens Wingenbach <clemens.wingenbach@uni-flensburg.de>, University of Flensburg, Germany

ABSTRACT: The global energy system is undergoing a major transition, and in energy planning and decision-making across governments, industry and academia, models play a crucial role. Because of their policy relevance and contested nature, the transparency and open availability of energy models and data are of particular importance. Here we provide a practical how-to guide based on the collective experience of members of the Open Energy Modelling Initiative (Openmod). We discuss key steps to consider when opening code and data, including determining intellectual property ownership, choosing a licence and appropriate modelling languages, distributing code and data, and providing support and building communities. After illustrating these decisions with examples and lessons learned from the community, we conclude that even though individual researchers' choices are important, institutional changes are still also necessary for more openness and transparency in energy research.

Keywords: open source, open data, energy modelling

1. Introduction

The history of energy system planning is primarily closed and proprietary, having been pursued by research institutions and large, vertically-integrated utilities that were under no obligation to reveal their modelling assumptions or methodologies. This may have been acceptable in a conventional energy system with only a few players, but the requirements on energy system planning are changing significantly driven by the advent of liberalised, regulated markets and the need for deep reductions in greenhouse gas emissions [1]. In addition, the rapid decline in costs of wind and photovoltaics (PV) [2] as well as advances in energy storage [3] mean that new modelling methods are required.

Open energy modelling is desirable for many reasons [4]. First, open code and data improve scientific quality if they lead to more transparency and reproducibility, and thus permit effective collaboration across the science-policy boundary. This is particularly important in energy policy, an urgent and highly contested topic. More transparent modelling is desirable from a regulatory and political perspective, as opening up decision processes and the reasoning behind them may lessen public opposition to new legislation and infrastructure. By reducing parallel efforts and allowing researchers to collaborate and share the burden of developing and maintaining large code bases and datasets, openness also enables increased productivity. Ultimately, we believe research funded with public money should be freely available to the public.

The open code, open data and open science movements are only slowly leading to models and data being opened up [5]. In order to classify as "open", the data or model code needs to be both accessible and legally usable. Hence, we pragmatically define open code, open data, and open models as artefacts that are available under a commonly used licence, which allows re-use without undue restrictions[1]. The history of open energy modelling can be traced to the release of the model Balmorel in March 2001 [7] (albeit without a standard licence until 2017), followed by early attempts with a now abandoned GPL-licensed model called deeco in 2004 [8]. After a long pause, the release of the modelling frameworks TEMOA and OSeMOSYS in 2010 [9,10] has spearheaded several dozen open projects.

The mainstream approach to energy modelling is often still proprietary and opaque, even where it directly feeds into policy [11]. Underlying reasons are manifold; however, commercial sensitivity, lock-in to proprietary models, lack of awareness, institutional and personal inertia, and fear of losing competitive advantage are certainly involved [4]. Due to this multitude of challenges in opening the black box of energy modelling, a group of

1 This is by and large akin to the Open Source Definition used by the Open Source Initiative [6], to which all commonly used licences for code conform. The full list of OSI-approved licences is at https://opensource.org/licenses.

modellers founded the Open Energy Modelling Initiative, Openmod[2], in 2014. This initiative met a timely need and membership is now approaching 400.

Based on our experience and as members of the Openmod initiative, this article addresses what we perceive to be the crucial factors limiting openness of energy data and models: the lack of practical knowledge as well as personal and institutional inertia. The article proceeds as follows. First, we briefly introduce the key steps in energy modelling and how they link to aspects of openness. We then walk through the practical steps energy modellers must think about and choices they must make when deciding to go open. Finally, we describe three examples that provide further context for these key choices, before concluding with the most important challenges that remain to be overcome in the wider institutions that shape the research landscape.

2. The energy modelling process

In order to structure the debate, Figure 1 outlines how open data, open source and open access conceptually relate to the energy modelling process.

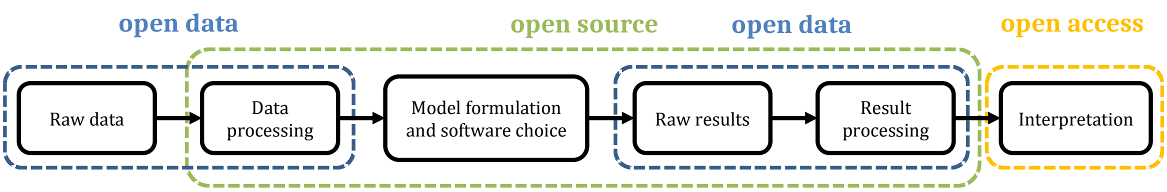

Figure 1: Open data, open source, and open access in relation to the energy modelling process.

Data is both an input and output of the process. Raw data in the energy field is spread widely and of varying quality, coming from academic sources, non-governmental bodies, markets, individuals and commercial entities. An obvious impediment to openness is the widespread use of non-disclosure agreements under which commercially sensitive data may be shared. A less obvious impediment is that in many cases, no explicit licence is attached to input data. Contrary to common practice, this does not imply the legal permission to use and share data, as discussed below. This is of crucial importance since the degree of openness and the licensing conditions of input data influence the degree to which a model based on them can be made open.

Raw data from various sources and in different formats must first be processed to become accessible to a model. Three kinds of data processing are usually necessary: (1) time series data: creating or cleaning up intermittent renewable generation and demand data, (2) geographic data (e.g. installed generation capacities): aggregation or disaggregation and

2 http://www.openmod-initiative.org/

other forms of geospatial analysis, (3) simple “tabular” data (e.g., costs of technologies or fuels): varied manual processing such as making assumptions where values are missing or converting currencies. Assumptions made during processing often go unreported and undocumented. As an example, recent work examining the effect of different time resolution reduction methods on model results found large enough effects to qualitatively affect conclusions [12–16]. This suggests that carefully documenting and making processing steps and code open is necessary not just for reproducibility, but also to allow users to assess the impact on results, their interpretation, and resulting policy implications.

Models are idealised representations of real systems built to perform a specific analysis or answer a specific question, and so usually include code (e.g. for reading data, constructing and solving equations) and data (e.g. technology costs). Implementing a model is to turn its conceptual components (such as the equations describing an energy system and the accompanying data parameters) into a computer program, for example, using a mathematical programming language. We distinguish *models* from *frameworks*. The latter are programs which are later populated with data to produce a model. They may contain structures designed to provide reusable functionality when building models (e.g., a general set of functions in a given programming language to read, process and analyse data). Models include data and assumptions, and are usually specific to one situation. The boundaries between model and framework are not always clear, but being aware of them helps when making decisions such as the choice of licence, because licensing code and data requires different considerations [17].

Model implementations vary greatly in complexity. Most simply, models can consist of a collection of spreadsheets. More complex ones might use commercial off-the-shelf tools such as Plexos, a mathematical programming language such as GAMS or GNU MathProg, or a general-purpose programming language such as Java, C++ or Python. Using a mathematical programming language keeps the model code at a higher level and focused on the actual mathematical model. On the other hand, building models directly with general-purpose programming languages allows common components of model implementation to be identified and extracted more easily, and made available for other models to use (i.e., as a framework).

Model results are published in scientific journals, access to which is frequently restricted to journal subscribers. Open access publishing changes the conditions under which publications are distributed. We include this final stage of openness in Figure 1, but do not further discuss it here.

3. Key considerations when going open

The multitude of choices and terminology a researcher faces when deciding how to open up code and data can be overwhelming. Based on our collective experience, we provide practical guidance covering the key considerations that arise during the process.

3.1. Who owns intellectual property

When making any code or data available, the first important step is to establish who owns the relevant intellectual property (i.e. who holds copyright): the researcher or their institution. In most cases where an employment contract is involved, the employer will own the intellectual property rights. Some institutions may automatically grant their researchers the right to open-source research software or have a fast-track process to grant approval for open-source release. Nevertheless, researchers should always confirm ownership regulations with their institutional technology transfer office or legal department [18]. "Provenance" is the history of a codebase and its contributions. Unless provenance can be conclusively determined it cannot be certain that code released under a specific licence is unencumbered by conflicting intellectual property ownership.

Proactively ensuring that contributors state that they have the right to contribute to an open-source project resolves potential legal concerns. For example, as of July 2017, the GitHub terms of service include a clause[3] stating that by contributing to a repository the user agrees to license their contribution under the licence specified in the repository, and that they are legally able to do so (i.e. they own intellectual property for their contributions). Another solution is to include a "certificate" alongside the licensed project that lists contributors including, where applicable, employers or clients that may own rights. Contributors can then be asked for written permission for their contributions where necessary [19], or requested to sign a contribution licence agreement (CLA), although the latter poses an additional barrier to contribution [20].

3.2. What and how much to publish

Every bit of information can be supportive when researchers try to reproduce or reuse the work of others. While true reproducibility requires complete openness, it is still valuable to open only parts of the model, data or data processing steps. Several tools exist to support the creation of reproducible research, like entire workflow systems [21], tools to track provenance [22], and more recently containerisation [23]. Research in this field is ongoing [24]. While complete and long-term reproducibility remains difficult and comprises multiple

3 GitHub Terms of Service, Contributions Under Repository License: https://help.github.com/articles/github-terms-of-service/#6-contributions-under-repository-license

aspects [25], researchers should not shy away from sharing code, even if they believe it is not yet comprehensive enough to result in fully replicable science [26].

When sharing either code or data, appropriate documentation for the target audience is important. With code, for example, will users have a graphical interface, or will they use the software as a library included in their own project? Is it desirable that users become collaborators that could extend and improve code? For possible collaborators, the internals of the code and/or the application programming interfaces (APIs) should be documented. In general, best practices from software engineering should be applied where possible [27]. Automated tests can provide a formal specification of the project, as they ensure certain features continue working after changes to the code. Code review can improve code readability [28] and can be used efficiently in a scientific context [29]. Guidelines for researchers new to software development are available in [30,31]. With data, it is primarily important to document where it has come from, what units and conventions are involved, and where the key uncertainties lie. The use of a standard metadata format, such as the Data Package standard [32], can help to ensure consistent and complete documentation. With full models, beyond documenting their code and data, even just documenting how the model was applied in specific studies can be of help to potential users.

3.3. Which licence to choose

To reiterate: when code or data are made available it is important to clarify the legal terms under which they are published. Applying a well-known licence is the easiest and preferred approach. It ensures interoperability between software projects and makes it straightforward for users and possible contributors to immediately understand the terms [33]. It is important to note that intellectual property rights and copyright protection always apply. Without a licence, code cannot be legally used [34,35], while the legal context under which data can be used remain unclear for potential users and contributors.

Two key considerations influence the choice of licence. First, different licences are more applicable depending on whether the content to be licensed is code, data, or other content. Second is the choice between two types of licences, often grouped into permissive and copyleft licences. Permissive licences generally allow all re-use, including integration into closed-source projects, without requiring improvements to the code to be released under the same licence. In contrast, copyleft licences require that derivative work is shared under the same or a compatible licence.

In the case of computer code, the most common permissive licences are the BSD, MIT and Apache V2.0 licences. The most common copyleft licence is the GNU GPL and related licences (such as the LGPL, AGPL) [18,36]. In the case of data and other content, common permissive licences are Creative Commons Attribution licences (CC BY) and common copyleft licences include other Creative Commons licences such as Creative Commons Attribution-ShareAlike (CC BY-SA) or the Open Data Commons Open Database License (ODbL). When licensing code, it is advisable to use licenses specifically developed for code, rather than other licences such as Creative Commons. The reason is that software-focused licences cover technical issues such as linking different pieces of code which do not arise with data and other content.

While some licences (like Creative Commons NonCommercial licences) specifically forbid commercial re-use of data, it is important to be aware that none of the commonly used licences for code prevent commercial use. They only specify whether users (both commercial and non-commercial) must make available their derivative works under the same licence or not. In addition, the definition of “commercial” as opposed to “non-commercial” can be difficult in practice [37]. For instance, contract research may be considered a commercial activity, even if conducted by a university.

The choice of permissive versus copyleft is more intricate. Permissive licences place few restrictions on users and thus make it more likely that code or data are re-used. Copyleft licences, especially the GPL, require that any derivative work is shared under a compatible licence. They are underpinned by a belief in the importance for all code and data to be open, but in practice can restrict the potential set of users: for example, the code licensed under the GPL licence cannot be integrated into permissively licensed code. Permissive licences have become the most popular, including in academia [18]. It is important to note that GPL-licensed code cannot be included in permissively licensed projects due to conflicting licensing conditions. Further guidance on the licence choice is given in [17,34,37].

While copyleft licences stipulate that if a derivative work is shared, it must be under the same licence, neither permissive nor copyleft licences require that derivative versions be shared at all. In other words, it is legally possible to publish results obtained with a modified version of a GPL-licensed model, without making available the modified model code. To address this, DeCarolis et al. [39] have proposed a new licence with provisions to enforce public release on formal publication of results (e.g. in a journal). However, licence proliferation and compatibility is already an issue [33], so instead, journal policies could require public availability of code and data on which a piece of submitted work is based.

3.4. Which modelling tools or language to choose

The type of modelling tool or programming language used to build a model defines, and to some extent limits, both the capabilities of the model and the degree to which it can easily be made open. The choice should also be considered in light of the user group being targeted. Choosing a specific (open-source) language and licence may be a reason to start building a model from scratch. Open languages, which include most common programming languages, guarantee that there are no financial barriers for new users. They are also often platform-independent, and can be run in containerised environments. On the other hand, commercial tools can offer improved performance [40], capabilities or usability benefits. The barrier to entry for non-programmers is also lower than with fully-featured programming languages. If a model requires a commercial tool such as a solution algorithm to run, it is still possible to provide the code or model definition under a permissive open licence. This does not avoid financial barriers for users, but the model itself can be scrutinised.

A widely-used tool is Microsoft Excel. Excel models come in many forms, from simple spreadsheet calculations to fully packaged applications and web services [41,42]. Its lack of version control and collaborative features makes it more difficult to use in common open workflows, but spreadsheets can nevertheless be made available, with the advantage of a low entry barrier for users.

3.5. How to distribute code and data

Ways of publishing open code and data range from compressed archives on personal or institutional websites to using code hosting platforms. The target audience and the intention behind open release will influence the choice. A compressed archive is sufficient, and possibly preferable, if minimum maintenance and no active participation is desired (for example, to release an archive of data or code alongside a specific publication). A platform makes sense if further development of code and data alongside contribution from others is envisaged. Platforms like GitHub or GitLab provide issue trackers for feature requests and bug reports, easy ways to contribute and review code, and hosting for wikis and documentation. GitHub in particular has emerged as a standard code sharing platform for many communities [43]. By providing a surface with minimum friction for re-use and collaboration, such platforms are arguably the most appropriate choice for code.

Similar choices have to be made for publishing data. The "good enough practice" described in [30] includes a description for what kinds of data version control is useful. For a comprehensive guide on scientific digital data storage, see [44]. For examples of platforms for energy-related data see [45,46]. A comprehensive overview of recommended online repositories for data is also provided by Nature Scientific Data[4]. Irrespective of repository choice, a DOI (Digital Object Identifier) makes code and data citable, even if it is not associated with a publication. Online archives such as Zenodo[5] or the Open Science

4 https://www.nature.com/sdata/policies/repositories
5 https://www.zenodo.org/

Framework[6] can mint DOIs. For code hosted on GitHub, a new Zenodo DOI can be automatically generated for each release.

3.6. How to provide support and build a community

Many open projects do not stop at providing access to code and data, but also offer structures to interact with the developers and support new users. This enables a community to form around a project, which can assist in further development, identifying bugs or other issues and helping people starting out with the model. The nucleus of a community can be formed through code hosting platforms with their issue trackers and wikis. For more on community-building, see [47].

A common concern is that by providing code and data openly, authors will be flooded with support requests, but that is unfounded in our experience. Conversely, the benefits even for an individual researcher – such as increased citations, requests for contract research and collaborations greatly outweigh the costs. Some projects also go beyond documenting the functionality of their model and put work into creating tutorials and simple examples for beginners; examples include Calliope[7], OSeMOSYS[8], PyPSA[9] and oemof[10]. Simple examples allow potential new users to quickly assess the functionality and usability of the model. A step further is to offer a forum or mailing list for people to ask questions, request new features and get update announcements (also see the four above projects for different approaches to doing this). This is common practice for large open software projects. While this may seem to create more time demands for developers and public pressure to respond to support requests, it can also help developers to see how people are using their software or data, may facilitate peer-to-peer support, and may improve documentation and tutorials.

4. Examples

The following uses three examples to illustrate the choices discussed above. First, the difference between starting a new open-source project versus opening up an existing closed model, and the implications of choosing a licence. Second, a case study of cross-pollination amongst power system tools written in Python. Third, lessons learned on how to build a community around a project. The projects mentioned in these examples are summarised in .

6 https://osf.io/
7 https://calliope.readthedocs.io/en/stable/user/tutorial.html
8 http://www.osemosys.org/get-started.html
9 https://pypsa.org/examples/
10 https://oemof.readthedocs.io/en/stable/getting_started.html#examples

Table 1: Overview and classification of the discussed models and frameworks. Type is classified by F (framework) or M (model). This list is non-exhaustive: consult other sources such as the Open Energy Modelling Initiative's wiki[11] for a more complete list of open modelling projects.

Name	Type	Description	Language	Licence	Release	Ref.	Website[12]
Balmorel	F	Energy system model framework	GAMS	ISC	2001[13]	[7]	Link
Calliope	F	Energy system model framework	Python	Apache 2.0	2013	[13]	Link
deeco	F	Operational optimisation model framework	C++	GPLv2	2004	[8]	Link
GridCal	F	Power system model framework based on PYPOWER	Python	GPLv3	2016	[48]	Link
oemof	F	Energy system model framework	Python	GPLv3	2015	[49]	Link
OSeMOSYS	F	Energy system model framework	MathProg, Python	Apache 2.0	2010	[9]	Link
pandapower	F	Power system model framework based on PYPOWER	Python	BSD 3-clause	2016	[50]	Link
PYPOWER	F	Power system model framework	Python	BSD 3-clause	2011	[51]	Link
PyPSA	F	Power system model framework inspired by PYPOWER	Python	GPLv3	2016	[52]	Link
renpass	M	Power system simulation model (Germany and neighbouring countries)	R	GPLv3	2014	[53]	Link
renpassGIS	M	German electricity market model	Python	GPLv3	2016	[54]	Link
SciGRID	M	Transmission grid model	Python	Apache 2.0	2015	[55]	Link
TEMOA	F	Energy system model framework	Python	GPLv2	2010	[10]	Link
UKTM	M	UK energy system model based on TIMES	GAMS	Unreleased	–	[56]	–

4.1. Starting open or closed and choosing a licence

Creating a new codebase with an open licence intuitively seems easier than retroactively opening a closed-source project. When starting open, all decisions can be made accordingly: for example, choosing third-party code and data based on licence compatibility. A recent flourishing of energy models and frameworks have begun as open-source projects from the outset, for example, OSeMOSYS, oemof, SciGRID, PyPSA, TEMOA, and Calliope (see Table 1). Opening up formerly closed models can be challenging, since it is possible that parts of the code cannot be released (for example, because the copyright holder of the code will not permit it or the provenance cannot be reconstructed). Such parts must be substituted or rewritten, which may not be practical if no open substitute exists.

11 http://wiki.openmod-initiative.org/
12 Main project website with access to code and documentation, or GitHub repository where no separate project website is available
13 Release 2001, but licensed only from 2017 onward

Even re-licensing existing open-source code after its release can be confusing and problematic to users. It also requires agreement of all copyright holders, which after several years of development may well include multiple institutions and individuals. Models like SciGRID and Calliope were therefore released under a permissive licence from the start (the Apache 2.0 licence). This permits the inclusion of all or parts of the code into closed-source projects as long as a copyright notice is included. Other models, like PyPSA or oemof, are licensed under the GPL. The GPLv3 is not bidirectionally compatible with the Apache 2.0 licence, for instance, and if GPLv3 code were added to a project licensed under a permissive licence such as Apache 2.0, the resulting combined codebase would need to be relicensed under the GPLv3. Furthermore, the GPL (both version 2 and 3) expressly forbid linking to non-GPL-compatible libraries and may therefore limit development options. The PyPSA developers therefore later decided to release part of their code that could be relevant for other projects under the Apache 2.0 licence, which was relatively easily possible as all three developers were working at the same institution at that point. The question of licensing can also veer into the discussion of viable open-source business models. For example, Renewables.ninja [57,58] makes data available under a non-commercial Creative Commons licence, so that commercial licensing can cross-subsidise the cost of providing a free service to academics.

Besides technical and legal difficulties, moving from closed to open can hit further stumbling blocks. The UK TIMES Model (UKTM) started closed-source and was scheduled to be released with an open licence in 2015 [56]. However, the intended release date has been successively pushed back, and is now anticipated in 2018. Part of the reason given is the additional effort required to meet government guidelines for presentation, documentation and plausibility of models [59]. The use of UKTM within policymaking also creates obstacles to openness. The UK government used it for modelling its response to the Carbon Budgets, the UK's official pathway for decarbonisation, and did not want the model to be openly and freely available whilst conducting this work [60].

A possible way to overcome this issue of culture can be to open things up incrementally. This is particularly pertinent with models that contain large amount of data, not just code. For example, the UK TIMES Model features around 1,600 technologies, each of which requires more than a dozen parameters. Unless these data were sourced very carefully, with provenance and attribution established in well-organised metadata associated with the data, opening up such a large model alongside its data might result in almost insurmountable challenges. Improved procedures for attribution, referencing, version control and documentation with such a large catalogue is necessary, in particular since typically, civil servants working with it are typically not trained scientists or data curators [60].

4.2. Cross-pollination: PYPOWER and its descendants

The range of power system tools written in the Python programming language provides an illustration of the downsides of uncoordinated development. Software projects were started for specific research questions, abandoned, and then further developed by other groups leading to fragmentation. That said, open licences have enabled code to be shared between the resulting independent projects. The first power system modelling tool in Python was a translation of Matlab-based Matpower [61] into Python called PYPOWER, developed by Richard Lincoln from 2009 onwards for his doctoral thesis in 2011 [51]. Active development ceased in 2014, although as of 2017 bug fixes from third-party developers are still merged into the code base.

In 2016 three independent new Python power system tools were announced, each of which continued the development of PYPOWER in a different direction. PyPSA [50] re-wrote the code base from scratch and added multi-period optimisation; pandapower [50] built on PYPOWER to further develop the modelling of distribution networks; GridCal [48] added new algorithms for power flow to PYPOWER and offered a graphical user interface. While it may seem counter-productive to have so many overlapping software projects for power simulations in Python, because of the open software licensing, it has been possible to share code and ideas between the projects. PyPSA borrowed some data structures from pandapower; PyPSA's handling of disconnected networks inspired pandapower's, and PyPSA is planning to implement some of GridCal's algorithmic work. Therefore, despite initial fragmentation, users benefit from this pooling of functionality.

This example demonstrates how in our view, the energy modelling field is currently in a phase of growth and experimentation, which may well be followed by a period of consolidation [62]. The example also shows the emergence of community efforts through different research groups developing closely related projects, then realising they can benefit from each other's work. It highlights the issue of projects with no active contribution management – after branching off into separate projects, reunification can be difficult. This implicit community growth contrasts with the next examples, which include the explicit, focused and labour-intensive effort to create a community around the OSeMOSYS energy modelling framework.

4.3. Community building: OSeMOSYS and oemof

Building and maintaining a community of users and contributors is not straightforward and requires careful consideration. Issues include incentivising users to contribute, attracting new contributors and streamlining the process of engaging with the community. Without focused personal involvement by at least one core developer or community organiser, we have found that this is difficult to achieve.

A good example is OSeMOSYS, a framework for long-term energy system planning optimisation models implemented in GNU MathProg and released in 2011 [9]. The core developers at KTH Stockholm have invested significant time and effort to build a user community. The community can be broadly divided into two types of users: academic researchers (including students) and policymakers, particularly in developing countries. To reach the first group, the OSeMOSYS developers hold regular side events at conferences such as the International Energy Workshop (IEW). These reach academics looking for open tools to teach energy system modelling, or for building new models for research purposes. For students, OSeMOSYS is useful as a 'gateway' modelling tool – its code is relatively straightforward and easy to customise.

To reach policymakers, the OSeMOSYS team developed a close relationship with the United Nations, specifically UNDESA and UNDP, to help drive adoption of OSeMOSYS by national governments for their energy sector planning. Notable examples of this include South Africa, Cyprus and Bolivia. Open models allow countries with a limited budget for energy policymaking to build their own domestic modelling capabilities. Through such collaboration, the developers engage directly with policy makers. Finally, OSeMOSYS sends out a monthly newsletter with feature additions, upcoming events, and the latest publications.

A slightly different approach was taken by the oemof project [49]. It is a general framework in Python inspired by and based on three earlier models. First, a model of the power system in Germany and neighbouring countries, renpass [53], written in R, as well as the Matlab-based MRESOM [63] and Python-based pahesmf [64]. Oemof, based on these three projects, included some of the original as well as new developers. The key focus was to first establish a working inter-institutional development process by following best practice from professional software development as much as possible, and only later shift to developing an active user community. Ultimately, renpass was then reimplemented using the oemof framework it inspired, as renpassGIS [54]. An initial problem-specific model leading to the development of a more general framework makes sense: indeed another framework, the Calliope project [13], also evolved from an earlier model written in GAMS [65], but in that case, the original inspiration was never re-implemented.

5. Discussion and conclusion

We are convinced that open energy system modelling – using open data in open-source models to produce open data that is discussed in open-access outlets – comes with many benefits: it increases the quality of research, reduces duplication of work, increases credibility and legitimacy, provides transparency to the policy discourse and makes high-quality data and planning tools accessible to researchers and government agencies without the funds for commercial options. There are also private benefits for individual researchers such as increased citations and future contract research.

Based on our own practical experience we have compiled strategies and lessons learned for researchers wanting to open their modelling black boxes. These include ensuring consent from all intellectual property holders, identifying those parts of the work that can be published, choosing an appropriate licence, considering the programming languages and frameworks to use, identifying an appropriate distribution channel and building a community of users and contributors.

That is not to say that the traditional tools of academic knowledge dissemination are not still useful: researchers should use existing fora such as academic conferences and workshops to engage with the community and discuss methodology, explain models and data accurately in publications, favour concise approaches over complicated ones to answer specific research questions and provide accessible and up-to-date documentation for their models and data. We also extend an invitation to academics and non-academics alike to make use of and contribute to the Open Energy Modelling Initiative for this purpose by joining our email list, forum and regular workshops[14].

The decisions we lay out above are predominantly those taken by individual researchers and their research groups. However, there are wider issues for which responsibility lies with the broader institutions which employ and fund these individuals. Three are of particular importance: first, data relevant for energy system research is often provided by institutions such as statistical offices, government agencies, or transmission system operators. These institutions should ensure that open licences are provided from the outset, which sadly today is still the exception rather than the rule. Second, research funding agencies have the power to change the current incentive structure by including open data and open-source requirements in their calls and contract research. Finally, there is still a lack of recognition for software and data development in the academic assessment system. Either they should be recognised alongside paper writing, or it should be easier to write journal papers on software and data. Ultimately, to uphold the trust and equality of academic research, we

14 http://www.openmod-initiative.org/

believe that the methods and results from research funded by public money should be openly available to the public.

6. Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

7. References

[1] IPCC, Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2014.

[2] IEA, World Energy Outlook 2016, IEA, 2016. http://www.oecd-ilibrary.org/energy/world-energy-outlook-2016_weo-2016-en (accessed January 3, 2017).

[3] O. Schmidt, A. Hawkes, A. Gambhir, I. Staffell, The future cost of electrical energy storage based on experience rates, Nature Energy. (2017). doi:10.1038/nenergy.2017.110.

[4] S. Pfenninger, J. DeCarolis, L. Hirth, S. Quoilin, I. Staffell, The importance of open data and software: is energy research lagging behind?, Energy Policy. 101 (2017) 211–215. doi:10.1016/j.enpol.2016.11.046.

[5] S. Pfenninger, Energy scientists must show their workings, Nature. 542 (2017) 393. doi:10.1038/542393a.

[6] Open Source Initiative, The Open Source Definition | Open Source Initiative, (2007). https://opensource.org/osd (accessed July 2, 2017).

[7] H.F. Ravn, et al., Balmorel: A model for analyses of the electricity and CHP markets in the Baltic Sea Region, Elkraft System, 2001. http://www.balmorel.com/index.php/balmorel-documentation (accessed July 10, 2017).

[8] R. Morrison, deeco: high-resolution energy system modeling framework (historical), 2017. https://github.com/robbiemorrison/deeco (accessed July 7, 2017).

[9] M. Howells, H. Rogner, N. Strachan, C. Heaps, H. Huntington, S. Kypreos, A. Hughes, S. Silveira, J. DeCarolis, M. Bazillian, A. Roehrl, OSeMOSYS: The Open Source Energy Modeling System: An introduction to its ethos, structure and development, Energy Policy. 39 (2011) 5850–5870. doi:10.1016/j.enpol.2011.06.033.

[10] J. DeCarolis, K. Hunter, S. Sreepathi, The TEMOA Project: Tools for Energy Model Optimization and Analysis, in: Stockholm, Sweden, 2010.

[11] K.-K. Cao, F. Cebulla, J.J. Gómez Vilchez, B. Mousavi, S. Prehofer, Raising awareness in model-based energy scenario studies—a transparency checklist, Energy, Sustainability and Society. 6 (2016) 28. doi:10.1186/s13705-016-0090-z.

[12] P. Nahmmacher, E. Schmid, L. Hirth, B. Knopf, Carpe Diem: A Novel Approach to Select Representative Days for Long-Term Power System Models with High Shares of

Renewable Energy Sources, Social Science Research Network, Rochester, NY, 2014. http://papers.ssrn.com/abstract=2537072 (accessed March 22, 2015).

[13] S. Pfenninger, Dealing with multiple decades of hourly wind and PV time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability, Applied Energy. 197 (2017) 1–13. doi:10.1016/j.apenergy.2017.03.051.

[14] C.F. Heuberger, I. Staffell, N. Shah, N.M. Dowell, A systems approach to quantifying the value of power generation and energy storage technologies in future electricity networks, Computers & Chemical Engineering. (2017). doi:10.1016/j.compchemeng.2017.05.012.

[15] J. Hörsch, T. Brown, The role of spatial scale in joint optimisations of generation and transmission for European highly renewable scenarios, arXiv:1705.07617 [Physics]. (2017). http://arxiv.org/abs/1705.07617 (accessed July 14, 2017).

[16] K. Poncelet, E. Delarue, J. Duerinck, D. Six, W. D'haeseleer, Impact of Temporal and Operational Detail in Energy-System Planning Models, KU Leuven Energy Institute, 2015. https://www.mech.kuleuven.be/en/tme/research/energy_environment/Pdf/wp-en201420-2.pdf (accessed April 23, 2015).

[17] D.S. Katz, K.E. Niemeyer, A.M. Smith, W.L. Anderson, C. Boettiger, K. Hinsen, R. Hooft, M. Hucka, A. Lee, F. Löffler, T. Pollard, F. Rios, Software vs. data in the context of citation, PeerJ Preprints, 2016. doi:10.7287/peerj.preprints.2630v1.

[18] A. Morin, J. Urban, P. Sliz, A Quick Guide to Software Licensing for the Scientist-Programmer, PLoS Computational Biology. 8 (2012) e1002598. doi:10.1371/journal.pcbi.1002598.

[19] K.E. Mitchell, /dev/lawyer License from Who?, (2016). https://writing.kemitchell.com/2016/05/13/License-from-Who.html (accessed July 1, 2017).

[20] G.R. Gangadharan, V. D'Andrea, S.D. Paoli, M. Weiss, Managing license compliance in free and open source software development, Inf Syst Front. 14 (2012) 143–154. doi:10.1007/s10796-009-9180-1.

[21] V. Curcin, M. Ghanem, Scientific workflow systems - can one size fit all?, in: 2008 Cairo International Biomedical Engineering Conference, 2008: pp. 1–9. doi:10.1109/CIBEC.2008.4786077.

[22] A. Davison, Automated Capture of Experiment Context for Easier Reproducibility in Computational Research, Computing in Science & Engineering. 14 (2012) 48–56.

[23] C. Boettiger, An Introduction to Docker for Reproducible Research, SIGOPS Oper. Syst. Rev. 49 (2015) 71–79. doi:10.1145/2723872.2723882.

[24] K. Hinsen, ActivePapers: a platform for publishing and archiving computer-aided research, F1000Research. (2015). doi:10.12688/f1000research.5773.3.

[25] J. Kitzes, D. Turek, F. Deniz, eds., The Practice of Reproducible Research: Case Studies and Lessons from the Data-Intensive Sciences, University of California Press, Oakland, CA, 2017. https://www.gitbook.com/book/bids/the-practice-of-reproducible-research/details (accessed July 5, 2017).

[26] N. Barnes, Publish your computer code: it is good enough, Nature News. 467 (2010) 753–753. doi:10.1038/467753a.

[27] D. Heaton, J.C. Carver, Claims about the use of software engineering practices in science: A systematic literature review, Information and Software Technology. 67 (2015) 207–219. doi:10.1016/j.infsof.2015.07.011.

[28] A. Bacchelli, C. Bird, Expectations, outcomes, and challenges of modern code review, in: 2013 35th International Conference on Software Engineering (ICSE), 2013: pp. 712–721. doi:10.1109/ICSE.2013.6606617.

[29] M. Petre, G. Wilson, Code Review For and By Scientists, arXiv:1407.5648 [Cs]. (2014). http://arxiv.org/abs/1407.5648 (accessed July 5, 2017).

[30] J. Bryan, L. Nederbragt, G. Wilson, Good Enough Practices in Scientific Computing, (2015).

[31] G.K. Sandve, A. Nekrutenko, J. Taylor, E. Hovig, Ten Simple Rules for Reproducible Computational Research, PLOS Computational Biology. 9 (2013) e1003285. doi:10.1371/journal.pcbi.1003285.

[32] Open Knowledge International, Data Packages, (2017). http://frictionlessdata.io/data-packages/ (accessed July 10, 2017).

[33] Open Source Initiative, Report of License Proliferation Committee, (2006). https://opensource.org/proliferation-report (accessed July 5, 2017).

[34] T. Jaeger, A. Metzger, Open Source Software: rechtliche Rahmenbedingungen der Freien Software, 4. Auflage, C.H. Beck, München, 2016.

[35] H. Meeker, Open source for business: a practical guide to open source software licensing, 2nd ed, CreateSpace Independent Publishing Platform, North Charleston, South Carolina, USA, 2017.

[36] J. Lerner, J. Tirole, The Scope of Open Source Licensing, JLEO. 21 (2005) 20–56. doi:10.1093/jleo/ewi002.

[37] G. Hagedorn, D. Mietchen, R. Morris, D. Agosti, L. Penev, W. Berendsohn, D. Hobern, Creative Commons licenses and the non-commercial condition: Implications for the re-use of biodiversity information, ZooKeys. 150 (2011) 127–149. doi:10.3897/zookeys.150.2189.

[38] GitHub, Choose an open source license, Choose a License. (2017). https://choosealicense.com/ (accessed July 6, 2017).

[39] J.F. DeCarolis, K. Hunter, S. Sreepathi, The case for repeatable analysis with energy economy optimization models, Energy Economics. 34 (2012) 1845–1853. doi:10.1016/j.eneco.2012.07.004.

[40] B. Meindl, M. Templ, Analysis of commercial and free and open source solvers for linear optimization problems, 2012. http://neon.vb.cbs.nl/cascprivate/..%5Ccasc%5CESSNet2%5Cdeliverable_solverstudy.pdf (accessed July 6, 2017).

[41] DECC, 2050 Pathways - Detailed guidance, (2013). https://www.gov.uk/guidance/2050-pathways-analysis (accessed March 14, 2016).

[42] T. Boßmann, I. Staffell, The shape of future electricity demand: Exploring load curves in 2050s Germany and Britain, Energy. 90, Part 2 (2015) 1317–1333. doi:10.1016/j.energy.2015.06.082.

[43] J.D. Blischak, E.R. Davenport, G. Wilson, A Quick Introduction to Version Control with Git and GitHub, PLOS Computational Biology. 12 (2016) e1004668. doi:10.1371/journal.pcbi.1004668.

[44] E. Hart, P. Barmby, D. LeBauer, F. Michonneau, S. Mount, P. Mulrooney, T. Poisot, K.H. Woo, N. Zimmerman, J.W. Hollister, Ten simple rules for digital data storage, PeerJ Preprints, 2016. doi:10.7287/peerj.preprints.1448v2.

[45] Open Power System Data, Open Power System Data – A platform for open data of the European power system., (2017). http://open-power-system-data.org/ (accessed July 6, 2017).

[46] OvGU, OpenEnergy Database (OEDB), (2017). http://oep.iks.cs.ovgu.de/dataedit/ (accessed July 18, 2017).

[47] J. Bacon, The Art of Community: Building the New Age of Participation, 2 edition, O'Reilly Media, Sebastopol, CA, 2012.

[48] S.P. Vera, GridCal, a power systems solver written in Python with user interface and embedded python console, 2017. https://github.com/SanPen/GridCal (accessed July 7, 2017).

[49] S. Hilpert, C. Kaldemeyer, U. Krien, S. Günther, C. Wingenbach, G. Plessmann, The Open Energy Modelling Framework (oemof) - A novel approach in energy system modelling, (2017). doi:10.20944/preprints201706.0093.v1.

[50] L. Thurner, A. Scheidler, Julian Dollichon, F. Schäfer, J.-H. Menke, F. Meier, S. Meinecke, others, pandapower - Convenient Power System Modelling and Analysis based on PYPOWER and pandas, University of Kassel and Fraunhofer Institute for Wind Energy and Energy System Technology, 2017. http://pandapower.readthedocs.io/en/v1.3.1/_downloads/pandapower.pdf.

[51] R.W. Lincoln, Learning to trade power, Ph.D., University of Strathclyde, 2011. http://oleg.lib.strath.ac.uk:80/R/?func=dbin-jump-full&object_id=15644 (accessed July 7, 2017).

[52] T. Brown, J. Hörsch, D. Schlachtberger, PyPSA: Python for Power System Analysis, Forthcoming. (2017).

[53] F. Wiese, G. Bökenkamp, C. Wingenbach, O. Hohmeyer, An open source energy system simulation model as an instrument for public participation in the development of strategies for a sustainable future, WIREs Energy Environ. 3 (2014) 490–504. doi:10.1002/wene.109.

[54] C. Kaldemeyer, M. Söthe, S. Hilpert, C. Wingenbach, renpassG!S - A Free and Open Python Tool for Simulating Energy Supply Systems, OSF. (2016). doi:10.17605/OSF.IO/KP4MH.

[55] C. Matke, W. Medjroubi, D. Kleinhans, S. Sager, Structure Analysis of the German Transmission Network Using the Open Source Model SciGRID, in: Advances in Energy System Optimization, Birkhäuser, Cham, 2017: pp. 177–188. doi:10.1007/978-3-319-51795-7_11.

[56] H.E. Daly, B. Fais, UK TIMES Model Overview, UCL Energy Institute, 2014. https://www.ucl.ac.uk/energy-models/models/uktm-ucl/uktm-documentation-overview (accessed July 7, 2017).

[57] I. Staffell, S. Pfenninger, Using bias-corrected reanalysis to simulate current and future wind power output, Energy. 114 (2016) 1224–1239. doi:10.1016/j.energy.2016.08.068.

[58] S. Pfenninger, I. Staffell, Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data, Energy. 114 (2016) 1251–1265. doi:10.1016/j.energy.2016.08.060.

[59] HM Treasury, The Aqua Book: guidance on producing quality analysis for government, HM Government, London, UK, n.d. https://www.gov.uk/government/publications/the-aqua-book-guidance-on-producing-quality-analysis-for-government (accessed July 10, 2017).

[60] P. Sargent, How government uses energy models, in: Imperial College London, 24 May 2016, 2016.

[61] R.D. Zimmerman, C.E. Murillo-Sanchez, R.J. Thomas, MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education, IEEE Transactions on Power Systems. 26 (2011) 12–19. doi:10.1109/TPWRS.2010.2051168.

[62] G. Varoquaux, Software for reproducible science: let's not have a misunderstanding, Computer / Data / Brain Science. (2015). http://gael-varoquaux.info/programming/software-for-reproducible-science-lets-not-have-a-misunderstanding.html (accessed June 1, 2016).

[63] G. Pleßmann, M. Erdmann, M. Hlusiak, C. Breyer, Global Energy Storage Demand for a 100% Renewable Electricity Supply, Energy Procedia. 46 (2014) 22–31. doi:10.1016/j.egypro.2014.01.154.

[64] G. Pleßmann, P. Blechinger, How to meet EU GHG emission reduction targets? A model based decarbonization pathway for Europe's electricity supply system until 2050, Energy Strategy Reviews. 15 (2017) 19–32. doi:10.1016/j.esr.2016.11.003.

[65] S. Pfenninger, P. Gauché, J. Lilliestam, K. Damerau, F. Wagner, A. Patt, Potential for concentrating solar power to provide baseload and dispatchable power, Nature Climate Change. 4 (2014) 689–692. doi:10.1038/nclimate2276.