Journal of
open research software

Bond-Lamberty, B, et al. 2019 *gcamdata*: An R Package for Preparation, Synthesis, and Tracking of Input Data for the GCAM Integrated Human-Earth Systems Model. *Journal of Open Research Software,* 7: 6. DOI: https://doi.org/10.5334/jors.232

SOFTWARE METAPAPER

gcamdata: An R Package for Preparation, Synthesis, and Tracking of Input Data for the GCAM Integrated Human-Earth Systems Model

Ben Bond-Lamberty[1], Kalyn Dorheim[1], Ryna Cui[1], Russell Horowitz[2], Abigail Snyder[1], Katherine Calvin[1], Leyang Feng[1,3], Rachel Hoesly[1], Jill Horing[4], G. Page Kyle[1], Robert Link[1], Pralit Patel[1], Christopher Roney[1], Aaron Staniszewski[1], Sean Turner[1], Min Chen[1], Felipe Feijoo[1,5], Corinne Hartin[1], Mohamad Hejazi[1], Gokul Iyer[1], Sonny Kim[1], Yaling Liu[6], Cary Lynch[1], Haewon McJeon[1], Steven Smith[1], Stephanie Waldhoff[1], Marshall Wise[1] and Leon Clarke[1]

[1] Joint Global Change Research Institute at Pacific Northwest National Laboratory (JGCRI/PNNL), US

[2] University of California at Los Angeles, US

[3] American University, US

[4] Stanford University, US

[5] Pontificia Universidad Católica de Valparaíso, CL

[6] Columbia University, US

Corresponding author: Ben Bond-Lamberty (bondlamberty@pnnl.gov)

The increasing data requirements of complex models demand robust, reproducible, and transparent systems to track and prepare models' inputs. Here we describe version 1.0 of the *gcamdata* R package that processes raw inputs to produce the hundreds of XML files needed by the GCAM integrated human-earth systems model. It features extensive functional and unit testing, data tracing and visualization, and enforces metadata, documentation, and flexibility in its component data-processing subunits. Although this package is specific to GCAM, many of its structural pieces and approaches should be broadly applicable to, and reusable by, other complex model/data systems aiming to improve transparency, reproducibility, and flexibility.

Keywords: Human-earth system modeling; data provenance; reproducibility; earth modeling; unit testing
Funding statement: Primary support for this work was provided by the U.S. Department of Energy, Office of Science, as part of research in Multi-Sector Dynamics, Earth and Environmental System Modeling Program. Additional support was provided by the U.S. Department of Energy Offices of Fossil Energy, Nuclear Energy, and Energy Efficiency and Renewable Energy and the U.S. Environmental Protection Agency.

(1) Overview

Introduction

Science is becoming increasingly collaborative and data-intensive [1], and many factors are pushing scientists to increase data access and use 'best practices' in dealing with data and code [2, 3]. There is also an increased emphasis on transparency, data provenance, and reproducibility from funders, governments, and scientists themselves [4, 5]. These factors have increased the need for reproducible, programmatic approaches, both to produce large-scale datasets and handle the increasing data demands of modern models. Examples include the software stack supporting globally gridded soils dataset products [6], the CEDS database of anthropogenic emissions of reactive gases and aerosols [7], and the Global Land Data Assimilation System [8]. These and many other regional to global data are used by a wide variety of earth system models, dynamic global vegetation models, and integrated human-earth system models [9].

Modern, integrated human-earth system models are typically complex and require correspondingly detailed input datasets. These models are sophisticated attempts to encapsulate relations between environmental, social and economic factors that are thought to drive future global

change, and assess the effectiveness of technologies and policies [10]. One integrated human-earth system model is GCAM, a model coupling representations of global and regional economies; energy systems; agricultural, water, and land use systems; and global climate [11]. GCAM's primary external assumptions include socioeconomic drivers (e.g., population and GDP), technology characterizations (e.g., cost and efficiency), and assumptions about regulations and policies that might influence the human systems represented in GCAM.

Currently GCAM requires over 200 Extensible Markup Language (XML) input files, detailing everything from future population projections to historical land allocation to emissions factors. These files create and describe six inter-dependent model modules: (1) agriculture and land use; (2) energy production, transformation, and consumption; (3) water demands; (4) socio-economic demand drivers; (5) non-CO_2 emissions; and (6) GCAM-USA, a state-level representation of the USA region. These modules' inputs include information for all time periods, including historical calibration data, characteristics of hundreds of modeled technologies, future assumptions, and other relevant data.

Earlier versions of the model [12] used spreadsheet-exported inputs but as the data volume increased a system of scripts was developed to generate and reconcile data [13]. Spreadsheets do not scale well, however, and in general impede reproducibility and transparency of data flow; thus there was an acute need for a data system that was open source and transparent, easy to install and use, flexible and robust in its assumptions, and well documented. There are general-purpose R packages to support reproducible, verifiable data processing and scientific research, including *madrat* (https://cran.r-project.org/package=madrat), *drake* (https://cran.r-project.org/package=drake), and *workflowr* (https://cran.r-project.org/package=workflowr). Our specific needs for extensive consistency-checking and error-handling, in addition to providing a platform for data and model exploration and reproducible, transparent scientific and policy research, led to the development of the system described here.

Implementation and architecture

As noted above, the design requirements for this software centered around clarity, ease of use, robustness, error checking, documentation, and flexibility. These criteria led us to select the R statistical programming language [14] as the programming language of choice. R has seen increasing use across many fields of science [15], is free and open source, and straightforward to install. Importantly, R's package system (along with optional tools such as *devtools*[1] and *roxygen2*)[2] offers extensive support for reproducible research [16]; for example, packages will not pass testing and continuous integration successfully if all user-facing functions do not have documentation, or if the package fails any one of a wide range of standard as well as user-defined tests.

Assuming that devtools is installed, the package can be installed and run by:

```
devtools::install_github("JGCRI/gcamdata")
library(gcamdata)
driver() # build the GCAM input data
```

The *gcamdata* system is conceptually organized into three levels of data: raw data (level 0) that are processed and aggregated/disaggregated into generic intermediate categories (level 1), which are then processed further to fit GCAM's specific structures and model time periods (level 2). There are ~30 major inventory data sources within GCAM's level 0 data (see data documentation at https://github.com/JGCRI/gcamdata/wiki), and hundreds of additional data sources are used for more specific information. Data sources consist of a blend of top-down inventories, bottom-up estimates, and information describing the characteristics of modeled technologies. The *gcamdata* package allows for users to update raw data, and modify assumptions and mappings, in order to generate alternative GCAM input scenarios. Internal consistency is enforced, i.e. modifying any calibrated flow estimate requires consideration of all affected sectors and processes, in order to ensure that all modeled flows remain balanced; this is automatically handled by the *gcamdata* code.

The units of code that handle these processing steps are termed 'chunks' and generally consist of a single function that takes inputs (data dependencies) and produce outputs, which are then available for processing by downstream chunks. On startup, chunks must declare all their required inputs, their optional inputs (see below), and their outputs. Two special classes of chunks also exist: "data" chunks are responsible only for loading and parsing specific datasets from disk, typically from a file with a nonstandard format; and "xml" chunks that construct the actual GCAM input files. The *gcamdata* code includes a facility for automatic generation of a chunk skeleton, i.e. the basic architecture of a chunk ready for coding; this provides a mechanism for extension of the data system's architecture.

A 'driver' routine is invoked by the user to start the data system. This function locates all the available chunks in the package namespace, and queries them for their dependencies and outputs, which allows for the construction of a full data-dependency graph (**Figure 1**). The driver enters its main run loop, in which it calls all chunks with currently-available inputs and verifies (see below) the chunks' outputs. Outputs are then added to the main data store or, if they will no longer be needed, written to disk and removed from working memory. This process continues until there are no more chunks to run.

An important question was how to handle proprietary data [17], specifically the International Energy Agency (IEA) Energy Balances [18], a data product that cannot be legally included in the open-source *gcamdata* package. This problem was solved by allowing chunks to have optional inputs, and including a cached copy of the summarized proprietary data in the package. (Distribution of these summarized versions of the IEA data is permitted under the terms of the license for the data.) For example, chunk X summarized the IEA data by GCAM regions and

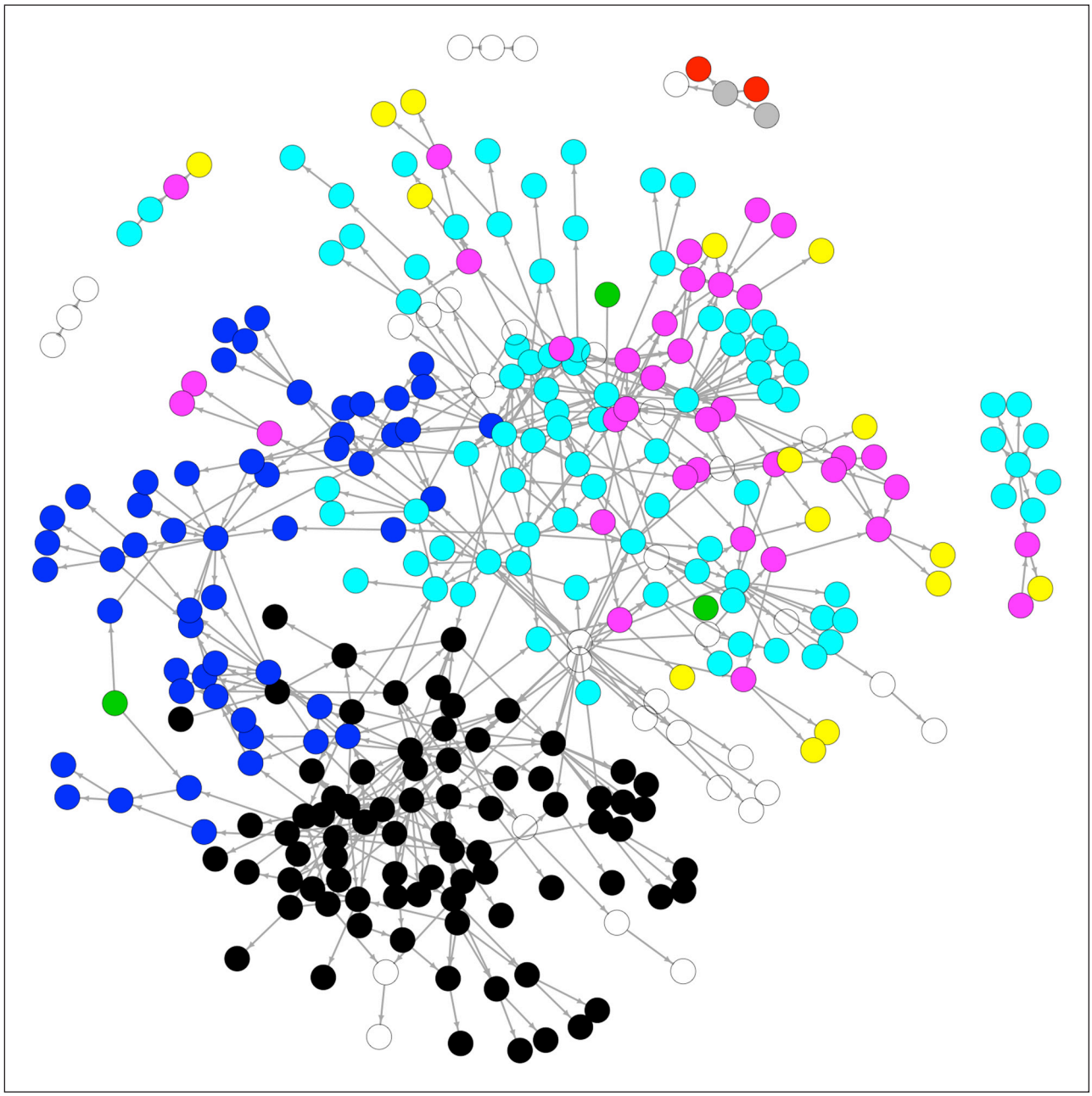

Figure 1: High level view of the code-data dependencies in the *gcamdata* package. This plot of the system architecture shows nodes ("chunks", units of code charged with processing data and producing specific outputs) and edges (data flows between chunks). Nodes are colored by discipline, e.g., agriculture and land use-related code is black, energy system code is blue, etc. For clarity neither the initial data inputs nor the final XML outputs (i.e. the GCAM input files) are shown; this means that seemingly isolated nodes or groups of nodes actually contribute data directly into the model.

technologies, producing output Y, which is then used by chunk Z. Y is cached (and included with the *gcamdata* download) and thus always available for Z, even if the source IEA data are not; when this occurs, a note is added to the downstream metadata indicating that cached data were used. The overall *gcamdata* license, the Educational Community License (http://opensource.org/licenses/ecl2.php), is close to the Apache license and chosen to match that of the GCAM model.

Data objects in *gcamdata* are required to have descriptive metadata attached, including title, units, description, and dependency information. (Most of these requirements are enforced on input data as well.) This allows us to track data provenance [19] throughout the system, and provide data tracing. Because all chunks declare their inputs and outputs to the driver, a full system-wide data map can be constructed (**Figure 1**) and then particular data dependencies, upstream and/or downstream, traced through the system (**Figure 2**). Because all data objects that flow between chunks are required to have extensive metadata (including title, units, source, and comments), this allows for easy and informative exploration of the data sources and dependencies of any object in the system.

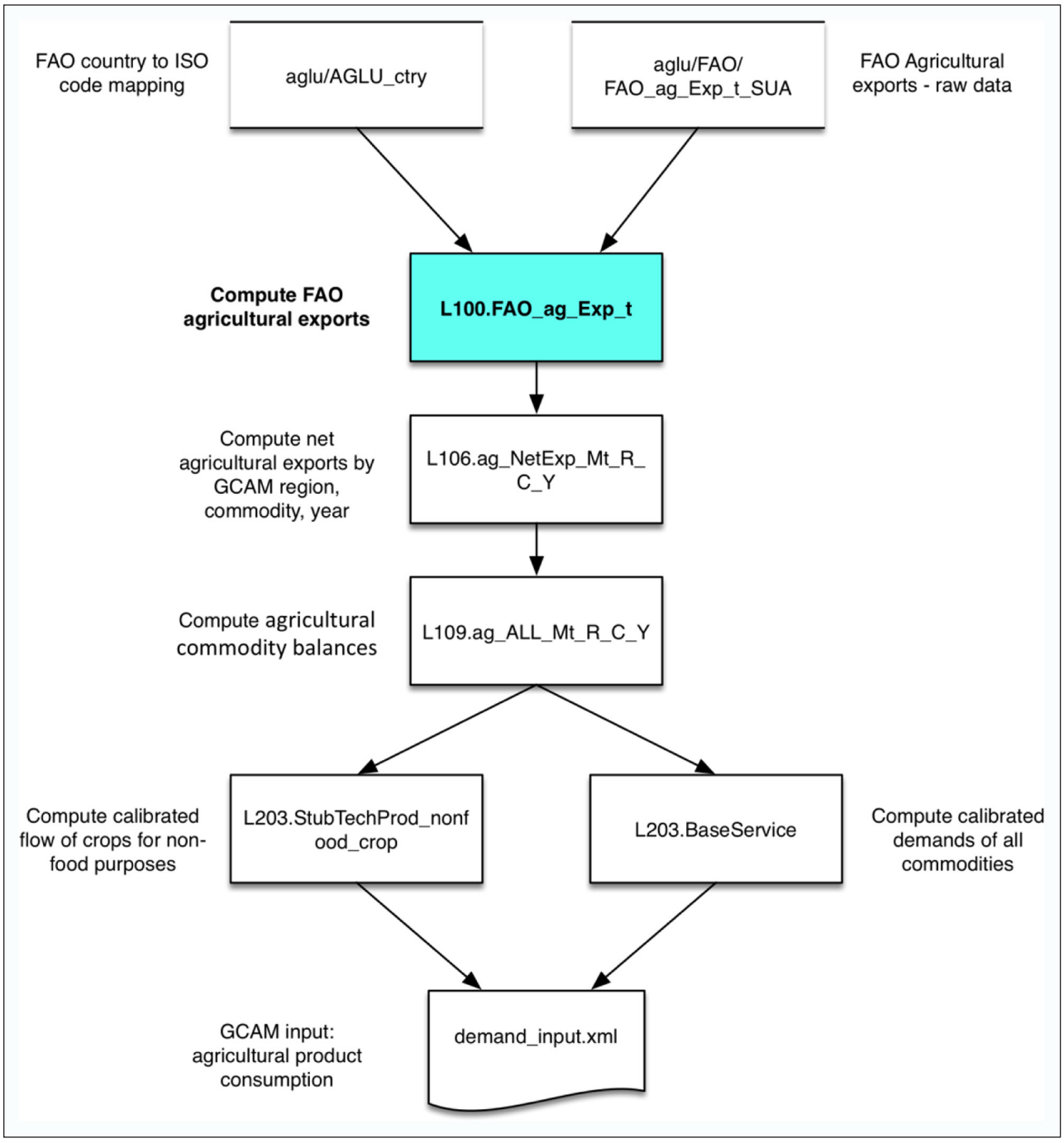

Figure 2: An example of tracing data flow. Here the user has requested a data trace on a particular data object "L100.FAO_ag_Exp_t" (FAO agricultural exports by country, item, and year). The package prints detailed information about this object and its upstream and downstream dependencies, and graphs these relationships to show data flow (arrows). Raw data inputs are at the top, and the final XML product that flows into the GCAM model is at the bottom. Explanatory notes describe each step.

Quality control

The package includes extensive functional and unit testing (**Table 1**) that verifies behavior of the data 'chunks', the supporting data system functions, the driver, and characteristics of data objects; testing considers both the chunks (units of code) and data (relationships) [20]. The current level of unit testing coverage is 92%. The full suite of tests is invoked every time a user builds the R package, at which point it is also subjected to a battery of standardized R checks (see http://r-pkgs.had.co.nz/check.html). The tests are part of the continuous integration [21] with the *gcamdata* repository (https://github.com/jgcri/gcamdata), meaning that they are invoked for every pull request (PR), and the PR cannot proceed without all tests being passed.

(2) Availability

Operating system

Mac OS X 10.6. or later; Unix-like operating systems; Windows 7 or later. See https://cran.r-project.org/doc/FAQ/R-FAQ.html#What-machines-does-R-run-on_003f.

Table 1: Automatic package-level checks performed on the *gcamdata* data-handling functions (termed "chunks") and their outputs.

Category	Test
Behavior	Chunk responds to required messages from driver (DECLARE_INPUTS, DECLARE_OUTPUTS, MAKE)
	Chunk doesn't make forbidden calls (e.g., slow or deprecated R routines)
	Chunk handles changes in model time settings
	Chunk (package-level) constants are correctly formatted
Data	Chunk declares a (possibly empty) list of input that can all be found, either as the product of another chunk or as a file input
	Chunk declares a valid list of outputs
	Chunk uses only its declared inputs
	Chunk produces exactly its declared outputs
	All file inputs have metadata headers and are encoded (e.g., standard line endings) correctly
	All chunk outputs have title, description, units, comments, and precursor information attached
	All declared precursors are in the chunk input list, and each chunk input is the precursor of at least one output
	Chunk outputs match known good output set

Programming language
R (version 3.1 or later).

Additional system requirements
The package uses and processes some large datasets, but efficiently prunes in-memory objects when they are no longer needed. Its memory usage during a run peaks at ~800 MB; the on-disk input data, many of which are compressed, are ~73 MB. The XML files written by the system are ~2.3 GB.

Dependencies
Required dependencies include the R packages *assertthat* (>=0.2), *dplyr* (>=0.7.0), *magrittr* (>=1.5), *tibble* (>=1.1), *tidyr* (>=0.7.1), *readr* (>=1.0.0), and *data.table* (>=1.10.4).

Optional dependencies include the R packages *igraph* (>=1.0.1), *mockr* (>=0.1), *testthat* (>=1.0.2), and *R.utils* (>=2.6.0), as well as Python version 3.

List of contributors
Package design and development was led by Ben Bond-Lamberty. Kalyn Dorheim was the verification lead, and with Ryna Cui, Russell Horowitz, and Abigail Snyder wrote the bulk of the code. Katherine Calvin, Leyang Feng, Rachel Hoesly, Jill Horing, Page Kyle, Robert Link, Pralit Patel, Chris Roney, Aaron Staniszewski, and Sean Turner contributed significantly to coding and/or design. Further contributions were made by Min Chen, Felipe Feijoo Palacios, Corinne Hartin, Mohamad Hejazi, Gokul Iyer, Sonny Kim, Yaling Liu, Cary Lynch, Haewon McJeon, Steve Smith, Stephanie Woldhoff, Marshall Wise. Katherine Calvin, Corinne Hartin, Gokul Iyer, Haewon McJeon, and Leon Clarke managed the various developments teams. Page Kyle developed the original R scripts that grew into *gcamdata* and made significant contributions in writing this manuscript.

Software location
Archive
Name: Zenodo
Persistent identifier: Version v1.0, DOI: https://doi.org/10.5281/zenodo.1249932
Licence: Educational Community License, Version 2.0 (ECL-2.0). See https://github.com/JGCRI/gcamdata/blob/master/LICENSE
Publisher: Pacific Northwest National Laboratory
Version published: 1.0
Date published: 19/05/2018

Code repository
Name: gcamdata
Identifier: https://github.com/JGCRI/gcamdata/
Licence: Educational Community License, Version 2.0 (ECL-2.0). See https://github.com/JGCRI/gcamdata/blob/master/LICENSE
Date published: 19/05/2018

Language
English

(3) Reuse potential
The *gcamdata* package maintains good separation between GCAM-specific code and infrastructure code, and reusing the infrastructure as a platform for building data preparation code for other scientific models would be straightforward; in particular, many of the intermediate processing steps and even particular data products are also required by other multi-sectoral models [22].

More generally, many of the package's concepts, structural design, and specific code elements may be broadly interesting to, and reusable by, other model/data teams interested in improving the transparency, reproducibility, and flexibility of their systems. Many parts of the *gcamdata*

package could be repurposed for any data system that involves multiple, potentially interacting, data processing steps. Given the wide diversity of human-earth system models and frameworks in use, and the resulting problems associated with separating model and scenario variability [23], standardizing on an open-source data processing platform would be valuable for the many communities in human-earth system modeling.

Key areas of interest and potential reuse include:

- An object-oriented approach to data processing, with chunks (units of code responsible for a specific data-processing step) called when needed by a controlling driver routine.
- Chunks are auto-discovered, so it is easy to add new ones. An empty chunk template is included in the *gcamdata* code.
- Data objects are passed between chunks. All data objects (including those from input files) must have attached metadata, and this is enforced by extensive checking by the driver.
- Enforcement of file encoding and structure. For example, our developers variously use Windows, Mac, and Unix, all of which have different line ending conventions, but *gcamdata* enforces a single standard.
- Chunks declare their inputs and outputs to a driver through a fixed Application Programming Interface. The resulting chunk and data dependency information allows for extensive visualization, data tracing, etc.
- A great deal of unit/functional testing. This is standard in the software design world [20], but much less so in scientific programming [24], and to our knowledge extremely rare in systems designed to process or produce datasets.

Notes

[1] https://cran.r-project.org/package=devtools.

[2] https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html.

Acknowledgements

The management and financial expertise of Ibimina Nweke and Kali Wood provided crucial support during *gcamdata* development. The package makes crucial use of land-use and land cover change data developed by Alan Di Vittorio of Lawrence Berkeley National Laboratory.

Competing Interests

The authors have no competing interests to declare.

References

1. **Adams, J** 2012 Collaborations: The rise of research networks. *Nature*, 490: 335–336. DOI: https://doi.org/10.1038/490335a
2. **Wilson, G, Aruliah, D A, Brown, C T,** et al. 2014 Best practices for scientific computing. *PLoS Biol*, 12: e1001745. DOI: https://doi.org/10.1371/journal.pbio.1001745
3. **Thornton, P E, Cook, R B, Braswell, B H,** et al. 2005 Archiving numerical models of biogeochemical dynamics. *EOS*, 86: 431–432. DOI: https://doi.org/10.1029/2005EO440003
4. **Ince, D C, Hatton, L** and **Graham-Cumming, J** 2012 The case for open computer programs. *Nature*, 482: 485–488. DOI: https://doi.org/10.1038/nature10836
5. **Peng, R D** 2011 Reproducible research in computational science. *Science*, 334: 1226–1227. DOI: https://doi.org/10.1126/science.1213847
6. **Hengl, T, Mendes de Jesus, J, MacMillan, R A,** et al. 2014 SoilGrids1km — global soil information based on automated mapping. *PLoS One*, 9: e114788. DOI: https://doi.org/10.1371/journal.pone.0105992
7. **Hoesly, R M, Smith, S J, Feng, L,** et al. 2018 Historical (1750–2014) anthropogenic emissions of reactive gases and aerosols from the Community Emissions Data System (CEDS). *Geoscientific Model Development*, 11: 369–408. DOI: https://doi.org/10.5194/gmd-11-369-2018
8. **Rodell, M, Houser, P R, Jambor, U,** et al. 2004 The Global Land Data Assimilation System. *Bull Am Meteorol Soc*, 85: 381–394. DOI: https://doi.org/10.1175/BAMS-85-3-381
9. **van Vuuren, D P, Bayer, L B, Chuwah, C,** et al. 2012 A comprehensive view on climate change: Coupling of earth system and integrated assessment models. *Environ Res Lett*, 7. DOI: https://doi.org/10.1088/1748-9326/7/2/024012
10. **van Vuuren, D P, Lowe, J A, Stehfest, E,** et al. 2011 How well do integrated assessment models simulate climate change? *Clim Change*, 104: 255–285. DOI: https://doi.org/10.1007/s10584-009-9764-2
11. **Calvin, K, Bond-Lamberty, B, Clarke, L,** et al. (2017/1) The SSP4: A world of deepening inequality. *Glob Environ Change*, 42: 284–296. DOI: https://doi.org/10.1016/j.gloenvcha.2016.06.010
12. **Kim, S H, Edmonds, J A, Lurz, J,** et al. 2006 The ObjECTS framework for integrated assessment: Hybrid modeling of transportation. *Energy J*, 27: 63–91. DOI: https://doi.org/10.5547/ISSN0195-6574-EJ-VolSI2006-NoSI2-4
13. **Kyle, G P, Luckow, P, Calvin, K V,** et al. 2011 GCAM 3.0 Agriculture and Land Use: Data Sources and Methods. College Park, MD: Pacific Northwest National Laboratory. DOI: https://doi.org/10.2172/1036082
14. **R Development Core Team** 2017 R: A language and environment for statistical computing. Version 3.3.3.
15. **Tippmann, S** 2014 Programming tools: Adventures with R. *Nature*, 517: 109–110. DOI: https://doi.org/10.1038/517109a
16. **Marwick, B, Boettiger, C** and **Mullen, L** 2017 Packaging data analytical work reproducibly using R (and friends). *PeerJ Preprints*.
17. **Heidorn, P B** 2008 Shedding Light on the Dark Data in the Long Tail of Science. *Libr Trends*, 57: 280–299. DOI: https://doi.org/10.1353/lib.0.0036
18. **IEA** 2012 Energy Balances of non-OECD Countries 2012. International Energy Agency.
19. **Buneman, P, Khanna, S** and **Tan, W-C** 2000 Data Provenance: Some Basic Issues. In: *FST TCS 2000: Foundations of Software Technology and*

Theoretical Computer Science, 87–93. Berlin, Heidelberg: Springer. DOI: https://doi.org/10.1007/3-540-44450-5_6

20. **Zhao, J** 2003 Data-flow-based unit testing of aspect-oriented programs. In: *Proceedings 27th Annual International Computer Software and Applications Conference*, 188–197. COMPAC 2003. DOI: https://doi.org/10.1109/CMPSAC.2003.1245340
21. **Humble, J** and **Farley, D** 2010 Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. *Pearson Education*.
22. **van Vuuren, D P, Edmonds, J, Kainuma, M,** et al. 2011 The representative concentration pathways: An overview. *Clim Change*, 109: 5. DOI: https://doi.org/10.1007/s10584-011-0148-z
23. **Krey, V** 2014 Global energy-climate scenarios and models: A review. *WIREs Energy Environ*, 3: 363–383. DOI: https://doi.org/10.1002/wene.98
24. **Wang, D, Xu, Y, Thornton, P,** et al. 2014 A functional test platform for the Community Land Model. *Environmental Modelling & Software*, 55: 25–31. DOI: https://doi.org/10.1016/j.envsoft.2014.01.015

How to cite this article: Bond-Lamberty, B, Dorheim, K, Cui, R, Horowitz, R, Snyder, A, Calvin, K, Feng, L, Hoesly, R, Horing, J, Kyle, G P, Link, R, Patel, P, Roney, C, Staniszewski, A, Turner, S, Chen, M, Feijoo, F, Hartin, C, Hejazi, M, Iyer, G, Kim, S, Liu, Y, Lynch, C, McJeon, H, Smith, S, Waldhoff, S, Wise, M and Clarke, L 2019 *gcamdata*: An R Package for Preparation, Synthesis, and Tracking of Input Data for the GCAM Integrated Human-Earth Systems Model. *Journal of Open Research Software,* 7: 6. DOI: https://doi.org/10.5334/jors.232

Submitted: 02 June 2018 **Accepted:** 18 February 2019 **Published:** 14 March 2019

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS