Ten Simple Rules for Reproducible Computational Research

Geir Kjetil Sandve , Anton Nekrutenko, James Taylor, Eivind Hovig

Published: October 24, 2013 • DOI: 10.1371/journal.pcbi.1003285 • Featured in PLOS Collections

Citation: Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten Simple Rules for Reproducible Computational Research. PLoS Comput Biol 9(
e1003285. doi:10.1371/journal.pcbi.1003285

Editor: Philip E. Bourne, University of California San Diego, United States of America

Published: October 24, 2013

Funding: The authors' laboratories are supported by US National Institutes of Health grants HG005133, HG004909, and HG006620 and US Nati
Foundation grant DBI 0850103. Additional funding is provided, in part, by the Huck Institutes for the Life Sciences at Penn State, the Institute for C
at Penn State, and a grant with the Pennsylvania Department of Health using Tobacco Settlement Funds. The funders had no role in the preparat
manuscript.

Competing interests: The authors have declared that no competing interests exist.

Replication is the cornerstone of a cumulative science [1]. However, new tools and technologies, massive amounts of data, interdisciplinary approach
complexity of the questions being asked are complicating replication efforts, as are increased pressures on scientists to advance their research [2]. A
studies on independently collected data is often not feasible, there has recently been a call for reproducible research as an attainable minimum stand
the value of scientific claims [3]. This requires that papers in experimental science describe the results and provide a sufficiently clear protocol to allo
repetition and extension of analyses based on original data [4].

The importance of replication and reproducibility has recently been exemplified through studies showing that scientific papers commonly leave out ex
essential for reproduction [5], studies showing difficulties with replicating published experimental results [6], an increase in retracted papers [7], and th
number of failing clinical trials [8], [9]. This has led to discussions on how individual researchers, institutions, funding bodies, and journals can establis
increase transparency and reproducibility. In order to foster such aspects, it has been suggested that the scientific community needs to develop a "cu
reproducibility" for computational science, and to require it for published claims [3].

We want to emphasize that reproducibility is not only a moral responsibility with respect to the scientific field, but that a lack of reproducibility can also
you as an individual researcher. As an example, a good practice of reproducibility is necessary in order to allow previously developed methodology to
applied on new data, or to allow reuse of code and results for new projects. In other words, good habits of reproducibility may actually turn out to be a
longer run.

We further note that reproducibility is just as much about the habits that ensure reproducible research as the technologies that can make these proce
realistic. Each of the following ten rules captures a specific aspect of reproducibility, and discusses what is needed in terms of information handling ar
procedures. If you are taking a bare-bones approach to bioinformatics analysis, i.e., running various custom scripts from the command line, you will p
handle each rule explicitly. If you are instead performing your analyses through an integrated framework (such as GenePattern [10], Galaxy [11], LON
Taverna [13]), the system may already provide full or partial support for most of the rules. What is needed on your part is then merely the knowledge c
these existing possibilities.

In a pragmatic setting, with publication pressure and deadlines, one may face the need to make a trade-off between the ideals of reproducibility and t
research out while it is still relevant. This trade-off becomes more important when considering that a large part of the analyses being tried out never e
any results. However, frequently one will, with the wisdom of hindsight, contemplate the missed opportunity to ensure reproducibility, as it may alread
take the necessary notes from memory (or at least much more difficult than to do it while underway). We believe that the rewards of reproducibility wi
the risk of having spent valuable time developing an annotated catalog of analyses that turned out as blind alleys.

As a minimal requirement, you should at least be able to reproduce the results yourself. This would satisfy the most basic requirements of sound rese
any substantial future questioning of the research to be met with a precise explanation. Although it may sound like a very weak requirement, even this
reproducibility will often require a certain level of care in order to be met. There will for a given analysis be an exponential number of possible combin

versions, parameter values, pre-processing steps, and so on, meaning that a failure to take notes may make exact reproduction essentially impossibl

With this basic level of reproducibility in place, there is much more that can be wished for. An obvious extension is to go from a level where you can r in case of a critical situation to a level where you can practically and routinely reuse your previous work and increase your productivity. A second exte that peers have a practical possibility of reproducing your results, which can lead to increased trust in, interest for, and citations of your work [6], [14].

We here present ten simple rules for reproducibility of computational research. These rules can be at your disposal for whenever you want to make y accessible—be it for peers or for your future self.

Rule 1: For Every Result, Keep Track of How It Was Produced

Whenever a result may be of potential interest, keep track of how it was produced. When doing this, one will frequently find that getting from raw data involves many interrelated steps (single commands, scripts, programs). We refer to such a sequence of steps, whether it is automated or performed r analysis workflow. While the essential part of an analysis is often represented by only one of the steps, the full sequence of pre- and post-processing critical in order to reach the achieved result. For every involved step, you should ensure that every detail that may influence the execution of the step step is performed by a computer program, the critical details include the name and version of the program, as well as the exact parameters and input

Although manually noting the precise sequence of steps taken allows for an analysis to be reproduced, the documentation can easily get out of sync analysis was really performed in its final version. By instead specifying the full analysis workflow in a form that allows for direct execution, one can en specification matches the analysis that was (subsequently) performed, and that the analysis can be reproduced by yourself or others in an automatec executable descriptions [10] might come in the form of simple shell scripts or makefiles [15], [16] at the command line, or in the form of stored workflo management system [10], [11], [13], [17], [18].

As a minimum, you should at least record sufficient details on programs, parameters, and manual procedures to allow yourself, in a year or so, to app reproduce the results.

Rule 2: Avoid Manual Data Manipulation Steps

Whenever possible, rely on the execution of programs instead of manual procedures to modify data. Such manual procedures are not only inefficient they are also difficult to reproduce. If working at the UNIX command line, manual modification of files can usually be replaced by the use of standard or small custom scripts. If working with integrated frameworks, there will typically be a quite rich collection of components for data manipulation. As a manual tweaking of data files to attain format compatibility should be replaced by format converters that can be reenacted and included into executab Other manual operations like the use of copy and paste between documents should also be avoided. If manual operations cannot be avoided, you sh minimum note down which data files were modified or moved, and for what purpose.

Rule 3: Archive the Exact Versions of All External Programs Used

In order to exactly reproduce a given result, it may be necessary to use programs in the exact versions used originally. Also, as both input and output change between versions, a newer version of a program may not even run without modifying its inputs. Even having noted which version was used o it is not always trivial to get hold of a program in anything but the current version. Archiving the exact versions of programs actually used may thus sa at later stages. In some cases, all that is needed is to store a single executable or source code file. In other cases, a given program may again have s requirements to other installed programs/packages, or dependencies to specific operating system components. To ensure future availability, the only may then be to store a full virtual machine image of the operating system and program. As a minimum, you should note the exact names and version programs you use.

Rule 4: Version Control All Custom Scripts

Even the slightest change to a computer program can have large intended or unintended consequences. When a continually developed piece of code script) has been used to generate a certain result, only that exact state of the script may be able to produce that exact output, even given the same in parameters. As also discussed for rules 3 and 6, exact reproduction of results may in certain situations be essential. If computer code is not systemat along its evolution, backtracking to a code state that gave a certain result may be a hopeless task. This can cast doubt on previous results, as it may know if they were partly the result of a bug or otherwise unfortunate behavior.

The standard solution to track evolution of code is to use a version control system [15], such as Subversion, Git, or Mercurial. These systems are rela up and use, and may be used to systematically store the state of the code throughout development at any desired time granularity.

As a minimum, you should archive copies of your scripts from time to time, so that you keep a rough record of the various states the code has taken development.

Rule 5: Record All Intermediate Results, When Possible in Standardized Formats

In principle, as long as the full process used to produce a given result is tracked, all intermediate data can also be regenerated. In practice, having ea intermediate results may be of great value. Quickly browsing through intermediate results can reveal discrepancies toward what is assumed, and can uncover bugs or faulty interpretations that are not apparent in the final results. Secondly, it more directly reveals consequences of alternative progran choices at individual steps. Thirdly, when the full process is not readily executable, it allows parts of the process to be rerun. Fourthly, when reproduci allows any experienced inconsistencies to be tracked to the steps where the problems arise. Fifth, it allows critical examination of the full process beh without the need to have all executables operational. When possible, store such intermediate results in standardized formats. As a minimum, archive result files that are produced when running an analysis (as long as the required storage space is not prohibitive).

Rule 6: For Analyses That Include Randomness, Note Underlying Random Seeds

Many analyses and predictions include some element of randomness, meaning the same program will typically give slightly different results every tim (even when receiving identical inputs and parameters). However, given the same initial seed, all random numbers used in an analysis will be equal, th identical results every time it is run. There is a large difference between observing that a result has been reproduced exactly or only approximately. W equal results is a strong indication that a procedure has been reproduced exactly, it is often hard to conclude anything when achieving only approxima results. For analyses that involve random numbers, this means that the random seed should be recorded. This allows results to be reproduced exactl same seed to the random number generator in future runs. As a minimum, you should note which analysis steps involve randomness, so that a certai discrepancy can be anticipated when reproducing the results.

Rule 7: Always Store Raw Data behind Plots

From the time a figure is first generated to it being part of a published article, it is often modified several times. In some cases, such modifications are adjustments to improve readability, or to ensure visual consistency between figures. If raw data behind figures are stored in a systematic manner, so data for a given figure to be easily retrieved, one can simply modify the plotting procedure, instead of having to redo the whole analysis. An additional is that if one really wants to read fine values in a figure, one can consult the raw numbers. In cases where plotting involves more than a direct visualiz underlying numbers, it can be useful to store both the underlying data and the processed values that are directly visualized. An example of this is the histograms, where both the values before binning (original data) and the counts per bin (heights of visualized bars) could be stored. When plotting is command-based system like R, it is convenient to also store the code used to make the plot. One can then apply slight modifications to these comma having to specify the plot from scratch. As a minimum, one should note which data formed the basis of a given plot and how this data could be recons

Rule 8: Generate Hierarchical Analysis Output, Allowing Layers of Increasing Detail to Be Inspected

The final results that make it to an article, be it plots or tables, often represent highly summarized data. For instance, each value along a curve may ir averages from an underlying distribution. In order to validate and fully understand the main result, it is often useful to inspect the detailed values unde summaries. A common but impractical way of doing this is to incorporate various debug outputs in the source code of scripts and programs. When th allows, it is better to simply incorporate permanent output of all underlying data when a main result is generated, using a systematic naming conventic data underlying a given summarized value to be easily found. We find hypertext (i.e., html file output) to be particularly useful for this purpose. This al results to be generated along with links that can be very conveniently followed (by simply clicking) to the full data underlying each summarized value. with summarized results, you should as a minimum at least once generate, inspect, and validate the detailed values underlying the summaries.

Rule 9: Connect Textual Statements to Underlying Results

Throughout a typical research project, a range of different analyses are tried and interpretation of the results made. Although the results of analyses a corresponding textual interpretations are clearly interconnected at the conceptual level, they tend to live quite separate lives in their representations: on a data area on a server or personal computer, while interpretations live in text documents in the form of personal notes or emails to collaborators. interpretations are not generally mere shadows of the results—they often involve viewing the results in light of other theories and results. As such, the information, while at the same time having their necessary support in a given result.

If you want to reevaluate your previous interpretations, or allow peers to make their own assessment of claims you make in a scientific paper, you will a given textual statement (interpretation, claim, conclusion) to the precise results underlying the statement. Making this connection when it is needed and error-prone, as it may be hard to locate the exact result underlying and supporting the statement from a large pool of different analyses with vari

To allow efficient retrieval of details behind textual statements, we suggest that statements are connected to underlying results already from the time are initially formulated (for instance in notes or emails). Such a connection can for instance be a simple file path to detailed results, or the ID of a resu framework, included within the text itself. For an even tighter integration, there are tools available to help integrate reproducible analyses directly into documents, such as Sweave [19], the GenePattern Word add-in [4], and Galaxy Pages [20]. These solutions can also subsequently be used in conne publications, as discussed in the next rule.

As a minimum, you should provide enough details along with your textual interpretations so as to allow the exact underlying results, or at least some be tracked down in the future.

Rule 10: Provide Public Access to Scripts, Runs, and Results

Last, but not least, all input data, scripts, versions, parameters, and intermediate results should be made publicly and easily accessible. Various solut
become available to make data sharing more convenient, standardized, and accessible in particular domains, such as for gene expression data [21]–
journals allow articles to be supplemented with online material, and some journals have initiated further efforts for making data and code more integra
publications [3], [24]. As a minimum, you should submit the main data and source code as supplementary material, and be prepared to respond to an
further data or methodology details by peers.

Making reproducibility of your work by peers a realistic possibility sends a strong signal of quality, trustworthiness, and transparency. This could increa
and speed of the reviewing process on your work, the chances of your work getting published, and the chances of your work being taken further and
researchers after publication [25].

References

1. Crocker J, Cooper ML (2011) Addressing scientific fraud. Science 334: 1182. doi: 10.1126/science.1216775
View Article • PubMed/NCBI • Google Scholar

2. Jasny BR, Chin G, Chong L, Vignieri S (2011) Data replication & reproducibility. Again, and again, and again…. Introduction. Science 334: 12
10.1126/science.334.6060.1225
View Article • PubMed/NCBI • Google Scholar

3. Peng RD (2011) Reproducible research in computational science. Science 334: 1226–1227. doi: 10.1126/science.1213847
View Article • PubMed/NCBI • Google Scholar

4. Mesirov JP (2010) Computer science. Accessible reproducible research. Science 327: 415–416. doi: 10.1126/science.1179653
View Article • PubMed/NCBI • Google Scholar

5. Nekrutenko A, Taylor J (2012) Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. Nat Rev Genet 13:
10.1038/nrg3305
View Article • PubMed/NCBI • Google Scholar

6. Ioannidis JP, Allison DB, Ball CA, Coulibaly I, Cui X, et al. (2009) Repeatability of published microarray gene expression analyses. Nat Genet
10.1038/ng.295
View Article • PubMed/NCBI • Google Scholar

7. Steen RG (2011) Retractions in the scientific literature: is the incidence of research fraud increasing? J Med Ethics 37: 249–253. doi: 10.1136
View Article • PubMed/NCBI • Google Scholar

8. Prinz F, Schlange T, Asadullah K (2011) Believe it or not: how much can we rely on published data on potential drug targets? Nat Rev Drug D
10.1038/nrd3439-c1
View Article • PubMed/NCBI • Google Scholar

9. Begley CG, Ellis LM (2012) Drug development: raise standards for preclinical cancer research. Nature 483: 531–533. doi: 10.1038/483531a
View Article • PubMed/NCBI • Google Scholar

10. Reich M, Liefeld T, Gould J, Lerner J, Tamayo P, et al. (2006) GenePattern 2.0. Nat Genet 38: 500–501. doi: 10.1038/ng0506-500
View Article • PubMed/NCBI • Google Scholar

11. Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, et al. (2005) Galaxy: a platform for interactive large-scale genome analysis. Genor
1451–1455. doi: 10.1101/gr.4086505
View Article • PubMed/NCBI • Google Scholar

12. Rex DE, Ma JQ, Toga AW (2003) The LONI Pipeline Processing Environment. Neuroimage 19: 1033–1048. doi: 10.1016/s1053-8119(03)001
View Article • PubMed/NCBI • Google Scholar

13. Oinn T, Addis M, Ferris J, Marvin D, Senger M, et al. (2004) Taverna: a tool for the composition and enactment of bioinformatics workflows. Bi
3045–3054. doi: 10.1093/bioinformatics/bth361
View Article • PubMed/NCBI • Google Scholar

14. Piwowar HA, Day RS, Fridsma DB (2007) Sharing detailed research data is associated with increased citation rate. PLoS ONE 2: e308
doi:10.1371/journal.pone.0000308.
View Article • PubMed/NCBI • Google Scholar

15. Heroux MA, Willenbring JM (2009) Barely sufficient software engineering: 10 practices to improve your cse software. In: 2009 ICSE Worksho
Engineering for Computational Science and Engineering. pp. 15–21.

16. Schwab M, Karrenbach M, Claerbout J (2000) Making scientific computations reproducible. Comput Sci Eng 2: 61–67. doi: 10.1109/5992.881
View Article • PubMed/NCBI • Google Scholar

17. Goble CA, Bhagat J, Aleksejevs S, Cruickshank D, Michaelides D, et al. (2010) myExperiment: a repository and social network for the sharinç
workflows. Nucleic Acids Res 38: W677–682. doi: 10.1093/nar/gkq429
View Article • PubMed/NCBI • Google Scholar

18. Deelman E, Singh G, Su M-H, Blythe J, Gil Y, et al. (2005) Pegasus: a framework for mapping complex scientific workflows onto distributed sy
Programming Journal 13: 219–237.
View Article • PubMed/NCBI • Google Scholar

19. Leisch F (2002) Sweave: dynamic generation of statistical reports using literate data analysis. In: Härdle W, Rönz B, editors. Compstat: proce
computational statistics. Heidelberg, Germany: Physika Verlag. pp. 575–580.

20. Goecks J, Nekrutenko A, Taylor J (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent compu
in the life sciences. Genome Biol 11: R86. doi: 10.1186/gb-2010-11-8-r86
View Article • PubMed/NCBI • Google Scholar

21. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, et al. (2001) Minimum information about a microarray experiment (MIAME)-
for microarray data. Nat Genet 29: 365–371.
View Article • PubMed/NCBI • Google Scholar

22. Brazma A, Parkinson H, Sarkans U, Shojatalab M, Vilo J, et al. (2003) ArrayExpress–a public repository for microarray gene expression data
Nucleic Acids Res 31: 68–71. doi: 10.1093/nar/gkg091
View Article • PubMed/NCBI • Google Scholar

23. Edgar R, Domrachev M, Lash AE (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. Nucleic
207–210. doi: 10.1093/nar/30.1.207
View Article • PubMed/NCBI • Google Scholar

24. Sneddon TP, Li P, Edmunds SC (2012) GigaDB: announcing the GigaScience database. Gigascience 1: 11. doi: 10.1186/2047-217x-1-11
View Article • PubMed/NCBI • Google Scholar

25. Prlić A, Procter JB (2012) Ten simple rules for the open development of scientific software. PLoS Comput Biol 8: e1002802 doi:10.1371/journ
View Article • PubMed/NCBI • Google Scholar