Manuscript Title

This manuscript (permalink) was automatically generated from ferenckata/SQSeminarPaper@f42fcd0 on September 20, 2023.

Authors

- **Katalin Ferenc** ✉
  0000-0002-3006-4297 · ferenckata
  Centre for Molecular Medicine Norway, University of Oslo · Funded by Grant XXXXXXXX

- **Ieva Rauluseviciute**
  0000-0001-9253-8825 · ievarau
  Centre for Molecular Medicine Norway, University of Oslo

- **Ladislav Hovan**
  XXXX-XXXX-XXXX-XXXX · ladislav-hovan
  Centre for Molecular Medicine Norway, University of Oslo

- **Vipin Kumar**
  XXXX-XXXX-XXXX-XXXX · princeps091-binf
  Centre for Molecular Medicine Norway, University of Oslo

✉ — Correspondence possible via GitHub Issues or email to Katalin Ferenc <k.t.ferenc@ncmm.uio.no>.

Abstract

Introduction

This should probably be about bioinformatics and its software problems. Industry is the place with experience -> learn from them just because of that, not glorifying their ideas just acknowledging that they thought more

Bioinformatics and computational biology are continuously gaining importance in biological research. These fields are heavily relying on inventions of computer science and software engineering. As Goble pointed out in 2014, about 90% of researchers used or relied on results produced by scientific software [1]. This implies that incorrect software results in invalid scientific findings.

However, bioinformaticians lack formal education in computer science and related subjects [2]. Such lack of theoretical and practical foundations hinders the adoption of good practices (eg. continuous integration, unit tests, code reviews, planning of software architecture) established in other software-heavy endeavors. Many of these practices rely on redundancy of knowledge within team members, supported by practices such as pair programming, daily stand-up meetings and code reviews.

One main feature of academic research groups is the research projects conducted by a single PhD student or post-doc. In fact, academia prides itself on enabling individual achievements, and generally measures the success of individual effort while disregarding team effort. This often results in very few people maintaining the software product, which might be used by thousands of researchers worldwide [3,4]. Lack of funding contributes to the poor maintenance status of much of scientific software [1] , which has been recognized and addressed by the Chan Zuckerberg Initiative Essential Open Source Software for Science fund [5].

The concept of a team is therefore different in a science project from a software project. While group members help each other with scientific suggestions, most often there is a single responsible person for the design and implementation of the code base. As official guidelines are rarely available on coding practices, the actual craft of software engineering is treated as secondary and up to individual judgment. We can observe (can we?) that these guidelines naturally emerge in larger groups (Seurat?), this requires a form of team organization not intrinsic to academic groups. In line with this, Hannay et al. [2] found that researchers tended to rank software engineering concepts higher if they worked in a team. Russel et al. [4] showed that high-profile code bases feature larger development teams than low-profile ones, and their activities are also associated with longer commit messages as a sign of better communication and documentation of changes. We asked ourselves whether a form of team structure organized around individual software products could improve the quality of our work.

In this work we review the findings of software engineer researchers on bioinformatics practitioners, and their suggestions for improving the overall quality of scientific software in bioinformatics and related fields (citations?). Inspired by the findings, especially related to process quality, our research group incorporated weekly discussions on code quality. We noticed that good practice requires investment in time and effort that may not pay off on an individual basis. Sharing the onboarding effort of new tools, and normalizing discussions on implementation details resulted in an overall better perceived process quality and code quality of the members.

We implemented a form of code review that fits our specific needs and context. We used our experience from these sessions, with special focus on team-based software development practices to ensure good code quality during in the update of the curation analysis software of the 2024 release of the JASPAR database (citation). In this project, as well as in some of our own individual research projects, we introduced user stories when documenting the assumptions, current features and new ideas, and we relied on development tools such as Jira and git. We note that the usage of these tools is not necessarily aligned with industry practices, due to the experimental nature of scientific software.

Nevertheless, as bioinformatics becomes a more and more software-heavy field, we believe a good direction is to collectively lower the barrier to adapting to new technologies.

We aim to provide guidelines on how to get started with improving the process quality of bioinformatics groups, even without members who have formal training in computer science or software engineering.

Review of existing suggestions for improving software quality in biomedical sciences

We reviewed the existing literature that focuses on providing guidelines for programming practices for scientists without extensive training in computational sciences. We used several key phrases to search for papers: “guidelines for bioinformatics software”, “rules for biologists learning bioinformatics”. Such papers are quite abundant and have a number of things in common. For example, they can focus on specific suggestions, often referred to as rules or “tips & tricks”, or they can more broadly direct towards good practices of coding, which are put together into “guidelines”.

Papers often choose a main focus or a theme and then distinguish a set of rules or guidelines. That focus can be very specialized, such as particular data analysis in a single disease [6]. Or the theme can be more general, for example setting the rules for next-generation sequencing (NGS) data analysis or outlining tips on how to start on computational analysis of the experimental data [7,8,9]. Both types of papers emphasize the need to learn how to analyze data properly and provide good suggestions to do that, based on the chosen topic. However, one needs to be aware when reading specialized focus guidelines that some of them can be much less applicable if the context of the analysis is different. Finally, the reference point matters. There is a difference, if the guidelines are read by a person with less experience in computational data analysis, where even a small set of rules can mean a steep learning curve, because some skills that are dependencies for the rules might be common knowledge for those with more experience. For example, a guideline to use version control systems, such as git, is a very important one, but for that a reader should most likely know the basics of the unix command line, which is not necessarily a rule on itself. One of the first things that should be done while suggesting your set of guidelines is clearly defining your target audience, even if they are meant to be broader suggestions.

As most guidelines tend to be written for researchers or students with minimal coding experience, suggestions often overlap. Most commonly highlighted are documentation and version control [10,11]. These are basic aspects of the software quality, which are still often overlooked in the publications making it hard to reproduce the research or use a published software [12]. Even the people with the least experience in computational analyses should always document their code by writing extensive README documentation. Documentation practice is very strong in the wet lab, where having a lab journal is unquestionable. This should not be different when working with computational tools. Version control of the code improves research reproducibility and the usage of the software. The most popular way to do that is by using git and remote repositories, for example, GitHub or Bitbucket [6,11,13]. Less commonly emphasized are testing of the developed code or software and optimization of pipelines [11]. These are very important rules to follow to end up with a proper collection of code or a stable software [12]. However, it might be considered to be too advanced for “beginners” guidelines. Unfortunately, not being aware of such requirements can lead to a code that either takes a long time to run or is buggy, which accumulates the technical debt and, in a way, encourages the bad practices. To avoid that it is important to at least be aware of the caveats of your code and document them for fellow researchers that might use your code in the future.

As code testing and origanization can be overwhelming for a less experienced computational scientist, there are multiple available tools to help organize the coding tasks and the code itself. For example,

using task management through Jira or GitHub issues, which are commonly used in team projects, where multiple people work on the same code. However, these resources are not emphasized in the guideline's literature as this literature is most commonly focused on the individual persons and their personal practices. Often "other people" are only mentioned when guidelines suggest where to seek help when encountering a problem with the code. This includes consulting with colleagues, finding a mentor or participating in online communities (for example, Stack Overflow or Biostars) [11]. More recently, new tools are introduced that integrate artificial intelligence (AI) as a helper in coding. Tools such as GitHub copilot or editor extensions with access to ChatGPT allow to ask the models to write a piece of code to address a problem. To our knowledge bioinformatics literature almost never presents suggestions how to code in a team setting and utilize multiple people's expertise on software development. In contrary to sofware engineering-oriented literature, where there is a lot of focus on coding in a team practices [14,15]. Hagan et al. described Code Clubs - the practice in their research lab, where group members are collectively engaged in software development through code reviews and pair coding and software engineering education through workshops or seminars [16]. The authors give tips on how to organize such meetings and what should be the ground rules. Sharing your coding experience with others helps minimize the isolation, allows individuals to learn from their peers, and finally helps to write a better quality software.

References

1. **Better Software, Better Research**
Carole Goble
IEEE Internet Computing (2014-09) https://doi.org/vjz
DOI: 10.1109/mic.2014.88

2. **How do scientists develop and use scientific software?**
Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, Greg Wilson
2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (2009-05) https://doi.org/bw966x
DOI: 10.1109/secse.2009.5069155

3. **Empirical study on software and process quality in bioinformatics tools**
Katalin Ferenc, Konrad Otto, Francisco Gomes de Oliveira Neto, Marcela Dávila López, Jennifer Horkoff, Alexander Schliep
bioRxiv (2022-03-13) https://doi.org/grx4jr
DOI: https://doi.org/10.1101/2022.03.10.483804

4. **A large-scale analysis of bioinformatics code on GitHub**
Pamela H Russell, Rachel L Johnson, Shreyas Ananthan, Benjamin Harnke, Nichole E Carlson
PLOS ONE (2018-10-31) https://doi.org/gskr8b
DOI: 10.1371/journal.pone.0205898 · PMID: 30379882 · PMCID: PMC6209220

5. **CZI – Essential Open Source Software for Science**
Chan Zuckerberg Initiative
https://chanzuckerberg.com/eoss/

6. **Guidelines for bioinformatics of single-cell sequencing data analysis in Alzheimer's disease: review, recommendation, implementation and application**
Minghui Wang, Won-min Song, Chen Ming, Qian Wang, Xianxiao Zhou, Peng Xu, Azra Krek, Yonejung Yoon, Lap Ho, Miranda E Orr, … Bin Zhang
Molecular Neurodegeneration (2022-03-02) https://doi.org/gptgqt
DOI: 10.1186/s13024-022-00517-z · PMID: 35236372 · PMCID: PMC8889402

7. **A Clinician's Guide to Bioinformatics for Next-Generation Sequencing**
Nicholas Bradley Larson, Ann L Oberg, Alex A Adjei, Liguo Wang
Journal of Thoracic Oncology (2023-02) https://doi.org/gsm3mb
DOI: 10.1016/j.jtho.2022.11.006 · PMID: 36379355 · PMCID: PMC9870988

8. **Practice guidelines for development and validation of software, with particular focus on bioinformatics pipelines for processing NGS data in clinical diagnostic laboratories**
Nicola Whiffin, Kim Brugger, Joo Wook Ahn
PeerJ (2017-05-29) https://doi.org/gsm3mc
DOI: 10.7287/peerj.preprints.2996

9. **Standards and Guidelines for Validating Next-Generation Sequencing Bioinformatics Pipelines**
Somak Roy, Christopher Coldren, Arivarasan Karunamurthy, Nefize S Kip, Eric W Klee, Stephen E Lincoln, Annette Leon, Mrudula Pullambhatla, Robyn L Temple-Smolkin, Karl V Voelkerding, … Alexis B Carter
The Journal of Molecular Diagnostics (2018-01) https://doi.org/gcsstd
DOI: 10.1016/j.jmoldx.2017.11.003

10. **Top considerations for creating bioinformatics software documentation**
Mehran Karimzadeh, Michael M Hoffman
Briefings in Bioinformatics (2017-01-14) https://doi.org/bzmp
DOI: 10.1093/bib/bbw134 · PMID: 28088754 · PMCID: PMC6054259

11. **Ten simple rules for getting started with command-line bioinformatics**
Parice A Brandies, Carolyn J Hogg
PLOS Computational Biology (2021-02-18) https://doi.org/gh32h2
DOI: 10.1371/journal.pcbi.1008645 · PMID: 33600404 · PMCID: PMC7891784

12. **Improving bioinformatics software quality through incorporation of software engineering practices**
Adeeb Noor
PeerJ Computer Science (2022-01-05) https://doi.org/gsm3hg
DOI: 10.7717/peerj-cs.839 · PMID: 35111923 · PMCID: PMC8771759

13. **Practice guidelines for development and validation of software, with particular focus on bioinformatics pipelines for processing NGS data in clinical diagnostic laboratories**
Nicola Whiffin, Kim Brugger, Joo Wook Ahn
PeerJ (2017-05-29) https://doi.org/gsm3md
DOI: 10.7287/peerj.preprints.2996v1

14. https://faculty.washington.edu/ajko/books/cooperative-software-development/

15. **Studying the impact of social interactions on software quality**
Nicolas Bettenburg, Ahmed E Hassan
Empirical Software Engineering (2012-04-28) https://doi.org/f4mhdp
DOI: 10.1007/s10664-012-9205-0

16. **Ten simple rules to increase computational skills among biologists with Code Clubs**
Ada K Hagan, Nicholas A Lesniak, Marcy J Balunas, Lucas Bishop, William L Close, Matthew D Doherty, Amanda G Elmore, Kaitlin J Flynn, Geoffrey D Hannigan, Charlie C Koumpouras, … Patrick D Schloss
PLOS Computational Biology (2020-08-27) https://doi.org/gg92xw
DOI: 10.1371/journal.pcbi.1008119 · PMID: 32853198 · PMCID: PMC7451508

17. **Sci-Hub provides access to nearly all scholarly literature**
Daniel S Himmelstein, Ariel Rodriguez Romero, Jacob G Levernier, Thomas Anthony Munro, Stephen Reid McLaughlin, Bastian Greshake Tzovaras, Casey S Greene
eLife (2018-03-01) https://doi.org/ckcj
DOI: 10.7554/elife.32822 · PMID: 29424689 · PMCID: PMC5832410

18. **Reproducibility of computational workflows is automated using continuous analysis**
Brett K Beaulieu-Jones, Casey S Greene
Nature biotechnology (2017-04) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6103790/
DOI: 10.1038/nbt.3780 · PMID: 28288103 · PMCID: PMC6103790

19. **Plan S: Accelerating the transition to full and immediate Open Access to scientific publications**
cOAlition S
(2018-09-04) https://www.wikidata.org/wiki/Q56458321

20. **Open access**
Peter Suber
MIT Press (2012)
ISBN: 9780262517638

21. **Open collaborative writing with Manubot**
Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
Manubot (2020-05-25) https://greenelab.github.io/meta-review/

22. **Opportunities and obstacles for deep learning in biology and medicine**
Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, … Casey S Greene
Journal of The Royal Society Interface (2018-04) https://doi.org/gddkhn
DOI: 10.1098/rsif.2017.0387 · PMID: 29618526 · PMCID: PMC5938574

23. **Open collaborative writing with Manubot**
Daniel S Himmelstein, Vincent Rubinetti, David R Slochower, Dongbo Hu, Venkat S Malladi, Casey S Greene, Anthony Gitter
PLOS Computational Biology (2019-06-24) https://doi.org/c7np
DOI: 10.1371/journal.pcbi.1007128 · PMID: 31233491 · PMCID: PMC6611653

This manuscript is a template (aka "rootstock") for Manubot, a tool for writing scholarly manuscripts. Use this template as a starting point for your manuscript.

The rest of this document is a full list of formatting elements/features supported by Manubot. Compare the input (`.md` files in the `/content` directory) to the output you see below.

Basic formatting

Bold text

Semi-bold text

Centered text

Right-aligned text

Italic text

Combined *italics and* ***bold***

~~Strikethrough~~

1. Ordered list item
2. Ordered list item
    a. Sub-item
    b. Sub-item
        i. Sub-sub-item
3. Ordered list item
    a. Sub-item

- List item
- List item
- List item

subscript: H_2O is a liquid

superscript: 2^{10} is 1024.

unicode superscripts^{0123456789}

unicode subscripts$_{0123456789}$

A long paragraph of text. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Putting each sentence on its own line has numerous benefits with regard to editing and version control.

Line break without starting a new paragraph by putting
two spaces at end of line.

Document organization

Document section headings:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

A heading centered on its own printed page

Horizontal rule:

`Heading 1`'s are recommended to be reserved for the title of the manuscript.

`Heading 2`'s are recommended for broad sections such as *Abstract*, *Methods*, *Conclusion*, etc.

`Heading 3`'s and `Heading 4`'s are recommended for sub-sections.

Links

Bare URL link: https://manubot.org

Long link with lots of words and stuff and junk and bleep and blah and stuff and other stuff and more stuff yeah

Link with text

Link with hover text

Link by reference

Citations

Citation by DOI [17].

Citation by PubMed Central ID [18].

Citation by PubMed ID [**pubmed:30718888?**].

Citation by Wikidata ID [19].

Citation by ISBN [20].

Citation by URL [21].

Citation by alias [22].

Multiple citations can be put inside the same set of brackets [17,20,22]. Manubot plugins provide easier, more convenient visualization of and navigation between citations [18,22,23,**pubmed:30718888?**].

Citation tags (i.e. aliases) can be defined in their own paragraphs using Markdown's reference link syntax:

Referencing figures, tables, equations

Figure 1

Figure 2

Figure 3

Figure 4

Table 1

Equation 1

Equation 2

Quotes and code

> Quoted text

> Quoted block of text
>
> Two roads diverged in a wood, and I—
> I took the one less traveled by,
> And that has made all the difference.

Code `in the middle` of normal text, aka `inline code`.

Code block with Python syntax highlighting:

```
from manubot.cite.doi import expand_short_doi

def test_expand_short_doi():
    doi = expand_short_doi("10/c3bp")
    # a string too long to fit within page:
    assert doi == "10.25313/2524-2695-2018-3-vliyanie-enhansera-copia-i-
        insulyatora-gypsy-na-sintez-ernk-modifikatsii-hromatina-i-
        svyazyvanie-insulyatornyh-belkov-vtransfetsirovannyh-geneticheskih-
        konstruktsiyah"
```

Code block with no syntax highlighting:

```
Exporting HTML manuscript
Exporting DOCX manuscript
Exporting PDF manuscript
```

Figures

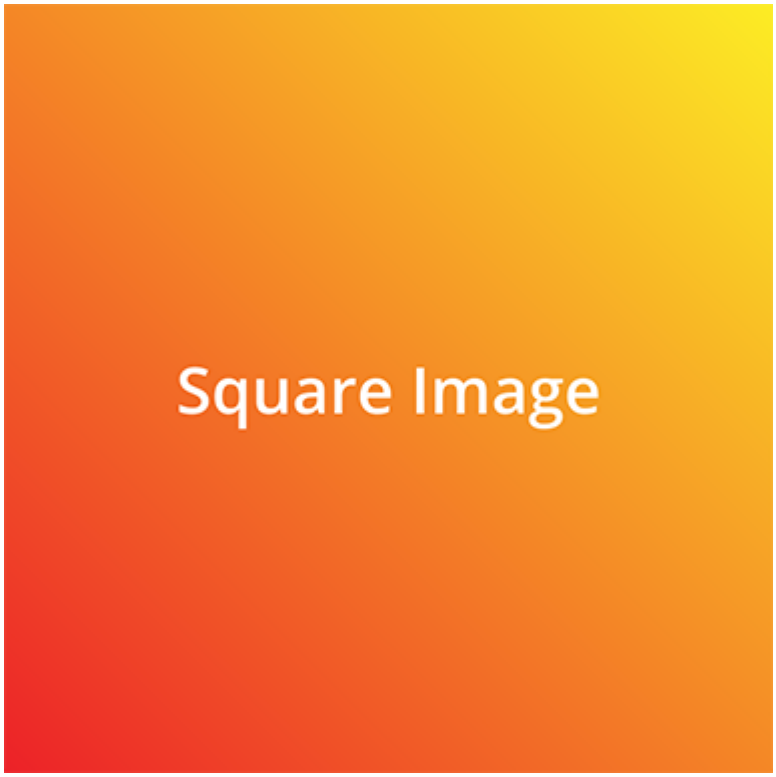

Figure 1: A square image at actual size and with a bottom caption. Loaded from the latest version of image on GitHub.

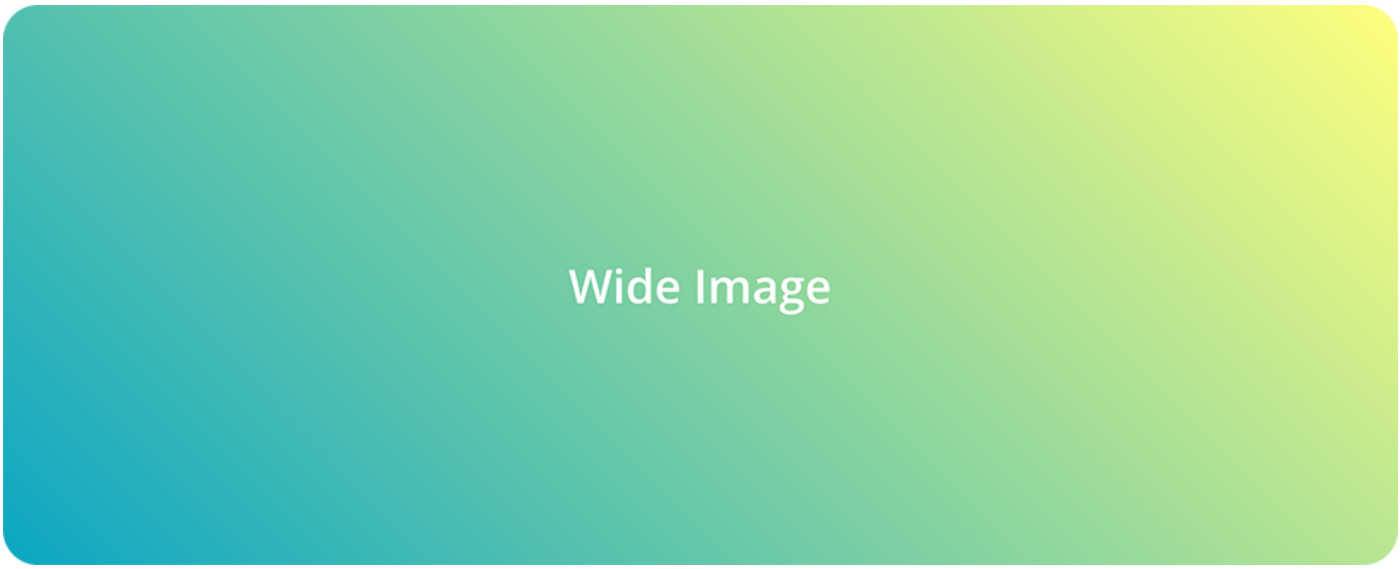

Figure 2: An image too wide to fit within page at full size. Loaded from a specific (hashed) version of the image on GitHub.

Figure 3: A tall image with a specified height. Loaded from a specific (hashed) version of the image on GitHub.

Figure 4: A vector `.svg` image loaded from GitHub. The parameter `sanitize=true` is necessary to properly load SVGs hosted via GitHub URLs. White background specified to serve as a backdrop for transparent sections of the image. Note that if you want to export to Word (`.docx`), you need to download the image and reference it locally (e.g. `content/images/vector.svg`) instead of using a URL.

Tables

Table 1: A table with a top caption and specified relative column widths.

Bowling Scores	**Jane**	**John**	**Alice**	**Bob**
Game 1	150	187	210	105
Game 2	98	202	197	102
Game 3	123	180	238	134

Table 2: A table too wide to fit within page.

	Digits 1-33	**Digits 34-66**	**Digits 67-99**	**Ref.**
pi	3.14159265358979323 846264338327950	28841971693993751 0582097494459230	78164062862089986 2803482534211706	`piday.org`
e	2.71828182845904523 536028747135266	24977572470936999 5957496696762772	40766303535475945 7138217852516642	`nasa.gov`

Table 3: A table with merged cells using the `attributes` plugin.

	Colors	
Size	**Text Color**	**Background Color**
big	blue	orange
small	black	white

Equations

A LaTeX equation:

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2} \tag{1}$$

An equation too long to fit within page:

$$x = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t + u + v + w + x + y + z + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 \tag{2}$$

Special

⚠ **WARNING** *The following features are only supported and intended for* `.html` *and* `.pdf` *exports. Journals are not likely to support them, and they may not display correctly when converted to other formats such as* `.docx`*.*

LINK STYLED AS A BUTTON

Adding arbitrary HTML attributes to an element using Pandoc's attribute syntax:

> Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot Manubot. Manubot.

Adding arbitrary HTML attributes to an element with the Manubot `attributes` plugin (more flexible than Pandoc's method in terms of which elements you can add attributes to):

> Manubot Manubot Manubot Manubot Manubot. Manubot Manubot Manubot Manubot. Manubot Manubot Manubot. Manubot Manubot. Manubot.

Available background colors for text, images, code, banners, etc:

`white` `lightgrey` `grey` `darkgrey` `black` `lightred` `lightyellow` `lightgreen` `lightblue` `lightpurple` `red` `orange` `yellow` `green` `blue` `purple`

Using the Font Awesome icon set:

✔ ? ★ 🔔 ⊗ •••

Light Grey Banner
useful for *general information* - manubot.org

Blue Banner
useful for *important information* - manubot.org

Light Red Banner
useful for *warnings* - manubot.org