

# Manuscript Title

This manuscript ([permalink](#)) was automatically generated from [computer-aided-biotech/better-cb@5ed987b](#) on November 26, 2020.

## Authors

---

- **Benjamín J. Sánchez**

 [0000-0001-6093-4110](#) ·  [BenjaSanchez](#) ·  [BenjaSanchez](#)

Department of Bioengineering, Technical University of Denmark, Kgs. Lyngby, 2800, Denmark

- **Daniela C. Soto**

 [0000-0002-6292-655X](#) ·  [dcsoto](#)

Genome Center, MIND Institute, and Department of Biochemistry & Molecular Medicine, Davis, CA 95616, USA

# Abstract

---

This should be the abstract.

## Introduction

---

Since Margaret Dayhoff pioneered the field of bioinformatics back in the sixties, the application of computational tools to the field of biology has vastly grown in scope and impact. Nowadays, biotechnological and biomedical research are routinely fed by the insights arising from novel computational approaches, machine learning algorithms and mathematical models. The ever increasing amount of biological data and the exponential growth in computing power will amplify this trend in the years to come.

The use of computers to address biological matters encompasses a wide array of applications usually grouped under the terms of “computational biology” and “bioinformatics”. Although distinct definitions have been delineated for each one [\[1,2\]](#), here we will consider both under the umbrella term “computational biology”, alluding to any application that involves the intersection of computing and biological data. As such, a computational biologist can be a data analyst, a data engineer, a statistician, a mathematical modeler, a software developer, and many others. In praxis, the modern computational biologist will be a “scientist of many hats”, taking on several of the duties listed above. But first and foremost, we will consider a computational biologist as a scientist whose ultimate goal is answering a biological question or addressing a need in the life sciences, by means of computation.

Scientific computing requires following specific practices to enable shareable, reproducible and sustainable outputs that stand the test of time. Computing-heavy disciplines such as software engineering and data science have already adopted practices addressing the need for collaboration, visualization, project management, and strengthening of online communities. However, as a highly interdisciplinary and evolving field, computational biology has yet to acquire a set of universal “best practices”. As most computational biologists come from diverse backgrounds and oftentimes lack formal computational training, the absence of guidelines can lead to disparate and unsustainable practices that hinder reproducibility and collaboration, and slow down biomedical and biotechnological research.

Over the last decade, several researchers have published advice directed to bench scientists starting in either scientific computing [\[3,4\]](#) or computational biology [\[5\]](#). The advice encompasses a wide range of topics ranging from programming to project organization to manuscript writing. Other works have adopted a different approach, fixating in one powerful tool and diving deeper into its scope and applications, such as the software development and version control cloud service GitHub [\[6\]](#) and the web-application Jupyter Notebooks [\[7\]](#). Similarly, recent works have chosen to comprehensively address one specific need in computational biology such as workflow automation [\[8\]](#) and software library development [\[9\]](#). Although this advice proves immensely helpful, several aspects of the computational biology “journey” remain uncovered. Specifically, there is a lack of a clear roadmap regarding best practices from fundamental to advanced topics, in addition to practical examples tackling the complexities of this kind of research.

We premise that best practices in computational biology lie within a continuum that traverses three “levels”: the individual’s internal practices, the collaborative practices of a team, and the practices that allow a broader scientific community to access and engage over time with the developed research. Each one of these levels has a different set of needs and challenges. Here, we compile a selected list of relevant practices and related tools advisable for each one of these levels, emphasizing their time

and place in a computational biology research project. Finally, we illustrate the utility of these practices in three case studies covering the broader spectrum of computational biology research.

## Level 1: Personal Research

---

Goal: How should you manage your computational biology project?

Topics: - Literate programming: R Markdown, Jupyter Notebooks - Version control: Git / GitHub (commits) - Software versioning and environment managers (Conda, python-env) - Modularize, snippets - Coding style: variable naming, linter, pep8, commenting - Programming practices: paradigms (object-oriented, procedural), assertions, pair-programming

## Level 2: Collaboration

---

Goal: How to allow your collaborators to reproduce and interact with your research/software?

Topics: - Sharing notebooks: R Markdowns and Jupyter Notebooks (Binder, Google CoLab) - Sharing apps: Shiny apps, Dashboard. - GitHub (branching, pull requests) - Workflow automation: Snakemake (NextFlow, Make, Bash script) - Sharing data and metadata

## Level 3: Community

---

Goal: How to develop and maintain a computational biology project with community feedback over time?

Topics: - GitHub releases and semantic versioning - Git Flow, GitHub Issues - Continuous integration and unit tests - Dependencies per user: pip-tools - Sharing software as Python packages, Conda/Bioconda or containers (Docker, Singularity) - Include license (MIT) and DOIs - Documentation: read the docs.

## Case studies

---

- Example of computational biology project 1: RNA-seq analysis (workflow)
- Example of computational biology project 2: Genome-scale metabolic model (systems biology project)
- Example of computational biology project 3: Software development (computational tool)

## Conclusion

---

Pending

## Acknowledgments

---

Pending

# References

---

## 1. NIH working definition of bioinformatics and computational biology

Huerta Michael, Downing Gregory, Haseltine Florence, Seto Belinda, Liu Yuan  
(2000-07-17)

<https://www.kennedykrieger.org/sites/default/files/library/documents/research/center-labs-cores/bioinformatics/bioinformatics-def.pdf>

## 2. What is bioinformatics? A proposed definition and overview of the field.

NM Luscombe, D Greenbaum, M Gerstein

*Methods of information in medicine* (2001) <https://www.ncbi.nlm.nih.gov/pubmed/11552348>

PMID: [11552348](https://pubmed.ncbi.nlm.nih.gov/11552348/)

## 3. Good enough practices in scientific computing

Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, Tracy K. Teal

*PLOS Computational Biology* (2017-06-22) <https://doi.org/gbkbwp>

DOI: [10.1371/journal.pcbi.1005510](https://doi.org/10.1371/journal.pcbi.1005510) · PMID: [28640806](https://pubmed.ncbi.nlm.nih.gov/28640806/) · PMCID: [PMC5480810](https://pubmed.ncbi.nlm.nih.gov/PMC5480810/)

## 4. Software engineering for scientific big data analysis

Björn A Grüning, Samuel Lampa, Marc Vaudel, Daniel Blankenberg

*GigaScience* (2019-05) <https://doi.org/gf4f4m>

DOI: [10.1093/gigascience/giz054](https://doi.org/10.1093/gigascience/giz054) · PMID: [31121028](https://pubmed.ncbi.nlm.nih.gov/31121028/) · PMCID: [PMC6532757](https://pubmed.ncbi.nlm.nih.gov/PMC6532757/)

## 5. So you want to be a computational biologist?

Nick Loman, Mick Watson

*Nature Biotechnology* (2013-11-08) <https://doi.org/p3j>

DOI: [10.1038/nbt.2740](https://doi.org/10.1038/nbt.2740) · PMID: [24213777](https://pubmed.ncbi.nlm.nih.gov/24213777/)

## 6. Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, Tobias Ternent, Stephen J. Eglén, Daniel S. Katz, ... Juan Antonio Vizcaíno

*PLOS Computational Biology* (2016-07-14) <https://doi.org/gbrb39>

DOI: [10.1371/journal.pcbi.1004947](https://doi.org/10.1371/journal.pcbi.1004947) · PMID: [27415786](https://pubmed.ncbi.nlm.nih.gov/27415786/) · PMCID: [PMC4945047](https://pubmed.ncbi.nlm.nih.gov/PMC4945047/)

## 7. Ten Simple Rules for Reproducible Research in Jupyter Notebooks

Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H. Nguyen, Sara Brin Rosenthal, Fernando Pérez, Peter W. Rose

*arXiv* (2018-10-13) <https://arxiv.org/abs/1810.08055v1>

## 8. Streamlining Data-Intensive Biology With Workflow Systems

Taylor Reiter, Phillip T. Brooks, Luiz Irber, Shannon E. K. Joslin, Charles M. Reid, Camille Scott, C. Titus Brown, N. Tessa Pierce

*Cold Spring Harbor Laboratory* (2020-07-01) <https://doi.org/gg353v>

DOI: [10.1101/2020.06.30.178673](https://doi.org/10.1101/2020.06.30.178673)

## 9. A Padawan Programmer's Guide to Developing Software Libraries

James T. Yurkovich, Benjamin J. Yurkovich, Andreas Dräger, Bernhard O. Palsson, Zachary A. King

*Cell Systems* (2017-11) <https://doi.org/gg8tqz>

DOI: [10.1016/j.cels.2017.08.003](https://doi.org/10.1016/j.cels.2017.08.003) · PMID: [28988801](https://pubmed.ncbi.nlm.nih.gov/28988801/)