BIODIVERSITY BUILDING BLOCKS FOR POLICY

M7 First quality assessment report of B3 software

19/08/2024

Author(s): Peter Desmet, Lissa Breugelmans, Shawn Dove, Pieter Huybrechts, Ward Langeraert, Sandra MacFadyen, Damiano Oldoni, Maarten Trekels, Toon Van Daele, Christophe Van Neste

Funded by the European Union

This project receives funding from the European Union's Horizon Europe Research and Innovation Programme (ID No 101059592). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the EU nor the EC can be held responsible for them.

Table of contents

Summary

Software developed by B3 partners needs to be open, portable and high-quality. These intentions are expressed in the B-Cubed software development guide, as 77 requirements that B3 software needs to meet. From June 19 to July 12, 2024, we reviewed ten software tools developed by or related to B3 and assessed how well these are currently meeting the requirements. Software scores fairly good overall (76%), especially regarding required metadata. More work is required for two Python tools and in creating or collecting software tutorials for the B-Cubed guides and tutorials website (https://docs.b-cubed.eu/). This first quality assessment report also acts as a benchmark to compare against in a second quality assessment report that will be produced towards the end of the project.

1. Introduction

Software is a crucial part of the B3 project. It is instrumental to many of the project deliverables and often a project deliverable itself, intended to be used by others, for different use cases and in different computing environments. That is why the software tools developed by B3 partners need to achieve a high level of quality, openness, and portability.

To aid software maintainers in accomplishing this, B3 published the **B-Cubed software development guide** in February 2024 (Huybrechts et al. 2024, initially created as deliverable D3.1). This document specifies high-level requirements for software, selected from numerous existing best practices and guidelines. It also provides hands-on instructions and examples. B3 partners have reported this guide to be useful in developing their software. Some of these tools were created or extended in the B-Cubed hackathon (April 2-5, 2024, see deliverable D1.7).

To assess how well B3 software currently meets the requirements laid out in the guide, we peer-reviewed the currently available tools. This first quality assessment acts as a benchmark for our software and will be repeated towards the end of the project.

2. Methodology

At the B3 General Assembly meeting (May 28-29, 2024, Montpellier) we identified the reviewers and the software to review for this first quality assessment. Ten people (the authors of this document) volunteered for review. All partners could suggest software to review, as long as it met the following criteria: 1) it is maintained by a B3 partner and/or is instrumental to the B3 project and 2) its maintainer agrees with a review. Based on these criteria, we identified ten software tools for review (see Table 1).

Each reviewer was assigned two software tools to review, based on expertise and not being a contributor. Review took place from **June 19 to July 12, 2024**. The results of the review were collected in a review spreadsheet.

Table 1: Software reviewed for this milestone.

Name	Language	Type	Repository
b3gbi	R	package	https://github.com/b-cubed-eu/b3gbi
binfrastructure	Python	other	https://github.com/AgentschapPlantentuinMeise/binfrastructure
ebvcube	R	package	https://github.com/LuiseQuoss/ebvcube
fowlplay	R	analysis	https://github.com/b-cubed-eu/fowlplay
gcube	R	package	https://github.com/b-cubed-eu/gcube
indicators	R	other	https://github.com/trias-project/indicators
occurrence-cube	Java	other	https://github.com/gbif/occurrence-cube
pygbif	Python	package	https://github.com/gbif/pygbif
rgbif	R	package	https://github.com/ropensci/rgbif
trias	R	package	https://github.com/trias-project/trias

For each software, the reviewers assessed **77 requirements**, i.e. the requirements as defined in the B-Cubed Software development guide (Huybrechts et al. 2024). Each criterion is associated with a chapter in the guide, e.g. “An installable software tool MUST be maintained in its own repository.” is part of the “Code repositories” chapter. The reviewer had to assess each requirement and indicate a result (see Table 2), with an option to leave comments.

Table 2: Possible results for each requirement.

Name	Description
Complete	Software meets the requirement completely.
Not entirely	Software almost meets the requirement.
Lacking	Software does not meet the requirement.
Not applicable	The requirement does not apply for this software. For example “R code MUST be placed in the R/ directory of the repository.” does not apply for Python software.
Cannot assess	This requirement cannot be assessed for this software. This may indicate a poorly phrased requirement in the software development guide.
To assess	Requirement has not been assessed. A review is incomplete as long as there are requirements with this result.

Once reviews were completed, the first author of this document investigated the results for (obvious) discrepancies and wrong assessments, and notified reviewers if these were altered. Three scores were then calculated per software (see Table 3):

1. An **average count per result**, i.e. how often a certain result (e.g. “Complete”) was assigned to the software. This gives an indication how the reviewers assessed the

software overall. For example, if reviewer 1 assessed the requirements as “Not entirely” three times and reviewer 2 two times, then the average count for “Not entirely” was 2.5.

2. An **average score per chapter in the guide**. This gives an indication how well the software scores for each chapter.

   First, a score was assigned per result per reviewer:
   - Completely: 100% (1.0)
   - Not entirely: 50% (0.5)
   - Lacking: 0% (0.0)
   - Not applicable: NA
   - Cannot assess: empty value

   Second, the scores were averaged for the two reviewers, where a numeric score by one reviewer trumps an NA or empty by the other reviewer.

   Finally, the scores were averaged by the number of applicable criteria per chapter. For example, reviewer 1 assigns NA, NA, 1.0 to the three criteria in the chapter “Tutorials”, while reviewer 2 assigns 1.0, 0.0, 1.0. NA values are ignored, so the average score is 1.0, 0.0, 1.0. Divided by the number of criteria for this chapter, the final score is 67% (2.0/3). When both reviewers assigned NA to all criteria in a chapter, the final score is NA.

3. A **total score**. This gives an indication how well the software scores overall. The same method is used as the average score per chapter in the guide, but across chapters.

3. Results

4.1 Review process

Reviewers reported “Cannot assess” very rarely and when they did it was not shared by the other reviewer (one case notwithstanding). This indicates that the requirements in the guide are well-defined and testable. Some minor updates were made to the guide to clarify edge cases.

Reviewers also reported that the review process was informative for their own software development. Software review can thus act as a learning experience.

4.2 Overall results

Software received an **average total score of 76%**, with none below 50% or reaching 100% (see Table 3). This indicates a good baseline for software quality, with room for improvement.

Chapters that score well were The README file (94%, n = 10), R analysis code (94%, n = 1), Code repositories (87%, n = 10), R functions (85%, n = 6), and R packages (77%, n = 5). Improvements can be made in Versioning (73%, n = 9), R (72%, n = 7), Code collaboration (69%, n = 10), and Python (56%, n = 3). The Tutorials requirements (46%, n = 10) scored the

lowest, which is expected since 1) tutorials are often only written when software is reaching a mature state and 2) the tutorial website https://docs.b-cubed.eu has only been set up recently.

Table 3: Overall scores for the software reviewed for this milestone

	b3gbi	binfrastructure	ebvcube	fowlplay	gcube	indicators	occurrence-cube	pygbif	rgbif	trias	total
Complete	44	12	41.5	12.5	49.5	31.5	11.5	19.5	46.5	38.5	30.7
Not entirely	7.5	2	10.5	3	3	3	2.5	4.5	9.5	12	5.75
Lacking	6	10.5	8	5.5	2	8.5	3.5	3.5	3	7	5.75
Not applicable	17.5	52.5	17	56	21	33	58.5	48	17.5	19	34
Cannot assess	2	0	0	0	1.5	1	1	1.5	0.5	0.5	0.8
Total	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**	**77**
Code repositories	100%	86%	71%	79%	100%	83%	96%	79%	86%	86%	87%
The README file	100%	83%	100%	92%	100%	100%	83%	92%	100%	92%	94%
Code collaboration	100%	38%	69%	69%	63%	67%	50%	56%	81%	100%	69%
Versioning	100%	0%	88%	NA	100%	0%	100%	100%	100%	67%	73%
R	83%	NA	88%	0%	95%	69%	NA	NA	82%	88%	72%
R functions	84%	NA	87%	NA	100%	81%	NA	NA	89%	68%	85%
R packages	68%	NA	63%	NA	96%	NA	NA	NA	86%	71%	77%
R analysis code	NA	NA	NA	NA	NA	94%	NA	NA	NA	NA	94%
Python	NA	38%	NA	45%	NA	NA	NA	85%	NA	NA	56%
Tutorials	67%	0%	67%	67%	67%	67%	25%	25%	75%	0%	46%
Total	**83%**	**52%**	**78%**	**66%**	**94%**	**73%**	**72%**	**75%**	**86%**	**77%**	**76%**

4.3 Results per software

See Table 3 for the scores per software. Software maintainers and contributors are welcome to consult the review spreadsheet, read the B-Cubed software development guide (Huybrechts et al. 2024) and/or contact the authors to see how their software can be improved.

4.4 Next steps

We plan a “Second quality assessment report of B3 software” (M8), by February 2026. This will include all software reviewed for this milestone (enabling score improvements/declines) and any additional software that is within scope.

4. Acknowledgements

We thank the software maintainers (and contributors) for developing and maintaining tools that are instrumental to the B3 project. It is not always easy to have your software scrutinized by others, so we thank you to be open for review: Matt Blissett (occurrence-cube), Shawn Dove (b3gbi), GBIF helpdesk (pygbif), Quentin Groom (fowlplay), Ward Langeraert (gcube), Damiano Oldoni (indicators, trias), Luise Quoß (ebvcube), Christophe Van Neste (binfrastructure), and John Waller (rgbif).

5. References

1. Huybrechts P, Trekels M, Abraham L, Desmet P (2024). B-Cubed software development guide. https://docs.b-cubed.eu/dev-guide/