Author reply to the comment by Handling Editor Chiel van Heerwaarden of the manuscript

gmd-2016-318

"eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5"

by S. Metzger et al.

We thank the Handling Editor for his encouraging summary:

> *"The paper has received very detailed reviews with excellent suggestions to improve the paper. The reviewers see the novelty an applicability of the presented framework, but demand improvements in order to accept the paper. The authors have given excellent answers to the reviewers' concerns and by incorporating their suggested improvements, this paper will most likely meet the standards of GMD. Important here is that the reviewers provide a working example as requested by the third reviewer. This will greatly improve the quality of the paper. I suggest that the authors proceed with improving the manuscript.*"

In response we have addressed all Referee comments (please see individual replies to the Referees), and have prepared an executable example workflow accompanying the revised manuscript.

Author reply to the comment by Executive Editor Astrid Kerkeweg of the manuscript gmd-2016-318

"eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5"

by S. Metzger et al.

We thank the Executive Editor for catching our oversight regarding "*The main paper must give the model name and version number (or other unique identifier) in the title.*" In response we have adjusted the manuscript title accordingly: "eddy4R **0.2.0**: A DevOps model for community-extensible processing and analysis of eddy-covariance data based on R, Git, Docker and HDF5".

Author reply to the comments by Anonymous Referee #1 of the manuscript

gmd-2016-318

"eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5"

by S. Metzger et al.

We thank Anonymous Referee #1 for the valuable feedback, which helped to improve the manuscript. Please find below the Referee comments recited in *blue, italics font*, followed by our point-by-point replies and corresponding changes in the manuscript in black, upright font.

1 General

- *The paper describes the development of an open source framework for the processing of eddy-covariance data. The framework is developed to deal with the wealth of data that will collected within the National Ecological Observatory Network (which is located in the US, a geographical specification not given in the manuscript).*
- *The paper both addresses the development process as well as some higher level details of the framework being developed. Little specific information is provided about the actual processing algorithms.*
- *Three case studies are presented in which the processing framework, in its present state, has been used.*
- *My general impression is that the work presented in itself is very worthwhile, and -at least in our field- quite novel.*

Author reply: We thank the Referee for acknowledging the significance of this work.

Changes in the manuscript: We have performed multiple changes resulting from these comments, which are detailed in response to the specific recommendations below.

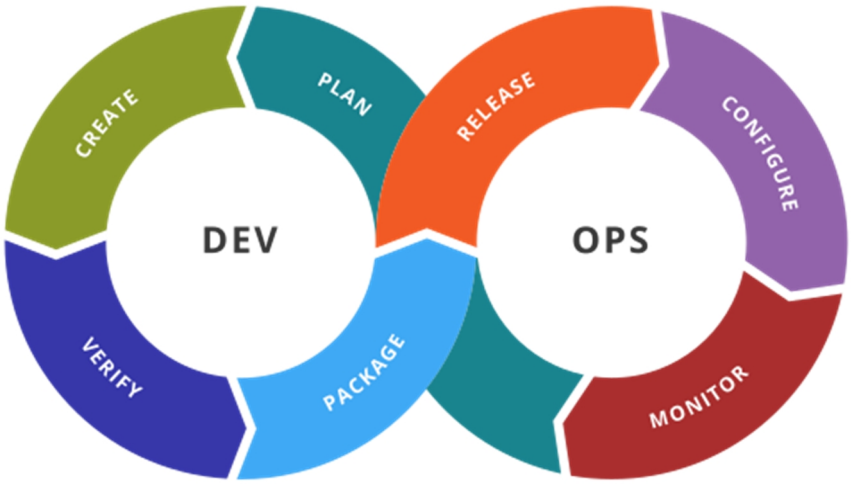
However, I do have some comments:

1. *The Development and Operations (DevOps) framework is mentioned as an essential characteristic of the work discussed. However, the DevOps framework is not clearly introduced to the reader (except for one paragraph in the introduction). Even there, most emphasis is placed on the tools, rather than the essential steps or processes that are part of DevOps. I would have expected some sort of a list of steps, or a schematic that shows*

general characteristics of the workflow in a generic DevOps development process (Wikipedia already shows some colorful examples). Then these generic characteristics could be translated into the specific characteristics of the project that is the subject of the paper. Furthermore, although a number of problems in current practice of EC-data processing are identified (lines 48-62), it remains unclear why DevOps would be the answer.

Author reply: We agree that the generic DevOps methodology was not introduced in sufficient clarity to the reader: an overview of DevOps steps and how they translate into the specific characteristics of the presented software framework is needed. In addition, we agree that clarification is needed on the problem statement of the paper (see Referee comment 5 below) and why DevOps is the answer.

Changes in the manuscript: To address these points, we added text and a figure in Sect. 2. These provide an overview of the DevOps methodology, and throughout the paper we now tie the eddy4R-Docker software development model to it. The adjusted text reads:



"Figure 1. Stages of the general DevOps workflow (source: Kharnagy via Wikimedia Commons [CC BY-SA 4.0]).

DevOps promotes collaboration and tight integration between software development, testing, and operational deployment by following a core workflow (e.g., Wurster et al., 2015): Plan, Create, Verify, Package, Release, Configure, and Monitor. The text below describes these stages and Figure 1 shows the general sequence and overlap of these stages between software developers (Dev) and operators (Ops).

Plan involves focusing and prioritizing new software features or capabilities based on their enhancement of value. **Create** is the activity of designing and writing the code that delivers a new feature. **Verify** tests the new software feature against established standards for accuracy and performance (e.g. does it unexpectedly alter the output of pre-existing features? Does it produce the expected result?). **Package** involves the compilation of the code once it is ready for deployment, including all data and software

dependencies, and gathers necessary approvals. The **Release** stage deploys the software into production. **Configure** involves supplying and configuring the IT infrastructure required to operate the code at scale, including storage, database operations, and networking. Finally, **Monitor** observes and tracks the use, performance, and end-user impact of the release.

Variants of this workflow exist (e.g., Chen, 2015), but the general components and sequence are retained. In addition, there is no single set of tools accompanying the DevOps approach. Rather, many tools exist that facilitate the execution of one or more of these workflow steps, often through automation.

NEON's DevOps framework consists of a periodic sequence (Figure 2) that incorporates these workflow steps: The science community contributes algorithms and best practices (1). Implicitly or explicitly, this embodies the DevOps: Plan stage – the algorithms most valued by the community are being incorporated. Together with NEON Science (2), these algorithms are coded in the open-source R computational environment (DevOps: Create stage). DevOps: Verify (testing) and Package (packaging) are performed as the code is compiled into eddy4R packages via the GitHub distributed version control system (3). NEON Science releases an eddy4R version from GitHub, which automatically builds an eddy4R-Docker image on DockerHub as specified in a "Dockerfile" (4; DevOps: Release stage). The eddy4R-Docker image is immediately available for deployment by NEON Cyberinfrastructure (CI; 5; DevOps: Configure & Monitor stages), the Science Community (1) and NEON Science (2) alike. Here the DevOps: Configure (computational resource allocation) & Monitor stages occur. Monitoring of end-user experience is also performed in GitHub (3) via issue-tracking.

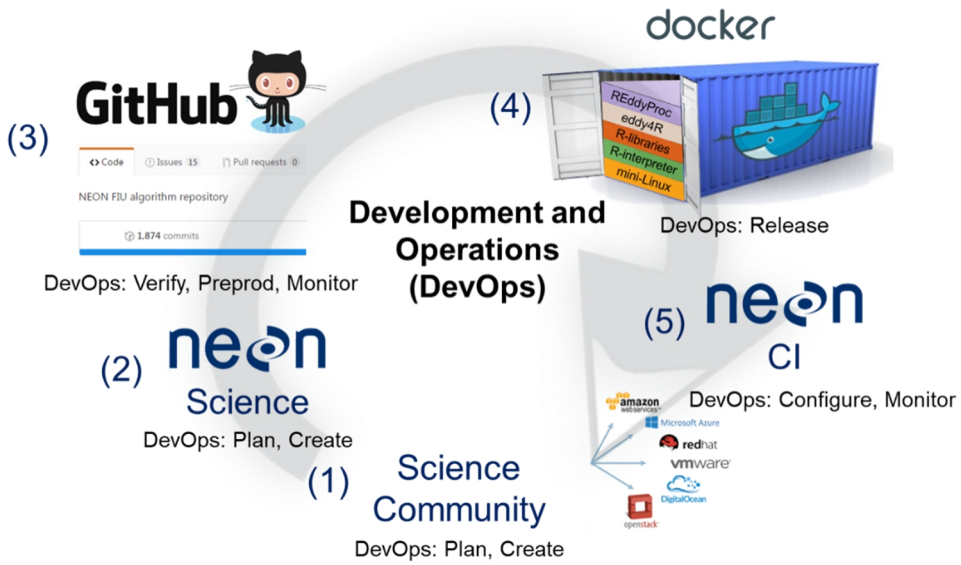


Figure 2. NEON-specific DevOps workflow. DevOps workflow steps are called out in parentheses. Please see text in Sect. 2 for detailed explanation."

In addition, we added text throughout Sects. 2.1 – 2.5, referencing the DevOps workflow steps that each presented tool facilitates.

To address why the DevOps methodology is the answer to some of the problems plaguing EC data processing, we first clarified the problem statement in the introduction: ”The question we ask in this paper is: How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of eddy covariance data products?”

Then, we added the following text later in Sect. 2: ”Thus, the DevOps model serves as the framework within which the scientific community can efficiently and robustly collaborate to produce, manage, and iterate community software. Through choosing appropriate tools to implement the DevOps workflow steps, the reproducibility, scalability and extensibility needs of software development communities (including EC) can be met.”

References:

Chen, L.: Continuous delivery: Huge benefits, but challenges too, IEEE Softw., 32, 50-54, doi:10.1109/ms.2015.27, 2015.

Wurster, L. F., Colville, R. J., and Duggan, J.: Market Trends: DevOps - not a market, but a tool-centric philosophy that supports a continuous delivery value chain, Gartner, Inc., G00274555, Stamford, U.S.A., 14 pp., 2015.

2. *The description of the DevOps framework in section 2 is in fact a description of the collection of tools. It is hard to recognize the DevOps characteristics of it.*

Author reply: We agree, and believe that the addition of the general DevOps overview and translation into the software development model presented in the paper as discussed above addresses this Referee comment.

Changes in the manuscript: Please see the response to the Referee comment 1 for changes made in the manuscript.

3. *The description of the tools in section 2 remains rather vague on one hand (2.1, 2.2), and becomes very (too) specific at other points (2.3). For example, to me the message of section 2.1 is that the authors have implemented regular EC- processing software in R. It remains unclear what are the special properties of this implementation that make it so much better/easier/more flexible/.... than any existing EC-processing toolchain (except that in a number of places terms are used that at least suggest that something special happens in the coding, while it remains unclear what this means in terms of code quality, re-usability etc. e.g. lines 131-134). Similar comments would hold for the other parts of section 2 (see below).*

Author reply: The level of detail provided in Sects. 2.1 – 2.5 was tailored to call out the particular elements of the tools that addressed the portability, reproducibility, and extensibility needs of the EC community. For example, in Sec. 2.1 (previously lines 132-

134), we stated "Following best practices, eddy4R is written in controlled and strictly hierarchical terminology consisting of base names and modifiers, which ensures modular extensibility over time." We agree that this was not clear to the reader as a result of an ambiguous problem statement and an insufficient overview of DevOps and how DevOps was translated into the specific software development model presented in the paper (addressed in response to Referee comment 1 above). We believe the changes made in response to these points now highlight the special properties of this implementation that improve upon existing EC processing toolchains.

Changes in the manuscript: Please see changes made in response to Referee comment 1.

4. *The various sections in in section 2 (except 2.1) in my view insufficiently address what the role of the different tools is (on a conceptual level, not an implementation level): who/why are these tools part of the proposed system. Furthermore, some of the sections are not very specific for the proposed application of the tool (section 2.3 could be used in nearly any paper that describes the use of Docker). The same holds to a large extent for sections 2.2 and 2.4.*

Author reply: We believe that this Referee comment is addressed by the responses above.

Changes in the manuscript: Please see changes made in response to Referee comment 1.

5. *To summarize my main concern: the paper misses a clear problem statement and as a result it is unclear why the presented software development would be the answer. The paper would be worth reconsidering for publication if the authors would be able to reformulate the paper in such a way that:*
   - *there is a clear problem statement (which may, or may not be related directly to the NEON network);*
   - *it is clear that the DevOps methodology is needed to tackle that problem (including a clear general introduction to DevOps, irrespective of the tools used);*
   - *that this set of proposed tools and methods would enable a DevOps methodology for the task at hand (EC-data processing);*
   - *the presented cases (section 3) clearly illustrate why a software infrastructure as proposed in the paper is needed.*

Author reply: We believe that our responses to Referee comments, especially comment 1, and the associated changes made to the manuscript, address these concerns.

Changes in the manuscript: Please see changes made in response to Referee comment 1.

Below I will provide detailed comments

Note: in the comments below, the comment is preceded by the line number.

2 Detailed comments

1. *56: I do not see why the ultimate goal should be a universal EC processing environment. As long as the software that is used in papers is described in open literature and the source code is openly available, readers will be able to assess the results of the EC-processing used in the described research. Research groups will always have reasons to do things their own way: because of instrument or site specifics, or due to specific requirements in output and analysis. And from software intercomparisons performed in the past it is quite clear in which aspects the main differences between different processing packages occur (e.g. Mauder et al., 2008; Fratini and Mauder, 2014).*

Author reply: By universal processing environment, we mean the capability of processing software to be portable, reproducible, and extensible such that it can be reliably and consistently applied at scale across most, if not all, computer platforms. This would better allow research groups to tailor existing software to their needs instead of re-creating code or kludging together multiple software outputs to realize an algorithmic chain for data analytics. Even though source code is available, it does not guarantee reproducibility: The subsequent sentence in the manuscript (previously, line 58) points out that 50% of published scientific code cannot even be run due to missing software dependencies. We adjusted the text to better clarify this point.

The Referee notes:" *As long as the software that is used in papers is described in open literature and the source code is openly available"*. Our claim is that this is generally not the case, and we argue that the DevOps framework helps groups achieve that aim and demonstrate such with our work with eddy4R. Even if source code is provided, code is not sufficiently updated to reflect e.g. new developments in processing or firmware bugs. End-to-end processing from raw data to fluxes is not commonly available except in a few softwares, usually written in compiled programming languages that are not easily modifiable by a general user. Further, it is not easy to modify them for site-specific conditions, and finally, they are rarely inter-compared.

Changes in the manuscript: The passage in question now reads: "Still, large differences in instrumentation, site setup, data format, and operating system stymie the adoption of a universal EC processing environment: one that is portable, reproducible, and extensible to allow tailored workflows that incorporate additional data streams, to automate and scale processing across large compute facilities, or to inject additional algorithms that address specific needs or synergistic research questions. In 50% of published scientific code, one cannot even replicate the necessary software dependencies (Collberg et al., 2014), and even widely used and well-documented EC processing software packages have shown substantial inconsistencies in flux estimates (e.g. Fratini and Mauder, 2014). A universal EC processing environment that enables these capabilities would better allow research groups to tailor existing software to their needs (and contribute new algorithms) instead of re-creating code or kludging together multiple software outputs to realize an algorithmic chain for data analytics."

References:

Collberg, C., Proebsting, T., Moraila, G., Shankaran, A., Shi, Z., and Warren, A. M.: Measuring reproducibility in computer systems research, University of Arizona, Department of Computer Science, Tucson, USA, 37, 2014.

Fratini, G., and Mauder, M.: Towards a consistent eddy-covariance processing: An intercomparison of EddyPro and TK3, Atmos. Meas. Tech., 7, 2273-2281, doi:10.5194/amt-7-2273-2014, 2014.

2. *71: it is unclear why DevOps is being embraced: why is this methodology so suited for the problem at hand (the problem is not clearly stated, but I assume that the 'strong need' expressed in lines 66-70 is 'the problem')?*

Author reply: In response to the points above we clarified the problem statement and provided a general overview of DevOps that links the NEON software framework to DevOps methodology. We believe this has clarified why DevOps is suited to address the problem statement.

Changes in the manuscript: Please see changes made in response to major Referee comment 1.

3. *83: indeed, I agree that the fact that the proposed work enables an exact reconstruction of the system that was used to construct certain derived data is an important asset (having the data and the software is -in some cases- insufficient to enable exact reproducibility). On the other hand, in the application at hand, bit-wise identical results are usually not needed: the statistical errors in the derived quantities are usually orders of magnitude larger than the differences that occur do to differences in details of the computational environment.*

Author reply: Our point was that the reproduction of the computational environment, namely the data and software dependencies required to run the software successfully, is the most important. Given that we do not mention bit-wise identical results and the preceding sentence states "The recipe automates the loading of the software including all dependencies so that the most significant hurdle of reproducing the computational environment is overcome.", we do not think the manuscript needs further clarification of this point.

Changes in the manuscript: No changes.

4. *96-106: this 'cycle' suggests some regularity and pace of development. Is that what is intended, or does it simply mean that once someone has a suggestion for a new feature or new implementation, the code has to go through the cycle? My own experience with code development is that many users are using very different versions of the code (from very old to cutting edge) which may yield impractical surprises when these different users supply new code. The advantage of the -apparently well-funded- NEON network*

is that paid personal is available to oversee new code submissions. In this section it is not clearly defined who/what is the 'science community' on one hand, and 'NEON science' on the other hand.

Author reply: The subsequent sentence (previously lines 106-108) described the pace of iteration "This DevOps cycle can be repeated for continuous development and integration of requests and future methodological improvements, resulting in the next release." Meaning, the cycle begins anew when new code features are submitted or desired by the scientific community. As this is already explained in the manuscript, we do not think further clarification is needed.

Regarding new code submissions arising from out-of-date versions: The advantage of the NEON-DevOps methodology is that users stay up-to-date with the code-base by 'forking' the repository in GitHub (previously lines 184-186). This avoids the issue of users modifying "offline" versions of the code that easily become out-of-date. As this is already described in the manuscript, we do not think further clarification is needed.

We agree that clarification of 'NEON science' vs. the 'science community' may be helpful to readers.

Changes in the manuscript: No changes were made with regard to the cycle of development and staying up-to-date with the code base.

We added the following text to Sect. 2 to clarify 'NEON Science' vs. the 'Science Community': "Here, we define NEON Science as personnel working directly on the NEON project, and the Science Community as anyone producing or using data, algorithms, or research products related to NEON data themes (Atmosphere; Biogeochemistry; Ecohydrology; Land Cover and Processes; Organisms, Populations, and Communities), regardless of whether they also work on the NEON project."

5. *As explained in my main comment: section 2.1 seems to mainly discuss 'just another EC-processing tool'. Please focus on those aspects that are new and make it particularly suited for NEON, and for use in the DevOps methodology. My impression is that the direct implementation of the option for parallel processing of EC-data is one important aspect (which may not be relevant if a group only operates a limited number of towers, but which may be relevant if in 10 years time NEON decides to reprocess all EC-data of all stations according to the latest insights). But if this parallelization is so central, I would like to see some more explanation/example/tests of it (performance, scalability). Another aspect would be indeed the modularity if it allows -as advertised- for the easy adaptation of new hardware. I suppose that you use some sort of instrument abstraction which makes it possible to describe the properties of any thinkable instrument in such a way that the data can be ingested and processed in a correct way. Again, more details on this potentially unique aspect would make the paper worthwhile to read.*

Author reply: As detailed above, the DevOps model has been centralized in the paper to demonstrate the novelty of this approach to scientific computation in general, and to EC specifically. An executable example workflow is provided as part of the revised manuscript in order to demonstrate some of the novel aspects of the eddy4R software itself. The example workflow covers read-in and write-out of self-describing HDF5 files, the modular and nested application of data processing models, as well as interactive visualization capabilities.

Changes in the manuscript: We have added additional context to the HDF5 Sect. 2.4, please see our reply to Referee comment 13 below. Furthermore, we have adjusted the installation and operation Sect. 2.6 to incorporate the executable example workflow.

6. *168: indeed git is open source and free, but Github is not (unless you use a public repository). That brings me to the question in which way eddy4R will be open-sourced. Will the github repository be opened up for everyone (by now I cannot find it)? Or will you publish certain stable versions to the public and keep the development closed? Reliance on Github may seem a risk when planning for the processing of data from a project that will run for 30 years. But since git repositories are stored locally as well, no risk exists in case Github would go out of business.*

Author reply: The current, stable snapshot of the GitHub repository is published and archived incl. DOI (links provided in Sect. 5 "Code and data availability"). As NEON Construction completes, the GitHub repository itself will be made publicly accessible. The central component with regard to contingency planning is distributed version control, of which Git is one implementation. Should Git fail in the next 30 years, it can be replaced with another version control system.

Changes in the manuscript: We added the following text to the end of Sect. 2.4: "We note that the DevOps workflow is robust to the business viability of the particular tools used for implementation. Git is simply one instance of a version control system, which could be replaced with another similar tool should Git fail at some point in the future."

7. *181: who will do the review and testing? Do you have full-time staff for that? Will the testing be automated in the sense that when the new code is included, all prior test cases should still run without errors, while the new code should also provide it's own test case (if it implements a new feature)?*

Author reply: As facilitator of the DevOps implementation of eddy4R, NEON provides the software change control board necessary for continuous development and integration. Testing is automated, and when new code is included, all prior test cases must run without errors and reproduce benchmark results. Including a test case for new code is strongly encouraged, but not mandatory.

Changes in the manuscript: Changed accordingly in Sect. 2.2.: "After (3) 'committing' or creating a new feature, the developer (4) can propose the feature for inclusion in the

official eddy4R source code by issuing a pull request to (5) NEON's change control board. After (6) thorough review and all prior test cases reproducing benchmark results (DevOps: Verify stage), the feature can be 'merged' or integrated into the next release of (1) the official, stable eddy4R source code (DevOps: Package stage). This cycle can be repeated to accommodate requests and future developments, resulting in subsequent releases. Including a test case for new code is strongly encouraged to ensure sustainability, but is not mandatory."

8. *198: The phrase 'minimal context' suggests that Dockers are small. However, in my experience Docker files are usually quite bulky (as they contain a complete operating system + the software needed to run the scripts). Although storage is not an issue nowadays, the wording suggests something different than what readers might expect.*

   Author reply: From https://www.docker.com/what-docker: "Using containers, everything required to make a piece of software run is packaged into isolated containers. Unlike VMs, containers do not bundle a full operating system - only libraries and settings required to make the software work are needed. This makes for efficient, lightweight, self-contained systems…". In consequence, the eddy4R Docker image is approximately one order of magnitude smaller compared to a virtual machine with comparable functionality.

   Changes in the manuscript: Clarified in Sect. 2.3: "Compared to the similar but more cumbersome virtual machine approach, a Docker image is an order or magnitude smaller (eddy4R-Docker: 2 GB without example data). Also, by running as native processes it bypasses the virtual machine overhead."

9. *211: it is unclear to what extent hub.docker.com provides versioning (in the sense that I would be able to exactly reproduce a docker that was produced 5 years ago (with the same Debian version and the same R version with the same package versions). In lines 324 and further it becomes clear that indeed you can specify a version of the docker.*

   Author reply: As mentioned by the Referee, versioning of Docker is addressed in Sect. 2.6: "The release version of the Docker image can be specified, or alternatively the specifier `latest` provides the most up-to-date development image." In addition, it is even possible to pull and run a specific digest using the `docker run stefanmet/eddy4r@sha256:` command.

   Changes in the manuscript: Added early reference to versioning in Sect. 2.3: "Using e.g. a cloud hosting platform like DockerHub (https://hub.docker.com/), the image build, versioning and distribution can be automated." And added in Sect. 2.6: "In addition, it is possible to pull and run a specific digest using the `docker run stefanmet/eddy4r@sha256:` command."

10. *I do understand that a docker image provides a machine that can do the data processing in a way that is independent of the underlying hardware and software (OS). But you do not specify how this machine (which I still would consider 'virtual') talks to the local file system (outside the docker) to read the raw EC data and write the results. Or is the data flow always supposed to pass through the NEON databases (which are accessed over a network connection).*

    Author reply: The Docker container can be mounted to a local file system when initially calling the docker run command. We have updated to the manuscript to clarify the ability to mount a local file system.

    Changes in the manuscript: In Sect. 2.6 (Installation and operation) we have added: "If data is not directed from/to cloud hosting, a physical file system location on the host computer (`local/dir`) can be mounted to a virtual file system location inside the Docker container (`docker/dir`). This is achieved with the `docker run` option `-v local/dir:docker/dir`."

11. *249: does 'uncompressed' mean 'ASCII', or does it mean 4 or 8 byte binary real values. In the first case a 1:10 compression ratio is not unexpected, in the second case I would be surprised.*

    Author reply: The data was saved in comma-delimited or .csv files in ASCII format.
    Changes in the manuscript: In Sect. 2.4 we added for clarity: "For the tower datasets analyzed in this study, including sonic anemometer, infrared gas analyzer and mass flow controller data, file sizes ranged from 1 GB for the uncompressed data in comma-delimited ASCII files to 0.1 – 0.2 GB in HDF5 format, depending on the amount of missing data."

12. *251: I am surprised that only the variable name and units are added as meta data. Since an important reason for the proposed methodology is reproducibility and traceability, I would expect that also metadata on the processing itself (software versions, tool chain, processing configurations) would be included. In that way a file with processed EC-data, when 'found in the wild' would still tell the story of how it was produced.*

    Author reply: These software configuration metadata are also included in the HDF5 files. In Sect. 2.5 they are referred to as processing parameters and described in more detail "This includes EC raw data (Level 0, or L0 data) alongside contextual information on measurement site (ParaSite), environment (ParaEnv), sensor (ParaSens), calibration (ParaCal), as well as processing parameters (ParaProc)."

    Changes in the manuscript: We have added mention of this additional metadata, now already in the HDF5 Sect. 2.4 of the manuscript: "Additional metadata are attributed

to various hierarchical groups throughout the file, including environmental parameters, sensor metadata, and processing parameters."

13. *253: I would say that text-based data storage of raw EC-data is already something of the distant past. NetCDF has gained significant usage since 15 years. When properly used, NetCDF files are self-descriptive as well (the same reservation would hold for HDF files: the meta data have to be filled). Otherwise many people use the binary file formats produced by their data logging infrastructure. So again, please indicate the real advantages of the HDF format as compared to others, vis-a-vis the requirements of your application, in combination with the DevOps framework. I could think of two aspects: compression of data (which is not possible in NetCDF) and the hierarchical data structure.*

Author reply: While the use of NetCDF has gained in popularity for some use cases, it does not appear to be prevalent throughout the community. On the other hand, storing data in binary formats necessitates converting the data before interacting with it for data analysis, which can be very cumbersome. We agree that the data compression, hierarchical data and metadata structure are important proponents for the choice of HDF5. Both, NetCDF-4 and HDF5 build on the HDF data model and are interoperable.

Changes in the manuscript: We have restructured Sect. 2.4 and added several sentences to make this clearer:

> "The capability to process large data sets is reliant upon efficient input and output of data, data compressibility to reduce compute resource loads, and the ability to easily package and access metadata. The Hierarchical Data Format (HDF5) is a file format that can meet these needs, and is a key tool aiding the DevOps: Configure (computational resource allocation) stage. A NEON standard HDF5 file structure and metadata attributes allow users to explore larger data sets in an intuitive "directory-like" structure that is based upon the NEON data product naming convention (see Figure 4). Group level 1 separates data by site and site level metadata are attributed at that level. Group level 2 separates data by data product level (DPL) and DPL metadata are attributed at that level, where DPLs correspond to the amount of processing performed. DPL1 are calibrated descriptive statistics, DPL2 are temporally interpolated, DPL3 are spatially interpolated, and DPL4 are further-derived quantities. Group level 3 are the individual data products, for instance CO_2 concentration. Lastly, replicates in the horizontal and vertical are separated as individual data tables.
>
> This provides a streamlined data-delivery mechanism for the eddy4R-Docker processing framework. For the tower datasets analyzed in this study, including sonic anemometer, infrared gas analyzer and mass flow controller data, file sizes ranged from 1 GB for the uncompressed data in comma-delimited ASCII files to 0.1 – 0.2 GB in HDF5 format, depending on the amount of missing data. The HDF5 files can be written in a simple format where data are stored

as single 1-dimensional arrays to maximize compression and efficiency, or the data can be stored as compound datatables that allow multiple datatypes to be written together in columnar format for ease of navigation when data size is not an issue.

Another important function of the HDF5 file format is the ability to attach metadata as attributes, further promoting reproducibility. The data in this study has the units and variable names as metadata attached to the data tables in the HDF5 file. Additional metadata are attributed to various hierarchical groups throughout the file, including environmental parameters, sensor metadata, and computational workflow parameters. As a result, HDF5 and similar self-documenting hierarchical data formats are gaining traction in a community that has traditionally relied on ASCII text column or comma-delimited files, especially as tools for viewing, manipulating, and extracting data from HDF5 become more commonplace. The utility of HDF5 file format is demonstrated in the executable example workflow that accompanies this manuscript (see Sects. 2.6, 5)."

14. *258: in the figure it is unclear if this depicts one file, or multiple files. Furthermore, the terminology ('group', 'Level 1,2,3', 'DPL', ...) is not explained. It is insufficient to keep the reader in the dark and just refer to the 'NEON data product naming convention'. To summarize: I do not understand what I am looking at.*

Author reply: Agreed. We have updated the manuscript to give a better description of the file structure.

Changes in the manuscript: Please see our reply to Referee comment 13 (above).

15. *268 and related text: in principle, the configuration shown here is not much different from the scripted use of a compiled program (or interpreted script): what is done by the docker images is described in the parameter files. What may be interesting is that the first (left-most) docker-image spawns the second (right-most) images, which are identical in implementation, but serve a different purpose be- cause they are instructed to do so (either the 'turbulence' task or the 'storage' task). Another important asset of the workflow is that apparently meta-data on the processing is included in the HDF files (although line 251 suggests otherwise). It would be of interest to give more details about that (how self-contained are the HDF-files?).*

Author reply: Many thanks for pointing out the one-to-many spawning property. Regarding the inclusion of metadata, please see our response to Referee comment 12.

Changes in the manuscript: We added in Sect. 2.5: "It should be noted that the "turbulence", "storage" and "combined" Docker containers (Figure 5 right) are all spawned from the same eddy4R Docker image (Figure 5 center): each container

includes the same underlying functionality (eddy4R packages), but serves a different purpose by being fed the "turbulence", "storage" or "combined" workflow files."

Regarding the inclusion of metadata, please see our response to Referee comment 13.

16. *283: I wonder why apparently a single day is used as the unit of storage when it comes to storage of L1-L4 data. In many applications I would think that longer time series are needed. Or does the data-portal glue these daily datafiles together when the data-request to the portal asks for e.g. a full year of data?*

Author reply: One day was chosen as a simply digestible and scalable unit for data processing. The NEON data portal also provides monthly concatenated files.

Changes in the manuscript: Added in Sect. 2.5: "In addition to the daily output files, monthly concatenated files are also available for download from the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0)."

17. *284: As it is shown here, the workflow is tightly integrated with the data portal. It seems to be the only use case. But what if a given researcher wants to (re-)process her/his data on her/his desktop for private use only? Is that possible as well? And how would that alter the use of the presented infrastructure?*

Author reply:

The Sect. 2.5 in question addresses "Modular compatibility with existing compute infrastructure", of which the NEON application is one example. This integration is optional, and each user can utilize eddy4R-Docker for their own purposes and in their own configuration. To highlight this aspect of extensibility, we created an executable example workflow accompanying the revised manuscript. It utilizes the functionality of both R-packages presented here, eddy4R.base and eddy4R.qaqc, and contains a user-extensible data read-in, processing and plotting workflow.

Changes in the manuscript: Sect. 2.6 now introduces the executable example workflow. Therein we demonstrate the utility of the HDF5 file format incl. example HDF5 input and output files that are already pre-compiled into the Docker image. The following paragraph was added to Sect. 2.6:

> "To demonstrate some basic capabilities and provide a template for potential eddy4R-Docker users, an executable example workflow and data are included in the eddy4R-Docker image. Once the eddy4R container is started, the example workflow, input data (NEON dp0p HDF5 file) and output data (NEON dp01 HDF5 file) are available from the Docker-internal directory `/home/eddy/`. The example workflow is located at `/home/eddy/flowExmp/flow.turb.tow.neon.exmp.dp01.R`, and provides a selection of the processing steps that yield the EC dp01 data on the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-

Docker/portal/0.2.0). The example workflow is fully documented to guide readers through the various processing steps. These include data and metadata import from the input HDF5 file, data assignment to file-backed objects, processing of 1 minute and 30 minute data statistics and data quality, and writing the output HDF5 file. In addition, outputs from the quality flag and quality metric model are visualized."

18. *294-297: this is the first point where I clearly see a reference to DevOps that links the presented software infrastructure to the advantages that DevOps could provide.*

Author reply: We thank the Referee for providing an example of how to link the DevOps methodology and associated advantages to the presented software infrastructure. As addressed in other Referee comments, we added text throughout the manuscript that provides an overview of DevOps and how it forms the basis for our software development model.

Changes in the manuscript: Changes regarding this topic have been addressed in previous replies.

19. *300: again, this figure contains a number of unexplained acronyms and terms. Furthermore, I wonder what the added value of this figure is for the paper as a whole. Finally, a part of the text is very hard to read.*

Author reply: The added value of this figure is to demonstrate the modular compatibility of the eddy4R-Docker framework with existing compute infrastructure, which is the focus of Sect. 2.5. We double-checked, and all acronyms are explained in the figure itself. The figure resolution appears sufficient for all text to be clearly legible.

Changes in the manuscript: No changes.

20. *303: section 2.6 reads as a user manual, rather than the description of the logic of operation. To me it is unclear why you would need to login to the docker machine (unless you want to develop and test new R-code. On the other hand, figure 5 suggests that the docker-images can be instructed 'from the outside' using parameter files.*

Author reply: The GMD instructions for "model description papers" require a "user manual"-like component.

The eddy4R Docker image can be used from the command line to perform batch processing. Alternatively, the RStudio interactive development environment (IDE) can be used for code development through accessing a running eddy4R Docker container via a web browser. To clarify access to the Docker container we prepared a executable example of a (simple) data read-in, processing and plotting workflow accompanying the revised manuscript. Readers can interactively run this example workflow, which

utilizes the functionality of both R-packages presented here, eddy4R.base and eddy4R.qaqc.

Changes in the manuscript:

We addressed the scaled deployment from command line options in Sect. 2.6 and linked it to the DevOps cycle:

> "The eddy4R Docker image can be used for code development (DevOps: Create stage) through accessing a running eddy4R Docker container via a web browser, as described above. Alternatively, the eddy4R Docker image can be used from the command line to perform scaled batch processing (DevOps: Configure & Monitor stages). Deployment from the command line consists of passing the R workflow file to the Docker container being deployed. This is achieved by using the `docker run` command with the additional argument `Rscript docker/dir/filename.R`. Thus, the eddy4R Docker image can be used to simultaneously deploy multiple Docker containers to process data for multiple days or sites to the capacity of the computational platform."

In addition, we introduce the executable example workflow in Sect. 2.6. Please see our reply to Referee comment 17 for details.

21. *41 and further: to me it is unclear what the function of the three case studies is. In the introductory paragraph details on the computing infrastructure is given. This suggests that some benchmarking in terms of performance will be done. However, this only makes sense to me if there is a standard against which the new setup can be compared (e.g. current practice of reading ASCII files on a single processor machine?).*

Author reply: The purpose of the test applications in Sect. 3 is to demonstrate the applicability and usefulness of the DevOps software development model to various EC use cases. Software benchmarking is one of several aspects of such evaluation, and absolute performance results are provided in Sect. 3.1.1. As explained in more detail in our response to Referee comment 34, a direct performance intercomparison from these results is not meaningful, as hardware and software process implementation differed among and within test applications.

Changes in the manuscript: Added clarification in Sect. 3 "In the following we present three test applications of eddy4R-Docker to evaluate whether the NEON DevOps model can indeed produce collaborative, portable, reproducible, and extensible eddy covariance software."

In addition, throughout Sect. 3 we now use our results to evaluate the desired DevOps software properties of portability, reproducibility and extensibility.

22. *349 and further: if indeed the performance of the software (in terms of speed, and perhaps also ease of use, flexibility . . .) is the objective of section 3, I do not see why so much attention needs to be paid to the experimental setup (including photographs!)*

as well as the interpretation of the results (section 3.1.2). Any EC dataset with a length of 1 to 2 weeks would have sufficed.

Author reply: As explained in our response to Referee comment 21, software performance is one of several aspects to evaluate the usefulness of the DevOps software development model for EC applications. More importantly, the modular applicability of eddy4R-Docker to substantially different site setups, measurement platforms and even end-to-end scientific hypothesis testing sets it apart from existing software solutions. As such, clearly identifying the differences in setups is an important component of demonstrating the value of the DevOps approach. Moreover, Referee #3 has inquired additional detail, and we feel that the current presentation strikes a balance between these viewpoints.

Changes in the manuscript: Please see our response to Referee comment 21.

23. *381: what kind of additional complexity? It could be that the presented processing infrastructure makes it easy to logically define different processing levels (L1 to L4). In that case it would be of interest to the reader to clarify which are the differences between the various processing levels, and how, conceptually, they can be defined and configured with the current setup. Perhaps this is an important added flexibility? In line 400 it becomes at least clear what is the distinction between L1 and L4.*

Author reply: Complexity to test assumptions of the EC theory. The full sentence reads "The algorithmic processing for the L4 flux calculations requires additional scientific and procedural complexity to test assumptions of the EC theory."

Changes in the manuscript: For additional definition of the processing levels, please see our reply to Referee comment 13.

24. *386: what is the difference between a simple data format and a compound data format. Since the use of the HDF5 file format is such an important aspect, I would expect a clearer description of the way in which the added possibilities of HDF5 files are used (which meta-data, mixes of L1, L2 . . . data . . . .).*

Author reply: We are using the simple HDF5 format to produce 1-dimensional arrays, each with a single datatype. In contrast, HDF5 compound datatables allow multiple datatypes to be written together in columnar format.
Changes in the manuscript: In Sect. 2.4 we added information to describe this functionality, please see our response to Referee comment 13 above.

25. *389-390: does this mean that the calibration step (to go from L0 to L0p) takes 4.8 PU minutes? Compared to the actual processing this seems long. But perhaps there is a good explanation for it. Please provide the reader with one (or perhaps my interpretation of the numbers is incorrect).*

Author reply: In the processing from L0 to L0p the plausibility tests per Taylor and Loescher (2013) are performed. Of these, the step-test is the main resource consument.

Changes in the manuscript: Added in Sect. 3.1.1 “This difference arises mainly from application of plausibility tests per Taylor and Loescher (2013) in the transition from L0 to L0p.”

References:

Taylor, J. R., and Loescher, H. L.: Automated quality control methods for sensor data: A novel observatory approach, Biogeosciences, 10, 4957-4971, doi:10.5194/bg-10-4957-2013, 2013.

26. *399 and further: only include these results if they show something that only this software can do, and other EC-processing methodologies cannot.*

Author reply: To our knowledge, no other software methodology currently permits modularly producing results for the tower and aircraft test applications. Please also see our reply to Referee comment 21.

Changes in the manuscript: No changes.

27. *423: the fact that the presented methodology apparently is equally well able to deal with aircraft data as it is can deal with tower data is an interesting added value, worth advertising! However, I would have appreciated it if the authors would have clearly shown which parts of the processing would work the same for tower and aircraft data, and which steps would be different (e.g. by referring to figure 5). Again, the details of the particular data set (conditions of collection) is less important than the type of data and instruments involved.*

Author reply: In contrast to existing flux processors, eddy4R-Docker enables any arbitrary data analysis. This is accomplished by spawning a Docker container with user-defined workflow files. These R-files define not only the parameter settings for the various processing steps and models, but the presence, absence and sequence of the individual processing steps themselves. In comparison to existing methodologies, the underlying eddy4R definition and wrapper functions are pre-compiled into the eddy4R-Docker image, and modularly shared and identical among the various applications. As such, each container spawned from the eddy4R-Docker image includes the same underlying functionality (eddy4R packages), but can serve a different purpose by being fed a different workflow file. Please also see our reply to Referee comment 15. The functions contained in the eddy4R.base and eddy4R.qaqc packages are published here, alongside an executable example workflow. How these functions are used together depends on the specific application and is user-defined. The example workflow

accompanying the revised manuscript serves as a template for user-specified implementations such as tower, aircraft, and various other potential use cases.

For the description of the datasets and site conditions, please see our reply to Referee comment 22.

Changes in the manuscript: We now introduce the executable example workflow in Sect. 2.6. Please see our reply to Referee comment 17 for details.

28. *442: it is unclear whether the 100Hz → 20Hz reduction is done by the same software or was part of the preparation of the L0 data (i.e. from L(-1) to L0.*

Author reply: These steps were performed in pre-processing, and were not part of the eddy4R processing.

Changes in the manuscript: Added clarification in Sect. 3.2 "These steps were performed prior to import into eddy4R processing, but could equally well be performed therein."

29. *445: it is unclear whether the software is able to perform all the corrections that are related to the accelerations of the aircraft (the wind speeds derived from the pressure readings are not the wind speeds). I could imagine that this is part of the L0 → L0p conversion, but this is not specified.*

Author reply: In the presented case, derivation of the 3-D wind vector was part of the L0 → L0p processing prior to ingest into eddy4R. Functionality to convert pressure and inertial motion readings to wind speeds exists in eddy4R (e.g., Metzger et al, 2011), but are not part of this release.

Changes in the manuscript: Clarified in Sect. 3.2 "The input data used in this study included the pre-derived 3-D wind vector from 5-hole probe and aircraft position, velocity and attitude."

References:

Metzger, S., Junkermann, W., Butterbach-Bahl, K., Schmid, H. P., and Foken, T.: Measuring the 3-D wind vector with a weight-shift microlight aircraft, Atmos. Meas. Tech. Discuss., 4, 1303-1370, doi:10.5194/amtd-4-1303-2011, 2011.

30. *449: it was earlier suggested that the workflow is based on HDF5 files, whereas here gzipped ASCII files are used. I understand that R is able to read all kinds of files, but is reading the ASCII files a standard feature of the methodology or did you first convert the gzipped ASCII files to HDF5 (to produce real L0 data) before ingesting them into the processing software?*

Author reply: The data ingest mechanism into eddy4R is user-definable. For the aircraft test application in Sect. 3.2, gzipped ASCII files were directly imported (without prior conversion to HDF5). In contrast, for NEON's test application in Sect. 3.1 HDF5 files are used. While HDF5 is conceptually preferable because self-describing and read/write efficient, the usability of various file formats highlights the modularity of eddy4R. As Sect. 3.2.1 explains "…standard R capabilities for data ingest can be used to read data in various formats, frequencies and units."

Changes in the manuscript: No changes.

31. *451-460: This is an interesting, perhaps non-standard, chain of processing steps. For the reader this would be an interesting example to see how easily this can be configured in your software. Is it just a matter of setting a number of flags in your configuration files? How easily can you change the order of various steps that are applied to high rate data or reduced data?*

Author reply: Please see our reply to Referee comment 27.

Changes in the manuscript: Please see our reply to Referee comment 27.

32. *Section 3.2.2: again (like for 3.1.2): presentation of the results is only relevant if your methodology can do something that existing software cannot (or if yours can do it better/faster/easier/more insightful). For instance, if figure 11 would be output that can be produced automatically it could be of added value. In that respect it would also be interesting to know if the software could be used in a scenario where certain processing has already been done, and the user decides that he/she wants additional output. Would the software be able to figure out which data (L1. . . L4) is already there, and which is the minimum set of additional processing steps needed to produce the additional output. Again, in that sense it would be worthwhile for the reader (and potential user) to know which output your software can produce, based on which type of input data (type as in L0, L0p, etc). A well-organized table would be more informative than three case studies. Similarly a clear definition of which types of input data can be ingested would be helpful.*

Author reply: Please see our replies to Referee comments 21, 22 and 27. One key attribute of the eddy4R-Docker methodology is its user-extensibility per requirements of the desired application. As such, no default inputs exist. To demonstrate the complementarity of eddy4R-provided functions and user-supplied workflow files, input data is instead defined individually for each test case in Sects. 3.1, 3.2 and 3.3.

Changes in the manuscript: Please see our replies to Referee comments 21 and 27.

33. *499: this is an interesting application! Does the script automatically select the EddyPro settings that would match the eddy4R settings? If so, providing the difference statistics*

with your L1-L4 output in the HDF5 file would be useful added value, since users of your L1-L4 data would have an indication how sensitive the resulting aggregated data are to details of the processing (he/she does not know who is wrong and who is right, but a sense of the sensitivity can be helpful).

Author reply: The comparison between EddyPro and eddy4R outputs is a test application and not done automatically. The results have been calculated based on the same input dataset and processing settings, but by separate institutions and on different computer platforms. Please see our reply to Referee comment 34 for details.

Changes in the manuscript: No changes.

34. *519: Do you run EddyPro in the same docker as the R-framework? It would be interesting to know what the performance difference (if any) is between EddyPro and your R-based software.*

Author reply:

EddyPro was run natively in Windows. The calculations were performed independently at LI-COR (EddyPro) and U Wisconsin (eddy4R-Docker), with identical settings and based on the same input dataset as specified in Sect. 3.3. Absolute benchmarking is possible and summarized in below table. However, a direct performance intercomparison from these results is not meaningful, as both, the hardware and the software process implementation differed substantially: EddyPro used two independent processes and required 33 s to process one day of data on a laptop with a solid-state drive. On the other hand, eddy4R was run as a single process and required 68 s to analyze one day of data on a server with 100 Mbit interconnect speed to a RAID Level 5 data share. Put against a rapid-access solid-state drive, the interconnect speed alongside the single/dual process implementation is likely dominating the difference in processing times.

Metric	**EddyPro**	**eddy4R**
File size per day and format	182 MB in ASCII format	182 MB in ASCII format
Utilized processor type and number	2 x Intel (R) Core (TM) i7-4600U CPU @2.1GHz	1 x Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
Available memory	16 GB	200 GB
Storage type and size	512 GB SSD	40 TB in RAID Level 5 configuration
Interconnect speed (if server)	NA	100 Mbit/s
Operating system	Windows 10 enterprise 64-bit	Linux 2.6.32-642.1.1.el6.centos.plus.x86_64

CPU time per day of data	33 s	68 s
Maximum memory use	120 MB	200 MB

Changes in the manuscript: No changes, as a direct performance intercomparison from the benchmarking results is not meaningful.

35. *550-560: it is OK for me that in the paper you present the workflow as it is envisaged. But it would be good to already clarify in the main text which steps have already been implemented and which are underway (for the eddy4R part you clearly made this distinction, but for the other parts not). Furthermore, it remains unclear to me to what extent the workflow you propose is strongly tied to the NEON infrastructure. Or could I, as a scientist not working within NEON, be able to adopt your methodology as well. To pose it differently: where does your methodology end and where does the NEON infrastructure start?*

Author reply: The eddy4R-Docker methodology is a generalized implementation of an EC software using the DevOps model. eddy4R-Docker is thus not tied to NEON, and openly available to everyone. Rather, NEON's implementation is one example of a scaled application of tower EC processing using the eddy4R-Docker methodology (Sect. 3.1). As such, the current status of NEON's implementation does not contribute to the objective of the manuscript, to "demonstrate the applicability of the DevOps model to science code development". eddy4R-Docker can be easily adopted for various use cases and applications by directly adjusting its workflow file. Please see our detailed replies to Referee comments 15, 17 and 27.

Changes in the manuscript: Please see our replies to Referee comments 15, 17 and 27.

3 Very detailed comments

1. *63: please add that NEON is being developed in the US.*

Author reply: We agree this detail is appropriate to include.

Changes in the manuscript: The sentence in question now reads: "The U.S.-based National Ecological Observatory Network"

2. *68: what do you mean by 'tight hardware-software integration'? And which code is 'distributed' and where?*

Author reply: By 'distributed and dynamic community developed code', we mean that code is written by multiple people in multiple places. By tight integration of software

and hardware we mean that improvements in data collection (e.g. spectral characteristics) and data processing (e.g., time-frequency-based spectral correction) can be closely aligned. This yields an overall optimal performance.

Changes in the manuscript: The sentence in question now reads: "This capability is accompanied by a strong need for a flexible and scalable processing framework that can incorporate specific data streams, take advantage of close alignment of hardware and software for problem tracking and resolution, provide traceability and reproducibility of outputs, and seamlessly integrate distributed and dynamic community-developed code (written by multiple people in multiple places) within existing cyberinfrastructure.

3. *189: I think figure 3 is superfluous. The message conveyed by the caption could be included in the text. No illustration needed.*

Author reply: We agree that Figure 3 can be removed without losing much information.

Changes in the manuscript: Removed Figure 3.

4. *277: for people working in the field of micrometeorology (and particularly when involved in CO2 exchange) the terms "turbulence" and "storage" will ring a bell. But for people somewhat more remote, but interested in the proposed collection of tools, it will be less clear (in the context of a strongly IT-oriented paper "storage" might mean something completely different to the reader). For the clarity of the paper the distinction between the turbulence and storage workflows is in fact irrelevant. The message is simply that based on the same code/machinery you can do different things with the same data Unless you want to show that you can have various parallel workflows which could also interact (but then you have to clarify the potential interactions).*

Author reply: Turbulent exchange and storage exchange computations are basic micrometeorological capabilities of the eddy4R family of R-packages for EC data processing, and introduced in Sect. 2.1. In the paragraph in question in Sect. 2.5 these are set in relation to each other (Fig. 5), incl. their interface through the Docker container "derived". Lastly, they are set in context once more during the outlook in Sect. 4 "Summary and conclusions".

Changes in the manuscript: Given that the terminology is central to the developed EC capability and described in text, no changes.

5. *317: I think that figure 7, apparently just a collection of screenshots, is superfluous.*

Author reply:

The GMD instructions for "model description papers" require a "user manual"-like component, and this figure presents our user interface: it permits to show how easy the Docker approach makes preparing the development environment. We are under the impression that retaining this figure provides clarity for some readers.

Changes in the manuscript: No changes.

6. *376: it is unclear to me how the current setup would lead to 50 high-rate data streams.*

Author reply: Including sensor diagnostics, these are ~10, ~10, ~20 and ~10 data streams from the sonic anemometer, attitude and motion reference system, infrared gas analyzer, and mass flow controllers and solenoids, respectively.

Changes in the manuscript: No changes

7. *385: the 0.1 to 0.2 Gb: I suppose that this refers to the input files (L0 or L0p).*

Author reply: Correct.

Changes in the manuscript: Clarified: "…the L0p input file sizes ranged from 0.1 – 0.2 GB…"

8. *459: the performance is only relevant if it can be compared to something else*

Author reply: Please see our reply to Referee comment 34 (above).

Changes in the manuscript: Please see our reply to Referee comment 34 (above).

9. *460: what do you mean with 'file-backed objects'?*

Author reply: R's ff file-backed object (https://cran.r-project.org/package=ff) facilities.

Changes in the manuscript: Now introduced in more detail in Sect. 2.1: "In addition, eddy4R is fully parallelized and memory efficient leveraging R's snowfall parallelization (https://cran.r-project.org/package=snowfall) and ff file-backed object (https://cran.r-project.org/package=ff) facilities, respectively."

10. *565: what do you mean by defensible?*

Author reply: Thanks for pointing this out.

Changes in the manuscript: Changed to "…for generating reproducible net ecosystem exchange data products…"

References

Mauder, M., Foken, T., Clement, R., Elbers, J. A., Eugster, W., Gr, T., ... & Kolle, O. (2008). Quality control of CarboEurope flux data–Part 2: Inter-comparison of eddy-covariance software. Biogeosciences, 5, 451-462.

Author reply to the comments by Anonymous Referee #2 of the manuscript

gmd-2016-318

"eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5"

by S. Metzger et al.

We thank Anonymous Referee #2 for the valuable feedback, which helped to improve the manuscript. Please find below the Referee comments recited in *blue, italics font*, followed by our point-by-point replies and corresponding changes in the manuscript in black, upright font.

Metzger et al. describe a data processing framework which is illustrated on ad- hoc examples from NEON's eddy covariance tower and airborne measurement datasets. Overall this technical concept seems potentially valuable for streamlining automation of specific data processing steps from different measurement stations but it is extremely difficult to recognize the broader scientific values in the current version of the paper as written. I must admit that I was rather disappointed to find the description of the tools to be fragmented and poorly supported by the scientific results and conclusions. The whole analysis is very descriptive and in many cases misleading as to what is possible. There is little effort to synthesize what an be actually learnt from using the tool other than what its potential applications might be in the future. Most importantly, the paper does not specify scientific goals and does not even address the scope of modeling which is what the main interest of the journal's audience is. The manuscript seems to need much work to make the results and discussion useful for the scientific community but could be worthwhile to reconsider after major clarifications. The other reviewer has already provided a useful detailed guidance how this could be achieved and I agree with her/him. I also have other concerns which hopefully can be addressed in the revision.

Author reply: Many thanks for your summary. Please find specific replies below. As detailed in the responses to Referee #1, we have better clarified the problem statement of the paper and why the DevOps approach and specific tools used to implement are the answer. The problem statement is: "How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of eddy covariance data products?" We believe this clarification addresses the main concern of this Referee – that there is little of use to the scientific community presented by the description of software tools – by better communicating that the focus of the paper is not on the specific software implemented, but a model of

how the EC community can go about creating portable, reproducible, and extensible software.

As such the manuscript addresses a methodological rather than scientific question. For this reason, the GMD journal was chosen, and three tests of geoscientific applications are provided in favor of a single in-depth scientific survey. These applications serve not as scientific results, but as tests that the software is portable, reproducible, and extensible. One core component of GMD model description papers is the "…evaluation against standard benchmarks…" which is addressed in Sect. 3.3.

Changes in the manuscript: Please see responses below and also to Referee #1.

General issues:

*1) One fundamental issue in this paper intended for GMDD is that the work is not even connected to any model or modeling framework. The journal scope does not overlap with what paper is about or at least the connection is not made clear. Because there is no model, there is no model version – a requirement of the journal. There are only two words "model" in the whole paper, one of which is included in the last sentence of conclusions but probably in a different meaning: "We hope this framework can serve as a *model* for implementing community-sourced, distributed-development scientific code while combatting the deficiencies of current computational frameworks that limit accessibility, reproducibility, and extensibility."*

Author reply: The authors considered several journals before deciding where to submit our manuscript, and we came to this decision through taking into account the manuscript types requested on the Geoscientific Model Development (GMD) webpage. Specifically, we felt that our paper provides "…utility tools … such as coupling frameworks … with a geoscientific application".

In addition, as detailed in the replies to Referee #1, we clarified the problem statement of the paper: "The question we ask in this paper is: How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of eddy covariance data products?" We then introduce the DevOps approach in more detail and how it, along with the specific tools implemented in the eddy4R-Docker development model, solves this problem. The framework provides modular processing for surface-atmosphere exchange data with quality assurance and quality control as foundation for modelling exercises such as the test application in Sect. 3.2. This includes footprint modeling (GMD: Kljun et al., 2015), evaluation of large eddy simulations (GMD: Maronga et al., 2015), machine learning etc. The result is an end-to-end framework for model building, parameterization and assessment considering the large amounts of theoretical assumptions in eddy-covariance technique that require corrections to the data. The combination of these tools to address the concern of reproducibility was a major consideration when submitting to GMD.

Per suggestion of Referee #2 as well as the executive editor, in addition to Sect. 5 Code and data availability we now include the eddy4R-Docker development model version (now: 0.2.0) also in the manuscript title.

We further clarify in the revised manuscript that eddy-covariance data processing consists of employing a sequence of model algorithms. These often originate from scientific sub-fields with corresponding publications, and eddy4R-Docker provides an integrative, yet modular and extensible framework for their concerted application and continued development. In its current form eddy4R-Docker 0.2.0 encompasses the following models: plausibility tests (Taylor and Loescher, 2013), de-spiking (Brock, 1986), lag correction, data aggregation, and QA/QC budgeting (Smith et al., 2014).

Additional models are in preparation for future extension of the eddy4R-Docker framework presented here: coordinate rotation (Wilczak et al., 2001), spectral correction (Nordbo and Katul, 2012), turbulent mixing and stationarity (Foken and Wichura, 1996), detection limit (Billesbach, 2011), turbulent sampling error (Lenschow et al., 1994), footprint analysis (Kljun et al., 2015), storage flux term, and uncertainty budgeting.

Please note that e.g. Kljun et al. (2015) is itself published in GMD.

Changes in the manuscript: We clarify the objective of the paper in the introduction: ”The question we ask in this paper is: How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of eddy-covariance data products?”

We then added the following text later in Sect. 2: ”Thus, the DevOps model serves as the framework within which the scientific community can efficiently and robustly collaborate to produce, manage, and iterate community software. Through choosing appropriate tools to implement the DevOps workflow steps, the reproducibility, scalability and extensibility needs of software development communities (including EC) can be met.”

Lastly, we link the manuscript to the GMD literature realm and further clarify in Sect. 2.1: “Eddy-covariance data processing consists of employing a sequence of models. These often originate from scientific sub-fields with corresponding publications, and eddy4R provides an integrative, yet modular and extensible framework for their concerted application and continued development: eddy4R.base provides natural constants and basic functions for usability, regularization, transformation, lag-correction, aggregation and unit conversion ensuring consistency of internal units at any point in the workflow. Next, eddy4R.qaqc provides the general quality assurance and quality control (QA/QC) tests of Taylor and Loescher (2013), along the Smith et al. (2014) model for tracking quality information in large datasets, and functions for de-spiking (Brock, 1986; Fratini and Mauder, 2014; Mauder et al., 2013; Mauder and Foken, 2015; Metzger et al., 2012; Vickers and Mahrt, 1997). eddy4R.turb provides standard, Reynolds-decomposed turbulent flux calculation (Foken, 2017), accompanied by models for planar fit transformation (Wilczak et al., 2001) and spectral correction (Nordbo and Katul, 2012). Additional functionalities include Fourier transform, the determination of detection limit (Billesbach, 2011),

integral length scales and statistical sampling errors (Lenschow et al., 1994), and flux-specific QA/QC models (Foken and Wichura, 1996; Vickers and Mahrt, 1997). Also, basic scaling variables, atmospheric stability and roughness length (Stull, 1988), as well as the flux footprint (Kljun et al., 2015; Kormann and Meixner, 2001; Metzger et al., 2012) can be determined. Lastly, edd4R.erf provides time-frequency de-composed flux processing and artificially intelligent functionality to determine an environmental response function model and project the flux fields underlying the EC observations (Metzger et al., 2013; Xu et al., 2017)."

References:

Billesbach, D. P.: Estimating uncertainties in individual eddy covariance flux measurements: A comparison of methods and a proposed new method, Agric. For. Meteorol., 151, 394-405, doi:10.1016/j.agrformet.2010.12.001, 2011.

Brock, F. V.: A nonlinear filter to remove impulse noise from meteorological data, J. Atmos. Oceanic Technol., 3, 51-58, doi:10.1175/1520-0426(1986)003<0051:anftri>2.0.co;2, 1986.

Foken, T., and Wichura, B.: Tools for quality assessment of surface-based flux measurements, Agric. For. Meteorol., 78, 83-105, doi:10.1016/0168-1923(95)02248-1, 1996.

Kljun, N., Calanca, P., Rotach, M. W., and Schmid, H. P.: A simple two-dimensional parameterisation for Flux Footprint Prediction (FFP), Geosci. Model Dev., 8, 3695-3713, doi:10.5194/gmd-8-3695-2015, 2015.

Lenschow, D. H., Mann, J., and Kristensen, L.: How long is long enough when measuring fluxes and other turbulence statistics?, J. Atmos. Oceanic Technol., 11, 661-673, doi:10.1175/1520-0426(1994)011<0661:HLILEW>2.0.CO;2, 1994.

Maronga, B., Gryschka, M., Heinze, R., Hoffmann, F., Kanani-Sühring, F., Keck, M., Ketelsen, K., Letzel, M. O., Sühring, M., and Raasch, S.: The Parallelized Large-Eddy Simulation Model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, recent developments, and future perspectives, Geosci. Model Dev., 8, 2515-2551, doi:10.5194/gmd-8-2515-2015, 2015.

Nordbo, A., and Katul, G.: A wavelet-based correction method for eddy-covariance high-frequency losses in scalar concentration measurements, Boundary Layer Meteorol., 146, 81-102, doi:10.1007/s10546-012-9759-9, 2012.

Smith, D. E., Metzger, S., and Taylor, J. R.: A transparent and transferable framework for tracking quality information in large datasets, PLoS One, 9, e112249, doi:10.1371/journal.pone.0112249, 2014.

Taylor, J. R., and Loescher, H. L.: Automated quality control methods for sensor data: A novel observatory approach, Biogeosciences, 10, 4957-4971, doi:10.5194/bg-10-4957-2013, 2013.

Wilczak, J. M., Oncley, S. P., and Stage, S. A.: Sonic anemometer tilt correction algorithms, Boundary Layer Meteorol., 99, 127-150, doi:10.1023/A:1018966204465, 2001.

2) It is not apparent how exactly this technical set of workflows adopted by NEON can be useful for a broader scientist/modeler community and what scientific problems it can solve as the idea wraps around different open-source products dedicated essentially to crunching of eddy covariance measurement data. In the abstract, it is promised that the framework is applicable beyond EC but it is completely unclear how. Maybe one way to overcome this issue would be to make a strong connection to a modeling framework where measurement and model outputs are evaluated together or elucidate aspects where this data processing framework would add to novelty and usefulness for the broader GMD community

Author reply: This suite of packages and Docker image is meant to provide a modularly extensible flux processing platform as foundation for modeling exercises (see reply above). The presented framework is motivated by a lack of collaborative coding and processing code development in the eddy-covariance community.

To address the Referee comment and demonstrate the ease of applied modelling, we prepared an executable example workflow that accompanies the revised manuscript and executes the QA/QC model by Smith et al. (2014).

The underlying functions are already included in the eddy4R.qaqc package. In addition, we highlight the extensibility that can be achieved with the modular packaging of the eddy4R-Docker framework: the eddy4R family of packages already includes the Environmental Response Function (ERF) model for flux upscaling to the landscape (manuscript Sect. 3.2.2), scheduled for future release.

Changes in the manuscript: We now introduce the executable example workflow in Sect. 2.6. Please see our reply to Referee #1, comment 17 for details.

References:

Smith, D. E., Metzger, S., and Taylor, J. R.: A transparent and transferable framework for tracking quality information in large datasets, PLoS One, 9, e112249, doi:10.1371/journal.pone.0112249, 2014.

3) There are no clear scientific objectives of the paper and the title does not help either "eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5". The use of "modeling framework" is misleading (see also comment 1) because the paper fails to present any modeling or prediction which could be achieved from this framework.

Author reply: The aim of manuscript is to introduce the novel eddy4R-Docker software framework to address a methodological rather than scientific question: the portable, reproducible and extensible processing of eddy-covariance data. For this reason, the GMD journal was chosen, and test applications to three geoscientific use cases are provided in favor of a single in-depth scientific survey.

Based on the GMD manuscript types specifications, as well as existing papers from our community (e.g., Kljun et al, 2015; Maronga et al., 2015), we are under the impression that scientific hypothesis testing is not a typical component of a GMD model / framework description paper. On the other hand a core component of GMD model description papers is "…evaluation against standard benchmarks…" which is addressed in Sect. 3.3.

Changes in the manuscript: Please see reply to Referee comment 1) (above).

References:

Kljun, N., Calanca, P., Rotach, M. W., and Schmid, H. P.: A simple two-dimensional parameterisation for Flux Footprint Prediction (FFP), Geosci. Model Dev., 8, 3695-3713, doi:10.5194/gmd-8-3695-2015, 2015.

Maronga, B., Gryschka, M., Heinze, R., Hoffmann, F., Kanani-Sühring, F., Keck, M., Ketelsen, K., Letzel, M. O., Sühring, M., and Raasch, S.: The Parallelized Large-Eddy Simulation Model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, recent developments, and future perspectives, Geosci. Model Dev., 8, 2515-2551, doi:10.5194/gmd-8-2515-2015, 2015.

4) The story basically presents a rather ambitious idea of automating data processing including quality control. The latter is not shown yet that it already works well so the product is not yet ready to be fully useful for the community. Once QC is implemented it could be interesting to see how it is done and how flexible the options are for the user. For instance, on page 14 L32 it is concluded "Once scientific QA/QC and uncertainty budget is implemented, the computational expense will likely increase by a factor of two to three. This suggests that eddy4R performs comparably to other flux processors." As presented, the value from another EC flux processor tool is unclear in where it would really help but what is interesting is that the development is directed to a modeling audience who might also be able to use this tool if it was better explained. However, without clearly stated goals and sufficient supporting material to assess its quality and usefulness, it is difficult to evaluate the code framework for all its ambitious features. The paper is incoherent in its presentation (e.g. different components, datasets are presented separately without a clear thread creating multiple fragmented methods and results) and in many places the quality is diverging from the standards of a scientific paper.

Author reply: We agree that the implementation of the QA/QC framework substantially adds to the novelty of the methods included in this initial release of the eddy4R software. The QA/QC framework to deal with plausibility tests on the data is now fully implemented. Additional flux QA/QC tests are still be refined to accompany the full suite of eddy4R packages that are being released with the completion of NEON Construction. We hope to address the main concern by providing an example workflow accompanying the revised manuscript, which include the Taylor and Loescher (2013) high-frequency plausibility test model alongside the Smith et al. (2014) model for consolidating the results to a final quality flag. This highlights some of the capabilities of the eddy4R.base and eddy4R.qaqc packages, and provides a user-accessible and

modifiable workflow template. Please also see our detailed QA/QC replies to Referee comment 6).

The different test applications are central to proving the flexibility of the eddy4R-Docker framework to process both tower and aircraft flux data. They demonstrate that the DevOps approach can be used in scientific software development.

Changes in the manuscript: We now introduce the executable example workflow in Sect. 2.6. Please see our reply to Referee #1, comment 17 for details.

References:

Smith, D. E., Metzger, S., and Taylor, J. R.: A transparent and transferable framework for tracking quality information in large datasets, PLoS One, 9, e112249, doi:10.1371/journal.pone.0112249, 2014.

Taylor, J. R., and Loescher, H. L.: Automated quality control methods for sensor data: A novel observatory approach, Biogeosciences, 10, 4957-4971, doi:10.5194/bg-10-4957-2013, 2013.

Specific issues

5) The number of figures seems rather large and not all of them seem necessary. A heavy detail from different settings and configurations (e.g. Sect. 3.1.1, 3.2.1) could be nicely summarized in a table. The examples in Figures 9-13 require specific understanding of eddy covariance and do not help a modeler to adjust the framework for their needs. Even for the eddy covariance community, it might seem surprising that airborne and tower data can be automatically compared, because there is a comprehensive quality control that needs to be performed on these data and it is not easy to automate such EC comparison at different scales (e.g. Mahrt, 1998), at least without multiple user interactions. For example, in Sect. 3.3 "Validation and verification" it is stated: "eddy4R includes a verification script which automatically processes subsets of the tower and aircraft data introduced in Sect. 3.1 and Sect. 3.2, and verifies the results against a reference, e.g. generated with a different software." Where is this validation shown? Do you actually mean that you duplicate the processing (e.g. also with Eddy Pro) or just check selected files for consistency? The agreement in Figure 13 definitely seems surprising. It almost looks like the same dataset was plotted against the same dataset? The significant figures inconsistently range from 1 to 5. R2=1 is surprisingly good but not too meaningful (did you mean 1.0000, 0.9999 or 0.99)? I am also confused why the measured variables (e.g. w, q, CO2 mixing ratio) are compared with each other as they should have been the same unless the software interferes with the measurement data.

Author reply:

We agree that Figure 3 can be removed without losing much information.

The intent of this paper has been clarified to demonstrate the applicability of the DevOps model to EC science code development (please see our reply to Referee comment 3 for details). One key attribute of the eddy4R-Docker methodology is its

user-extensibility per requirements of the desired application. As such, no default workflow or settings exist that could be easily tabulated across applications. As mentioned by the Referee, specifics differ substantially e.g. among the tower and aircraft use cases. To demonstrate the complementarity of the eddy4R-provided functions and user-supplied workflow files, the corresponding workflows and settings are thus documented individually for each test case in Sects. 3.1, 3.2 and 3.3. The test applications in Figures 9 – 13 are central evidence to the claim of adjustability and expansibility.

We agree with the Referee that tower and aircraft data require careful QA/QC and interpretation, and in no part of these test applications airborne and tower data are automatically compared.

Rather, a comparison takes place as part of the DevOps **Verify** step: reference datasets generated with EddyPro have been stored and automated tests are performed prior to new code being incorporated. The results are shown in Sect. 3.3: calculations were performed independently at LI-COR (EddyPro) and U Wisconsin, with identical settings and based on the same input dataset as specified in the manuscript. Four significant digits were specified in the plotting routine for Figure 13, and the output of any uninformative zeroes is consequently suppressed. As discussed e.g. in Mauder et. al. (2008), discrepancies exist among software implementations also for the calculation of averages and variances, which we thus show alongside their corresponding fluxes.

Changes in the manuscript: Removed Figure 3.

References:

Mauder, M., Foken, T., Clement, R., Elbers, J. A., Eugster, W., Grunwald, T., Heusinkveld, B., and Kolle, O.: Quality control of CarboEurope flux data - Part 2: Inter-comparison of eddy-covariance software, Biogeosciences, 5, 451-462, 2008.

6) The data quality control does not seem careful. For example, in Figure 9 the periods of latent heat and CO2 flux were not rejected when the friction velocities were at their minima which look definitely below 0.1 m/s on days 114, 115, 116, 119, and other. It is also unclear what the gaps correspond to (rejected data, power interruption). I would be surprised if it was not possible to choose an uninterrupted dataset in the NEON's large EC measurement network. It also seems weird in the same figure that the general temperature trend is anticorrelated with sensible heat flux (doy 115-118). Does the data output use normalized flux units for CO2?

Author reply:

We absolutely agree that thorough QA/QC is paramount for evaluating scientific findings. This is especially true for measurement techniques involving substantial theoretical models and assumptions, such as eddy-covariance. However, the objective of this paper is to demonstrate the applicability of the DevOps model to scientific code development, and not the publication of scientific findings. Please refer to our responses to previous Referee comments for this clarification as part of the manuscript revision.

NEON sites are just beginning to collect eddy-covariance data, which are considered engineering-grade until Construction and Commissioning of the Observatory is complete. At this time, first, provisional NEON eddy-covariance data products produced by eddy4R-Docker are available for download on the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0).
Analogous to the manuscript presentation, these data are accompanied by both, provisional quality and uncertainty summaries following the models of Salesky et al. (2012), Smith et al. (2014) and Taylor and Loescher (2013).

The results in this manuscript are shown identical to directly displaying the data downloadable from the NEON data portal, without sub-setting for quality or uncertainty. This highlights the need for the data user to determine and select the acceptable level of quality and uncertainty based on the particular use case (analogous to e.g. MODIS quality flags). During the remaining software development steps throughout NEON Construction, dedicated flux QA/QC metrics are being added to the already implemented plausibility tests. These are currently residing in the eddy4R.turb package, which is not released, and hence not applied here. Please also see our reply to Referee comment 4).

To address your last question about the temperature trend and sensible heat flux, we see the DOY 117 night was a very turbulent night with a lot of mixing resulting in less radiative cooling. This may explain some of the decorrelation.

Changes in the manuscript: In Sect. 3.1.2 we have clarified that the full flux QA/QC and uncertainty budget needs to be applied: “The spiky results preceding and following periods with >10% invalid data highlight the need for applying the full flux QA/QC and uncertainty budget to provide science-grade fluxes.”

Many thanks for catching the mal-formatted unit of the CO_2 flux. We have corrected the figure axis label to µmol m^{-2} s^{-1}.

Additionally, please also see replies to above Referee comments for changes to the manuscript.

References:

Salesky, S., Chamecki, M., and Dias, N.: Estimating the random error in eddy-covariance based fluxes and other turbulence statistics: The filtering method, Boundary Layer Meteorol., 144, 113-135, doi:10.1007/s10546-012-9710-0, 2012.

Smith, D. E., Metzger, S., and Taylor, J. R.: A transparent and transferable framework for tracking quality information in large datasets, PLoS One, 9, e112249, doi:10.1371/journal.pone.0112249, 2014.

Taylor, J. R., and Loescher, H. L.: Automated quality control methods for sensor data: A novel observatory approach, Biogeosciences, 10, 4957-4971, doi:10.5194/bg-10-4957-2013, 2013.

7) The results and discussion also do not focus on the science but rather on what the software can do before the QC/QA are implemented. The QC/QA are the most important component of

any data processing, so I am a little bit shocked that this has not been done before the submission and only raw data are reported. It is all about QC/QA so if it does not work well, the whole infrastructure could be in vain. Was it not possible to wait until the QC steps are implemented? When will it happen?

Author reply: Please see our replies to Referee comments 4) and 6).

Changes in the manuscript: Please see our replies to Referee comments 4) and 6).

8) Sect. 3.2.1 "Algorithm setting and profiling". Can you define algorithm setting? There is no model algorithm here. By algorithm you probably mean the data processing routine which deals with technical issues of EC data handling such as "despiking of unphysical data". This and other similar sections can be confusing for the journal readers. It is also unclear what you mean by profiling in this context as it can also have different meanings (I suppose you meant vertical profiling of EC fluxes rather than algorithm profiling). The authors should be careful not to use ambiguous terms and define clearly what they mean by model, algorithm, and other terms where the meaning is not unambiguous in the modeling context.

Author reply: An algorithm setting encompasses the selection of a specific model (e.g., Salesky et al., 2012) for a general application (e.g., random error calculation), alongside the corresponding parameter settings (e.g., number of low-pass filters). The eddy4R-Docker framework encompasses a multitude of these models, which are summarized in Sect. 2.1. The Sect. 3.2.1 in question describes these settings in detail for the aircraft test application, incl. literature references.

Changes in the manuscript: We now provide additional clarification in Sect. 2.1: "Eddy-covariance data processing consists of employing a sequence of model algorithms. These often originate from scientific sub-fields with corresponding publications, and eddy4R provides an integrative, yet modular and extensible framework for their concerted application and continued development:"

Given this clarification and that vertical soundings are nowhere mentioned in the manuscript, no changes were performed with regard to "algorithm profiling". The section heading in question reads "Algorithm settings and profiling": "algorithm" is the noun-modifier adjective to "profiling". That is, profiling is performed with regard to a piece of code, and profiling results are provided in the same section: "The analysis took 56 minutes with 8-fold parallelization and consumed <3 GB RAM thanks to the use of fast access file-backed objects."

References:

Salesky, S., Chamecki, M., and Dias, N.: Estimating the random error in eddy-covariance based fluxes and other turbulence statistics: The filtering method, Boundary Layer Meteorol., 144, 113-135, doi:10.1007/s10546-012-9710-0, 2012.

9) The choice of example figures 10 and 11 is not optimal because they are not well explained or sufficiently informative for the story. The figures are described only superficially what they represent but are not interpreted scientifically. The blue areas represent high deposition of methane? I am not convinced it is fair to show these data without the discussion of uncertainties which are definitely different in airborne and ground fluxes. The results section 3.3.2 are less than a paragraph so it cannot be informative. This should be made more general or explained much better for general audience of GMD.

Author reply: The intent of this paper has been clarified to demonstrate the applicability of the DevOps model to science code development. For this reason, test applications to three geoscientific use cases are provided in favor of a single in-depth scientific survey. Please see our responses to Referee comments 1) and 3) for additional detail. The figures in question represent the second test application: the analysis of airborne EC data which has been emphasized by Referee #1 as a highlight of the eddy4R-Docker development model. For this test application, uncertainties are discussed just below Figure 11: "Corresponding systematic and random statistical errors are calculated following Lenschow and Stankov (1986) and Lenschow et al. (1994), and the flux detection limit is calculated after Billesbach (2011)." The uncertainty results are then provided in Figure 12.

Changes in the manuscript: We have added explanation to the caption of Figure 10: "For each combination of aircraft position and eddy size, blue and red areas indicate transport toward and away from the surface, respectively."

10) There are other issues which are uncommon to see in a peer-review paper. For example, on Page 7, L225-239 the information is shown as bullet points more like a web-based manual or technical report which almost feels like from a magazine advertising IT Systems. It is interesting, that not even one paper is cited from the GMD community, and majority of the references are authors' own papers published in specialist eddy covariance journals. I think it would make more sense to send the paper to one of those journals or make a better and balanced connection to the GMD literature realm. The EC data handling does seem promising but the approaches vary in various details among the groups so I found the author's EC method review particularly unbalanced.

Author reply: The GMD instructions for "model description papers" require a "user manual"-like component. As such, list-style presentations are not untypical in the peer-reviewed literature, see e.g. Maronga et al. (2015) in GMD. Regarding the literature review and connection to GMD literature realm, please see our replies to Referee comment 1): out of 10 EC papers introduced there, only one is co-authored, while two others are GMD papers. Lastly, the intent of this paper has been clarified to demonstrate the applicability of the DevOps model to science code development. Manuscript Sect. 2 has been substantially expanded specifically to address this model framework aspect of GMD (please see replies to Referee #1 for details).

Changes in the manuscript: Please see our replies to Referee comment 1).

References:

Maronga, B., Gryschka, M., Heinze, R., Hoffmann, F., Kanani-Sühring, F., Keck, M., Ketelsen, K., Letzel, M. O., Sühring, M., and Raasch, S.: The Parallelized Large-Eddy Simulation Model (PALM) version 4.0 for atmospheric and oceanic flows: model formulation, recent developments, and future perspectives, Geosci. Model Dev., 8, 2515-2551, doi:10.5194/gmd-8-2515-2015, 2015.

Overall, I like the general concept of the real-time data/model processing framework, but I expected the paper would be much more than just a teaser of an EC data- processing framework in progress. The revised paper should be guided by clearly defined science question(s) through a coherent story thread throughout the paper. If the intention is to publish the science in GMD, I would strongly recommend the authors to refocus the story on a solid connection between measurement data and modeling. One example could be a model-measurement testbed to validate models on observation data which could be very novel and useful for a larger audience including GMD.

Author reply: Thank you for your thorough review. We hope that the proposed improvements by focusing on the DevOps approach for collaborative coding and including the QA/QC framework in the executable example workflow, in conjunction with our responses would sufficiently address your concerns.

Changes in the manuscript: Please see responses above for changes made to the manuscript.

References:

Mahrt L. Flux sampling errors for aircraft and towers. Journal of Atmospheric and Oceanic technology. 1998 Apr;15(2):416-2

Author reply to the comments by Anonymous Referee #3 of the manuscript

gmd-2016-318

"eddy4R: A community-extensible processing, analysis and modeling framework for eddy-covariance data based on R, Git, Docker and HDF5"

by S. Metzger et al.

We thank Anonymous Referee #3 for the valuable feedback, which helped to improve the manuscript. Please find below the Referee comments recited in *blue, italics font*, followed by our point-by-point replies and corresponding changes in the manuscript in black, upright font.

This study present a radically new way to process eddy-covariance data. It combines R-coded EC software that are wrapped in a portable Docker image that can be used on various platforms. It is meant to be scalable and to make use of parallel processing of large quantities of data.

Author reply: Many thanks for this succinct summary.

Changes in the manuscript: No changes performed.

Major comments

In line with the other reviewers, I think that the paper currently lacks a clear scientific question. I could image that for GMD a clear description of a software environment would suffice, but this paper seems to describe "work in progress".

Author reply: As stated by the Referee, the aim of manuscript is to introduce the novel eddy4R-Docker software development model to address a methodological rather than scientific question: the portable, reproducible and extensible processing of eddy-covariance data. For this reason, the GMD journal was chosen, and three tests of geoscientific applications are provided in favor of a single in-depth scientific survey. One core component of GMD model description papers is "…evaluation against standard benchmarks…" which is addressed in Sect. 3.3. To demonstrate completion we created an executable example workflow accompanying the revised manuscript.

In addition, as detailed in the replies to Referee #1, we clarify the problem statement of the paper: "The question we ask in this paper is: How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of eddy covariance data products?" We then introduce the DevOps approach in more detail and how it, along with the specific tools implemented in the eddy4R-Docker development model, solves this problem. In doing so, we clarify that although the specific software implemented by this developmental model is a work in progress, the developmental model itself is complete and robust, as shown by the test applications.

Changes in the manuscript: Please see the detailed responses below as well as the responses to Referee #1 for changes made in the manuscript.

I am a big fan of Docker and directly downloaded the Docker image. I was disappointed in the fact that the image did not contain clear examples (e.g. the three examples outlined in the paper). I could see that the eddy4R.base and eddy4R.qaqc packages were part of the Docker image. I think it is a missed opportunity not to provide examples of (simple) data processing and plotting. Now the advantage of Docker images remains untraceable to the readers and remains rather theoretical.

Author reply: We could not agree more with the Referee in that an application example would add much value for the reader and potential user. For this reason, we created an executable example workflow accompanying the revised manuscript. It utilizes the functionality of both R-packages presented here, eddy4R.base and eddy4R.qaqc, and contains a user-extensible data read-in, processing and plotting workflow.

Changes in the manuscript: Sect. 2.6 now introduces the executable example workflow:

> "To demonstrate some basic capabilities and provide a template for potential eddy4R-Docker users, an executable example workflow and data are included in the eddy4R-Docker image. Once the eddy4R container is started, the example workflow, input data (NEON dp0p HDF5 file) and output data (NEON dp01 HDF5 file) are available from the Docker-internal directory `/home/eddy/`. The example workflow is located at `/home/eddy/flowExmp/flow.turb.tow.neon.exmp.dp01.R`, and provides a selection of the processing steps that yield the EC dp01 data on the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0). The example workflow is fully documented to guide readers through the various processing steps. These include data and metadata import from the input HDF5 file, data assignment to file-backed objects, processing of 1 minute and 30 minute data statistics and data quality, and writing the output HDF5 file. In addition, outputs from the quality flag and quality metric model are visualized."

For instance, the HDF5 section (2.4) is clear but a rather standard description that is available on internet (meta-data, directory structure, self-documenting). Again, this is a missed opportunity to guide users through an example (download raw data, process the data, and HDF5 output and visualization of results). You want to convince the "traditional ASCII" community.

Author reply: Agreed. The executable example workflow includes HDF5 read-in, write-out examples with attributed metadata to demonstrate the utility of having metadata attached to the data.

Changes in the manuscript: Please see our replies to the Referee comment above. Among others we demonstrate the utility of the HDF5 file format in the executable example workflow, and example HDF5 input and output files are already pre-compiled into the Docker image.

Section 2.5 presents the way NEON wants to deploy Docker images. Again, this remains rather high level, while the stated goal is to "empower the Science community at large by putting the key to the scientific algorithms into the hand of scientists". Again, a clear running example in a Docker container would convince these scientists more than a NEON brochure.

Author reply: We believe that this concern is addressed through the executable example workflow, which is described in Sect. 2.6.

Changes in the manuscript: No changes to Sect. 2.5.

Section 2.6 would be an ideal starting point for further "Docker-assisted" data analysis, but unfortunately stops at a reference to the eddy4R wiki pages.

Author reply: In response to the Referee comment, we introduce the executable example workflow incl. "Docker-assisted" data analysis in Sect. 2.6.

Changes in the manuscript: Please see our replies to the Referee comments above.

In section 5 there is a reference to the raw data, but again unfortunately no examples are given in which a Docker image automatically reads, processes, and presents results. In the remainder of the paper, three examples are given, which is basically fine, but without a traceable and "hands-on" exercise does not add much. It is (and should be) part of the standard software testing.

Author reply: We address this concern through providing the executable example workflow.

Changes in the manuscript: Please see our replies to the Referee comments above.

In summary, I very much like the concept presented in this paper. However, without more in depth possibilities for potential users of the software, the papers seems more suitable for internal documentation than convincing readers that this is a promising way for the community to process eddy covariance data.

Author reply: We thank the Referee for sharing the positive impression of the paper's concept. We agree that more in-depth possibilities for potential users of the software will help demonstrate the utility of the software development approach.

Changes in the manuscript: We have created an executable example workflow accompanying the revised manuscript.

Minor comments

Page 1: line 34: mention where the NEON site is and also where the aircraft data were collected.

Author reply: This information is provided as part of the test applications in Sects. 3.1 and 3.2.

Changes in the manuscript: Added "USA" for the aircraft test application in Sect. 3.2.

Page 1, line 38: "streaming generation of science-grade EC fluxes": please explain better what this means.

Author reply: Adjusted.

Changes in the manuscript: Changed to "…automated generation of science-grade EC fluxes…"

Page 6, line 185: current recent

Author reply: Adjusted.

Changes in the manuscript: Changed to "…most recent eddy4R source code…"

Page 6, Figure 3, introduced at line 191. This hardly adds anything. A link would do here. Also figure 4 and figure 7 seem illustrations that do not add much.

Author reply: We agree that Figure 3 can be removed without losing much information. The GMD instructions for "model description papers" require a "user manual"-like component: Figure 4 introduces the HDF5 format and structure used in this study and for NEON data portal downloads of EC data (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0). Figure 7 presents the development environment user interface. As both of these are fairly new to the EC community we are under the impression that retaining Figure 4 and Figure 7 provides clarity for some readers.

Changes in the manuscript: Removed Figure 3.

Page 7, line 231: CI?

Author reply: Cyberinfrastructure, as introduced in Sect. 2.

Changes in the manuscript: No changes.

eddy4R 0.2.0: A DevOps model for community-extensible processing and analysis of eddy-covariance data based on R, Git, Docker and HDF5

Stefan Metzger[1,2], David Durden[1], Cove Sturtevant[1], Hongyan Luo[1], Natchaya Pingintha-Durden[1], Torsten Sachs[3], Andrei Serafimovich[3], Jörg Hartmann[4], Jiahong Li[5], Ke Xu[2], Ankur R. Desai[2]

[1]National Ecological Observatory Network, Battelle, 1685 38th Street, Boulder, CO 80301, USA

[2]University of Wisconsin-Madison, Dept. of Atmospheric and Oceanic Sciences, 1225 West Dayton Street, Madison, WI 53706, USA

[3]GFZ German Research Centre for Geosciences, Telegrafenberg, 14473 Potsdam, Germany

[4]Alfred Wegener Institute – Helmholtz Centre for Polar and Marine Research, Am Handelshafen 12, 27570 Bremerhaven, Germany

[5]LI-COR Biosciences, 4647 Superior Street, Lincoln, NE 68504, USA

Correspondence to: Stefan Metzger (smetzger@battelleecology.org)

Keywords: computing, container, continuous development, continuous integration, devOps, eddy4R, eddy-covariance, EddyPro, EdiRe, EddyUH, image, NEON, REddyProc, reproducibility, science code, TK3

Deleted: ,

Deleted: and modeling framework for

Formatted: Superscript

Deleted: Sachs[2]

Deleted: Serafimovich[2]

Deleted: Hartmann[3]

Deleted: Li[4]

Deleted: Xu[5]

Deleted: Desai[5]

Deleted: [1]Battelle Ecology

Moved (insertion) [1]

Deleted: [2]GFZ

Deleted: [3]Alfred

Deleted: [4]LI

Moved up [1]: of Wisconsin-Madison, Dept. of Atmospheric and Oceanic Sciences, 1225 West Dayton Street, Madison, WI 53706, USA¶

Deleted: [5]University

Formatted: Left, Space After: 8 pt, Line spacing: Multiple 1.08 li

Formatted: Font: 12 pt

Abstract

Large differences in instrumentation, site setup, data format, and operating system stymie the adoption of a universal computational environment for processing and analyzing eddy-covariance (EC) data. This results in limited software applicability and extensibility in addition to often substantial inconsistencies in flux estimates. Addressing these concerns, this paper presents the systematic development of portable, reproducible, and extensible EC software achieved by adopting a Development and Systems Operation (DevOps) approach. This software development model is used for the creation of the eddy4R family of EC code packages in the open-source R Language for Statistical Computing. These packages are community-developed, iterated via the Git distributed version control system, and wrapped into a portable and reproducible Docker filesystem that is independent of the underlying host operating system. The HDF5 hierarchical data format then provides a streamlined mechanism for highly compressed and fully self-documented data ingest and output.

The usefulness of the DevOps approach was evaluated for three test applications. First, the resultant EC processing software was used to analyze standard flux tower data from the first EC instruments installed at a National Ecological Observatory (NEON) field site. Second, through an aircraft test application we demonstrate the modular extensibility of eddy4R to analyze EC data from other platforms. Third, an intercomparison with commercial-grade software showed excellent agreement (R^2=1.0 for CO_2 flux). In conjunction with this study, a Docker image containing the first two eddy4R packages and an executable example workflow, as well as first NEON EC data products are released publicly. We conclude by describing the work remaining to arrive at the automated generation of science-grade EC fluxes, and benefits to the science community at large.

This software development model is applicable beyond EC, and more generally builds the capacity to deploy complex algorithms developed by scientists in an efficient and scalable manner. In addition, modularity permits meeting project milestones while retaining extensibility with time.

Deleted: study

Deleted: an open-source, flexible and modular eddy-covariance (EC) data processing framework. This is

Deleted: through

Deleted: philosophy, building on

Deleted: as foundation.

Deleted: GitHub

Deleted: framework

Deleted: The efficiency and consistency of this framework is demonstrated in the form of three application examples. These include tower EC data from first instruments installed at a National Ecological Observatory (NEON) field site, aircraft flux measurements in combination with remote sensing data, as well as a software intercomparison. In conjunction with this study, the first two eddy4R packages and simple NEON EC data products are released publicly. While this proof-of-concept represents a significant advance, substantial work remains to arrive at the automated framework needed for the streaming generation of science-grade EC fluxes.

1 Introduction

Answering grand challenges in earth system science and ecology requires combining information from hierarchies of environmental observations (tower, aircraft, satellite; Raupach et al., 2005; Running et al., 1999; Turner et al., 2004). Eddy-covariance (EC) measurements serve as crucial observations in this hierarchy to study landscape-scale surface-atmosphere exchange processes that both inform and anchor earth system models. Networks of EC towers such as FLUXNET (Baldocchi et al., 2001), AmeriFlux (Law, 2007), ICOS (Sulkava et al., 2011), and others are vital for providing the necessary distributed observations covering the climate space, with the longest running towers now reaching two decades of observations.

A current challenge for EC tower networks in informing regional and continental scale processes is instrument and computational compatibility. The computations involved in EC processing are complex and developmentally dynamic, making code portability, extensibility, and documentation paramount. Much progress has been made in developing community standards for processing algorithms and workflows (Aubinet et al., 2012; Papale et al., 2006). Many authors have included code in publication, or have developed sharable tools (e.g. EddyPro and TK3 by Fratini and Mauder (2014), EddyUH by Mammarella et al. (2016), EdiRe by Clement et al. (2009), despite the significant and often unfunded effort required to adequately document and generalize code. Still, large differences in instrumentation, site setup, data format, and operating systems stymie the adoption of a universal EC processing environment; one that is portable, reproducible, and extensible to allow tailored workflows that incorporate additional data streams, to automate and scale processing across large compute facilities, or to inject additional algorithms that address specific needs or synergistic research questions. In 50% of published scientific code, one cannot even replicate the necessary software dependencies (Collberg et al., 2014), and even widely used and well-documented EC processing software packages have shown substantial inconsistencies in flux estimates (e.g. Fratini and Mauder, 2014). A universal EC processing environment that enables these capabilities would better allow research groups to tailor existing software to their needs (and contribute new algorithms) instead of re-creating code or kludging together multiple software outputs to realize an algorithmic chain for their data analytics.

The U.S.-based National Ecological Observatory Network (NEON), once fully operational, will represent the largest single-provider EC tower network globally, with a standardized measurement suite designed explicitly for cross-site comparability and analysis of continental-scale ecological change (Schimel et al., 2007). This capability is accompanied by a strong need for a flexible and scalable processing framework that can incorporate specific data streams, take advantage of close alignment of hardware and software for problem tracking and resolution, provide traceability and reproducibility of outputs, and seamlessly integrate distributed and dynamic community-developed code (written by multiple people in multiple places) within existing cyberinfrastructure (CI). In sum, NEON needs what the EC community is currently lacking.

The question we ask in this paper is: How do we collaboratively create portable, reproducible, open-source, scalable, and extensible software that improves reliability and comparability of

Deleted: (tower, aircraft, satellite; Raupach et al., 2005; Running et al., 1999; Turner et al., 2004)

Deleted: (Law, 2007)

Deleted: (Sulkava et al., 2011)

Deleted: Papale et al., 2006

Deleted: . However, the computations involved in EC processing are complex and developmentally dynamic, making code portability, extensibility, and documentation paramount.

Deleted: Fratini and Mauder (2014)

Deleted: Mammarella et al. (2016)

Deleted: Clement et al. (2009)

Deleted: .

Deleted: system

Deleted: , exacerbated by the significant and often unfunded effort required to adequately document and generalize code.

Deleted: (Collberg et al., 2014)

Deleted: let alone develop tailored workflows to incorporate additional data streams, automate and scale processing across large compute facilities, or inject additional algorithms to address specific needs or synergistic research questions.

Deleted: (Schimel et al., 2007)

Deleted: tight

Deleted: -

Deleted: integration

Deleted: Here, we describe and demonstrate a framework

EC data products? Here, we describe and demonstrate a developmental model that enables these capabilities by embracing a Development and Systems Operation (DevOps) approach. DevOps is a philosophy arising from the software development community that emphasizes collaboration among developers and operators to continuously iterate the development, building, testing, packaging, and release of software (Erich et al., 2014; Loukides, 2012). Tools are adopted that control and automate these processes, allowing distributed development and rapid iteration. Applied to the scientific community, developers are the multitude of scientists creating and improving the scientific algorithms that form the developmentally dynamic community standard. Operators are those deploying the algorithms to process and analyze data, and can be the same or different people as those creating the algorithms. A key aspect of DevOps is the recipe- or script-based generation and packaging of computation environments rather than abstracted documentation, which improves accessibility, extensibility, and reproducibility of scientific software (Boettiger, 2015; Clark et al., 2014). The recipe automates the loading of the software including all dependencies so that the most significant hurdle of reproducing the computational environment is overcome. At the same time, the recipe serves as explicit documentation, and can be easily extended (added to or changed), shared, and versioned. The entire computational environment including any necessary data are packaged into Docker images that work identically across different computers and operating systems, can be deployed at scale, and archived for ultimate reproducibility.

In the following we present this framework and demonstrate its success in producing EC data products via a family of modular, open-source R packages wrapped in Docker images. We emphasize that this paper is not a presentation of EC processing software (although this is the ultimate application). Rather, it is a presentation of the development model that facilitates portability, reproducibility, and extensibility of EC processing software. In the following, Sect. 2 describes the DevOps framework, and Sect. 3 provides three core tests of the applicability of this framework: 1) processing tower-based flux data, including NEON's first set of EC data, 2) processing and footprint modeling of aircraft-based flux data, and 3) a software cross-validation. Section 4 summarizes the work remaining to operationally produce EC fluxes from 47 NEON sites, and provides an outlook on future capabilities and science community benefits. Code and data availability information is provided in Sect. 5.

2 The development and operations (DevOps) model

DevOps promotes collaboration and tight integration between software development, testing, and operational deployment by following a core workflow (e.g., Wurster et al., 2015): Plan, Create, Verify, Package, Release, Configure, and Monitor. The text below describes these stages and **Error! Reference source not found.** shows the general sequence and overlap of these stages between software developers (Dev) and operators (Ops).

Plan involves focusing and prioritizing new software features or capabilities based on their enhancement of value. **Create** is the activity of designing and writing the code that delivers a new feature. **Verify** tests the new software feature against established standards for accuracy and performance (e.g. does it unexpectedly alter the output of pre-existing features? Does it produce the expected result?). **Package** involves the compilation of the code once it is ready

Deleted: within

Deleted: (Erich et al., 2014; Loukides, 2012)

Deleted: A key aspect of DevOps that can aid improving accessibility, extensibility, and reproducibility of scientific software is through

Deleted: Clark et al., 2014

Deleted: developmental framework

Deleted: scalability,

Deleted: Section 2

Deleted: Sect. 3

Deleted: and aircraft example applications

Deleted: as well as

Deleted: .

Deleted: framework

for deployment, including all data and software dependencies, and gathers necessary approvals. The **Release** stage deploys the software into production. **Configure** involves supplying and configuring the computational infrastructure required to operate the code at scale, including storage, database operations, and networking. Finally, **Monitor** observes and tracks the use, performance, and end-user impact of the release.

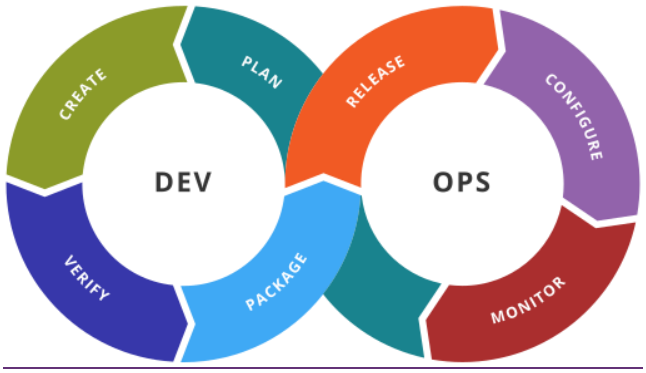


Figure 1. Stages of the general DevOps workflow (source: Kharnagy via Wikimedia Commons [CC BY-SA 4.0]).

Variants of this workflow exist (e.g., Chen, 2015), but the general components and sequence are retained. In addition, there is no single set of tools accompanying the DevOps approach. Rather, many tools exist that facilitate the execution of one or more of these workflow steps, often through automation.

NEON's DevOps framework consists of a periodic sequence (Figure 2) that incorporates these workflow steps. For this purpose we define NEON Science as personnel working directly on the NEON project, and the Science Community, regardless of whether they also work on the NEON project, as anyone producing or using data, algorithms, or research products related to the NEON data themes (Atmosphere; Biogeochemistry; Ecohydrology; Land Cover and Processes; Organisms, Populations, and Communities): The science community contributes algorithms and best practices (1). Implicitly or explicitly, this embodies the DevOps: Plan stage – the algorithms most valued by the community are being incorporated. Together with NEON Science (2), these algorithms are coded in the open-source R computational environment (DevOps: Create stage). DevOps: Verify (testing) and Package (packaging) are performed as the code is compiled into eddy4R packages via the GitHub distributed version control system (3). NEON Science releases an eddy4R version from GitHub, which automatically builds an eddy4R-Docker image on DockerHub as specified in a "Dockerfile" (4; DevOps: Release stage). The eddy4R-Docker image is immediately available for deployment by NEON CI (5; DevOps: Configure & Monitor stages), the Science Community (1) and NEON Science (2) alike. Here the DevOps: Configure (computational resource allocation) & Monitor stages occur.

Moved (insertion) [2]

Deleted: Figure 1):

Deleted:) which together

Deleted:) are

Deleted:).

Deleted: ¶
¶

Moved up [2]: Figure 1.

Deleted: DevOps workflow of the eddy4R-Docker

Moved down [3]: . Please see text in Sect. 2 for detailed explanation.

Deleted: Page Break
¶
image

Deleted: Cyberinfrastructure (

Deleted: ;

Monitoring of end-user experience is also performed in GitHub (3) via issue-tracking. This DevOps cycle can be repeated for continuous development and integration of requests and future methodological improvements by the scientific community, resulting in the next release. Two principal types of releases are provided: stable versions are tagged with "0.2.0", "0.2.1" etc., and the most recent development built is tagged with "latest". Thus, the DevOps model serves as the framework within which the scientific community can efficiently and robustly collaborate to produce, manage, and iterate software. Through choosing appropriate tools to implement the DevOps workflow steps, the reproducibility, scalability and extensibility needs of software development communities (including EC) can be met.

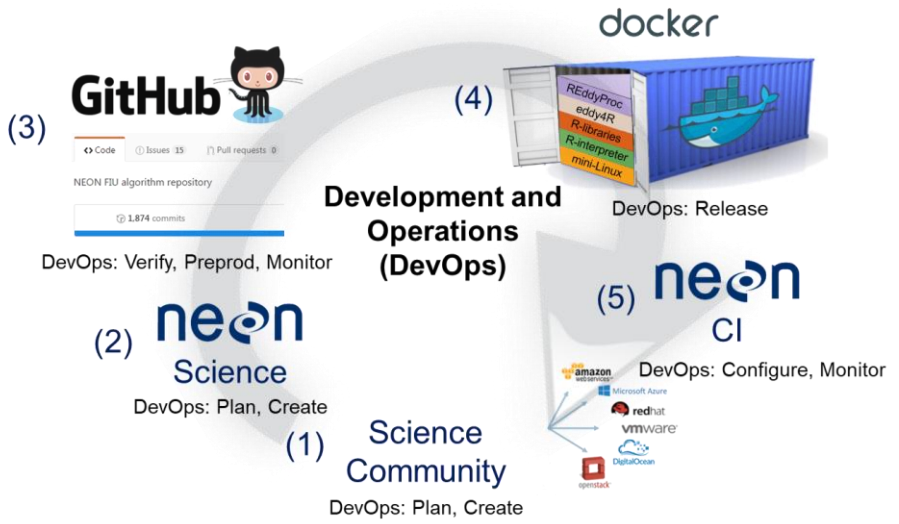


Figure 2. NEON-specific DevOps workflow. DevOps workflow steps are called out in parentheses. Please see text in Sect. 2 for detailed explanation.

In the following we describe the key components and tools of this NEON-specific DevOps model, namely the eddy4R family of code packages (Sect. 2.1), Git-based distributed code development (Sect. 2.2), packaging of the computational environment in Docker images (Sect. 2.3), hierarchical data formats (Sect. 2.4), integration with NEON's CI (Sect. 1.1), and installation and deployment (Sect. 2.6).

2.1 The eddy4R family of R-packages (DevOps: Plan & Create)

eddy4R is a family of open-source packages for EC raw data processing, analyses and modeling in the R Language for Statistical Computing (R Core Team, 2016). Forming the DevOps: Plan & Create stages, it is being developed by NEON scientists with wide input from the

Deleted: v1.

Deleted: v1.

Moved (insertion) [3]

Deleted: infrastructure

Deleted: framework

Deleted: cyberinfrastructure

Deleted: (R Core Team, 2016)

Deleted: . It

micrometeorological community (e.g., De Roo et al., 2014; Kohnert et al., 2015; Lee et al., 2015; Metzger et al., 2012; Metzger et al., 2013; Metzger et al., 2016; Sachs et al., 2014; Salmon et al., 2015; Serafimovich et al., 2013; Starkenburg et al., 2016; Vaughan et al., 2015; Xu et al., 2017). eddy4R currently consists of four packages eddy4R.base, eddy4R.qaqc, eddy4R.turb, and eddy4R.erf. Of these, eddy4R.base and eddy4R.qaqc are published here in conjunction with NEON's release of EC Level 1 data products (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0): descriptive statistics of calibrated instrument output. In addition, previews of eddy4R.turb and eddy4R.erf are provided, which will be published along NEON's upcoming release of EC Level 4 data products (derived quality-controlled fluxes and related variables). Development of two additional R-packages eddy4R.stor and eddy4R.ucrt has started, which provide functionalities for storage flux computation and uncertainty quantification, respectively. These packages are not covered here, and will be published once available.

Each eddy4R package consists of a hierarchical set of reusable definition functions, wrapper functions and workflows. Following best practices, eddy4R is written in controlled and strictly hierarchical terminology consisting of base names and modifiers, which ensures modular extensibility over time. Interactive documentation is provided through the use of Roxygen tags (http://roxygen.org/) during development, and follows the Comprehensive R Archive Network (CRAN; https://cran.r-project.org/) guidelines for package dissemination. In addition, expanded documentation is available in the form of Algorithm Theoretical Basis Documents from the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0).

EC data processing consists of employing a sequence of model algorithms. These often originate from scientific sub-fields with corresponding publications, and eddy4R provides an integrative, yet modular and extensible framework for their concerted application and continued development: eddy4R.base provides natural constants and basic functions for usability, regularization, transformation, lag-correction, aggregation and unit conversion ensuring consistency of internal units at any point in the workflow. Next, eddy4R.qaqc provides the general quality assurance and quality control (QA/QC) tests of Taylor and Loescher (2013), along the Smith et al. (2014) model for tracking quality information in large datasets, and functions for de-spiking (Brock, 1986; Fratini and Mauder, 2014; Mauder et al., 2013; Mauder and Foken, 2015; Metzger et al., 2012; Vickers and Mahrt, 1997). eddy4R.turb provides standard, Reynolds-decomposed turbulent flux calculation (Foken, 2017), accompanied by models for planar fit transformation (Wilczak et al., 2001) and spectral correction (Nordbo and Katul, 2012). Additional functionalities include Fourier transform, the determination of detection limit (Billesbach, 2011), integral length scales and statistical sampling errors (Lenschow et al., 1994), and flux-specific QA/QC models (Foken and Wichura, 1996; Vickers and Mahrt, 1997). Also, basic scaling variables, atmospheric stability and roughness length (Stull, 1988), as well as the flux footprint (Kljun et al., 2015; Kormann and Meixner, 2001; Metzger et al., 2012) can be determined. Lastly, edd4R.erf provides time-frequency de-composed flux processing and data mining functionalities to determine an environmental

Deleted: (e.g., De Roo et al., 2014; Kohnert et al., 2015; Lee et al., 2015; Metzger et al., 2012; Metzger et al., 2013; Metzger et al., 2016; Sachs et al., 2014; Salmon et al., 2015; Serafimovich et al., 2013; Starkenburg et al., 2016; Vaughan et al., 2015; Xu et al., 2017)

Deleted: :

Deleted: workflow templates.

Deleted: tags

Deleted: (data.neonscience.org/home).

Deleted: Taylor and Loescher (2013)

Deleted: Smith et al. (2014)

Deleted: framework

Deleted: Fratini and Mauder, 2014; Mauder and Foken, 2011

Deleted: Mauder et al., 2013

Deleted: Metzger et al., 2012

Deleted: Vickers and Mahrt, 1997

Deleted: (Foken, 2008), accompanied by facilities

Deleted: (Wilczak et al., 2001)

Deleted: (Nordbo and Katul, 2012)

Deleted: (Lenschow et al., 1994)

Deleted: (Foken and Wichura, 1996; Vickers and Mahrt, 1997)

Deleted: (Stull, 1988)

Deleted: (Kljun et al., 2015; Kormann and Meixner, 2001; Metzger et al., 2012)

Deleted: artificially intelligent functionality

response function model and project the flux fields underlying the EC observations (Metzger et al., 2013; Xu et al., 2017).

eddy4R can be used with a fully adaptive single-pass workflow (Sect. 3.1), which makes it computationally efficient compared to the multiple passes required by other flux processing schemes. In addition, eddy4R is fully parallelized and memory efficient leveraging R's snowfall parallelization (https://cran.r-project.org/package=snowfall) and ff file-backed object (https://cran.r-project.org/package=ff) facilities, respectively. This makes eddy4R seamlessly scalable from local laptop development to deployment across massively parallel computing facilities. Lastly, its unique modularity permits straightforward adjustments (extensibility) and versioning as science and/or hardware progresses.

2.2 Git distributed version control (DevOps: Verify & Package)

The eddy4R source code resides on a version-controlled Git repository on the hosting service GitHub (https://github.com/). In general, a developer community uses a version control system to manage and track different states of their works over time. GitHub provides distributed version control and has become widely used by scientific research groups because it is free, open-source, and provides several features that make it useful for managing artifacts of scientific research (Ram, 2013).

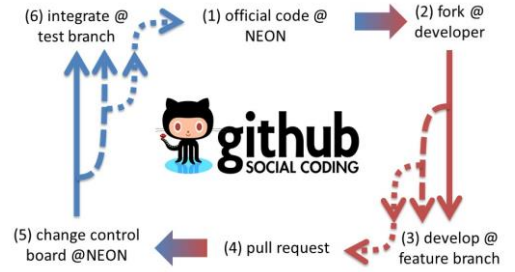


Figure 3. NEON's Git workflow. Please see text in Sect. 2.2 for detailed explanation.

Git allows multiple users and developers to simultaneously access and collaborate on a remote repository by means of independent 'forks' or replicas of the entire repository (Paarsch and Golyaev, 2016). Figure 3 shows NEON's Git workflow: At any given time (1) the official, stable eddy4R source code resides on NEON's GitHub repository. A user can install the eddy4R packages directly from there, and (2) a developer can 'fork' or copy the repository and create 'branches' for modification. After (3) 'committing' or creating a new feature, the developer (4) can propose the feature for inclusion in the official eddy4R source code by issuing a pull request to (5) NEON's change control board. After (6) thorough review and all prior test cases reproducing benchmark results (DevOps: Verify stage), the feature can be 'merged' or integrated into the next release of (1) the official, stable eddy4R source code (DevOps: Package

Deleted: functions
Deleted: Metzger et al., 2013
Field Code Changed
Deleted: template
Field Code Changed
Moved (insertion) [4]
Moved (insertion) [5]
Deleted: ¶
Moved up [4]: ¶
Figure
Deleted: 2.
Moved up [5]: NEON's Git workflow. Please see text in Sect. 2.2 for detailed explanation.¶
Deleted: (Paarsch and Golyaev, 2016)
Deleted: testing,
Deleted: .

stage). This cycle can be repeated to accommodate requests and future developments, resulting in subsequent releases. Including a test case for new code is strongly encouraged to ensure sustainability over time, but is not mandatory. The developers can periodically update their 'forks' from the remote repository, ensuring that they always work on basis of the most recent eddy4R source code.

The ultimate advantages of Git are provenance, reproducibility and extensibility. Every copy of the code repository includes the complete history of all changes and authorship that can be viewed and searched by anyone (Ram, 2013). This allows developers to build from any stage of the versioned project and makes it easy to collaborate as an integrated scientific community. We note that the DevOps workflow is robust to the business viability of the particular tools used for implementation. Git is simply one instance of a version control system, which could be replaced with another similar tool should Git fail at some point in the future.

2.3 Docker image build and deployment (DevOps: Release)

Facilitating the DevOps: Release stage, Docker images (https://www.docker.com/what-docker) wrap a piece of software in a complete filesystem that contains only the minimal context an application needs to run: code, runtime, system libraries and tools. This guarantees that it always performs the same, regardless of the compute environment it is deployed in (i.e. ultimate reproducibility). Compared to the similar but more cumbersome virtual machine approach, a Docker image is an order or magnitude smaller (eddy4R-Docker: 2 GB without example data). Also, by running as native processes it bypasses the virtual machine overhead. Docker is used by many organizations (e.g., National Center for Atmospheric Research, National Snow and Ice Data Center, NSF Agave API), and widely supported across large-scale cloud compute environments (e.g., Amazon EC2 Container Service, Google Container Engine, NSF Xsede). It is particularly well suited to NEON's DevOps strategy: combining development, operation and quality assurance to enable creating, testing, deploying and updating scientific software rapidly and reliably (Figure 2).

Docker can build images automatically by reading the instructions from a Dockerfile. A Dockerfile is a text document that contains all the instructions a user would call on the command line to assemble an image. Using e.g. a cloud hosting platform like DockerHub (https://hub.docker.com/), the image build, versioning and distribution can be automated. This is realized through executing the series of command-line instructions defined in the Dockerfile whenever a new eddy4R source code version is available on GitHub. A key feature of eddy4R-Docker is that it builds upon "Rocker" pre-built Docker images, maintained by the rOpenSci group (https://ropensci.org/). This ensures access to stable, up-to-date base images containing R and a variety of packages commonly used. The eddy4R-Docker image (0.2.0) released in this study was built based on the rocker/ropensci/latest image containing R (3.4.0; (https://hub.docker.com/r/rocker/ropensci/builds/). As specified in the eddy4R Dockerfile, our R packages eddy4R.base (0.2.0) and eddy4R.qaqc (0.2.0) and their dependencies were automatically built on top of this base image. To complete the eddy4R-Docker processing, analysis and modeling environment, the NEON data portal API Client nneo (0.1.0) as well as

Deleted: current

Deleted: ¶

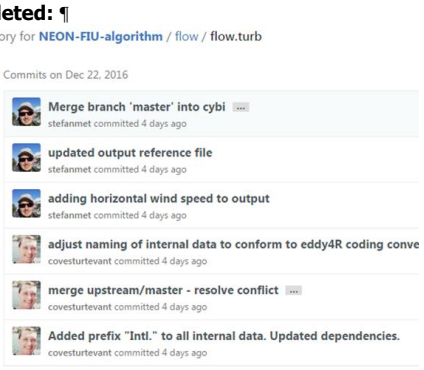


Figure 3. Example of GitHub facilities for exploring code authorship, development history.¶

¶

Deleted: (Figure 3):

Deleted: ,

Deleted: (Ram, 2013)

Deleted: the

Formatted: Not Strikethrough

Deleted: tools, system

Deleted: . By running as native processes, Docker bypasses the overhead encountered in

Field Code Changed

Deleted: commands

Deleted: 1

Deleted: 3.2

Deleted: 1

Deleted: 1

Deleted: .3.9100

the REddyProc (1.0.0) high-level utilities for aggregated EC data were also included. In addition, the user can install any desired R packages to customize the environment.

Docker's benefits to scientific software development are described in detail in Boettiger (2015). For NEON's purposes, several Docker properties are particularly important:

- **Portability**: Docker images are portable and independent of the underlying operating system. This enables scientists to develop code on local computers or virtual machines without worrying about the deployment architecture.
- **Reproducibility**: The DevOps principles are ingrained into the Docker build process, thus ensuring a fully traceable and documented Docker image.
- **Streamlined interface between Science and CI**: Defined inputs, outputs and instructions provide an ideal framework to isolate and package algorithmic services for operational deployment.
- **Continuous development and integration**: Docker provides a modular and extensible framework, permitting NEON's data processing to remain up-to-date with the latest algorithmic developments. As shown be the nneo and REddyProc examples, it enables directly leveraging community-developed code. In this way eddy4R-Docker is functionally extensible, while making it easy for the community to incorporate NEON-developed code into their own data processing.

2.4 Hierarchical Data Format version 5 (DevOps: Configure)

The capability to process large data sets is reliant upon efficient input and output of data, data compressibility to reduce compute resource loads, and the ability to easily package and access metadata. The Hierarchical Data Format (HDF5) is a file format that can meet these needs, and is a key tool aiding the DevOps: Configure (computational resource allocation) stage. A NEON standard HDF5 file structure and metadata attributes allow users to explore larger data sets in an intuitive "directory-like" structure that is based upon the NEON data product naming convention (see Figure 4). Group level 1 separates data by site and site level metadata are attributed at that level. Group level 2 separates data by data product level (DPL) and DPL metadata are attributed at that level, where DPLs correspond to the amount of processing performed. DPL1 are calibrated descriptive statistics, DPL2 are temporally interpolated, DPL3 are spatially interpolated, and DPL4 are further-derived quantities. Group level 3 are the individual data products, for instance CO_2 concentration. Lastly, replicates in the horizontal and vertical are separated as individual data tables.

This provides a streamlined data-delivery mechanism for the eddy4R-Docker processing framework. For the tower datasets analyzed in this study, including sonic anemometer, infrared gas analyzer and mass flow controller data, file sizes ranged from 1 GB for the uncompressed data in comma-delimited ASCII files to 0.1 – 0.2 GB in HDF5 format, depending on the amount of missing data. The HDF5 files can be written in a simple format where data are stored as single 1-dimensional arrays to maximize compression and efficiency, or the data can be stored as compound datatables that allow multiple datatypes to be written together in columnar format for ease of navigation when data size is not an issue.

Deleted: 8-2

Deleted: .

Deleted: Figure 4).

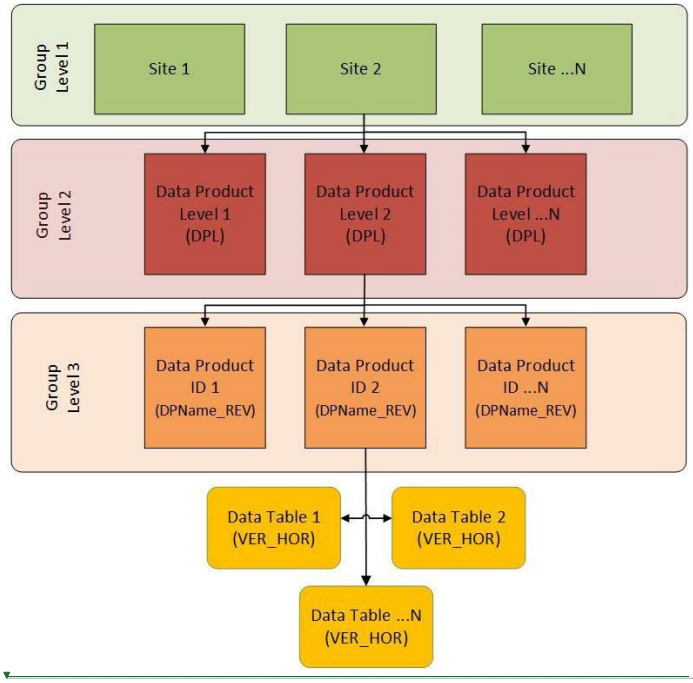


Figure 4 The NEON HDF5 file structure based on the NEON data product naming convention.

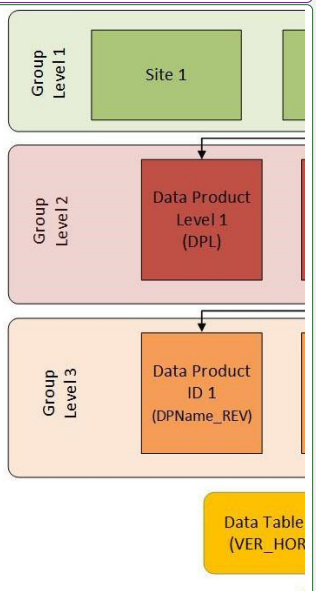
Another important function of the HDF5 file format is the ability to attach metadata as attributes, further promoting reproducibility. The data in this study has the units and variable names as metadata attached to the data tables in the HDF5 file. Additional metadata are attributed to various hierarchical groups throughout the file, including environmental parameters, sensor metadata, and processing parameters. As a result, HDF5 and similar self-documenting hierarchical data formats are gaining traction in a community that has traditionally relied on ASCII text column or comma-delimited files, especially as tools for viewing, manipulating, and extracting data from HDF5 become more commonplace. The utility of HDF5 file format is demonstrated in the executable example workflow that accompanies this manuscript (see Sects. 2.6, 5).

2.5 Modular compatibility with existing compute infrastructure (DevOps: Configure & Monitor)

To perform a defined series of processing steps, a Docker image is called with a workflow file, resulting in a running instance called Docker container (Figure 5). Through this mechanism, an arbitrary number of Docker containers can be run simultaneously performing identical or different services depending on the workflow file. This provides an ideal framework for scaled deployment using e.g. high-throughput compute architectures, cloud-based services etc.

Moved (insertion) [6]

Deleted: ¶



Moved up [6]: Figure 4 The NEON HDF5 file structure based on the NEON data product naming convention.¶
¶

Deleted: an instruction

Deleted: instruction

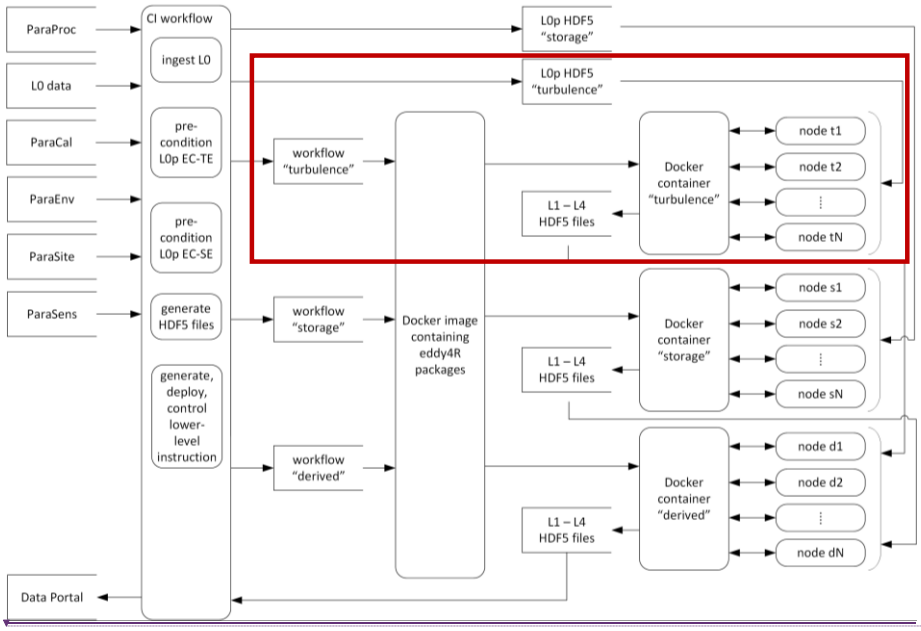
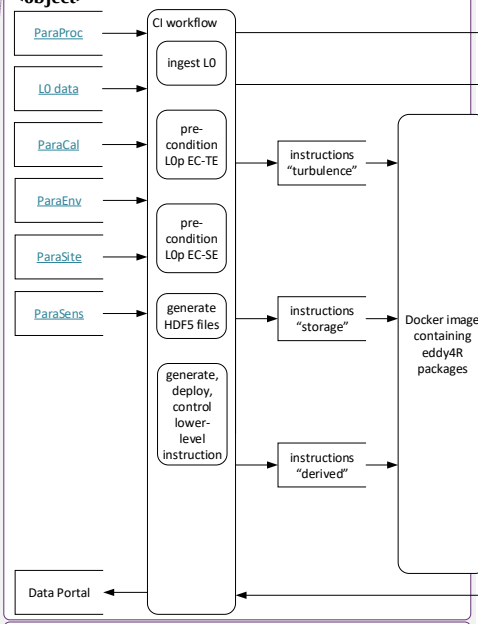


Figure 5. NEON's eddy4R-Docker EC processing framework. The red box visualizes the scope of the present study, and individual components are described in the text.

Embodying the DevOps: Configure stage, NEON's eddy4R-Docker EC processing framework begins with ingesting information from various data sources on a site-by-site basis (Figure 5 top left). This includes EC raw data (Level 0, or L0 data) alongside contextual information on measurement site (ParaSite), environment (ParaEnv), sensor (ParaSens), calibration (ParaCal), as well as processing parameters (ParaProc). Next, the raw data is preconditioned and all information is hierarchically combined into a compact and easily transferable HDF5 file (Figure 5 panel "CI workflow"). Each file contains the calibrated raw data (L0 prime, or L0p) and metadata for one site and one day, either for EC turbulent exchange or storage exchange. In this manuscript, we focus on demonstrating the turbulence data process and analysis in the red box of Figure 5. Together with the "turbulence" workflow file the HDF5 L0p data file is passed to the eddy4R-Docker image, where a running Docker container is spawned that scales the computation over a specified number of compute nodes (Figure 5 top right). The resulting higher-level data products (Level 1 – Level 4, or L1-L4) are collected from the compute nodes and, together with all contextual information, are combined into a daily L1-L4 HDF5 data file that is served on the data portal (Figure 5 bottom left). In addition to the daily output files, monthly concatenated files are also available for download from the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0). This sequence is performed analogously for different combinations of workflows and data, and it is possible for the workflow instruction sets to interact with each other. For example, the "turbulence" and

Deleted: ¶
<object>



Deleted: workflow.

Deleted: workflow

Deleted: workflow

Deleted: instruction

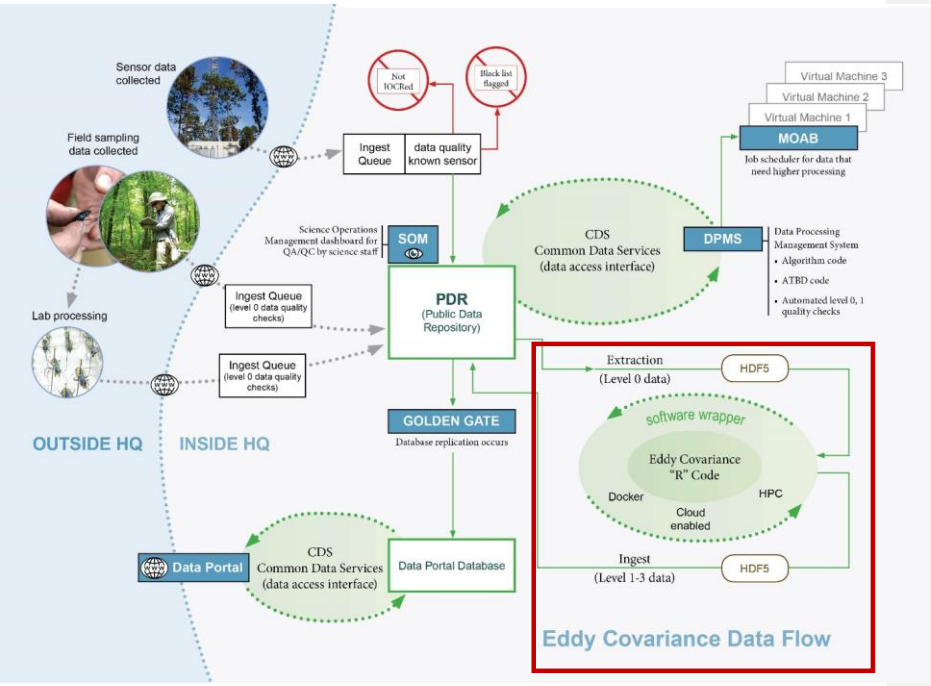
Deleted: created

Deleted: instructions

563 "storage" containers are processing in parallel, and starting the "derived" container once all
564 intermediary results are available (Figure 5 bottom right). It should be noted that the
565 "turbulence", "storage" and "combined" Docker containers (Figure 5 right) are all spawned
566 from the same eddy4R-Docker image (Figure 5 center): each container includes the same
567 underlying functionality (eddy4R packages), but serves a different purpose by being fed the
568 "turbulence", "storage" or "combined" workflow files.

569 This eddy4R-Docker EC processing framework modularly integrates into pre-existing data
570 processing pipelines, such as NEON's CI (Figure 6): in NEON's pre-existing framework the
571 CI group encoded simple algorithms (e.g. temporal means) in Java, based on algorithm
572 documentation provided by Science staff. The key difference of the eddy4R-Docker EC
573 processing framework is that instead of algorithm documentation, NEON Science staff now
574 provide documented algorithms that perform a complex series of processing steps, which can
575 be directly deployed by CI. Not only does this adoption of the NEON-DevOps workflow
576 (Figure 2) streamline end-to-end operational implementation and efficiency, it empowers the
577 Science community at large by putting the key to the scientific algorithms into the hand of
578 scientists.

579



580

581 Figure 6. NEON's CI for streaming data processing. The red box visualizes the eddy4R-Docker
582 EC processing framework within the overall CI.

583

Deleted: workflow
Deleted: the one of NEON
Deleted: workflow
Deleted: provides
Deleted: Figure 1
Deleted: processing framework for EC
Deleted: deployment
Deleted: framework
Deleted: ¶

To address the DevOps: Monitor stage, the computational resource load and performance statistics of operating eddy4R-Docker can easily be monitored with standard profiling procedures within NEON's CI or other compute infrastructures. eddy4R-Docker further utilizes the R logging package (0.7-103) to provide hierarchical logging, multiple handlers and formattable log records. Finally, end-user experience is monitored via the Issues feature in GitHub, where users can report code bugs, deployment problems, etc.

2.6 Installation and operation

One source of resistance to reproducible research is the initial burden of learning a new workflow. The eddy4R-Docker image aims to reduce the initial setup effort and learning requirements. This is achieved by providing a computational environment that contains all the necessary software dependencies, the Rstudio graphical development environment (https://www.rstudio.com/), and a code base consisting of example workflows and easily accessible functions. Combined with a simple and thoroughly documented installation procedure it provides a similar feel to working locally.

To work with the eddy4R-Docker image, one first needs to sign up at DockerHub (https://hub.docker.com/) and install the Docker host software following the Docker installation instructions (https://docs.docker.com/engine/getstarted/step_one/). Next, the download of the eddy4R-Docker image and subsequent creation of a container can be performed by two simple commands in an open shell (Linux/Mac) or the Docker Quickstart Terminal (Windows):

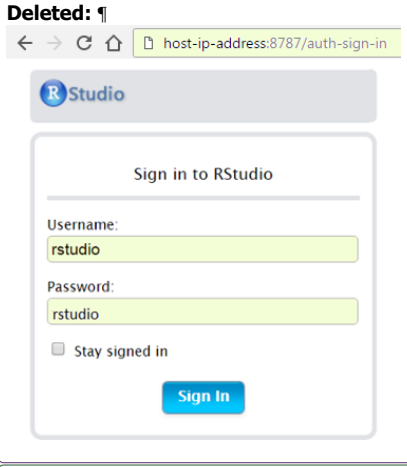
```
docker login
docker run -d -p 8787:8787 stefanmet/eddy4r:0.2.0
```

The first command will prompt for the user's DockerHub ID and password. The second command will download the latest eddy4R-Docker image and start a Docker container that utilizes port 8787 for establishing a graphical interface via web-browser. The release version of the Docker image can be specified, or alternatively the specifier `latest` provides the most up-to-date development image. In addition, it is possible to download and run a specific digest using the `docker run stefanmet/eddy4r@sha256:` command. If data is not directed from/to cloud hosting, a physical file system location on the host computer (`local/dir`) can be mounted to a file system location inside the Docker container (`docker/dir`). This is achieved with the `docker run` option `-v local/dir:docker/dir`.

The interactive Rstudio Server session running inside the Docker container can then be accessed via a web browser at http://host-ip-address:8787, using the IP address of the Docker host machine. The IP address of the Docker host can be determined by typing `localhost` in a shell session (Linux/Mac) or by typing `docker-machine ip default` in cmd.exe (Windows). Lastly, in the web browser the user can log into the RStudio session with username and password `rstudio` (see Figure 7 top left panel). Figure 7 also shows the Rstudio integrated development environment and interactive help for the eddy4R.base package in the bottom and top left panels, respectively. Additional information about the use of Rstudio and eddy4R packages in Docker containers can be found on the rocker-org/rocker website

Deleted: workflow templates

Deleted: ¶

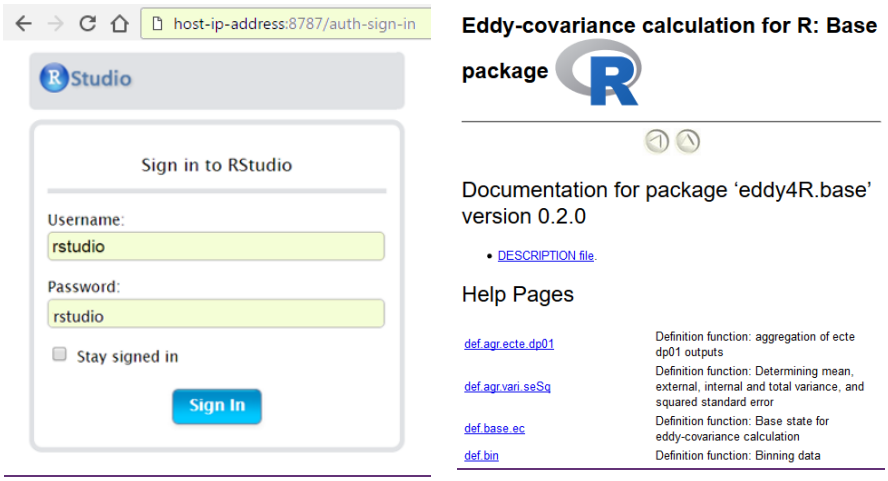


...

Moved down [7]: Figure 7 Docker-based Rstudio server session login via web-browser. Top left panel: Sign-in screen with highlighted areas showing information to input by the user. Top right panel: Interactive help for the eddy4R.base package. Bottom panel: Integrated development environment with workflow template, R console, Git staging area and eddy4R packages.¶
¶

Deleted: `v0.1`

(https://github.com/rocker-org/rocker/wiki/Using-the-RStudio-image) and the eddy4R Wiki pages.



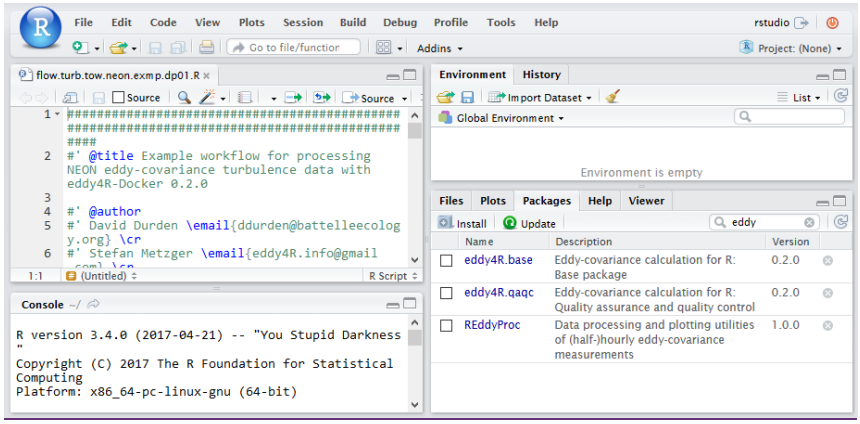


Figure 7 Docker-based Rstudio server session login via web-browser. Top left panel: Sign-in screen with highlighted areas showing information to input by the user. Top right panel: Interactive help for the eddy4R.base package. Bottom panel: Integrated development environment with workflow template, R console, Git staging area and eddy4R packages.

To demonstrate the ease of "Docker-assisted" data analysis and provide a template for potential eddy4R-Docker users, an executable example workflow and data are included in the eddy4R-Docker image. Once the eddy4R container is started, the example workflow, input data (NEON dp0p HDF5 file) and output data (NEON dp01 HDF5 file) are available from the Docker-

Deleted: , such as scaled deployment from the command line without graphical user interface

Moved (insertion) [7]

Deleted: Example

internal directory `/home/eddy/`. The example workflow is located at `/home/eddy/flowExmp/flow.turb.tow.neon.exmp.dp01.R`, and provides a selection of the processing steps that yield the EC dp01 data on the NEON data portal (https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0). The example workflow is fully documented to guide readers through the various processing steps, and employs key functionalities of the eddy4R.base and eddy4R.qaqc packages. These include data and metadata import from the input HDF5 file, data assignment to file-backed objects, processing of 1 minute and 30 minute data statistics and data quality, and writing the output HDF5 file. In addition, outputs from the quality flag and quality metric model are visualized.

As described above, the eddy4R Docker image can be used for code development (DevOps: Create stage) through accessing a running eddy4R Docker container via a web browser. Alternatively, the eddy4R Docker image can be used from the command line to perform scaled batch processing (DevOps: Configure & Monitor stages). Deployment from the command line consists of passing the R workflow file to the Docker image. This is achieved by using the `docker run` command with the additional argument `Rscript docker/dir/filename.R`, with `filename.R` being the desired workflow. Thus, the eddy4R Docker image can be used to simultaneously deploy multiple Docker containers to process data for multiple days or sites to the capacity of the computational platform.

3 Test applications

In the following we present three test applications of eddy4R-Docker to evaluate whether the NEON DevOps model can indeed produce collaborative, portable, reproducible, and extensible EC software. Code development, packaging, release, and operation followed the NEON DevOps model presented in this paper. Code modules have been contributed by order 10 individuals, distributed across multiple institutions and utilizing various computer systems. Nevertheless, each contributor achieved identical results per validation scripts during the DevOps: Verify & Package stage (Sect. 2.2), emphasizing the achieved portability and reproducibility. The majority of the calculations presented here were performed on 12 Intel Xeon X5550 2.67GHz CPUs, 32 GB memory with 10 Mbit interconnects and 10 Mbit access to 8 TB storage on an Oracle Zettabyte File System. The software specifications were CentOS 7 (3.10.0-327.el7.x86_64) with docker-engine (1.11.0). In Sect. 3.1, results of processing 12 days of EC data from a fixed tower at a NEON field site are shown. Next, in Sect. 0, we present the processing of EC fluxes from a 1-hour recording of a moving platform: airborne observations in a convectively mixed boundary layer. Lastly, a validation via software intercomparison is provided in Sect. 3.3.

Deleted: In the following we present three example applications of eddy4R-Docker. The calculations were performed on 12

Deleted:

Deleted:

3.1 Tower eddy-covariance measurements

Here, we use tower EC measurements to test a typical implementation of the eddy4R processing framework. The Smithsonian Environmental Research Center (SERC) in Edgewater, MD, USA is located on the Rhode and West Rivers, and hosts the NEON SERC tower (38°53'24.29" N, 76°33'36.04" W; 30 m a.s.l.). The ecosystem at SERC is a closed-canopy hardwood deciduous

Deleted: illustrate

forest dominated by tulip popular, oak and ash, with a mean canopy height of approximately 38 m (Figure 8). EC turbulent flux sensors are mounted at the tower top at 62 m above ground or 24 m above the forest canopy.

An enclosed infrared gas analyzer (IRGA, LI-COR Biosciences, Lincoln, NE, USA, model: LI-7200, firmware version 7.3.1.) was used to measure the turbulent fluctuations of H_2O and CO_2. A mass flow controller (Alicat Scientific, Burlington, VT, USA, model: MCRW-20 SLPM-DS-NEON) was used to maintain a constant flow rate of 12 SLPM through the IRGA cell. A sonic anemometer (Campbell Scientific, Logan, UT, USA, model: CSAT3, firmware version 3) was used to measure the 3-dimensional turbulent wind components. Data from the IRGA and the sonic anemometer was synchronized using triggering and network timing protocol, and collected simultaneously at 20 Hz sampling rate.

Figure 8. Left panel: Ecosystem at the NEON SERC tower (credit: Stephen Voss Photography; http://www.stephenvoss.com/blog/neon-tower-smithsonian). Right panels: EC instrumentation on top of the NEON SERC tower. Right top panel: Campbell Scientific CSAT-3 three-dimensional sonic anemometer (front) and LI-COR Biosciences LI-7200 infrared gas analyser (back) on the retracted tower-top boom. Right bottom panel: Same instrumentation but with the tower-top boom extended at 230° from true north.

Deleted: v7

Deleted: .

Deleted: v3

Moved down [8]: Here, data from April 22 to May 3, 2016 were used. The mean temperature during this time period was 15°C, with a maximum temperature of 29°C and a minimum of 8°C. A total of 15 mm of precipitation was observed at nearby Annapolis Naval Academy.¶

Formatted: Font: 12 pt

Formatted Table

Here, data from April 22 to May 3, 2016 were used. The mean temperature during this time period was 15°C, with a maximum temperature of 29°C and a minimum of 8°C. A total of 15 mm of precipitation was observed at nearby Annapolis Naval Academy.

Moved (insertion) [8]

3.1.1 Algorithm settings and profiling

The eddy4R workflow file was configured to ingest on the order of 50 data streams at 20 Hz, including 3-D wind components, sonic temperature, and H_2O and CO_2 concentrations. The data were processed to half-hourly L1 data products and turbulent fluxes. The L1 data products are essentially state variables (wind, temperature, concentrations) with basic statistical products derived, i.e. mean, minimum, maximum, standard error of the mean and variance. The algorithmic processing for the L4 flux calculations requires additional scientific and procedural complexity to test assumptions of the EC theory. The resultant fluxes represent half-hourly vertical turbulent exchanges between the earth's surface and the atmosphere corresponding to these state variables.

Deleted: theoretical

For the datasets analyzed in this study, the L0p input file sizes ranged from 0.1 – 0.2 GB in HDF5 format depending on the amount of missing data, with metadata attached as attributes. We used the simple data format for our HDF5 files, as opposed to compound data type, this resulted in reduced read in time from 60 seconds to 3 seconds for 20 Hz IRGA data. Elementary testing indicates that in this framework 6 CPU-minutes were required to process 1 day of 20 Hz L0 data, and 1.2 CPU-minutes per 1 day of L0p data (100,000,000 observations). This difference arises mainly from application of plausibility tests per Taylor and Loescher (2013) in the transition from L0 to L0p. No reduction in efficiency was observed between direct software deployment and its Docker implementation. Once flux QA/QC and uncertainty budget is implemented, the computational expense will likely increase by a factor of two to three. This suggests that eddy4R performs comparably to other flux processors. Memory usage is kept below 2 GB through the use of fast access file-backed objects, enabling more sophisticated scientific analyses through access to multiple days of data without overloading random access memory (RAM) resources. Additionally, the snowfall R package allows for logical parallelization frameworks to be implemented in the processing framework, even at low-level analysis steps.

Deleted: scientific

3.1.2 Results and discussion

The time series ranging from April 22 to May 3, 2016 was processed to deliver both state (L1) and flux (L4) quantities; however, the initial eddy4R package release will only contain functions necessary to report state variables or L1 data products in the NEON data product description. During the processing of the proof-of-concept results averaging periods with >10% missing data (incl. bad sensor diagnostic flags) were removed, and dedicated flux QA/QC and uncertainty quantification were disabled.

Deleted: The QA/QC and uncertainty frameworks were not fully implemented during

Deleted: , but

Figure 9 shows the resultant time series of shear stress (friction velocity), sensible heat, latent heat and CO_2 flux. The derived values fall into typical ranges for mid-latitude hardwood forests in spring. As expected, fluxes follow the general trends in the scalar quantities. Good data

coverage can be seen for the LI-7200 measurements even during the rainy period at the end of the analysis. A footprint analysis revealed that 90% of the flux measurement signals were sourced within 800 m from the tower, and 80% were within 500 m from the tower at our site. Data coverage was reduced after day of year (DOY) 120 due to inclement weather conditions.

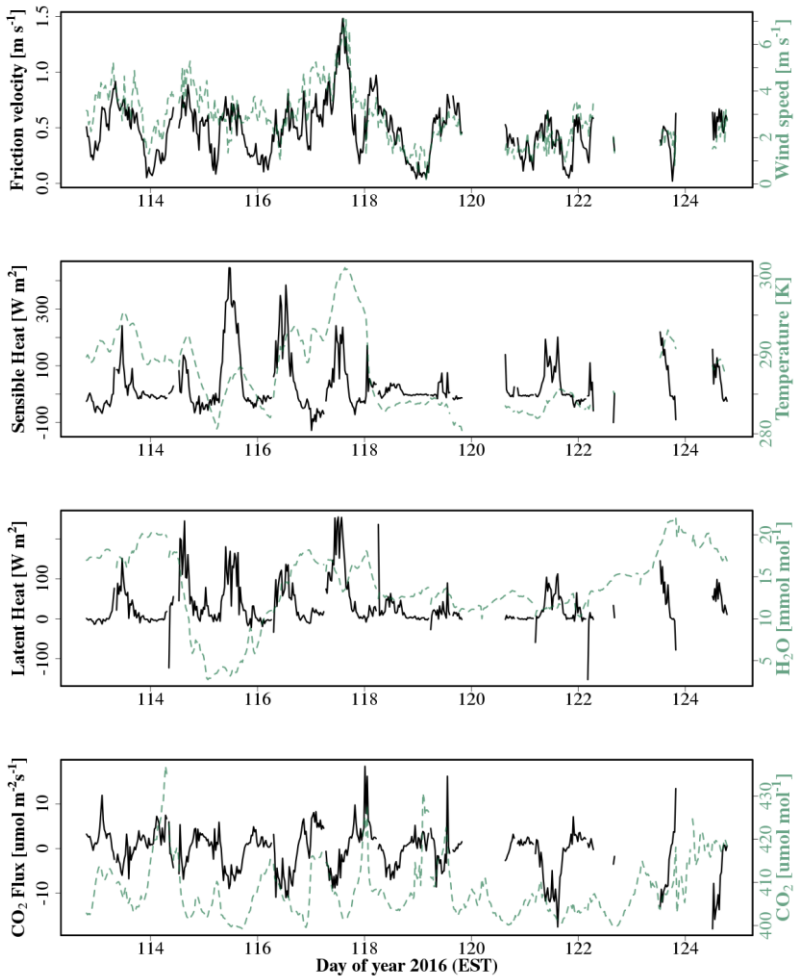
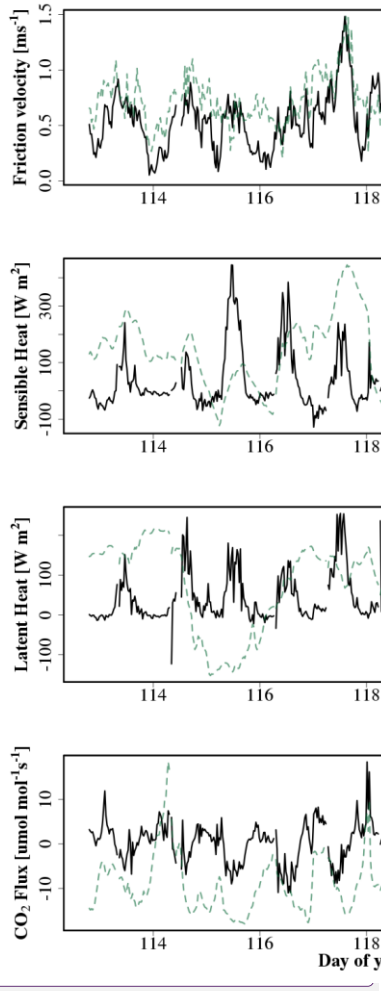


Figure 9. Time-series of turbulent fluxes derived from EC measurements atop the NEON SERC tower. Top to bottom: Vertical turbulent exchange of shear (friction velocity) and wind speed, sensible heat and temperature, latent heat and H_2O dry mole fraction and CO_2 flux and CO_2 dry mole fraction.

Deleted: The spiky results preceding and following periods with >10% invalid data highlight the need for scientific QA/QC and uncertainty budget to provide science-grade fluxes. Nonetheless, this implementation of eddy4R in a Docker image, as it will interact with NEON CI, clearly demonstrates its core capability to generate L1-L4 data products.



Deleted:

The spiky results preceding and following periods with >10% invalid data highlight the need for enabling the full flux QA/QC and uncertainty budget to subset science-grade fluxes. This implementation of eddy4R in a Docker image, as it will interact with NEON CI, clearly demonstrates the applicability of the DevOps model for generating EC L1-L4 data products.

3.2 Aircraft eddy-covariance measurements

Here, we use aircraft EC measurements to test more advanced scientific capabilities of the eddy4R processing framework. Airborne turbulent flux observations were performed along more than 3100 km of low level (i.e. 50 m above ground level) flights across the North Slope of Alaska, USA in July 2012, using the research aircraft Polar 5 (Tetzlaff et al., 2015). The example data used in this manuscript were recorded during a SSW-NNE flight line near the village of Atqasuk, Alaska, above tundra dominated by sedges and emerging herbaceous wetland vegetation. Large, often oriented lakes and the meandering Meade River characterize the surrounding landscape.

The aircraft was equipped with a 3 m nose boom holding a 5-hole probe for wind measurements, an open wire Pt100 in an unheated Rosemount housing for air temperature measurements, and an HMT-330 (Vaisala, Helsinki, Finland) in a Rosemount housing for relative humidity. Sample air was drawn from an inlet above the cabin at about 9.7 l s^{-1}, analysed in an RMT-200 cavity-ringdown trace gas sensor (Los Gatos Research Inc., Mountain View, California, USA) and recorded at 20 Hz. Aircraft position, velocity and attitude was provided by several Global Positioning Systems (NovAtel Inc., Calgary, Alberta, USA) and an Inertial Navigation System (Laseref V, Honeywell International Inc., Morristown, New Jersey, USA). Height above ground was determined by a radar altimeter (KRA 405B/ Honeywell International Inc., Morristown, New Jersey, USA) and a laser altimeter (LD90/ RIEGL Laser Measurements Systems GmbH, Horn, Austria). The input data used in this study included the pre-derived 3-D wind vector from 5-hole probe and aircraft position, velocity and attitude. After spike removal the sampling frequency of the original data was reduced from 100 Hz to 20 Hz resolution using block averaging. These steps were performed prior to import into eddy4R processing, but could equally well be performed therein.

3.2.1 Algorithm settings and profiling

Here, aircraft-measured vertical wind speed and CH_4 dry mole fraction were analysed to determine CH_4 emissions by means of a time-frequency-resolved version of the EC method (Metzger et al., 2013). For this purpose a combination of settings were chosen in the eddy4R workflow file that differ from Sect. 3.1: Initially the small (<1 MB) EC raw data file consisting of 17 variables and 12,800 data points (or 42 km flight data) was read in ASCII Gzip format – standard R capabilities for data ingest can be used to read data in various formats, frequencies and units. Aircraft-measured vertical wind speed and CH_4 dry mole fraction were then correlated using a Wavelet transform (Metzger et al., 2013). This process includes ranging and de-spiking of unphysical raw data values (Mauder et al., 2013; Metzger et al., 2012), fast dry mole fraction derivation (e.g., Burba et al., 2012) and spectroscopic correction (Tuzson et al., 2010) of CH_4 trace gas observations, and high-frequency spectral correction (Ammann et al.,

Deleted: illustrate

Deleted: (Tetzlaff et al., 2015)

Deleted: ,

Deleted:), altitude

Deleted: (Metzger et al., 2013)

Deleted:

Deleted:

Deleted: (Metzger et al., 2013)

Deleted: considers

Deleted: (Mauder et al., 2013; Metzger et al., 2012)

Deleted: (Tuzson et al., 2010)

Deleted: cavity-ringdown

2006) by means of applying a sigmoidal transfer function (Eugster and Senn, 1995) directly in Wavelet space. This permits estimating turbulent fluxes with improved spatial discretization and determining ~100 biophysically relevant surface properties in the flux footprint. The analysis took 56 minutes with 8-fold parallelization and consumed <3 GB RAM thanks to the use of fast access file-backed objects.

3.2.2 Results and discussion

The resulting Wavelet cross-scalogram (Figure 10) is integrated in frequency over transport scales up to 20 km, and along the flight path over a 1000 m moving window with 100 m step size, similar to the resolution of the land surface data. The result is an in-situ observed space-series of the CH_4 surface-atmosphere exchange at 100 m spatial resolution. Analogously, turbulence statistics characterizing shear stress and buoyancy are determined for characterizing the atmospheric transport between the emitting land surface and the aircraft position.

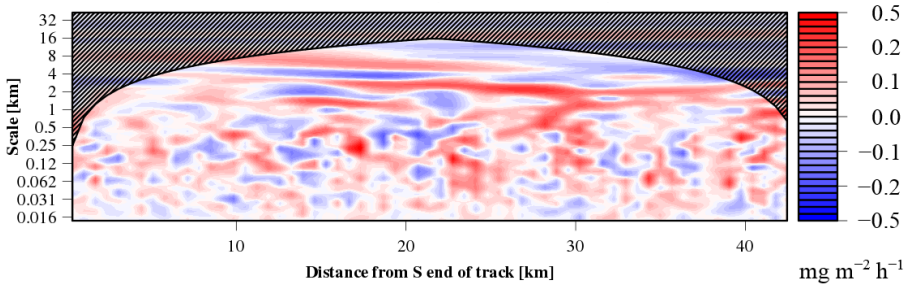


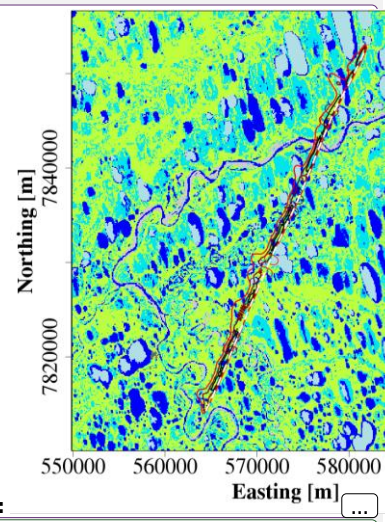
Figure 10. Wavelet cross-scalogram of the CH_4 flux equivalent to a time (x-axis) frequency (y-axis) resolved version of EC. For each combination of aircraft position and eddy size, blue and red areas indicate transport toward and away from the surface, respectively.

Corresponding systematic and random statistical errors are calculated following Lenschow and Stankov (1986) and Lenschow et al. (1994), and the flux detection limit is calculated after Billesbach (2011).

The relationship between the aircraft-observed CH_4 surface-atmosphere exchange and land surface properties is established through an atmospheric transport operator, the so-called flux footprint function (e.g., Schmid, 1994). Here we use a computationally efficient one-dimensional parameterization of a Lagrangian particle model for the along-wind footprint extent (Kljun et al., 2002; Kljun et al., 2004), combined with an analytical approach to determine cross-wind surface contributions to each 100 m aircraft measurement, depending on aircraft position (Figure 11; Metzger et al., 2012).

For each 100 m observation of the CH_4 surface-atmosphere exchange an individual footprint weight matrix derived from the footprint parameterization is convolved with the land surface

Deleted: (Eugster and Senn, 1995)



Deleted: ...

Moved down [9]: Figure 11. The composite flux footprint along the flight line (30 %, 60 %, 90% contour lines) superimposed over the National Land Cover Database. The white dashed line represents the aircraft flight track.

Deleted: Page Break

¶

Corresponding systematic and random statistical errors are calculated following Lenschow and Stankov (1986) and Lenschow *et al*. (1994),

Deleted: (e.g., Schmid, 1994)

Deleted: (Kljun et al., 2002; Kljun et al., 2004)

Deleted: (Figure 10; Metzger et al., 2012)

drivers. The results are space-series of land surface contributions accompanying the CH_4 measured surface-atmosphere exchange (Figure 12).

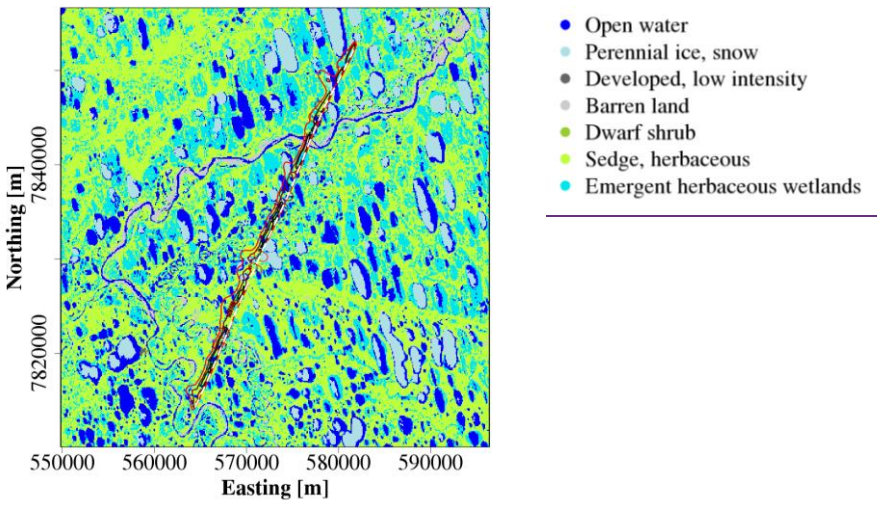


Figure 11. The composite flux footprint along the flight line (30 %, 60 %, 90% contour lines) superimposed over the National Land Cover Database. The white dashed line represents the aircraft flight track.

Moved (insertion) [9]

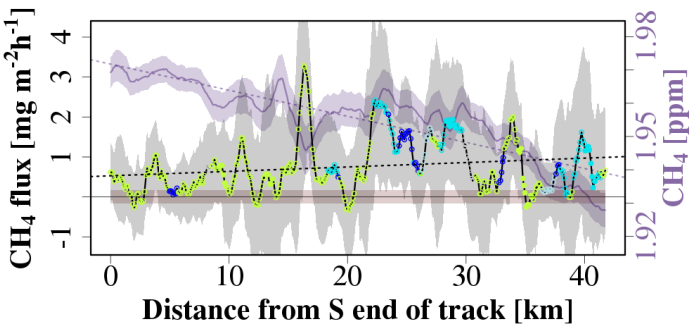


Figure 12. Space-series of 399 CH_4 concentration (purple line) and flux (black line) observations each 100 m, averaged over 1000 m windows. The random sampling errors are indicated by the shaded areas enveloping each line, and the flux detection limit is shown as salmon envelope around the abscissa. Circles indicate the dominating land cover in the footprint of each observation (Figure 11) with full circles corresponding to 'pure' fluxes (>80% surface contribution).

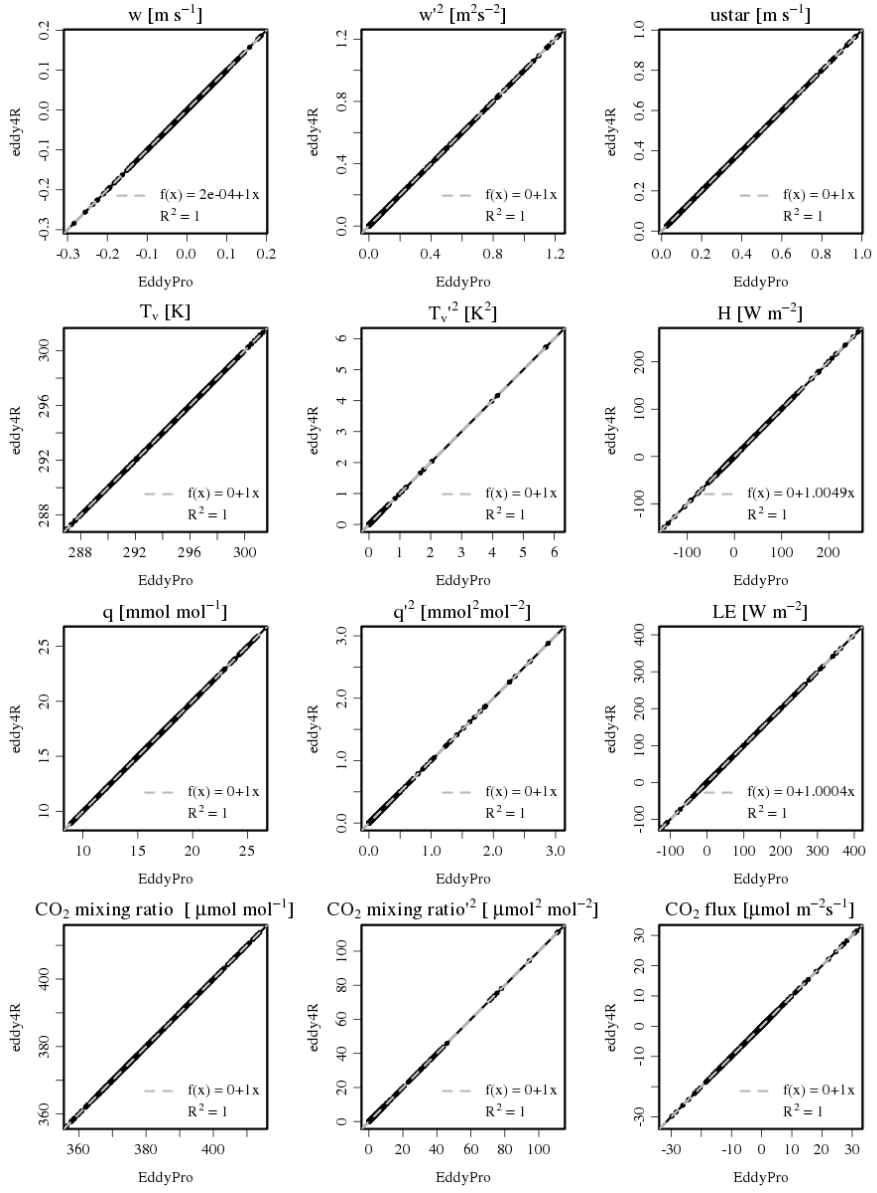


Figure 13. Scatterplot of means (vertical wind speed, w; sonic temperature, T_v; H_2O dry mole fraction (q); and CO_2 dry mole fraction), variances and fluxes (friction velocity, ustar; sensible heat flux, H; latent heat, LE; CO_2 flux). Data are generated from 2011 July to Aug WLEF data in EddyPro and eddy4R. Each point represents a one-hour averaging period. Black lines are 1:1 lines, and dashed lines are robust regressions (Salibian-Barrera and Yohai, 2006).

The successful application of eddy4R-Docker to both, basic tower and advanced aircraft EC data analyses, highlights how the DevOps model promotes modular extensibility.

3.3 Validation and verification

eddy4R includes a verification script which automatically processes subsets of the tower and aircraft data introduced in Sect. 3.1 and Sect. 3.2, and verifies the results against a reference, e.g. generated with a different software.

Here, we demonstrate such approach at the Park Falls, Wisconsin WLEF very tall tower Ameriflux site (US-PFa). The 447 m tall WLEF television tower (45.946◦N, 90.272◦W) has been instrumented for EC measurements in 1996, and is part of the AmeriFlux network. Flux measurements at 30 m, 122 m and 396 m sample a mixed landscape of forests and wetlands (Desai et al., 2015). The surrounding forest canopy has approximately 70% deciduous and 30% coniferous trees, and a mean canopy height of 20 m. The site has an interior continental climate. Instrumentation at each level consists of fast response wind speed and temperature from a sonic anemometer (Applied Technologies., Inc., Seattle, USA, ATI Type K). 10 Hz dry mole fraction of CO_2 and H_2O at the 122 m level used here were measured by a closed-path infrared gas analyzer (LI-COR, Inc., Lincoln, USA, LI-6262) located on the tower.

A data set from July 27 to August 19, 2011 was used in the intercomparison between eddy4R and the reference software EddyPro (LI-COR, Inc., Lincoln, USA, version 6.2.0). EddyPro was released in April 2011 and is widely used in the EC community.

3.3.1 Algorithm settings

Several preprocessing steps were applied, and the resulting data and settings were used in both, eddy4R and EddyPro: (i) The raw data was pre-cleaned in eddy4R using the Brock (1986) de-spiking algorithm with a filter width of 9 data points for all variables. (ii) EddyPro was used to calculate the planar-fit rotation parameters (Wilczak et al., 2001) over the entire dataset (offset = -0.06 ms^{-1}, pitch = $-5.27°$, roll = $-1.81°$). (iii) Time lags for dry mole fractions of CO_2 (0.8 s behind vertical wind) and H_2O (0.1 s behind vertical wind) were calculated in eddy4R using maximum correlation (median lag time over entire dataset).

Because CO_2 and H_2O fluxes were calculated from dry mole fractions, the Webb et al. (1980) density correction was not necessary and therefore not applied (Burba et al., 2012). Frequency response correction was not considered in this validation and therefore not applied. Means, variances and fluxes were calculated on the basis of one-hour block averages. Based on Schotanus et al. (1983), sensible heat flux was calculated from point-by-point conversion of sonic temperature in eddy4R, and with the half-hourly statistical correction in EddyPro.

3.3.2 Results and discussion

eddy4R and EddyPro produce nearly identical results (Figure 13), and the gain error is within 0.04% for most outputs. Sensible heat flux values produced by eddy4R have slightly larger magnitude compared to EddyPro, by 0.49%. This is likely a result of the different methods applied when converting sonic temperature to air temperature. This intercomparison confirms that applying the DevOps model to scientific EC software achieved results comparable to

Deleted: 3.2

Deleted: an

Deleted: Page Break ¶

Deleted: v6

Deleted: being

Deleted: (Wilczak et al., 2001)

Deleted: Webb et al. (1980)

Deleted: Schotanus et al. (1983)

commercial-grade software. A detailed end-to-end intercomparison considering additional processing steps and EC software is planned for a separate manuscript accompanying NEON's release of flux data products.

4 Summary and conclusions

Adopting a DevOps philosophy has facilitated the creation of a universal processing environment for producing NEON's EC data products. Portable, reproducible, and extensible software is reliably and efficiently created by incorporating the DevOps workflow steps of Plan, Create, Verify, Package, Release, Configure, and Monitor into a NEON-specific DevOps model based on the tools R, Git, HDF5, and Docker. Git-distributed version control facilitates simultaneous internal-external collaboration on scientific algorithms, the outcome being a modular family of open-source R packages. The use of Hierarchical Data Format allows for efficient, self-describing data input and output. Docker images package the entire processing environment for robust, scalable, and portable deployment. The capability of this framework was demonstrated with cross-validated tower and aircraft fluxes.

The results presented here are from a file-based implementation of the eddy4R Docker workflow, with EC instrument data accessed directly e.g. from the NEON site and manually processed into the HDF5 ingest format (Sect. 1.1). The subsequent focus is the operational implementation of the eddy4R-Docker workflow for reporting means and variances. This includes: (i) Automated ingest of streaming raw data into the NEON database; (ii) Processing of raw data into the standard, defined inputs required by the eddy4R-Docker in HDF5 format, and (iii) Developing the software and hardware infrastructure to pass data and instructions back and forth to the eddy4R-Docker workflow, and control program execution in a distributed computing framework.

Remaining scientific algorithms are being integrated into eddy4R-Docker for producing turbulent exchange data products. These algorithms include on-the-fly de-spiking, lag correction, planar-fit and spectral correction, flux QA/QC, and uncertainty budget estimation. Finally, eddy4R-Docker is being expanded to include "storage" and "derived" workflows (Figure 6) for generating reproducible net ecosystem exchange data products in 2018. Lessons learned here will profit the community at large, e.g. through enabling streaming processing directly at an EC site or over cellular modems with the same eddy4R-Docker open-source software as used for sophisticated analyses (Sect. 3.2). Already now, the executable example workflow and data included in eddy4R-Docker image invite the reader to realize their own end-to-end data analysis and apply it to their data (Sects. 2.6, 5).

While our sole focus in developing and implementing this model has been to generate EC data products with the unique capabilities and constraints of NEON, it has become clear that the NEON DevOps model enables the implementation of a suite of complex processing algorithms, such as temporal gap filling of sensor time series data or modeling re-aeration rates. There exist many potential synergies between NEON, other tower networks, and the user community for producing high level EC data products. We hope this framework can serve as a model for implementing community-sourced, distributed-development scientific code while combatting

Deleted: flexible, scalable, and extensible

Deleted: Focus now shifts to

Deleted: 3.2

Deleted: Thereafter, remaining scientific algorithms will be integrated in eddy4R-Docker for producing defensible turbulent exchange data products. These algorithms include on-the-fly de-spiking, lag correction, planar-fit and spectral correction, scientific QA/QC, and uncertainty budget estimation. Finally, the eddy4R-Docker will be expanded to include "storage" and "derived" workflows (Figure 6) for producing defensible net ecosystem exchange data products in 2018.¶

Deleted: framework

Deleted: facilitate generating

Deleted: framework has the potential for enabling

the deficiencies of current computational frameworks that limit accessibility, reproducibility, and extensibility.

5 Code and data availability

The source code packages eddy4R.base (0.2.0) and eddy4R.qaqc (0.2.0) used in this study are archived at https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/code/0.2.0, under the GNU Affero General Public License (GNU AGPLv3). Similarly, the corresponding eddy4R-Docker image (0.2.0), including an executable example workflow and data, is available at https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/docker/0.2.0. In addition, a data supplement is provided at https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/data/0.2.0, including an extended abstract and all NEON SERC raw data used in this study, accompanied by variable documentation. Lastly, NEON EC data products generated with eddy4R-Docker are available at https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/portal/0.2.0.

Deleted: 1

Deleted: 1

Deleted: https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/code

Deleted: at https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/docker

Deleted: 1.0)

Deleted: . Lastly

Deleted: https://w3id.org/smetzger/Metzger-et-al_2017_eddy4R-Docker/data

Acknowledgements

Many colleagues at Battelle Ecology supported this study. In particular, Santiago Bonarrigo provided pre-parsed high-frequency data from the SERC site, and Andrew Fox (now: National Center for Atmospheric Research), Mike SanClements and David Hulslander commented on an earlier version of the manuscript, Henry Loescher and Leslie Goldman designed Figure 6, and Andrea Thorpe (now: Washington Natural Heritage Program), Thomas Gulbransen and Michael Kuhlman helped shepherding this study and its publication through required administrative procedures. Special thanks goes to Timothy Brown at the National Oceanic and Atmospheric Administration, for numerous discussions and invaluable advice on scientific computing. The National Ecological Observatory Network is a project sponsored by the National Science Foundation and managed under cooperative agreement by Battelle Ecology, Inc. This material is based upon work supported by the National Science Foundation under the grant DBI-0752017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Ankur Desai acknowledges support from NSF DBI-1457897 and DOE Office of Science Ameriflux Network Management Project core site support to the ChEAS cluster. Torsten Sachs and Andrei Serafimovich are supported by the Helmholtz Association of German Research Centres through a Helmholtz Young Investigators Group grant to Torsten Sachs (grant VH-NG-821).

Deleted: , and Andrea Thorpe,

References

Ammann, C., Brunner, A., Spirig, C., and Neftel, A.: Technical note: Water vapour concentration and flux measurements with PTR-MS, Atmos. Chem. Phys., 6, 4643-4651, doi:10.5194/acp-6-4643-2006, 2006.

Aubinet, M., Vesala, T., and Papale, D., (Eds.): Eddy covariance: A practical guide to measurement and data analysis, Springer, Dordrecht, Heidelberg, London, New York, 438 pp., 2012.

Baldocchi, D., Falge, E., Gu, L., Olson, R., Hollinger, D., Running, S., Anthoni, P., Bernhofer, C., Davis, K., Evans, R., Fuentes, J., Goldstein, A., Katul, G., Law, B., Lee, X., Malhi, Y., Meyers, T., Munger, W., Oechel, W., U, K., Pilegaard, K., Schmid, H., Valentini, R., Verma, S., Vesala, T., Wilson, K., and Wofsy, S.: FLUXNET: A new tool to study the temporal and spatial variability of ecosystem-scale carbon dioxide, water vapor, and energy flux densities, Bull. Am. Meteorol. Soc., 82, 2415-2434, doi:10.1175/1520-0477(2001)082<2415:FANTTS>2.3.CO;2, 2001.

Billesbach, D. P.: Estimating uncertainties in individual eddy covariance flux measurements: A comparison of methods and a proposed new method, Agric. For. Meteorol., 151, 394-405, doi:10.1016/j.agrformet.2010.12.001, 2011.

Boettiger, C.: An introduction to Docker for reproducible research, with examples from the R environment, Operating Systems Review, 49, 71-79, doi:10.1145/2723872.2723882, 2015.

Brock, F. V.: A nonlinear filter to remove impulse noise from meteorological data, J. Atmos. Oceanic Technol., 3, 51-58, doi:10.1175/1520-0426(1986)003<0051:anftri>2.0.co;2, 1986.

Burba, G., Schmidt, A., Scott, R. L., Nakai, T., Kathilankal, J., Fratini, G., Hanson, C., Law, B., McDermitt, D. K., Eckles, R., Furtaw, M., and Velgersdyk, M.: Calculating CO_2 and H_2O eddy covariance fluxes from an enclosed gas analyzer using an instantaneous mixing ratio, Global Change Biol., 18, 385-399, doi:10.1111/j.1365-2486.2011.02536.x, 2012.

Chen, L.: Continuous delivery: Huge benefits, but challenges too, IEEE Softw., 32, 50-54, doi:10.1109/ms.2015.27, 2015.

Clark, D., Culich, A., Hamlin, B., and Lovett, R.: BCE: Berkeley's common scientific compute environment for research and education, Proceedings of the 13th Python in Science Conference (SCIPY 2014) Austin, USA, 2014.

Clement, R. J., Burba, G. G., Grelle, A., Anderson, D. J., and Moncrieff, J. B.: Improved trace gas flux estimation through IRGA sampling optimization, Agric. For. Meteorol., 149, 623-638, doi:10.1016/j.agrformet.2008.10.008, 2009.

Collberg, C., Proebsting, T., Moraila, G., Shankaran, A., Shi, Z., and Warren, A. M.: Measuring reproducibility in computer systems research, University of Arizona, Department of Computer Science, Tucson, USA, 37, 2014.

De Roo, F., Abdul Huq, S. U., Metzger, S., Desai, A. R., Xu, K., and Mauder, M.: On the benefit of driving large-eddy simulation with spatially resolved surface fluxes derived from environmental response functions, TERENO International Conference, Bonn, Germany, 29 September - 2 October, 2014.

Desai, A. R., Xu, K., Tian, H., Weishampel, P., Thom, J., Baumann, D., Andrews, A. E., Cook, B. D., King, J. Y., and Kolka, R.: Landscape-level terrestrial methane flux observed from a very tall tower, Agric. For. Meteorol., 201, 61-75, doi:10.1016/j.agrformet.2014.10.017, 2015.

Erich, F., Amrit, C., and Daneva, M.: A mapping study on cooperation between information system development and operations, 15th International Conference on Product-Focused Software Process Improvement, PROFES 2014, Helsinki, Finland, 2014.

Eugster, W., and Senn, W.: A cospectral correction model for measurement of turbulent NO_2 flux, Boundary Layer Meteorol., 74, 321-340, doi:10.1007/bf00712375, 1995.

Foken, T., and Wichura, B.: Tools for quality assessment of surface-based flux measurements, Agric. For. Meteorol., 78, 83-105, doi:10.1016/0168-1923(95)02248-1, 1996.

Foken, T.: Micrometeorology, 2 ed., Springer, Berlin, Heidelberg, 362 pp., 2017.

Fratini, G., and Mauder, M.: Towards a consistent eddy-covariance processing: An intercomparison of EddyPro and TK3, Atmos. Meas. Tech., 7, 2273-2281, doi:10.5194/amt-7-2273-2014, 2014.

Kljun, N., Rotach, M. W., and Schmid, H. P.: A three-dimensional backward lagrangian footprint model for a wide range of boundary-layer stratifications, Boundary Layer Meteorol., 103, 205-226, doi:10.1023/A:1014556300021, 2002.

Kljun, N., Calanca, P., Rotach, M. W., and Schmid, H. P.: A simple parameterisation for flux footprint predictions, Boundary Layer Meteorol., 112, 503-523, doi:10.1023/B:BOUN.0000030653.71031.96, 2004.

Kljun, N., Calanca, P., Rotach, M. W., and Schmid, H. P.: A simple two-dimensional parameterisation for Flux Footprint Prediction (FFP), Geosci. Model Dev., 8, 3695-3713, doi:10.5194/gmd-8-3695-2015, 2015.

Kohnert, K., Serafimovich, A., Metzger, S., Hartman, J., and Sachs, T.: Geogenic sources strongly contribute to the Mackenzie River Delta's methane emissions derived from airborne flux data, 48th AGU annual Fall Meeting, San Francisco, U.S.A., 14 - 18 December, 2015.

Kormann, R., and Meixner, F. X.: An analytical footprint model for non-neutral stratification, Boundary Layer Meteorol., 99, 207-224, doi:10.1023/A:1018991015119, 2001.

Law, B.: AmeriFlux network aids global synthesis, Eos, Transactions American Geophysical Union, 88, 286-286, doi:10.1029/2007eo280003, 2007.

Lee, J., Vaughan, A., Lewis, A., Shaw, M., Purvis, R., Carlslaw, D., Hewitt, C., Misztal, P., Metzger, S., Beevers, S., Goldstein, A., Karl, T., and Davison, D.: Spatially resolved emissions of NO_x and VOCs and comparison to inventories, 48th AGU annual Fall Meeting, San Francisco, U.S.A., 14 - 18 December, 2015.

Lenschow, D. H., and Stankov, B. B.: Length scales in the convective boundary layer, Journal Of The Atmospheric Sciences, 43, 1198-1209, doi:10.1175/1520-0469(1986)043<1198:LSITCB>2.0.CO;2, 1986.

Lenschow, D. H., Mann, J., and Kristensen, L.: How long is long enough when measuring fluxes and other turbulence statistics?, J. Atmos. Oceanic Technol., 11, 661-673, doi:10.1175/1520-0426(1994)011<0661:HLILEW>2.0.CO;2, 1994.

Loukides, M.: What is DevOps? Infrastructure as Code, O'Reilly Media, Ebook, Safari Books Online, 15 pp., 2012.

Mammarella, I., Peltola, O., Nordbo, A., Järvi, L., and Rannik, Ü.: Quantifying the uncertainty of eddy covariance fluxes due to the use of different software packages and combinations of

Deleted: 306

Deleted: 2008

processing steps in two contrasting ecosystems, Atmos. Meas. Tech., 9, 4915-4933, doi:10.5194/amt-9-4915-2016, 2016.

Mauder, M., Cuntz, M., Drüe, C., Graf, A., Rebmann, C., Schmid, H. P., Schmidt, M., and Steinbrecher, R.: A strategy for quality and uncertainty assessment of long-term eddy-covariance measurements, Agric. For. Meteorol., 169, 122-135, doi:10.1016/j.agrformet.2012.09.006, 2013.

Mauder, M., and Foken, T.: Eddy-covariance software TK3 [Data set]. Documentation and instruction manual of the eddy-covariance software package TK3 (update). University of Bayreuth, Bayreuth, Germany, doi:10.5281/zenodo.20349, 2015.

Metzger, S., Junkermann, W., Mauder, M., Beyrich, F., Butterbach-Bahl, K., Schmid, H. P., and Foken, T.: Eddy-covariance flux measurements with a weight-shift microlight aircraft, Atmos. Meas. Tech., 5, 1699-1717, doi:10.5194/amt-5-1699-2012, 2012.

Metzger, S., Junkermann, W., Mauder, M., Butterbach-Bahl, K., Trancón y Widemann, B., Neidl, F., Schäfer, K., Wieneke, S., Zheng, X. H., Schmid, H. P., and Foken, T.: Spatially explicit regionalization of airborne flux measurements using environmental response functions, Biogeosciences, 10, 2193-2217, doi:10.5194/bg-10-2193-2013, 2013.

Metzger, S., Burba, G., Burns, S. P., Blanken, P. D., Li, J., Luo, H., and Zulueta, R. C.: Optimization of an enclosed gas analyzer sampling system for measuring eddy covariance fluxes of H_2O and CO_2, Atmos. Meas. Tech., 9, 1341-1359, doi:10.5194/amt-9-1341-2016, 2016.

Nordbo, A., and Katul, G.: A wavelet-based correction method for eddy-covariance high-frequency losses in scalar concentration measurements, Boundary Layer Meteorol., 146, 81-102, doi:10.1007/s10546-012-9759-9, 2012.

Paarsch, H. J., and Golyaev, K.: A gentle introduction to effective computing in quantitative research: What every research assistant should know, MIT Press, Cambridge, USA, 776 pp., 2016.

Papale, D., Reichstein, M., Aubinet, M., Canfora, E., Bernhofer, C., Kutsch, W., Longdoz, B., Rambal, S., Valentini, R., Vesala, T., and Yakir, D.: Towards a standardized processing of Net Ecosystem Exchange measured with eddy covariance technique: algorithms and uncertainty estimation, Biogeosciences, 3, 571-583, doi:10.5194/bg-3-571-2006, 2006.

R Core Team: R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, 2016.

Ram, K.: Git can facilitate greater reproducibility and increased transparency in science, Source Code Biol. Med., 8, 7, doi:10.1186/1751-0473-8-7, 2013.

Raupach, M. R., Rayner, P. J., Barrett, D. J., DeFries, R. S., Heimann, M., Ojima, D. S., Quegan, S., and Schmullius, C. C.: Model-data synthesis in terrestrial carbon observation: Methods, data requirements and data uncertainty specifications, Global Change Biol., 11, 378-397, doi:10.1111/j.1365-2486.2005.00917.x, 2005.

Deleted: and Foken, T.: Documentation and instruction manual of the eddy-covariance software package TK3, Universität Bayreuth, Arbeitsergenisse Abteilung Mikrometeorologie, 46, Bayreuth, Germany, 60 pp., ISSN 1614-8924, 2011.¶
Mauder, M.,

Running, S. W., Baldocchi, D. D., Turner, D. P., Gower, S. T., Bakwin, P. S., and Hibbard, K. A.: A global terrestrial monitoring network integrating tower fluxes, flask sampling, ecosystem modeling and EOS satellite data, Remote Sens. Environ., 70, 108-127, doi:10.1016/S0034-4257(99)00061-9, 1999.

Sachs, T., Serafimovich, A., Metzger, S., Kohnert, K., and Hartmann, J.: Low permafrost methane emissions from arctic airborne flux measurements, 47th AGU annual Fall Meeting, San Francisco, U.S.A., 15 - 19 December, 2014.

Salibian-Barrera, M., and Yohai, V. J.: A fast algorithm for S-regression estimates, Journal Of Computational And Graphical Statistics, 15, 414-427, 2006.

Salmon, O., Caulton, D., Shepson, P., Brian, S., Metzger, S., and Musinsky, J.: Attributing airborne measurements of forest CO_2 exchange to finer spatial scales, 5th NACP Principal Investigators Meeting, Washington D.C., U.S.A., 26 - 29 January, 2015.

Schimel, D., Hargrove, W., Hoffman, F., and MacMahon, J.: NEON: a hierarchically designed national ecological network, Frontiers in Ecology and the Environment, 5, 59-59, doi:10.1890/1540-9295(2007)5[59:nahdne]2.0.co;2, 2007.

Schmid, H. P.: Source areas for scalars and scalar fluxes, Boundary Layer Meteorol., 67, 293-318, doi:10.1007/bf00713146, 1994.

Schotanus, P., Nieuwstadt, F. T. M., and Bruin, H. A. R.: Temperature measurement with a sonic anemometer and its application to heat and moisture fluxes, Boundary Layer Meteorol., 26, 81-93, doi:10.1007/BF00164332, 1983.

Serafimovich, A., Metzger, S., Kohnert, K., Hartmann, J., and Sachs, T.: The airborne measurements of methane fluxes (AIRMETH) arctic campaign, 46th AGU annual Fall Meeting, San Francisco, U.S.A., 9 - 13 December, 2013.

Smith, D. E., Metzger, S., and Taylor, J. R.: A transparent and transferable framework for tracking quality information in large datasets, PLoS One, 9, e112249, doi:10.1371/journal.pone.0112249, 2014.

Starkenburg, D., Metzger, S., Fochesatto, G. J., Alfieri, J. G., Gens, R., Prakash, A., and Cristóbal, J.: Assessment of de-spiking methods for turbulence data in micrometeorology, J. Atmos. Oceanic Technol., doi:10.1175/jtech-d-15-0154.1, 2016.

Stull, R. B.: An Introduction to Boundary Layer Meteorology, Kluwer Academic Publishers, Dordrecht, The Netherlands, 670 pp., 1988.

Sulkava, M., Luyssaert, S., Zaehle, S., and Papale, D.: Assessing and improving the representativeness of monitoring networks: The European flux tower network example, J. Geophys. Res., 116, G00J04, doi:10.1029/2010jg001562, 2011.

Taylor, J. R., and Loescher, H. L.: Automated quality control methods for sensor data: A novel observatory approach, Biogeosciences, 10, 4957-4971, doi:10.5194/bg-10-4957-2013, 2013.

Tetzlaff, A., Lüpkes, C., and Hartmann, J.: Aircraft-based observations of atmospheric boundary-layer modification over Arctic leads, Q. J. R. Meteorolog. Soc., 141, 2839-2856, doi:10.1002/qj.2568, 2015.

Turner, D. P., Ollinger, S. V., and Kimball, J. S.: Integrating remote sensing and ecosystem process models for landscape- to regional-scale analysis of the carbon cycle, BioScience, 54, 573-584, doi:10.1641/0006-3568(2004)054[0573:irsaep]2.0.co;2, 2004.

Tuzson, B., Hiller, R. V., Zeyer, K., Eugster, W., Neftel, A., Ammann, C., and Emmenegger, L.: Field intercomparison of two optical analyzers for CH_4 eddy covariance flux measurements, Atmos. Meas. Tech., 3, 1519-1531, doi:10.5194/amt-3-1519-2010, 2010.

Vaughan, A. R., Lee, J., Misztal, P., Metzger, S., Shaw, M. D., Lewis, A. C., Purvis, R., Carslaw, D., Goldstein, A., Hewitt, C. N., Davison, B., Beevers, S. D., and Karl, T.: Spatially resolved flux measurements of NO_x from London suggest significantly higher emissions than predicted by inventories, Faraday Discuss., doi:10.1039/c5fd00170f, 2015.

Vickers, D., and Mahrt, L.: Quality control and flux sampling problems for tower and aircraft data, J. Atmos. Oceanic Technol., 14, 512-526, doi:10.1175/1520-0426(1997)014<0512:QCAFSP>2.0.CO;2, 1997.

Webb, E. K., Pearman, G. I., and Leuning, R.: Correction of flux measurements for density effects due to heat and water vapour transfer, Q. J. R. Meteorolog. Soc., 106, 85-100, doi:10.1002/qj.49710644707, 1980.

Wilczak, J. M., Oncley, S. P., and Stage, S. A.: Sonic anemometer tilt correction algorithms, Boundary Layer Meteorol., 99, 127-150, doi:10.1023/A:1018966204465, 2001.

Wurster, L. F., Colville, R. J., and Duggan, J.: Market Trends: DevOps - not a market, but a tool-centric philosophy that supports a continuous delivery value chain, Gartner, Inc., G00274555, Stamford, U.S.A., 14 pp., 2015.

Xu, K., Metzger, S., and Desai, A. R.: Upscaling tower-observed turbulent exchange at fine spatio-temporal resolution using environmental response functions, Agric. For. Meteorol., 232, 10-22, doi:10.1016/j.agrformet.2016.07.019, 2017.