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.

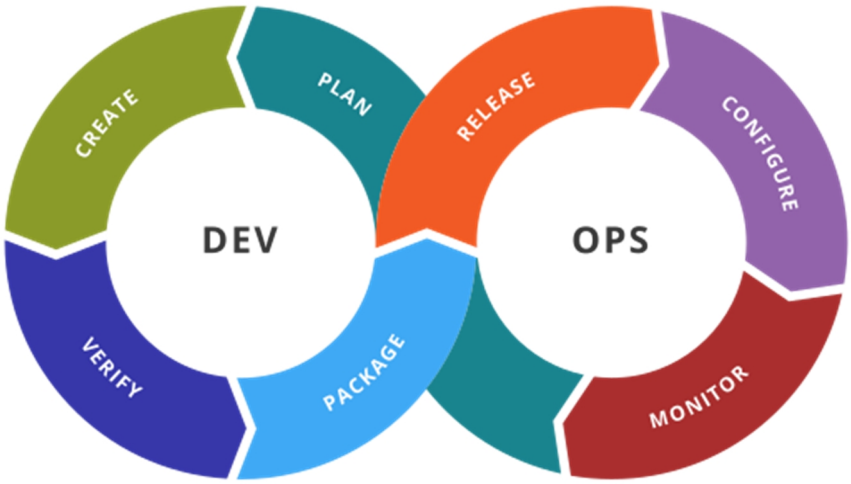
However, I do have some comments:

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.

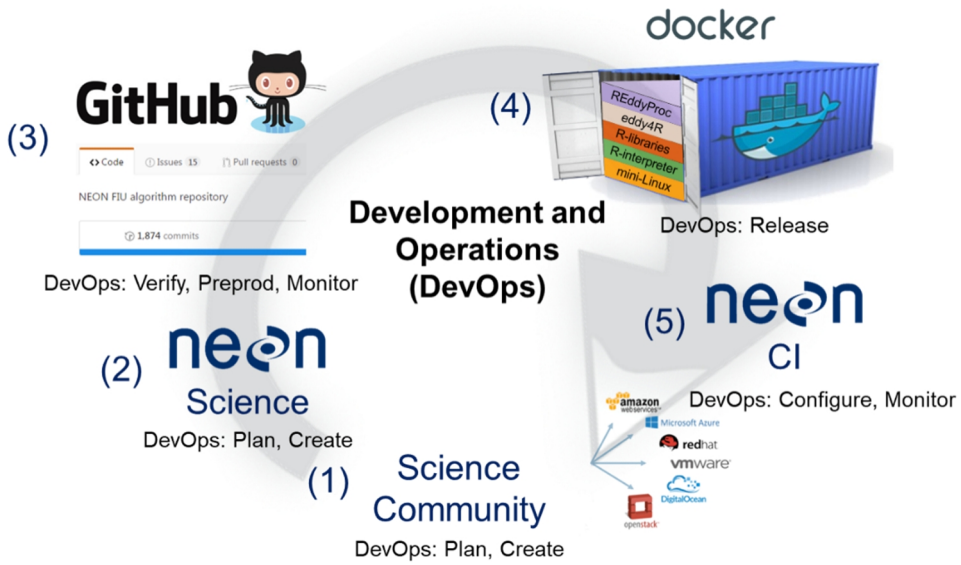


Figure 2. NEON-specific DevOps workflow. DevOps workflow steps are called out in parentheses. Please see text in Sect. 2 for detailed explanation."

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.