Lifting Industrial Ecology Modeling to a New Level of Quality and Transparency

A Call for More Transparent Publications and a Collaborative Open Source Software Framework

Stefan Pauliuk, Guillaume Majeau-Bettez, Christopher L. Mutel, Bernhard Steubing, and Konstantin Stadler

Keywords:
industrial ecology
input-output analysis (IOA)
LCA software
open source software
scientific computing
transparency and reproducibility

Supporting information is available on the *JIE* Web site

Summary

Industrial ecology (IE) is a maturing scientific discipline. The field is becoming more data and computation intensive, which requires IE researchers to develop scientific software to tackle novel research questions. We review the current state of software programming and use in our field and find challenges regarding transparency, reproducibility, reusability, and ease of collaboration. Our response to that problem is fourfold: First, we propose how existing general principles for the development of good scientific software could be implemented in IE and related fields. Second, we argue that collaborating on open source software could make IE research more productive and increase its quality, and we present guidelines for the development and distribution of such software. Third, we call for stricter requirements regarding general access to the source code used to produce research results and scientific claims published in the IE literature. Fourth, we describe a set of open source modules for standard IE modeling tasks that represent our first attempt at turning our recommendations into practice. We introduce a Python toolbox for IE that includes the life cycle assessment (LCA) framework Brightway2, the ecospold2matrix module that parses unallocated data in ecospold format, the pySUT and pymrio modules for building and analyzing multiregion input-output models and supply and use tables, and the dynamic_stock_model class for dynamic stock modeling. Widespread use of open access software can, at the same time, increase quality, transparency, and reproducibility of IE research.

Introduction

Computer Tools in Modern Science

The omnipresence of computer-based analysis in modern science comes with two challenges, which are the subject of this article. (1) Where science is based on computer models, the existing software infrastructure determines the range of research questions that can be tackled. Consequently, cutting-edge research requires software development, which, in turn, requires special skills and is time-consuming. Researchers are rarely software specialists, and their lack of knowledge may lead to suboptimal results, such as low readability and reusability of code (Hannay et al. 2009). (2) Published results and scientific claims must be reproducible and transparent. To ensure this, scientists have agreed on the general rules of good scientific record keeping (Macrina 2005) and on the right to give peers

Address correspondence to: Konstantin Stadler, Industrial Ecology Programme, NTNU; NO-7491 Trondheim, Norway. *Email*: konstantin.stadler@ntnu.no

© 2015 by Yale University
DOI: 10.1111/jiec.12316

Editor managing review: Manfred Lenzen

Volume 19, Number 6

access to raw data upon request (Ince et al. 2012). There is no agreement, however, on whether the software used to generate scientific results should be published along with the results. Withholding source code makes it more difficult to reproduce results, and several proposals for higher software availability have already been made (Peng 2011; Ince et al. 2012; Hanson et al. 2011).

Computer Tools in Industrial Ecology

Industrial ecology (henceforth IE) is a rapidly advancing field. Recent progress includes the compilation of an increasing number of databases (ecoinvent Center 2014; Lenzen et al. 2013; SERI/WU 2014; Tukker et al. 2013), use of increasingly sophisticated quantitative analysis and assessment methods (e.g., Hertwich et al. 2015), establishment of research paradigms (Pauliuk and Hertwich 2015; Ehrenfeld 2004), or development of common accounting frameworks (Suh et al. 2010; Fischer-Kowalski et al. 2011; Pauliuk et al. 2015a). Models currently used in IE require more data and collaboration than ever before. Has the development of software tools kept pace? How has the IE community responded to the two challenges above? To our knowledge, these issues have not yet been addressed in the IE literature. We attempt to answer these questions for our area of expertise, which comprises life cycle assessment (LCA), input-output (I-O) analysis, and dynamic material (MFA) or substance flow analysis (SFA), three core methods of IE. We focus on software used to tackle data- and computation-intensive research questions, that is, cases where simple spreadsheets are not enough.

Life Cycle Assessment

State-of-the-art LCA studies are most affected, because of the large number of processes in the background system (more than 10,000 for ecoinvent 3 [ecoinvent Center 2014]). Tools for LCA can be divided into closed-source commercial software (e.g., GaBi [GaBi 2014], SimaPro [Goedkoop et al. 2008], Umberto [ifu Hamburg 2014], and Aveny [Aveny GmbH 2014]), closed source freeware (e.g., CMLCA [Heijungs 2012]), and open source freeware (e.g., OpenLCA [OpenLCA 2014]). Designed mainly for industrial users, the commercial tools are essential to mainstreaming the practical application of LCA. Often, however, they prove to be too inflexible to serve the needs of cutting-edge research. This includes questions requiring region-specific emissions data and impact factors (Mutel et al. 2012; Cherubini et al. 2012; Saner et al. 2014; Geyer et al. 2010; Verones et al. 2012), dynamic LCA and temporal emission profiles of product systems (Levasseur et al. 2013; Guest et al. 2013; Beloin-Saint-Pierre et al. 2014), large-scale prospective assessments and hybridization (Hertwich et al. 2015; Lenzen and Crawford 2009), efficient uncertainty analysis (Groen et al. 2014), especially with respect to allocation choices (Jung et al. 2013; Majeau-Bettez et al. 2014*b*), and the handling of very large, but fragmented and incompatible, data sets (Lenzen et al. 2014; Suh 2005, 2009; Lenzen et al. 2013). Many leading groups therefore undertake their own programming activities, and the code is usually written by LCA, rather than software, specialists.

Despite repeated calls for higher transparency and reproducibility (Frischknecht 2004; Finnveden et al. 2009; Finnveden and Ekvall 1998), most LCAs are not sufficiently transparent to allow for meaningful comparisons across studies or for the harmonization of their system definitions (Price and Kendall 2012; Burkhardt et al. 2012; Kim and Wallington 2013). For many studies, raw data are not published, or not published in a convenient format, which makes it impossible to reproduce the analysis. This has several reasons: (1) It is difficult to transparently and comprehensively document LCA analyses with commercial software; (2) there is a lack of transparency surrounding critical steps performed by database providers, such as the matching of characterization factors to emissions, or allocation; and (3) incentives for more transparent inventories are low. Moreover, recent analysis has shown that the choice of LCA software can impact assessment results (Speck et al. 2015; Herrmann and Moltesen 2015). As a result, many LCA studies are hard to compare, and they lack credibility if evaluated by high scientific standards.

Input-Output

Specific software is needed for most I-O models because of the large number of system variables. Aside from play models for illustrative purposes, the number of sectors in the system typically lies between several hundred (single-region I-O) and 10^4 to 10^5 (for integrated hybrid models such as THEMIS or the Eora MRIO table [Hertwich et al. 2015; Lenzen et al. 2013]). I-O modeling requires many intermediate steps that require software, including balancing algorithms, trade linking tools for multiregion input-output (MRIO), constructs to build I-O models from supply and use tables (SUTs), and aggregation/disaggregation routines (Wood 2015; Lenzen et al. 2009; Majeau-Bettez et al. 2014*b*; Miller and Blair 2009). Data processing and analysis scripts, however, are not generally made public, and we only know of two exceptions: De Koning and colleagues (2015) and CIRAIG (2015). A recent comparative study of six major MRIO frameworks by Lutter and Giljum (2014, 7) finds a general lack of transparency: "Procedures for manipulating IO tables, e.g. for disaggregating existing tables or harmonizing IO tables from different national sources, [are] often not well documented." This is a problem, given that these models are rapidly gaining relevance in climate and resource policy making.

Material Flow Analysis

The complexity of systems studied in MFA has increased significantly over the last few years. Systems containing between 1 and approximately 25 processes are often visualized with the freeware application STAN, which can also handle data reconciliation and uncertainty propagation (Vienna University of Technology 2012). Specialized software is indispensable for more comprehensive systems, including dynamic MFA and dynamic stock models (Hatayama et al. 2010; Modaresi et al. 2014; Elshkaki and Graedel 2013; Wiedenhofer et al.

2013; Müller 2006; Daigo et al. 2014; Busch et al. 2014), MFA models of complex supply chains (Wiedmann et al. 2013; Nakamura and Nakajima 2006; Nakamura et al. 2014; Kagawa et al. 2014), and quantification of stocks with high spatial resolution (Takahashi et al. 2009). Transparency of MFA and SFA studies is enhanced by explicit system definitions that accompany many publications. Comprehensive data and intermediate results are often not published, which makes many results opaque and hard to reproduce.

Model Combinations

Next to models that strictly follow one of the traditions discussed above, there are an increasing number of extensions and combinations of methods, such as hybrid LCA (Suh 2004; Suh et al. 2004; Heijungs and Suh 2002; Gibon et al. 2014), connections between MFA and I-O (Nakamura and Nakajima 2006; Nakamura et al. 2011; Kytzia et al. 2004; Lenzen and Reynolds 2014), and dynamic stock modeling (Pauliuk et al. 2015b; Nakamura et al. 2014). With the exception of CMLCA (Heijungs 2012), which can handle hybrid LCA, and the IELab (Lenzen et al. 2014), which can reconcile large, partially conflicting data sets, no specific software for any of these models is generally available.

Databases

Databases for IE are further developed than is the case for model software. Existing databases are accepted and widely used by the community; they are comprehensive and often well documented. Examples for such databases include ecoinvent for LCA (Frischknecht et al. 2005; Weidema et al. 2013); the Eora world MRIO model (Lenzen et al. 2013), IELab (Lenzen et al. 2014), or EXIOBASE (Wood et al. 2014) for I-O, as well as a global database of materials flows for MFA (SERI/WU 2014). Collaborative data frameworks have been proposed by several researchers (Davis et al. 2010; Lenzen et al. 2014).

Research Gap and Scope of the Article

1. **Level of software development:** The understanding of the importance of good software is ubiquitous within IE, but this is rarely reflected in common practice of our field. No widely accepted, readily available implementations of many common computational routines exist. At present, most IE models are coded in spreadsheets or form monolithic blocks of software in various programming languages. They are developed as in-house projects for single-case studies. Often, the quality of documentation does not match the complexity of the code. In many cases, the code is difficult to reuse and is therefore abandoned. We know of several instances where this led to double work. Professional software tools exist for specific LCA and MFA applications, but it is hard or impossible to adapt them to new types of research questions. There is no common public software repository for IE models to which every researcher can contribute.

   We see lifting IE software and the way it is used to a higher level as a key strategy to successfully mastering future challenges such as model integration, closer collaboration across research groups, and better interaction with neighboring fields, including integrated assessment, climate, and computable general equilibrium modeling.

2. **Level of software openness:** To our knowledge, there is neither an established standard nor a vivid debate regarding transparency and reproducibility of computations behind published quantitative research conducted under the label IE. A general lack of reproducibility may lower the scientific quality of the field as a whole, which, in the long run, can impede interaction with other scientific fields and acquisition of research funding. Low levels of reproducibility and transparency exclude noninsiders from verifying the conclusions drawn, which can undermine the credibility of our research. Given that policy guidance is a main motivation of many studies that use IE models (e.g., Wiedmann et al. 2011; Tukker et al. 2013; Wiedmann and Barrett 2013), this lack of credibility must be considered problematic in a democratic society. This becomes even more apparent in light of the proposed role of scientists as "honest brokers" (Pielke Jr 2007) and as information providers for evidence-informed policy development (Rose 2014).

In this article, we present our efforts to respond to the two challenges. First, we propose guidelines for how existing general principles for the development of good scientific software (Dubois 2005; Wilson et al. 2014) could be implemented in IE and related fields. Second, we propose guidelines for the development of open access software for IE, and, based on Peng (2011) and Ince and colleagues (2012), we argue that the IE community should set higher requirements regarding access to source code behind scientific computations. Finally, we present our first attempt at "walking the talk": We describe a set of open source modules for standard IE modeling tasks that are written in the general purpose language, Python. We present each topic in a separate section and discuss possible disadvantages and options for future development at the end.

Guidelines for Developing, Testing, and Documenting Software Tools in Industrial Ecology

We present guidelines for how IE software should be developed and organized to facilitate correctness and reusability. These guidelines are independent of the programming language chosen. Nor are they affected by the decision on whether to share the software with others. Our suggestions extend existing guidelines for best practice in scientific coding (Wilson et al. 2014) and for maintaining correctness in scientific

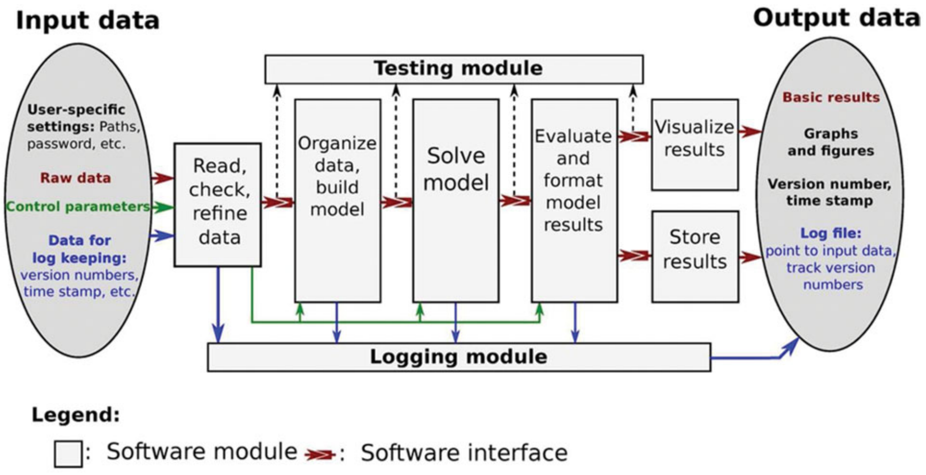
Input data
User-specific settings: Paths, password, etc.
Raw data
Control parameters
Data for log keeping: version numbers, time stamp, etc.
Testing module
Read, check, refine data
Organize data, build model
Solve model
Evaluate and format model results
Visualize results
Store results
Logging module
Output data
Basic results
Graphs and figures
Version number, time stamp
Log file: point to input data, track version numbers
Legend:
Software module
Software interface

Figure 1 Computational blocks of the scientific software for a typical model run in industrial ecology. Not all elements are always present, but a more elaborate structure, especially with logging and testing modules, is important for complex projects.

software (Dubois 2005). Our guidelines apply to software that is structured according to the scheme in figure 1, which we consider to be the general structure of computations in IE.

The software framework depicted in figure 1 breaks up the data flow into specific functional blocks. Software should be modular; this increases the clarity and reusability of the different components. Subsequent reuse for similar purposes should be considered. There should be a dedicated section or software module for each block in the diagram in figure 1. For example, data should be read by a designated input module, in contrast to the common practice of reading data whenever they are needed. Control parameters should not be hard coded in the model, but defined in data files, given that hard-coded parameters are prone to yield mistakes when modified and forgotten, especially when they enter the code at different places.

Confidence in final results is heightened by systematic and transparent quality checks. *Testing routines should therefore be an integral part of scientific IE software development*. Unit tests check whether each unit of code (usually each function) performs as intended. Unit tests include test cases that cover the typical use of a function, as well as special cases and situations that should raise errors (Beck 2003). These tests should always accompany the code and be rerun when the module is modified to ensure that the intended functionality is maintained. In contrast, *system tests*—such as performance, compatibility, load, and usability testing—are harder to automate. They verify that the software as a whole meets functional requirements and are therefore interesting for more mature IE projects aiming for mainstream use.

The next guideline is the key to successful modularization and reuse of the code: Development and documentation of the interfaces between software blocks is of central importance, as is the documentation of the functions of the code within the blocks. Documentation should cover the function of the code, not only its structure (Wilson et al. 2014). Moreover, good interfaces facilitate the use of different programming languages.

Once a module works correctly and is well documented, it should be formally released under a version number: *The different blocks of a model should be subject to a version control system to keep track of modifications*. Version control systems record the entire history of a piece of software. This is essential if past results based on older versions are to be reproduced, and it facilitates the consolidation of simultaneous contributions by multiple developers or research groups. Examples for distributed version control software can be found in the supporting information available on the Journal's website.

Finally, we need to keep track of the computations of the entire software to make sure that the results can be understood and reproduced, even years after the model was run: *Each proper model run should be documented in a log file that is stored together with the results*, *which traces the flow of computation from the input data to the results*. The log file should contain information such as a time stamp, the source location and version numbers of the input data and the modules used, the status or error messages of the different modules called, and key parameters and user choices defined during the model run.

The above recommendations are consensual and do not require much motivation. To develop a sophisticated piece of software following these rules, the programming community has articulated many complementary or partly overlapping software development strategies to efficiently reach high levels of quality. Notable examples include: the traditional waterfall model; test-driven development; extreme programming; and multiple agile development strategies (Beck 1999). We refrain from recommending a specific methodology to the IE community, given that the choice of an optimal strategy is largely project and resource dependent. We do recommend, however, that groups investing resources in software development apply a specific strategy for this process.

Guidelines for Collaborative Software Projects Within Industrial Ecology and a Call for More Transparency of Published Results

The benefits of pursuing high-quality standards for scientific software are hardly debatable. More sensitive is the question of whether or not to share software with peers and have them use and improve it. We argue that, in many cases, considerable synergies may arise by combining the pursuit of high-quality software and the willingness to publish it open source.

The first type of synergy concerns the productivity of IE research: Open source software can help the whole IE community to move faster and produce higher-quality work with less effort. Collaborative software projects encourage researchers to interact; they can also initiate scientific collaboration beyond mere programming. Contributing to such software increases a researcher's visibility. An example from another scientific field is the Coupled Model Intercomparison Project (CMIP), which includes a standard experimental protocol for studying the output of coupled atmosphere-ocean general circulation models. CMIP provides a community-based infrastructure in support of climate model diagnosis, validation, intercomparison, documentation and data access. Virtually the entire international climate modeling community has participated in this project since its inception in 1995 (WCRP 2015). Similar efforts are known from neuroscience (Hines et al. 2009), bioinformatics (Hamelryck and Manderick 2003), and astrophysics (Robitaille et al. 2013).

The second type of synergy concerns the reproducibility and transparency of published numerical results and the scientific claims based on them (Easterbrook 2014). If authors of scientific articles were required to publish the code they used to analyze the data and present the results, their work would gain substantial credibility, and the community could make much more use of it (Ince et al. 2012; Peng 2011). These obvious benefits of open scientific software have started to change journal policies regarding access to code. For example, in 2011, *Science* announced that it would extend its "data access requirement [...] to include computer codes involved in the creation or analysis of data" (Hanson et al. 2011, 649). In October 2014, *Nature* released an editorial on code sharing and announced that "editors [...] reserve the right to decline a paper if important code is unavailable" (Nature 2014, 536).

We believe that it is time for the IE community to embark on the discussion about open source software. To feed this debate, we propose guidelines for the development of open source software and more transparent publishing.

Guidelines for the Development and Deployment of Open Access Software in Industrial Ecology

Our first claim is that each piece of software that transforms a well-defined input into a well-defined output and that has potential for reuse is worth publishing under an open source license. That includes computation routines that implement a published model, but also routines for data parsing and formatting as well as visualization scripts.

Any published software code can be studied, used, and modified for private purposes, without limitations. If the modifications are to be published and redistributed, however, the original software needs to be published under a specific open source license, because default copyright rules would apply if this was not done (WIPO 2009). The open source license determines the type of applications the code can be used for; it therefore plays a crucial role in the success of the open source strategy. A large number of open source licenses exist, but they follow two basic patterns. (1) *Copyleft licenses* mandate modified code to be published under the same open source license. The most prominent example is the GNU General Public License (GPL), which requires that all modifications and improvements of open source software published under the GPL be published under the same license. This provides an incentive for further open source development, and thus it discourages commercial use (Free Software Foundation 2007). (2) *Permissive licenses*, such as the BSD and MIT licenses (Open Source Initiative 1988; The FreeBSD Project 1992), only have minimal constraints for how the software can be used, and their use may increase chances that the code is used for both scientific and commercial applications. In the Supporting Information on the Web, we provide links to different license guides. We assert that *the selection of the open source license has important implications for the success of the project and potential future commercial use*.

We continue by arguing that in order to avoid fragmentation, practitioners should make an effort to reuse existing code—expanding upon existing projects or creating derivatives—rather than starting from scratch and isolating their work from the rest of the IE code base. Interfaces to other modules should be built and published alongside with the actual functionality. Good documentation, examples, and tutorials should come along with the code.

Publishing scientific software under an open source license is no guarantee of better software, but it carries a large potential. It turns a piece of software into a publication in its own right, which encourages the publisher to properly modularize, test, document, and comment the software. This alone will raise practice standards by making the code better structured and less prone to mistakes. We believe that following the proposed guidelines will help to make open source publishing of code beneficial to the community as a whole, including the author of the code.

More Transparency and Reproducibility from Mandatory Code Sharing

In response to the increasing demand for publicly available software to reproduce scientific claims, the IE community should consider to set higher requirements regarding the provision of raw data and scientific software along with peer-reviewed publications. We believe that the editors of journals that publish IE research as well as the reviewers are key players in this process. They should consider asking authors for a major

Guidelines for software development	Guidelines for software sharing
• Build modular software and consider how it could be re-used. • Develop well-defined interfaces between modules and provide extensive documentation on modules and interfaces. • Perform unit-tests on all independent modules. • Use version control to keep the overview as the software modules change. • Apply logging routines to facilitate high transparency within a model run.	• Every piece of code that has well-defined functionality and interfaces and that can potentially be re-used is worth publishing. • Deliberate choice of the open source license is essential to the success of the software project. • Version control is central to keeping the overview as the model changes and to facilitate parallel development in different institutions. • Expand or modify existing code rather than start over. • Integrate your work: Build interfaces, provide documentation, tutorials, and examples.

Guidelines for publications and transparency
• Identify and exploit mutual benefits between open source software and reproducible, transparent scientific claims. • Electronic laboratory notebooks could become a standard to track computations. • Journal editors: Encourage higher standards for data and model transparency in peer-reviewed publications.

Figure 2 Summary of our proposed guidelines for software development and sharing. These guidelines were developed for industrial ecology, but they also comprise other fields where the typical program structure follows the flow diagram in figure 1.

revision of manuscripts whose claims are not reproducible because of lack of documentation, data, or software. This would encourage authors to document their scientific claims with higher transparency and reproducibility.

Laboratory notebooks are a basic element of scientific work in many established fields of science. To increase the reproducibility of scientific computations in IE, the IE community should consider promoting the preparation of electronic lab books that track computations (Mutel and Müller 2013; Shen 2014).

A general requirement that scientific claims of IE publications be fully reproducible would be a strong incentive for a more open software culture. Authors could adhere to the requirement by not only publishing the essential scientific software along with peer-reviewed publications, but also by modularizing and releasing the software under an open source license with all benefits this brings about. On the other hand, if scientific software is already released open source, an author writing an article could merely point to a set of software modules and their version numbers. This could be recognized as a sufficient level of reproducibility by the community. Open source software and reproducible scientific claims are two sides of the same coin. We close our explanations by summarizing our recommendations on software development, sharing, and publication in tabular form (figure 2).

The Python Toolbox for Industrial Ecology

There are many ways of turning the general principles into practice, and we refrain from promoting a "right way" of doing this. Starting points for the choice of open source programming frameworks can be found in the Supporting Information on the Web. Here, we present the results of an effort made during recent years and invite others to use the tools and to share their own developments with the community. Using the general purpose language, Python, we developed and published a number of software modules to handle a large number of typical tasks in the main fields of quantitative IE research: LCA, I-O, and MFA/SFA. Though developed mostly independently of one another, we strived to make them compatible to one another by developing interfaces and making them adhere to common data formats. We first present the modules and then the connections between them (figure 3).

Life Cycle Assessment (Brightway2 and Extensions)

Brightway2 is a framework for LCA, covering everything from data I-O and processing to calculations and interpretation (Mutel 2014). The software itself is split into different modules, each with a specific focus and limited set of capabilities. In addition to the core components, extension modules provide user interfaces, regional and dynamic LCA, and data interfaces

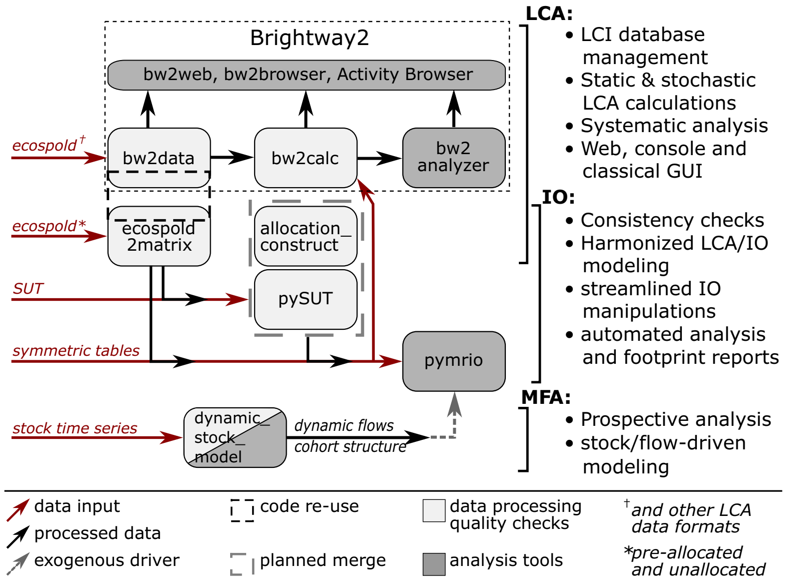
Brightway2
bw2web, bw2browser, Activity Browser
ecospold†
bw2data
bw2calc
bw2 analyzer
ecospold*
ecospold 2matrix
allocation_ construct
SUT
pySUT
symmetric tables
pymrio
stock time series
dynamic_ stock_ model
dynamic flows
cohort structure
LCA:
• LCI database management
• Static & stochastic LCA calculations
• Systematic analysis
• Web, console and classical GUI
IO:
• Consistency checks
• Harmonized LCA/IO modeling
• streamlined IO manipulations
• automated analysis and footprint reports
MFA:
• Prospective analysis
• stock/flow-driven modeling
data input
processed data
exogenous driver
code re-use
planned merge
data processing quality checks
analysis tools
†and other LCA data formats
*pre-allocated and unallocated

Figure 3 Relations between the different modules. We show input data (red), data flow between modules (black), and typical applications (right). From left to right, this figure follows the progression of figure 1 from raw data input to final analysis. SUT = supply and use table; LCA = life cycle assessment; IO = input-output; MFA = material flow analysis; LCI = life cycle inventory; GUI = graphical user interface.

for external programs. Brightway2 is extensively documented and is designed to be easy to use and adapt. An example for the application of Brightway2LCA is given by Mutel and colleagues (2012).

Brightway2LCA includes the following major features:

- Import of ecospold 1 and 2 (ecoinvent 2 and 3) and SimaPro files and export to multiple formats
- Very fast static and stochastic LCA calculations and supply-chain graph traversal
- Web-based and classical graphical user interfaces (GUIs) enable data management as well as visualization of LCA results including networks, Hinton matrices, and treemaps.

The Activity Browser is a free open source LCA software that builds upon brightway2 (Steubing 2014). It provides a GUI to brightway2 and enables, among others, a browser-inspired navigation through LCA databases, the creation and modification of activities and databases, as well as fast LCA calculations for multiple functional units and impact assessment methods at a time. The Activity Browser also includes an extension for a meta-level modeling of life cycle inventories (LCIs) that enables grouping several processes into so-called metaprocesses, which can be linked based on user-defined product names to screen alternative life cycles and which provide the basis for optimization problems (Steubing 2015).

Matrix System Representations (ecospold2matrix)

Ecospold2matrix (Majeau-Bettez 2014a) is partly based on the ecospold-parsing functionality of Brightway2, but it is much simpler and more narrow in scope. To help bridge the gap between typical LCA and environmentally extended input-output workflows, this module reorganizes the ecoinvent 3 database as a collection of matrices. It can notably:

- assemble the unallocated dataset in a supply and use table framework (see pySUT).
- perform basic quality checks on the preallocated data sets and arrange them as Leontief technical coefficient matrices with environmental extensions.
- optionally change sign conventions for waste flows and properties to align with the waste input-output (WIO) model (Nakamura and Kondo 2002).

Supply and Use Tables (pySUT), Allocations, and Constructs

SUTs are a widely used accounting framework for flows within society's metabolism and its exchange with nature (Miller and Blair 2009; European Commission 2008; Schmidt et al. 2010; Majeau-Bettez et al. 2014a). SUTs are the starting point for the construction of symmetric input-output tables (IOTs) (United Nations 1999; Lenzen and Reynolds 2014; Lenzen and Rueda-Cantuche 2012).

The open source python class, pySUT, provides the following group of routines common to the System of National Accounts and I-O modeling (Pauliuk 2014a):

- Balance checks, including the market and industry balances
- Aggregation and resorting of products, industries, and regions
- Application of allocations and constructs to build Leontief-**A**-matrices and extensions.

The LCA community has articulated the treatment of multifunctional activities in terms of allocation models, whereas the

I-O community treats coproducing industries with construct models (Heijungs and Suh 2002). The allocation_construct.py module (Majeau-Bettez 2014b) reflects the recent harmonization of these modeling practices (Suh et al. 2010) and offers the functionality of both. It applies the general and specific allocations and constructs that follow the framework developed by Majeau-Bettez and colleagues (2014b).

Multiregional Input-Output Analysis (pymrio)

Approximately half a dozen environmentally extended MRIO tables were published over the last 2 years (Tukker and Dietzenbacher 2013). Most of these tables are freely available. Each model has its own file format, classification, and indexing; efficient handling and analyzing of MRIO models therefore requires a certain degree of training. The pymrio module (Stadler 2014a) allows for easy handling of global MRIO models. It provides a comprehensive, well-documented (Stadler 2014b) set of commands for manipulating and analyzing (MR)IO tables, including:

- Parsing global MRIO tables
- Modifying region and sector classification
- Restructuring extensions
- Calculating various accounts (footprint, territorial, impacts embodied in trade)
- Exporting to various formats (csv, html, MS Excel)
- Visualization and automated report generation

Dynamic Stock Modeling (dynamic_stock_model)

Age-cohort–based models are standard in modeling of material, product, and capital stocks (Pauliuk et al. 2015b). The open source python class, dynamic_stock_model, provides three standard routines for dynamic stock modeling (Pauliuk 2014b):

- Inflow-driven model: (Van der Voet et al. 2002; Pauliuk et al. 2012; Nakamura et al. 2014; Fishman et al. 2014; Glöser et al. 2013).
- Stock-driven model: (Müller 2006; Modaresi and Müller 2012; Modaresi et al. 2014; Hatayama et al. 2010; Busch et al. 2014).
- Historic inflow from given age-structure of a stock. This method is used in cases where an initial stock is present, for example, when processing scenario results from the TIMER model (de Vries et al. 2001).

Connections Between the Different Modules

The different modules are related to one another in several ways (figure 3). SUTs are one central nexus: They are an output of ecospold2matrix and an input to pySUT and allocation_construct. Leontief IOTs (for I-O and LCA) are the other nexus; they are an output of bw2data, ecospold2matrix, pySUT, and allocation_construct, as well as an input to bw2calc and pymrio. At present, there is no connection between the modules for LCA and I-O and the dynamic_stock_model class. This will change once dynamic stock models will be used as exogenous drivers for dynamic I-O models (Nakamura et al. 2014; Pauliuk et al. 2015b; Kagawa et al. 2014).

Access, Use, and Collaboration

This brief description should give an impression of the capability of the toolbox and describe how the different modules are connected. We refer to the specific documentations for further detail. The toolboxes can be downloaded from their respective home pages or from GitHub and Bitbucket. Most modules are published under the BSD or similar licenses, which allows third parties to use the code for commercial and noncommercial applications. The branching functionality of distributed version control allows for several versions to be modified in parallel. The modules follow our guidelines for good coding: They offer specific functions, and they come along with unit tests, logging routines (where applicable), and version control. They all come with an online documentation, are published under an open source license, and the software releases are version controlled.

Options for Future Development

Further development of the LCA, I-O, and MFA toolboxes is planned. There are many options for expansion, including:

- Standardized and transparent compilation and reconciliation of SUTs from primary data sources
- Implementation of tools for tiered hybrid, integrated hybrid, and WIO models (Heijungs and Suh 2002; Nakamura and Kondo 2002)
- Creation of good visualization and data handling routines
- Dynamic-stock driven LCA/I-O analyses
- Structural analysis of MRIO systems

We invite interested scholars to join our efforts to create a more open, more professional toolbox for IE.

Another important step to create an active software developer community for IE is to provide good overview of the different tools, their capabilities, and interfaces between them. This will become necessary at some point and could happen through a dedicated website such as IndEcol-software-Wikibook, an open content online textbook that any user can edit (Wikimedia Foundation 2014).

Final Considerations

Concerns About Open Source Software

Can open source software also be commercially successful? How does it impact the dynamics of science, especially competition among researchers? How can researchers reconcile their wish to publish open source with their employers' regulations on intellectual property? These questions are the subject of a broad debate, and we restrict ourselves to providing a few arguments in favor of open source software: Publishing software under certain open source licenses (e.g., BSD) does not hinder commercialization of the software later on. Major information technology (IT) companies, including Red Hat Inc., Oracle Corporation, Google Inc, and Facebook Inc, publish the source code of their products or actively participate in open source projects. We believe that publishing open source software for IE does not preclude successful commercialization given that many end users in industrial or consulting businesses will prefer using high-end commercial software and related training services to working with the source code themselves. Neither does it hinder competition among researchers in the short run, given that a published software project may be split into different versions ("forked"), and developed in different directions by different research groups. It would impede, however, the development of large nontransparent software projects and the accumulation of knowledge biases over many years, as seen in the integrated assessment modeling community (Schneider 1997). Finally, from our experience, we can tell that universities generally accept the release of code or materials in support of research, provided that they benefit financially from successful commercialization.

On the Choice of Software

We have been careful not to make too specific recommendations regarding which language or development environment to use. We see this as a secondary issue, surpassed by the need to build well-structured, tested, and well-documented software. All available languages and tools have their advantages and disadvantages. In many cases, they can be embedded into one another, given that versatile interfaces between most languages exist.

An important issue, which requires careful deliberation, concerns the openness of the development environment. Whereas for many general purpose languages, including Java, Python, C++, or Fortran, compilers and integrated development environments are freely available under an open source license, this is not the case for proprietary languages such as Matlab or Mathematica. Thus, users of the latter languages are subject to vendor lock-in, which raises issues over the longevity of the code base and may be cost prohibitive for some institutions. This economic advantage and the independence of a specific provider should be taken into consideration when deciding which programming environment to use. We consider the use of a free and open programming environment to be more appropriate for an emerging research field with high ambitions and a strong footing in the developing world.

Conclusion

IE needs better software to mature further. The proposed guidelines for the development of more professional software for the IE community cover modularization, testing, interfaces, documentation, version control, and logging routines. To follow these guidelines, additional effort related to modularization, testing, and documentation is needed. According to our experience, however, this effort is promptly offset by the benefits of reusability and ease of collaboration.

Open source software for IE research has a large potential to make our field more productive and to produce higher-quality results. The proposed guidelines for open source IE software may help to realize this potential, and first steps in that direction have already been taken. It may take some time, however, until a comprehensive and mature open source software framework will be at disposal to the community.

Transparent and reproducible results are another aspect of a mature research field. Published IE research often falls short of the standards set in other fields, and the IE community should consider setting higher requirements regarding general access to software tools and/or documentation of data flows related to published IE research.

We identified considerable synergies between high-quality software, open source software, and reproducible results and believe that the recommendations given in the article will eventually help IE researchers make more solid contributions to understanding and solving the environmental challenges faced by modern society. Clearly, we are only at the beginning of such a process.

Acknowledgments

The work of Stefan Pauliuk and Konstantin Stadler was funded by the European Commission under the DESIRE Project (grant no.: 308552). The work of Bernhard Steubing was funded within the National Research Programme ≪Resource Wood≫ (NRP 66; www.nfp66.ch) by the Swiss National Science Foundation (project no.: 136612). The research was conducted without involvement of the funding sources. The authors thank two reviewers for their helpful feedback.

References

Aveny GmbH. 2014. Aveny LCA. www.aveny.ch/. Accessed 20 October 2014.

Beck, K. 1999. *Extreme programming explained: Embrace change*. Boston, MA, USA. Addison-Wesley.

Beck, K. 2003. *Test-driven development by example*. Boston, MA, USA: Addison-Wesley.

Beloin-Saint-Pierre, D., R. Heijungs, and I. Blanc. 2014. The ESPA (Enhanced Structural Path Analysis) method: A solution to an implementation challenge for dynamic life cycle assessment studies. *The International Journal of Life Cycle Assessment* 19(4): 861–871.

Burkhardt, J. J., G. Heath, and E. Cohen. 2012. Life cycle greenhouse gas emissions of trough and tower concentrating solar power electricity generation. *Journal of Industrial Ecology* 16(S1): S93–S109.

Busch, J., J. K. Steinberger, D. A. Dawson, P. Purnell, and K. E. Roelich. 2014. Managing critical materials with a technology-specific stocks and flows model. *Environmental Science & Technology* 48(2): 1298–1305.

Cherubini, F., R. M. Bright, and A. H. Strømman. 2012. Site-specific global warming potentials of biogenic CO_2 for bioenergy:

Contributions from carbon fluxes and albedo dynamics. *Environmental Research Letters* 7(4): 045902.

CIRAIG (The Internation Centre for the Life Cycle of Products, Processes and Services). 2015. Open IO-Canada: Open source Input-output LCA model and tool to estimate life cycle impacts of products and services. http://www.ciraig.org/en/open_io_canada/. Accessed 28 May 2015.

Daigo, I., S. Osako, Y. Adachi, and Y. Matsuno. 2014. Time-series analysis of global zinc demand associated with steel. *Resources, Conservation and Recycling* 82: 35–40.

Davis, C., I. Nikolic, and G. P. J. Dijkema. 2010. Industrial ecology 2.0. *Journal of Industrial Ecology* 14(5): 707–726.

Dubois, P. F. 2005. Maintaining correctness in scientific programs. *Computing in Science and Engineering* 7(3): 80–85.

Easterbrook, S. M. 2014. Open code for open science? *Nature Geoscience* 7(11): 779–781.

ecoinvent Center. 2014. ecoinvent version 3. Dübendorf, Switzerland: ecoinvent Center. www.ecoinvent.org/database/ecoinvent-version-3/. Accessed 10 May 2014.

Ehrenfeld, J. R. 2004. Industrial ecology: A new field or only a metaphor? *Journal of Cleaner Production* 12(8–10): 825–831.

Elshkaki, A. and T. E. Graedel. 2013. Dynamic analysis of the global metals flows and stocks in electricity generation technologies. *Journal of Cleaner Production* 59: 260–273.

European Commission. 2008. *Eurostat manual of supply, use and input-output tables, 2008 edition*. Luxembourg: Eurostat.

Finnveden, G. and T. Ekvall. 1998. Life-cycle assessment as a decision-support tool—The case of recycling versus incineration of paper. *Resources Conservation and Recycling* 24(3): 235–256.

Finnveden, G., M. Z. Hauschild, T. Ekvall, J. Guinée, R. Heijungs, S. Hellweg, A. Koehler, D. Pennington, and S. Suh. 2009. Recent developments in life cycle assessment. *Journal of Environmental Management* 91(1): 1–21.

Fischer-Kowalski, M., F. Krausmann, S. Giljum, S. Lutter, A. Mayer, S. Bringezu, Y. Moriguchi, H. Schütz, H. Schandl, and H. Weisz. 2011. Methodology and indicators of economy-wide material flow accounting. *Journal of Industrial Ecology* 15(6): 855–876.

Fishman, T., H. Schandl, H. Tanikawa, P. Walker, and F. Krausmann. 2014. Accounting for the material stock of nations. *Journal of Industrial Ecology* 18(3): 407–420.

Free Software Foundation. 2007. GNU general public license. www.gnu.org/copyleft/gpl.html. Accessed 28 October 2014.

Frischknecht, R. 2004. Transparency in LCA—A heretical request? *The International Journal of Life Cycle Assessment* 9(4): 211–213.

Frischknecht, R., N. Jungbluth, H. J. Althaus, G. Doka, R. Dones, T. Heck, S. Hellweg, et al. 2005. The ecoinvent database: Overview and methodological framework. *International Journal of Life Cycle Assessment* 10(1): 3–9.

GaBi. 2014. GaBi software—A product sustainability performance solution. www.ecoinvent.org/database/ecoinvent-version-3/. Accessed 24 January 2014.

Geyer, R., D. M. Stoms, J. P. Lindner, F. W. Davis, and B. Wittstock. 2010. Coupling GIS and LCA for biodiversity assessments of land use. *The International Journal of Life Cycle Assessment* 15(5): 454–467.

Gibon, T., E. G. Hertwich, R. Wood, J. Bergesen, and S. Suh. 2014. *A methodology for scenario analysis in hybrid input-output analysis: Case study on energy technologies*. Trondheim, Norway: NTNU.

Glöser, S., M. Soulier, and L. A. Tercero Espinoza. 2013. Dynamic analysis of global copper flows. Global stocks, postconsumer material flows, recycling indicators, and uncertainty evaluation. *Environmental Science & Technology* 47(12): 6564–6572.

Goedkoop, M., M. Oele, A. de Schryver, and M. Vieira. 2008. *SimaPro 7, database manual, methods library*. Amersfoors, the Netherlands: PRé Consultants.

Groen, E. A., R. Heijungs, E. A. M. Bokkers, and I. J. M. de Boer. 2014. Methods for uncertainty propagation in life cycle assessment. *Environmental Modelling & Software* 62: 316–325.

Guest, G., F. Cherubini, and A. H. Strømman. 2013. Global warming potential of carbon dioxide emissions from biomass stored in the anthroposphere and used for bioenergy at end of life. *Journal of Industrial Ecology* 17(1): 20–30.

Hamelryck, T. and B. Manderick. 2003. PDB file parser and structure class implemented in Python. *Bioinformatics* 19(17): 2308–2310.

Hannay, J. E., C. Macleod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson. 2009. How do scientists develop and use scientific software? In ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. SECSE '09, 23 May, Vancouver, British Columbia, Canada.

Hanson, B., A. Sugden, and B. Alberts. 2011. Making data maximally available. *Science* 331(11): 649.

Hatayama, H., I. Daigo, Y. Matsuno, and Y. Adachi. 2010. Outlook of the world steel cycle based on the stock and flow dynamics. *Environmental Science & Technology* 44(16): 6457–63.

Heijungs, R. 2012. CMLCA: Scientific software for LCA, IOA, EIOA, and more. www.cmlca.eu/. Accessed 10 October 2014.

Heijungs, R. and S. Suh. 2002. *Computational structure of life cycle assessment*. Dordrecht, the Netherlands: Kluwer Academic.

Herrmann, I.T. and A. Moltesen. 2015. Does it matter which Life Cycle Assessment (LCA) tool you choose? – a comparative assessment of SimaPro and GaBi. *Journal of Cleaner Production* 86: 163–169.

Hertwich, E. G., T. Gibon, E. A. Bouman, A. Arvesen, S. Suh, G. A. Heath, J. D. Bergesen, A. Ramirez, M. I. Vega, and L. Shi. 2015. Integrated life-cycle assessment of electricity-supply scenarios confirms global environmental benefit of low-carbon technologies. *Proceedings of the National Academy of Sciences of the United States of America* 112(20): 6277–6282.

Hines, M. L., A. P. Davison, and E. Muller. 2009. NEURON and Python. *Frontiers in Neuroinformatics* 3: 1.

ifu Hamburg. 2014. Umberto NXT LCA. www.umberto.de/en/. Accessed 20 October 2014.

Ince, D. C., L. Hatton, and J. Graham-Cumming. 2012. The case for open computer programs. *Nature* 482(7386): 485–488.

Jung, J., N. Assen, and A. Bardow. 2013. Sensitivity coefficient-based uncertainty analysis for multi-functionality in LCA. *The International Journal of Life Cycle Assessment* 19(3): 661–676.

Kagawa, S., S. Nakamura, Y. Kondo, K. Matsubae, and T. Nagasaka. 2014. Forecasting replacement demand of durable goods and the induced secondary material flows: A case study of automobiles. *Journal of Industrial Ecology* 19(1): 10–19.

Kim, H. C. and T. J. Wallington. 2013. Life-cycle energy and greenhouse gas emission benefits of lightweighting in automobiles: Review and harmonization. *Environmental Science & Technology* 47(12): 6089–6097.

Koning, A. De, G. Huppes, S. Deetman, and A. Tukker. 2015. Scenarios for a 2°C world: A trade-linked input-output model with high sector detail. *Climate Policy* 1–17.

Kytzia, S., M. Faist, and P. Baccini. 2004. Economically extended—MFA: A material flow approach for a better understanding of food production chain. *Journal of Cleaner Production* 12(8–10): 877–889.

Lenzen, M. and R. Crawford. 2009. The path exchange method for hybrid LCA. *Environmental Science & Technology* 43(21): 8251–8256.

Lenzen, M., B. Gallego, and R. Wood. 2009. Matrix balancing under conflicting information. *Economic Systems Research* 21(1): 23–44.

Lenzen, M., A. Geschke, T. O. Wiedmann, J. Lane, N. Anderson, T. Baynes, J. Boland, et al. 2014. Compiling and using input-output frameworks through collaborative virtual laboratories. *The Science of the Total Environment* 485–486C: 241–251.

Lenzen, M., D. D. Moran, K. Kanemoto, and A. Geschke. 2013. Building Eora: A global multi-region input-output database at high country and sector resolution. *Economic Systems Research* 25(1): 20–49.

Lenzen, M. and C. J. Reynolds. 2014. A supply-use approach to waste input-output analysis. *Journal of Industrial Ecology* 18(2): 212–226.

Lenzen, M. and J. M. Rueda-Cantuche. 2012. A note on the use of supply-use tables in impact analyses. *Statistics and Operations Research Transactions* 36(2): 139–152.

Levasseur, A., P. Lesage, M. Margni, and R. Samson. 2013. Biogenic carbon and temporary storage addressed with dynamic life cycle assessment. *Journal of Industrial Ecology* 17(1): 117–128.

Lutter, S. and S. Giljum. 2014. *Demand-based measures of material flows. A review and comparative assessment of existing calculation methods and data options*. Working Party on Environmental Information. Paris: OECD.

Macrina, F. L. 2005. Scientific record keeping. In *Scientific integrity*, 3rd ed., edited by F. L. Macrina (pp. 269–295). Washington, DC: ASM.

Majeau-Bettez, G. 2014a. ecospold2matrix: A Python class for recasting Ecospold2 LCA datasets into Leontief matrix representations or supply and use tables. https://github.com/majeau-bettez/ecospold2matrix. Accessed 14 November 2014.

Majeau-Bettez, G. 2014b. allocation_construct: A Python module for lifecycle assessment allocations and input-output constructs. https://github.com/majeau-bettez/allocation_construct. Accessed 17 November 2014.

Majeau-Bettez, G., R. Wood, E. G. Hertwich, and A. H. Strømman. 2014a. When do allocations and constructs respect material, energy, financial, and production balances in LCA and EEIO? *Journal of Industrial Ecology* Forthcoming. DOI: 10.1111/jiec.12273.

Majeau-Bettez, G., R. Wood, and A. H. Strømman. 2014*b*. Unified theory of allocations and constructs in life cycle assessment and input-output analysis. *Journal of Industrial Ecology* 18(5): 747–770.

Miller, R. E. and P. D. Blair. 2009. *Input-output analysis: Foundations and extensions*. New York: Cambridge University Press.

Modaresi, R. and D. B. Müller. 2012. The role of automobiles for the future of aluminum recycling. *Environmental Science & Technology* 46(16): 8587–8594.

Modaresi, R., S. Pauliuk, A. N. Løvik, and D. B. Müller. 2014. Global carbon benefits of material substitution in passenger cars until 2050 and the impact on the steel and aluminum industries. *Environmental Science & Technology* 48(18): 10776–10784.

Müller, D. B. 2006. Stock dynamics for forecasting material flows—Case study for housing in the Netherlands. *Ecological Economics* 59(1): 142–156.

Mutel, C. and S. Müller. 2013. Using online scientific notebooks for LCA calculations. http://chris.mutel.org/static/images/ipython-notebooks-handout.pdf. Accessed 24 October 2014.

Mutel, C. L. 2014. Brightway2: A new open source framework for advanced life cycle assessment calculations. http://brightwaylca.org/. Accessed 11 October 2014.

Mutel, C. L., S. Pfister, and S. Hellweg. 2012. GIS-based regionalized life cycle assessment: How big is small enough? Methodology and case study of electricity generation. *Environmental Science & Technology* 46(2): 1096–1103.

Nakamura, S. and Y. Kondo. 2002. Input-output analysis of waste management. *Journal of Industrial Ecology* 6(1): 39–63.

Nakamura, S., Y. Kondo, S. Kagawa, K. Matsubae, K. Nakajima, and T. Nagasaka. 2014. MaTrace: Tracing the fate of materials over time and across products in open-loop recycling. *Environmental Science & Technology* 48(13): 7207–7214.

Nakamura, S., Y. Kondo, K. Matsubae, K. Nakajima, and T. Nagasaka. 2011. UPIOM: A new tool of MFA and its application to the flow of iron and steel associated with car production. *Environmental Science & Technology* 45(3): 1114–1120.

Nakamura, S. and K. Nakajima. 2006. Waste input-output material flow analysis and its application to quantity metals. *Journal of the Japan Institute of Metals* 70(6): 505–510.

Nature. 2014. Nature editorial: “Code share: Papers in Nature journals should make computer code accessible where possible.” *Nature* 514(7524): 536.

Open Source Initiative. 1988. The MIT license. http://opensource.org/licenses/MIT. Accessed 28 October 2014.

OpenLCA. 2014. Open LCA Nexus—Your source for LCA datasets. https://nexus.openlca.org/brands. Accessed 24 January 2013.

Pauliuk, S. 2014a. pySUT—A Python class for handling supply and use tables. https://github.com/stefanpauliuk/pySUT. Accessed 8 October 2014.

Pauliuk, S. 2014b. dynamic_stock_model—A Python class for dynamic stock modelling. https://github.com/stefanpauliuk/dynamic_stock_model. Accessed 6 October 2014.

Pauliuk, S. and E. G. Hertwich. 2015. Prospective models of society’s future metabolism—What industrial ecology has to contribute. In *Taking stock of industrial ecology*, edited by R. Clift and A. Duckmann. Dordrecht, the Netherlands: Springer, Netherlands. Forthcoming.

Pauliuk, S., G. Majeau-Bettez, and D. B. Müller. 2015*a*. A general system structure and accounting framework for socioeconomic metabolism. *Journal of Industrial Ecology* Forthcoming.

Pauliuk, S., T. Wang, and D. B. Müller. 2012. Moving toward the circular economy: The role of stocks in the Chinese steel cycle. *Environmental Science & Technology* 46(1): 148–154.

Pauliuk, S., R. Wood, and E. G. Hertwich. 2015b. Dynamic models of fixed capital stocks and their application in industrial ecology. *Journal of Industrial Ecology* 19(1): 104–116.

Peng, R. D. 2011. Reproducible research in computational science. *Science (New York, N.Y.)* 334(6060): 1226–1227.

Pielke Jr, R. A. 2007. *The honest broker: Making sense of science in policy and politics*. Cambridge, UK; New York: Cambridge University Press.

Price, L. and A. Kendall. 2012. Wind power as a case study. *Journal of Industrial Ecology* 16(S1): S22–S27.

Robitaille, T. P., E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, et al. 2013. Astrophysics astropy: A community Python package for astronomy. *Astronomy and Astrophysics* 558(A33): 1–9.

Rose, D. C. 2014. Five ways to enhance the impact of climate science. *Nature Climate Change* 4(7): 522–524.

Saner, D., C. Vadenbo, B. Steubing, and S. Hellweg. 2014. Regionalized LCA-based optimization of building energy supply: Method and case study for a Swiss municipality. *Environmental Science & Technology* 48(13): 7651–7659.

Schmidt, J. H., B. P. Weidema, and S. Suh. 2010. *EU-FORWAST project. Deliverable no. 6.4. Documentation of the final model used for the scenario analyses*. Aalborg, Denmark. http://forwast.brgm.fr/Documents/Deliverables/Forwast_D64.pdf. Accessed 28 May 2015.

Schneider, S. H. 1997. Integrated assessment modeling of global climate change: Transparent rational tool for policy making or opaque screen hiding value-laden assumptions? *Environmental Modeling and Assessment* 2(4): 229–249.

SERI/WU (Sustainable Europe Research Institute/Vienna University of Economics and Business). 2014. Global material flow database. www.materialflows.net. Accessed 28 May 2015.

Shen, H. 2014. Nature toolbox: "Interactive notebooks: Sharing the code." *Nature* 515(7525): 151–152.

Speck, R., S. Selke, R. Auras, and J. Fitzsimmons. 2015. Life cycle assessment software: Selection can impact results. *Journal of Industrial Ecology*. DOI: 10.1111/jiec.12245.

Stadler, K. 2014a. pymrio—Multi regional input output analysis in Python. http://dx.doi.org/10.6084/m9.figshare.1209339. Accessed 20 October 2014.

Stadler, K. 2014b. pymrio: A python module for automating input output calculations and generating reports. https://github.com/konstantinstadler/pymrio. Accessed 11 October 2014.

Steubing, B. 2014. Activity browser—A free and extendable LCA software. https://bitbucket.org/bsteubing/activity-browser. Accessed 2 February 2015.

Steubing, B. 2015. Meta-process concept. http://activity-browser.readthedocs.org/en/latest/metaprocess_introduction.html. Accessed 17 February 2015.

Suh, S. 2004. Functions, commodities and environmental impacts in an ecological-economic model. *Ecological Economics* 48(4): 451–467.

Suh, S. 2005. Developing a sectoral environmental database for input-output analysis: The comprehensive environmental data archive of the US. *Economic Systems Research* 17(4): 449–469.

Suh, S. 2009. Developing the sectoral environmental database for input-output analysis: Comprehensive environmental data archive of the U.S. In *Handbook of input-output economics in industrial ecology*, edited by S. Suh (pp. 689–712). Dordrecht, the Netherlands: Springer.

Suh, S., M. Lenzen, G. J. Treloar, H. Hondo, A. Horvath, G. Huppes, O. Jolliet, et al. 2004. System boundary selection in life-cycle inventories using hybrid approaches. *Environmental Science & Technology* 38(3): 657–664.

Suh, S., B. P. Weidema, J. H. Schmidt, and R. Heijungs. 2010. Generalized make and use framework for allocation in life cycle assessment. *Journal of Industrial Ecology* 14(2): 335–353.

Takahashi, K. I., R. Terakado, J. Nakamura, I. Daigo, Y. Matsuno, and Y. Adachi. 2009. In-use stock of copper analysis using satellite nighttime light observation data. *Materials Transactions* 50(7): 1871–1874.

The FreeBSD Project. 1992. The FreeBSD copyright. www.freebsd.org/copyright/freebsd-license.html. Accessed 28 October 2014.

Tukker, A. and E. Dietzenbacher. 2013. Global multiregional input-output frameworks: An introduction and outlook. *Economic Systems Research* 25(1): 1–19.

Tukker, A., A. de Koning, R. Wood, T. Hawkins, S. Lutter, J. Acosta, J. M. Rueda Cantuche, et al. 2013. Exiopol—Development and illustrative analyses of a detailed global MR EE SUT/IOT. *Economic Systems Research* 25(1): 50–70.

United Nations. 1999. *Handbook of input-output-table compilation and analysis. Studies in methods*. Vol. series F. New York: United Nations.

Van der Voet, E., R. Kleijn, R. Huele, M. Ishikawa, and E. Verkuijlen. 2002. Predicting future emissions based on characteristics of stocks. *Ecological Economics* 41(2): 223–234.

Verones, F., K. Bartl, S. Pfister, R. Jimenez Vilchez, and S. Hellweg. 2012. Modeling the local biodiversity impacts of agricultural water use: Case study of a wetland in the coastal arid area of Peru. *Environmental Science & Technology* 46(9): 4966–4974.

Vienna University of Technology. 2012. STAN. www.stan2web.net/. Accessed 10 October 2014.

Vries, B. J. M. de, D. P. van Vuuren, M. G. J. den Elzen, and M. A. Janssen. 2001. *The Targets IMage Energy Regional (TIMER) Model: Technical documentation*. www.pbl.nl/en/publications/2001/TheTargetsIMageEnergyRegionalTIMERModelTechnicalDocumentation. Accessed 28 May 2015.

WCRP (World Climate Research Program). 2015. CMIP Coupled Model Intercomparison Project. http://cmip-pcmdi.llnl.gov/. Accessed 12 February 2015.

Weidema, B. P., C. Bauer, R. Hischier, C. L. Mutel, T. Nemecek, J. Reinhard, C. Vadenbo, and G. Wernet. 2013. *Overview and methodology. Data quality guideline for the ecoinvent database version 3*. Zürich, Switzerland. www.ecoinvent.org/files/dataqualityguideline_ecoinvent_3_20130506.pdf. Accessed 28 May 2015.

Wiedenhofer, D., E. Rovenskaya, W. Haas, F. Krausmann, I. Pallua, and M. Fischer-Kowalski. 2013. Is there a 1970s syndrome? Analyzing structural breaks in the metabolism of industrial economies. *Energy Procedia* 40: 182–191.

Wiedmann, T. O. and J. Barrett. 2013. Policy-relevant applications of environmentally extended MRIO databases—Experiences from the Uk. *Economic Systems Research* 25(1): 143–156.

Wiedmann, T. O., H. Schandl, M. Lenzen, D. D. Moran, S. Suh, J. West, and K. Kanemoto. 2013. The material footprint of nations. *Proceedings of the National Academy of Sciences of the United States of America* 1–6.

Wiedmann, T. O., H. C. Wilting, M. Lenzen, S. Lutter, and V. Palm. 2011. Quo vadis MRIO? Methodological, data and institutional requirements for multi-region input–output analysis. *Ecological Economics* 70(11): 1937–1945.

Wikimedia Foundation. 2014. Wikibooks. http://en.wikibooks.org/wiki/Main_Page. Accessed 11 October 2014.

Wilson, G., D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S. H. D. Haddock, et al. 2014. Best practices for scientific computing. *PLoS Biology* 12(1): e1001745.

WIPO (World Intellectual Property Association). 2009. *Understanding copyright and related rights*. Geneva, Switzerland: WIPO.

Wood, R. 2011. Construction, stability and predictability of an input-output time-series for Australia. *Economic Systems Research* 23(2): 175–211.

Wood, R., K. Stadler, T. Bulavskaya, S. Lutter, S. Giljum, A. de Koning, J. Kuenen, et al. 2014. Global sustainability accounting—Developing EXIOBASE for multi-regional footprint analysis. *Sustainability* 7(1): 138–163.

About the Authors

Stefan Pauliuk is a post-doctoral researcher and **Konstantin Stadler** is a researcher at the Industrial Ecology Programme at the Department of Energy and Process Engineering at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway. **Guillaume Majeau-Bettez** is a post-doctoral researcher jointly hired by the Industrial Ecology Programme at the Department of Energy and Process Engineering at NTNU and by CIRAIG (Interuniversity Research Centre for the Life Cycle of Products, Processes and Services) at the École Polytechnique de Montréal, Quebec, Canada. **Christopher L. Mutel** is a staff scientist at the Paul Scherrer Institute (PSI), Villigen, Switzerland. **Bernhard Steubing** is a post-doctoral researcher at the Institute of Environmental Engineering at the Swiss Federal Institute of Technology (ETH Zurich), Zurich, Switzerland.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

Supporting Information S1: This supporting information contains a detailed list of online references to programming practice in other fields, links to some software tools mentioned in the paper, and links to several blog entries on relevant topics.