

The computer code of computational research is vital paradata, the academic paper optional metadata

Richèl J.C. Bilderbeek¹

¹Department of Immunology, Genetics and Pathology, Science for Life Laboratory, Uppsala University, Uppsala, Sweden

May 31, 2022

Contents

9	1 Introduction	4
10	2 Example study	5
11	3 Code is more important than the (English) paper	8
12	4 There are multiple way to publish code	12
13	4.1 Publishing raw code	14
14	4.2 Publishing hosted code	15
15	4.3 Publishing a virtual environment	20
16	5 Conclusions	21

17	6 Discussion	22
18	7 Data Accessibility	24
19	A Supplementary materials	29
20	A.1 Funding	29

Abstract

22 Here we define paradata as the data that describes the generation of
 23 data. In genetic epidemiology, the data generated is mostly the results of
 24 an analysis (e.g. predicting a person having a disease), as done by com-
 25 puter code. In such context, paradata is usually the scientific paper that
 26 describes what the computer code does. However, this has the unrealistic
 27 assumption that there is a perfect match between the paper and the code.
 28 In this chapter it is argued that the source code should be supplied,
 29 as this is the true paradata: if the paper and code disagree, it is the code that
 30 has generated the results. The chapter concludes by some rules how to bet-
 31 ter code to serve as paradata, and hence allowing computational research
 32 to be truly reproducible.

33 **Keywords:** paradata, reproducible research, code, software, FAIR data,
 34 computational research, Open Science, best practices, genetic epidemiology

35 **Definitions**

Term	Definition	Example
Code	Body of text to be run by a computer	The scripts run by a computational experiment
Data	Individual facts, statistics, or items of information	A SNP that has a significant association
Genotype	The DNA allele at a certain location	AA, AC, CG, GT, ...
Paradata	Data that describes the generation of data	The code to conclude that a SNP has a significant association
Phenotype	How an organism looks like in the broadest sense	The concentration of IL6RA in the blood
Metadata	Data that provides information about other data	The article that describes an experiment
Trait	A phenotype	The concentration of IL6RA in the blood

³⁶ 1 Introduction

³⁷ The currency that universities generate is knowledge, which in turn is dis-
³⁸ tributed in mostly lectures and academic papers. Academic papers contain
³⁹ knowledge, either as the report of measurements or the outcome of an experi-
⁴⁰ ment. These reports are described in English, the current lingua franca of sci-
⁴¹ ence.

⁴² A paper contains and/or describes data, where we define data as 'indi-
⁴³ vidual facts, statistics, or items of information, often numeric' [1]. To ensure
⁴⁴ unhampered knowledge growth, this data should [TODO: Why? Reference!] follow the FAIR principles. The FAIR principles [2] are that data should be
⁴⁵ [TODO: there are probably better examples] Findable (e.g. found by web), Ac-
⁴⁶ cessible (e.g. DataDryad and or Zenoda), Interoperable (e.g. comma-separated
⁴⁷ files) and Reusable (e.g. work on all operating systems)

⁴⁹ A paper in itself is data, as it is a collection of text and figures that we itself
⁵⁰ can do measurements upon. Also here, a paper should be FAIR, [TODO: there
⁵¹ are probably better examples] i.e. Findable (e.g. Google Scholar), Accessible
⁵² (e.g. view it online), Interoperable (e.g. written in English) and Reusable (e.g.
⁵³ it is valid to cite and build upon earlier academic papers). This paper itself
⁵⁴ will later specify the standpoint that a paper is metadata (i.e. data about data),
⁵⁵ instead of paradata (i.e. data that describes the collection of data)

⁵⁶ This paper focusses on research fields that uses computation to do exper-
⁵⁷ iments and uses genetic epidemiology as an example. Genetic epidemiology
⁵⁸ is a field within biology that, among others, measures the spread (hence 'epi-
⁵⁹ demiology') of heritable (i.e. genetic) traits, as well as the relation between
⁶⁰ having a certain genetic makeup and a certain trait, where the trait can be any
⁶¹ human property, such as weight, height, the amount of metabolites and having
⁶² a disease yes/no.

63 **2 Example study**



Figure 1: Picture of Karesuando's church, the village where the Northern Swedish Population Health Study started. From [3]

64 As an example, we'll use a pseudorandomly selected paper from [4]. The
65 data used by that paper is from a population study called the Northern Swedish
66 Population Health Study (NSPHS) that started in 2010 [5]. The approximately
67 1000 participants were initially mostly surveyed about lifestyle [5] and follow-
68 up studies provided the type of data relevant for this paper, which are (1) the
69 genetic data [6], (2) the concentration of certain proteins in the blood [7, 8].

70 The first type of data, the (final form of the) genetic data, consists out of sin-
71 gle nucleotide polymorphisms (SNPs). SNPs consist out of a name, a position
72 and a nucleotide. DNA (organized into chromosomes and present in every (nu-
73 cleated) cell, see figure 2) consists out of many nucleotides, of which there are
74 four types, called adenine, cytosine, guanine and thymine, all commonly
75 abbreviated as A, C, G and T respectively. One SNP example is rs12133641,
76 which is a SNP located at position 154,428,283, where some people have a cer-
77 tain nucleotide. In this case, 67 percent of the people within this study have an
78 A, and 33 percent have a G (also from [4], Table S3).

79 The second type of data are concentrations of certain proteins in the blood.

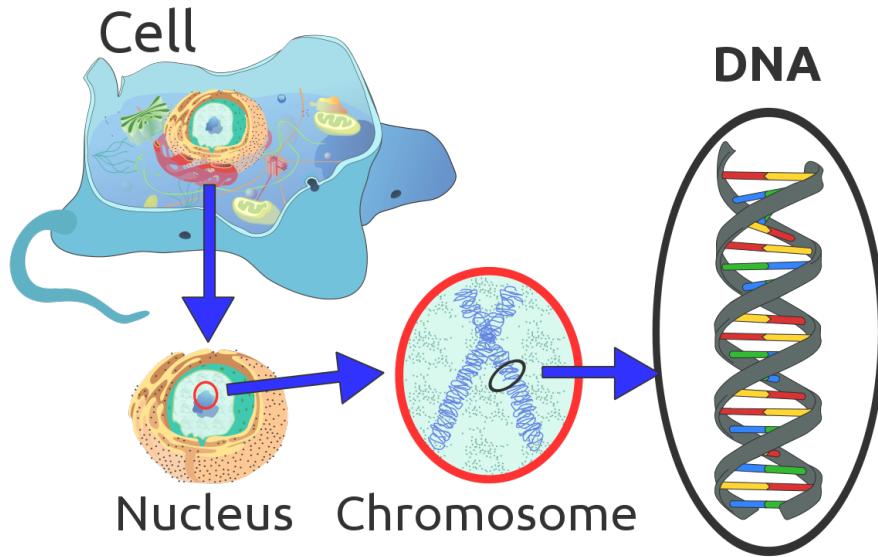


Figure 2: A cell has a nucleus that contains chromosomes. Each of these chromosomes (46 in humans) consist out of DNA. DNA itself consists out of 4 nucleotides, as depicted by the horizontal sticks with the colors red, yellow, green and blue. From [9]

80 DNA contains the code for building proteins (see Figure 3), as well as the rate
 81 at which a protein is created. Some proteins end up in the blood and their
 82 presence can be used to assess the health of an individual. IL6RA is a protein
 83 [TODO: more info]
 84 The field of genetic epidemiology looks -among others- for correlations be-
 85 tween genetic data and biological traits. For example, figure 4 (from [4]) shows
 86 that SNP rs12133641 is highly correlated (p value is $3.0 * 10^{-73}$, 961 individ-
 87 uals) with protein IL6RA. What this results does not yet teach us, is how this
 88 correlation works, yet figure 5 shows us the direction of the association: the X
 89 axis shows the possible genetic make-ups (aka the genotype) of the individuals,
 90 where 0 denotes the individuals with genotype AA (the individual inherited
 91 one adenosine from his/her mother and one adenosine from his/her father),
 92 1.0 denotes AG (one A is inherited from one parent, where the G is inherited

6
 then

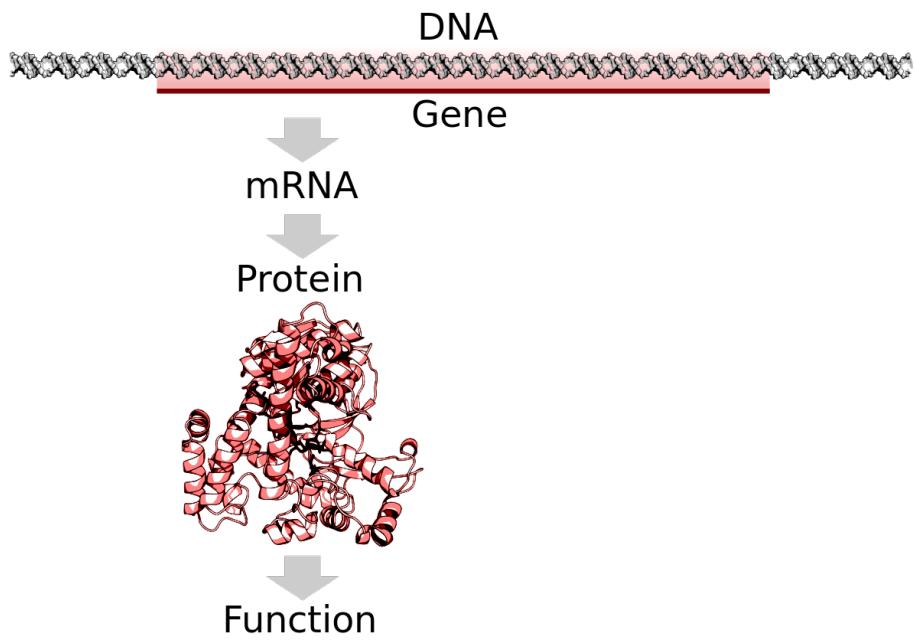


Figure 3: Parts of DNA (so-called 'genes') code for proteins. The DNA, that always stays put in the cell's nucleus, is transcribed to messenger RNA (mRNA). mRNA leaves the nucleus and its code gets translated to a protein sequence. Near the start of a gene are regions that determine the amount of proteins produced (not shown in figure). Adapted from [10]

93 from the other parent) and 2.0 denotes GG. From Figure 5 we can conclude
94 that, on average, the more guanines are inherited at that SNPs location, the
95 higher concentration of IL6RA can be found in a human's blood.

96 The amount of variance that can be explained by an association (i.e. the
97 R squared value) is rarely 100 percent, which means that a trait (in this case,
98 IL6RA concentrations) cannot be perfectly explained from the genotype alone.
99 As we can see in figure 5, 43 percent of the variance can be attributed to an
100 individuals' genotype and additional factors, such as the effect of the environ-
101 ment (e.g. geographic location, time of day), lifestyle (e.g. smoking yes/no)
102 or having a disease (e.g. diabetes) are needed to explain the variation between
103 individuals better.

Biomarker	Chr	SNP name	SNP position	P (GWAS) N = 961
IL6RA	1	rs12133641	154428283	3.0×10^{-73}

Figure 4: An example result of a genetic epidemiological research. It shows that the SNP named rs12133641 (located at position 154,428,283 of chromosome 1) is highly correlated (p value is 3.0×10^{-73} , 961 individuals) to the concentration of the protein IL6RA, as measured in blood. The table is a simplified result from [4].

104 3 Code is more important than the (English) paper

105 The experiment described above is run by code. There is no fieldwork, nor lab-
106 work involved. It is the researcher ~~that~~ ^{who} writes the code, after which the code
107 does the work. A genetic epidemiologist ~~that~~ ^{who} looks for associations between
108 genotype and phenotype, ~~one~~ does not need a lab.

109 To obtain the genetic data, yes, there may be fieldwork and/or lab-work
110 involved. For example, a researcher needs to go into the field to take samples
111 (e.g. blood) of humans or other species. To analyse these samples, a researcher

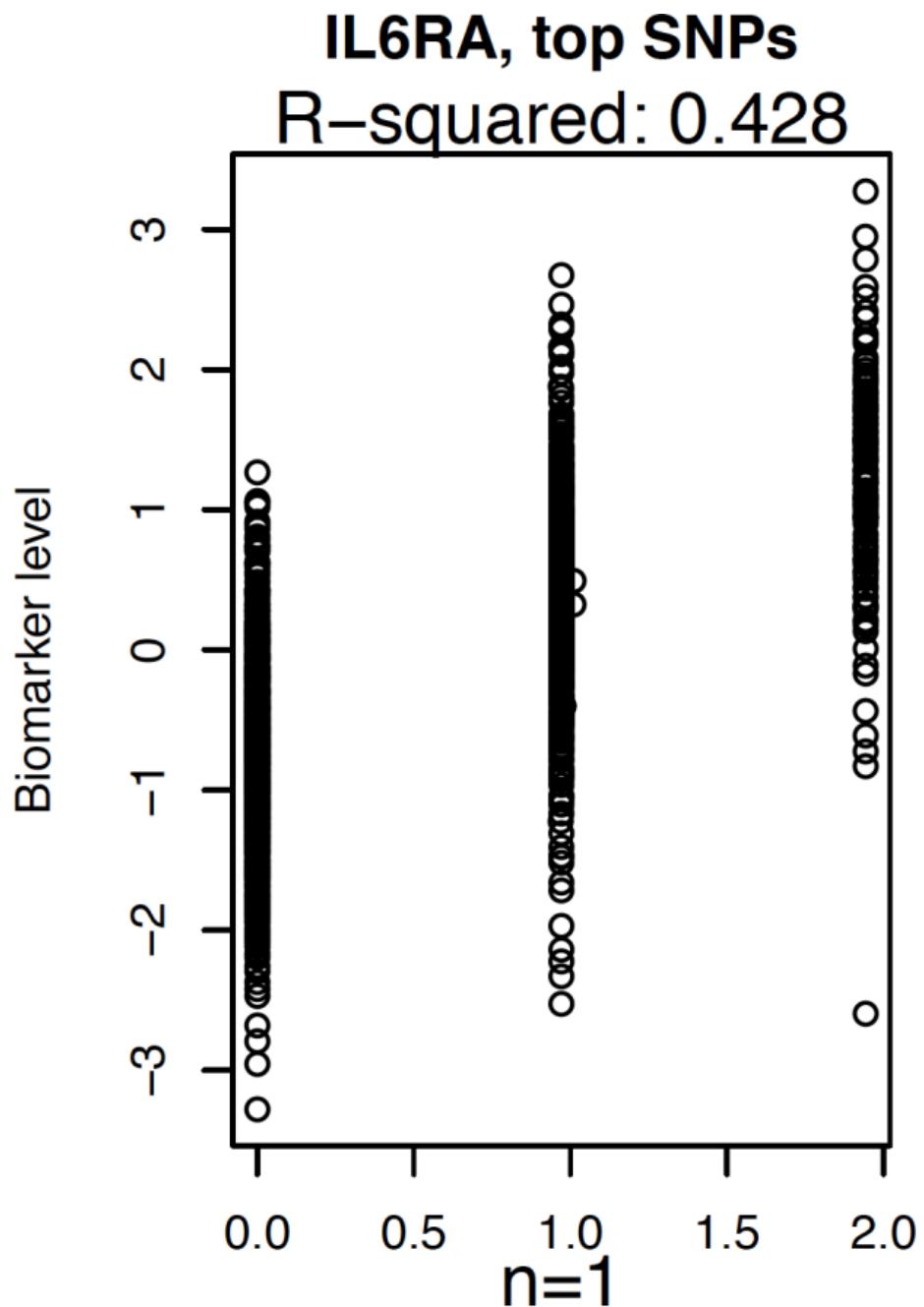


Figure 5: The relation between the genotype for SNP rs12133641 and the protein concentration of IL6RA is relatively strong. The X axis shows the the genotype of the individuals, where 0 denotes AA, 1.0 denotes AG and 2.0 denotes GG. The Y axis shows the concentration of the protein IL6RA as found in the participants' blood. n = 1 denotes the number of SNPs that were determined to be involved.

112 needs a lab. Depending on the technique being used on how to analyse the
113 samples, there may be a big bioinformatics step to aggregate the measurements
114 into useful genetic data. Although this paper does not focus on the code run
115 to aggregate the raw data into useful genetic data, the same arguments can be
116 made for that code as in the example: that code is the ground truth of how the
117 raw data is collected into useful genetic data.

118 Programming code is paradata, as it is data that describes how data is col-
119 lected, which is the definition of paradata [TODO: REF]. Programming code
120 is data that is usually in the form of text, spread over one or more files, that
121 describes the experiment. The experiment collects the data. The data collected
122 is the data we call the results of an experiment.

123 The article is metadata, as it is data that describes other data, which is the
124 definition of metadata [TODO: REF]: an article describes the experiment and
125 hence the programming code in English. However, it is not the best candidate
126 to describe how the data is collected, as it has a loose connection with collecting
127 the data. Again, if the code and article of an experiment disagree, it is the code
128 that actually ~~let~~ the data be collected. Instead, an article is metadata about ~~the~~ the
129 research.

130 A scientific claim must be reproducible before it is accepted by the scientific
131 community. Reproducibility (i.e. to reproduce the same results) and replicabil-
132 ity (i.e. to reconclude a conclusion) are fundamental characteristics of scientific
133 studies ([11]). For computational science, it is relatively easy to reproduce an
134 experiment, as all it takes is a computer, electricity, an optional internet connec-
135 tion, the code and the data. In computational science, however, it appears that
136 culture to reproduce results has been lost and to counter this, it has been sug-
137 gested to make reproducible research a minimal requirement for publication
138 ([12]).

*(oh: a research
project)*

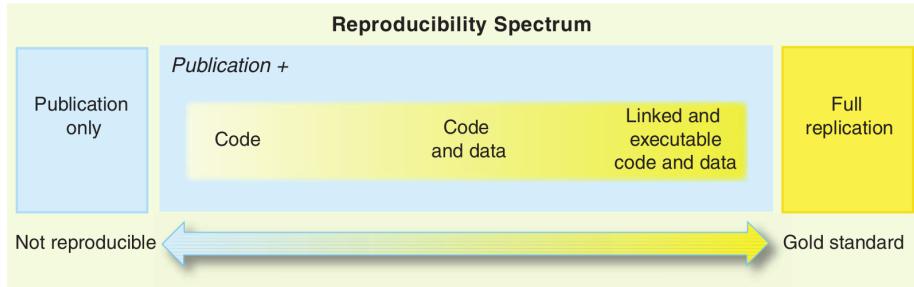


Figure 6: Levels of reproducibility, from [12]

139 The code of a computational experiment must be published for an experi-
 140 ment to reproducible. It is however (too) common that the original data and/or
 141 code are unavailable ([13]). One sad example is described in [14], where an
 142 algorithm that detects breast cancer better than a human expert, was irrepro-
 143 ducible. In the case of genetic epidemiology, it is a given that the original data
 144 cannot be published as this data is sensitive. The code of a computational ex-
 145 periment, however, can be distributed without such problems. It is, however,
 146 not code that is published, but an English description of what that code does
 147 instead. From a knowledge management's perspective, to conserve an experi-
 148 ment, it is the code that must be archived and not the English description of
 149 what it does: it is the code does the actual work. Hence, from a knowledge
 150 management perspective, emphasis should be put more on the preservation of
 151 code, as it is the most important actor in an experiment.

152 Code should be published, as it holds the ground truth of an experiment; it
 153 does the actual work. The more complex the computation pipeline is, the easier
 154 it is to have a mismatch between the article (that describes what the code does)
 155 and the code (that actually does the work). It is easy (tempting?) to overlook
 156 how easy it is to have a mismatch between paper and code.

157 To illustrate how easy it is to get a mismatch between a paper and the code,
 158 consider this fictional example of a text in a paper:

159 We compared the values of `x` and `y` using a one-tailed Student's T-test,
160 as we expect the average value of `x` to be less than the average value of `y`.

161 We ignore the choices of words and style of this sentence: what is important
162 is the content: a one-tailed T test is performed. Taking a look at the (in this case,
163 the programming language R) code, we find the following line:

164 `t.test(x, y)`

165 The line of code is simplified (as the result of the T test is never stored), but
166 the code is correct: `t.test` is indeed the name of an R function to do a T-test
167 and it is reasonable to assume that this code, when found in the code of an
168 article, is assumed to be correct.

169 Here, however, we see a mismatch between the English description and
170 the code: by default, the R function `t.test` does a **two-tailed** T-test. A conse-
171 quence of this fictional example is that the published p-values are higher than
172 needed, resulting in less significant findings, which results in less conclusions
173 drawn from a paper.

174 It is the paper that accompanies the code, as it is the code that generates
175 the results. When humans are fallible and code gets bigger, the likelihood of a
176 mismatch between the English paper and the code increases. We know that the
177 most common errors are simple (and may legitimately fear that those simple
178 errors are common) ([15]), and this example is not too far off from the examples
179 given in that paper.

180 But regardless of the size of the code, it is the code that is the ground truth.

181 **4 There are multiple way to publish code**

182 There are multiple ways of publishing code, that are increasingly hard to do,
183 yet result in more reproducible experiments: (1) publishing the code as text

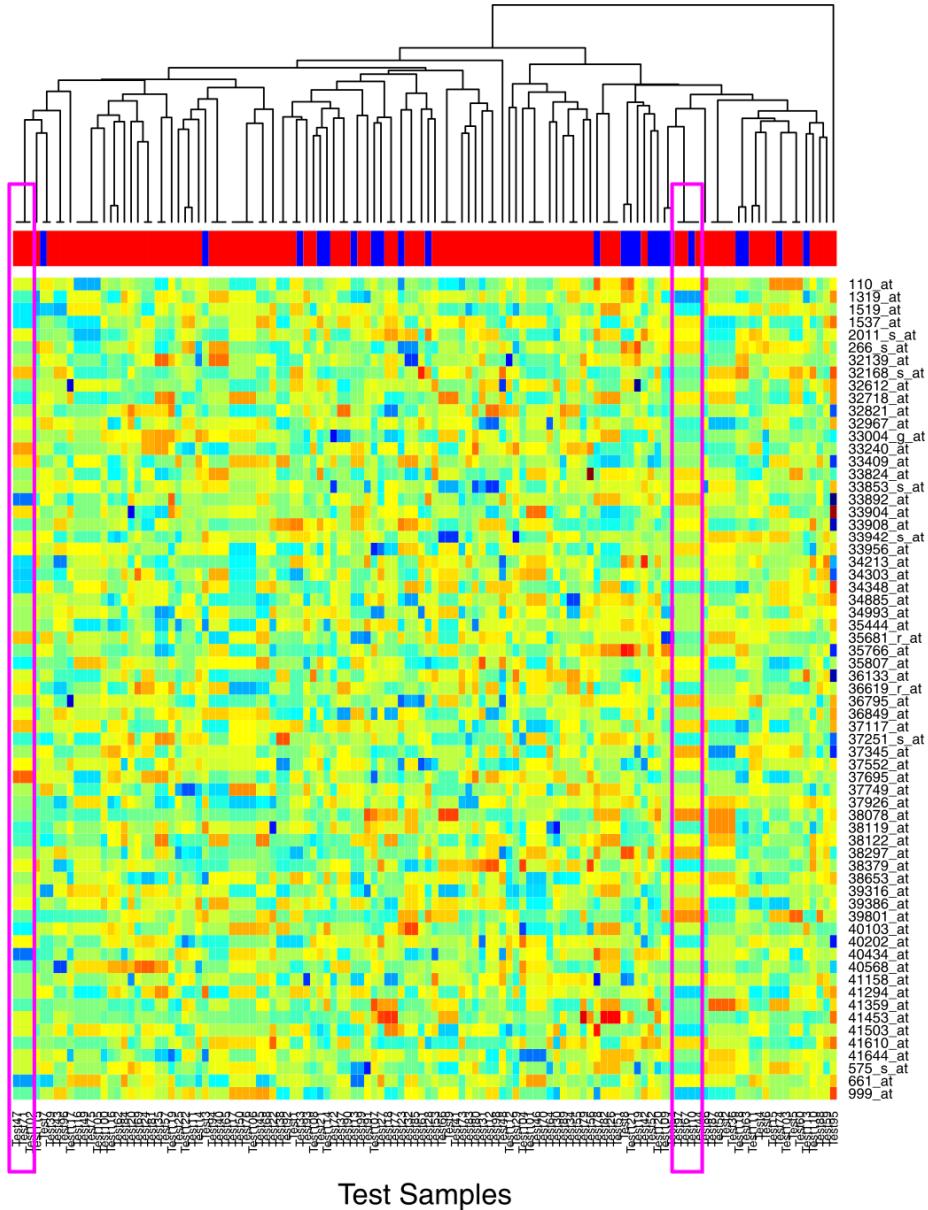


Figure 7: Gene expression result due to an off-by-one error. The cells in the main table show how much mRNA is produced per cell line (the columns) for different proteins (the rows). The colors above the cells, with the dendrogram, shows red for cells that do not respond (red) and do respond (blue) to an antibiotic. The purple rectangles show samples that are duplicated. Note that, due to this, some samples that behave identically are both non-responders and responders. From [15]

184 only, e.g. by printing it in a Supplementary Materials, (2) publishing the code
185 on a hosted website, e.g on GitHub, with added features (3) publishing the
186 installed code within a virtual environment, for example, using a Singularity
187 container. Here I will go into details of each of these ways.

188 4.1 Publishing raw code

189 Code is commonly published alongside of a paper, pasted as text in the Supple-
190 mentary Materials (for example, in [16]). This is the rawest form of publishing
191 code, as it is just text.

192 The raw code of a computational experiment is more important to preserve
193 than the article describing the experiment, even when both describe the exper-
194 iment identically, correctly and reproducibly, as it has/should been that code
195 that generated the results of an experiment. Again: if the paper and code differ,
196 it is the code that did the actual work with the actual/true results.

197 Raw code gives an indication of the quality of a computational experiment
198 and several metrics are devised to get an idea of the quality of code.

199 The simplest metric to assess the quality of code is the amount of (single)
200 lines of code (SLOC). Code that has a lower SLOC count, i.e. it is more concise,
201 is usually regarded as having a lower chance of containing bugs. Additionally,
202 following a style guide (e.g. [?]), improves software quality ([17]).

203 A more interesting metric is the cyclomatic complexity of the code. The
204 cyclomatic complexity is defined as the number of ways that the code can be
205 executed. For example, code that has only one `if` statement has a cyclomatic
206 complexity of 2, as the condition within the `if` statement can be true or false.
207 The cyclomatic complexity correlates with code complexity, where more com-
208 plex code is likelier to contain bugs.

209 A feature that deserves more attention is that code can actively teach what it

barely mention dependencies to strong Ken State next?

210 does. A mechanism to do so, which is present in most programming languages,
211 is error handling: errors are the mechanism by which a programmer can teach
212 the user what to do/expect. One example is when calculating the variance of a
213 distribution. To calculate the variance of a distribution, at least two values are
214 required. When the variance of only one value is requested, an error message
215 can helpfully indicate that one needs at least two values.

216 There are multiple problems with publishing raw code. First, there is no
217 guarantee that that code *can* actually run, for example, due to a typo added
218 after the code was pasted. Secondly, there is no way to determine if code that
219 worked yesterday *works today*. These problems can be prevented by, instead
220 of pasting raw code, using a website to host that code (see below).

221 Publishing raw code is already better than not publishing code at all, as that
222 raw code can be assumed to have run successfully at least once. To reproduce
223 an experiment from raw code can be painful, yet way better than rewriting the
224 same code from scratch.

225 **4.2 Publishing hosted code**

226 Code is sometimes published alongside a paper, in the form of a hosted web-
227 site. This way of publishing code gives additional options, both for the authors
228 of that code, as well as its users.

229 There are multiple websites for hosting code, with BitBucket, GitHub, Git-
230 Lab and SourceForge being the most well-known. Of these, GitHub is the most
231 popular website to host code in general, with millions of users. GitHub users
232 can create dedicated websites (called 'repositories') to upload code. GitHub
233 hosts that uploaded by and provides a user interface for every visitor of such
234 a website. The use of code hosting websites, such as GitHub, accommodates
235 collaboration ([18]).

236 <https://github.com/richelbilderbeek/babette>,
 237 and improves transparency ([19]). An example from the author is the web-
 238 site [20], that hosts part of the code for the paper [21], as shown in Figure 8.

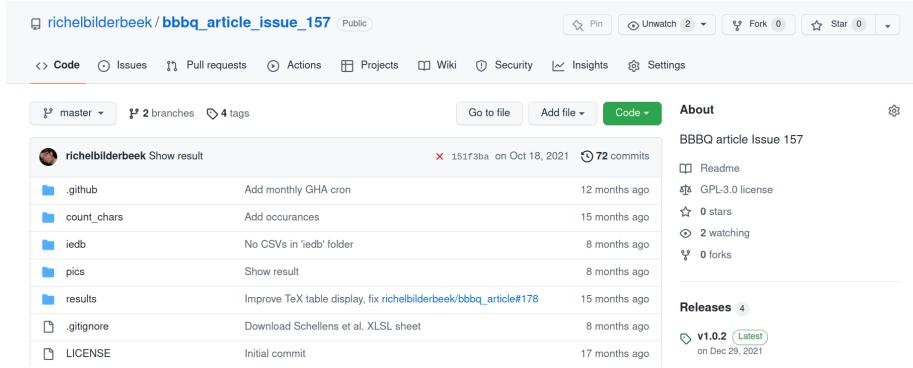


Figure 8: A typical GitHub repository ([20]), hosting the code for a computational experiment, as published with [21]. Note that this repository has 72 commits, where a commit is a change to the code.

239 Hosted code has many additional features, such as a **version-controlled history**, continuous integration scripts and automatically generated metrics.

240 *(of individual lines of code across many commits)*
 241 Hosted code commonly has a commit history. For example, GitHub uses a
 242 version control system called `git` (hence its name). In practice, this means that
 243 when a change is made to the code, a new version is created. In the case that
 244 the change was harmful, one can go back to an earlier version and continue
 245 again from there.

246 Hosted code improves transparency ([19]). The history of changes, called
 247 the commit history, helps to assess the honesty in the reporting of the code.
 248 Ideally, the full version history of the code has been generated, by hosting the
 249 code directly at its inception. However, in some repositories, only the final
 250 version of the code has been posted, which means all historical information is
 251 lost. From a code history, it can be observed, for example, how much statistical
 252 tests have really/actually been done and see if all are reported: not reporting

this only applies to hosting solutions that use control systems like git (specify)

253 a statistical tests is considered a Dubious Research Practice [RB: REF], as this
254 allows dishonest reporting of p-value hacking.

255 Some code hosts, among others, GitHub, allows the researcher to start soft-
256 ware scripts when uploading a new version. To determine the reproducibility
257 of the code, one such script should run the code (on trivial data) to verify if the
258 code works at all. If the functionality of the code is lost, the researcher can im-
259 mediately see which small change was the cause of this. A formalized version
260 of this workflow is called continuous integration (CI). Using a CI service (such
261 as GitHub Actions), is known to significantly increase the number of bugs ex-
262 posed ([22]) and increases the speed at which new features are added ([22]).
263 Using CI opens up the possibility to informally annotate features of a repos-
264 itory. For example, Figure 9 shows a build badge that indicates that a LaTeX
265 document (i.e. this paper) could be built.



Figure 9: A build badge for the GitHub repository of this article. The descrip-
tion indicates which process it is that the badge signals success (as in this case)
or failure of.

266 A more sophisticated check for the code's correctness is to run units tests
267 and determine the percentage of code that has been tested, which is called the
268 code coverage. Code coverage correlates with code quality ([23, 24]). The non-
269 profit organisation rOpenSci, which peer-reviews code, has made it a prerequi-
270 site to have a code coverage of 100%. Also here, CI opens up the possibility to
271 informally annotate the code coverage. For example, Figure 10 shows a build
272 badge that indicates that certain R package's code has a 100% code coverage.

273 Most code hosts, among others, GitHub and GitLab, automatically keep
274 track of certain metrics. One such metric is the amount of code each contributor
275 has made. In that way, authors involved in the writing of the software are



Figure 10: A build badge for the code coverage of an R package (<https://github.com/ropensci/beautier>). The umbrella signals that the code coverage is hosted by CodeCov (<https://codecov.io/>). The text indicates which process it is that the badge signals, which in this case, is that the code is fully covered by tests.

276 honestly acknowledged. Figure 11 shows the number of commits the authors
 277 of DAISIE ([25]) have made to its code.

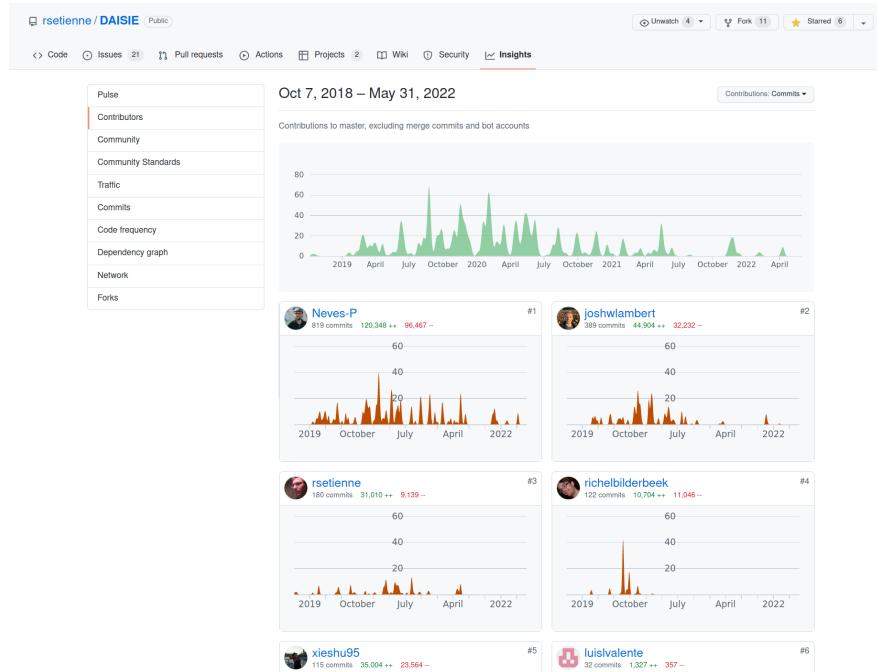


Figure 11: An overview of the commits that multiple contributors have made, in this case, to the code of DAISIE ([25])

278 A feature that gets too few attention is that code hosts allow users to post
 279 bug reports and/or ask questions. This allows other researchers to assess the
 280 health of the code.

281 Albeit hosted code is vastly superior over raw code in case of reproducib-

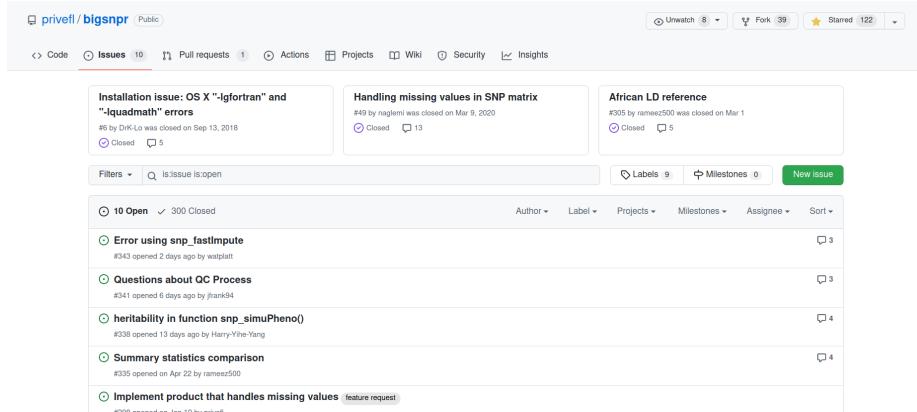


Figure 12: An overview of the issues for the code for LDpred2 ([26]). Of the 310 issues, 300 have been closed. The first four issues have been replied to, as indicated by the talk balloon. The fifth issues has a label with the text ‘feature request’ to signal the issue type.

282 lity and correctness, there are some problems with hosted code as well. The
 283 most important one is that hosted code needs maintanance: the libraries that
 284 are used by an experiment will change over time, inevitably causing the ex-
 285 periment to fail over time. This process is called bit rot and is a fundamental
 286 problem [RB: ref]. A solution that has more longevity is to publishing the code
 287 installed in a virtual environment, as described below.

288 A lesser problem, which is only a problem to some, is, that the author of the
 289 code can be contacted and that bugs can be reported. It can be argued that no
 290 energy should be wasted on published code. This is however, a rather selfish
 291 point: a bug should be reported and given proper attention, for the sake of the
 292 correctness of the results published.

293 Here are multiple suggestions to make hosted code FAIR. To make the
 294 hosted code findable, commonly a paper refers to its URL. At the homepage
 295 of hosted code, there is no standardized metadata, to, for example, link to the
 296 paper. Using a popular code host ensures the code is easily accessible. There is
 297 some interoperabiliby supplied by a version control manager, such as in obtain-

298 ing the history of the code. The reusability of code is reasonably easy to assess,
299 by using build badges. However, there is no guarantee that a build badge that
300 claimes success, actually is an honest signal (i.e. that code may not actually
301 build), nor are there standards for what ‘success’ entails.

302 **4.3 Publishing a virtual environment**

303 The most reproducible way of submitting the code of an experiment, is by pro-
304 viding the code and all it needs installed in a virtual container. This is close to
305 the golden standard suggested by [12] (see also Figure 6), however, that was
306 before virtualization existed. Virtualization allows one to provide a standard
307 environment, that can be used on any other computer. The best known ex-
308 amples of container software are Docker and Singularity, where the latter is
309 the only viable option to be able to run an experiment on a high performance
310 computing (HPC) environment, such as a computer cluster.

311 Unlike English, code is fragile in standing the test of time. Containers can
312 alleviate this, as these can be created at a point in time and then run later.
313 Because the container is stand-alone, all libraries the code depends on remain
314 in their current/old state. As with hosted code, an experiment may not run
315 with newer versions of libraries anymore, but a containerized experiment can
316 run regardless, if one refrains from updating.

317 Ideally, to re-do an experiment, one would need a one-liner to a container,
318 for example, `singularity run my_experiment.sif run`.

319 Containers allow a computation experiment to be highly reproducible: given
320 the same data, an experiment put into a container will give the same results on
321 different platforms, at least in theory. In practice, differences may be observed
322 when peripheral factors are different, such as the random numbers as gener-
323 ated by an operating system, or data that is downloaded from additional online

324 sources.

325 Containers allow a computation experiment to be highly re-usable, as any
326 scientist can work with it and tweak code.*He*

327 There are some problems with providing a container with a runnable con-
328 tainer. First, a container is several gigabytes and need to be stored somewhere
329 online. Ideally, this would be at Syslabs, as that one is free, yet there is a limit
330 of 15 Gb. Also, Syslabs does not allow much metadata to be entered, such as the
331 URL of the article, nor the URL of hosted code. Also, journals do not request
332 hosted code, nor runnable code.

333 Here are multiple suggestions to make containers with runnable code FAIR.
334 To make such containers findable, the most common environments (Syslabs
335 and DockerHub) to upload containers must be indexed. Uploading a container
336 to those environments is, however, limited by a restricted amount of free stor-
337 age, with the possible consequence that those containers are uploaded some-
338 where where they escape proper indexing. Having containers uploaded at the
339 existing websites ensures they are accessible, interoperable and reusable. One
340 could argue however, that the scientific community need its own, similar web-
341 site to upload containers, as the metadata for these sites are limited to free-text
342 comments and tags (DockerHub) or not metadata at all (Syslabs).

343 When research truly needs to be reproducible, putting the code of an exper-
344 iment into a container is a good solution, as it can stand the test of time well.
345 Rewarding this, however, is not done yet by journals.

346 5 Conclusions

347 The code that does the work in a computational experiment should always be
348 published together with a scholarly article. For research to be reproducible, one
349 ideally has access to both the data used and the code. In some fields, such as

³⁵⁰ genetic epidemiology, the data is sensitive, hence cannot be released, yet there
³⁵¹ are methods being devised to run code on sensitive data with assured privacy
³⁵² [27, 28].

³⁵³ Code has additional useful information, similar to confidence intervals, that
³⁵⁴ allow a reader to gauge how much he/she trusts the results. The most impor-
³⁵⁵ tant way to determine the quality of code is the amount of unit tests. When
³⁵⁶ following a set of best practices, such as DevOps, TDD, Agile, writing unit
³⁵⁷ tests is an essential part in writing code. The amount of unit tests is an honest
³⁵⁸ signal for code correctness (i.e. it does what it is supposed to do, as opposed
³⁵⁹ to 'it does something'). In academia, to uncover the truth, code correctness is
³⁶⁰ essential. To make a comparison with cell biologists, where working sterile is
³⁶¹ essential to have the correct results, unit tests allow a computational biologist
³⁶² to have the same.

³⁶³ Code is harder to preserve than an English text.

³⁶⁴ Although code is the primary actor in computational experiments, there is
³⁶⁵ no incentive to submit code alongside a publication. Most academic journal
³⁶⁶ do not require authors to submit their code, nor is the submitted code peer
³⁶⁷ reviewed.

³⁶⁸ Although the code of computation experiments can be archived well, there
³⁶⁹ is no incentive to do so.

³⁷⁰ Although a runnable version of a computation experiment can be archived
³⁷¹ well, there is no incentive to do so.

³⁷² 6 Discussion

³⁷³ Code may or may not be paradata, depending on how the definition is inter-
³⁷⁴ preted. Here we repeat the definition of 'paradata' and discuss its (numbered)
³⁷⁵ constituents. This paper defined paradata as 'data (1) about the collecting (2)

376 of the data (3)', where (1) code must be seen as data, (2) downloading raw
377 data and doing calculations must be seen as collecting, and (3) the results of
378 an experiment must be seen as data. This paper argues, that (1) code is data
379 in the form of text spread over one or more files, that has useful measurable
380 properties, (2) downloading raw datas and doing calculations, such as a T-test,
381 does describe how the bits and pieces of an end result is collected, and (3) an
382 experimental results is data, as it can be measured and used as the raw data of
383 a next experiment.

384 Publishing code may be disadvantageous for an author. For science, yes,
385 as this allows reproducible research. For an author, publishing code alongside
386 an experiment opens up the possibily to receive questions regarding that code.
387 Note, however, that not publishing code will always thwart the reproduction
388 of incorrect code, at the cost of a scientific career [15].

389 Is it worth it to publish version-controlled code? For an author, there is
390 additional training involved, and also here, publishing code alongside an ex-
391 periment opens up the possibily to receive questions regarding that code.

392 Is it worth it to archive running versions of code? For an author, there is
393 additional training involved. There is no FAIR infrastructure for Singularity
394 containers.

395 Being able to re-do an experiment is a core principle of the scientific method.
396 Publishing only a paper about a computational experiment is not enough, as
397 the results are too likely to mismatch that English description. Journal should
398 make it mandatory for authors to publish their code alongside a computational
399 experiment. Authors should follow the FAIR principles for their code as well,
400 as can be done using the infrastructure as supplied by, for example, GitHub
401 and GitLab, opening up new angles in metascience regarding computational
402 experiments. Knowledge managers should create the infrastructure for the

⁴⁰³ preservation of runnable experiments, to allow scientist to upload a Singularity
⁴⁰⁴ container, so these are as well following the FAIR principles.

Type of code	Recommendation
No code	Disallow papers that do not supply code publicly
Raw code	Don't publish raw code as text, host it on a code hosting website
Hosted code	Use a standarized badge for build status
	Use a standarized badge for code coverage
	Standardize to link between paper and code host
Runnable code	Allow to upload containers without limits
	Website to host containers must allow for annotation
	Use a standarized badge for the container to be functioning
	Standarize to link between container, paper and code repository

Table 1: Recommendations described in this paper

⁴⁰⁵ The world of science would be a more open, humble, trustworthy, truthful
⁴⁰⁶ and helpful would the code that accompanies a scientific paper be treated like
⁴⁰⁷ a first class citizen. As doing so in an exemplary way is yet to be rewarded,
⁴⁰⁸ hence it has to be the idealistic scientists to wage this battle. I feel the truth
⁴⁰⁹ and science are worth fighting for and I hope this paper helps others to join.

⁴¹⁰ 7 Data Accessibility

⁴¹¹ This article and its metadata can be found at

⁴¹² https://github.com/richelbilderbeek/chapter_paradata

⁴¹³ .

⁴¹⁴ References

⁴¹⁵ [1] OECD Glossary. The oecd glossary of statistical terms, 2003.

⁴¹⁶ [2] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg,
⁴¹⁷ Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-

- 418 Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The
419 fair guiding principles for scientific data management and stewardship.
420 *Scientific data*, 3(1):1–9, 2016.
- 421 [3] Nicolas Höpfner. Wooden church of Karesuando, Sweden,
422 2005. URL https://commons.wikimedia.org/wiki/File:Karesuando_church.jpg.
- 424 [4] Muhammad Ahsan, Weronica E Ek, Mathias Rask-Andersen, Torgny
425 Karlsson, Allan Lind-Thomsen, Stefan Enroth, Ulf Gyllensten, and Åsa
426 Johansson. The relative contribution of dna methylation and genetic vari-
427 ants on protein biomarkers for human diseases. *PLoS genetics*, 13(9):
428 e1007005, 2017.
- 429 [5] Wilmar Igl, Åsa Johansson, and Ulf Gyllensten. The northern swedish
430 population health study (nsphs)—a paradigmatic study in a rural popu-
431 lation combining community health and basic research. *Rural and remote*
432 *health*, 10(2):198–215, 2010.
- 433 [6] Åsa Johansson, Stefan Enroth, Magnus Palmblad, André M Deelder, Jonas
434 Bergquist, and Ulf Gyllensten. Identification of genetic variants influenc-
435 ing the human plasma proteome. *Proceedings of the National Academy of*
436 *Sciences*, 110(12):4673–4678, 2013.
- 437 [7] Stefan Enroth, Åsa Johansson, Sofia Bosdotter Enroth, and Ulf Gyllensten.
438 Strong effects of genetic and lifestyle factors on biomarker variation and
439 use of personalized cutoffs. *Nature communications*, 5(1):1–11, 2014.
- 440 [8] Stefan Enroth, Sofia Bosdotter Enroth, Åsa Johansson, and Ulf Gyllen-
441 sten. Effect of genetic and environmental factors on protein biomarkers
442 for common non-communicable disease and use of personally normalized
443 plasma protein profiles (pnppp). *Biomarkers*, 20(6-7):355–364, 2015.

- 444 [9] Sponk, Tryphon, Magnus Manske, User:Dietzel65, LadyofHats (Mar-
445 iana Ruiz), and Radio89. Diagram of DNA in a eukaryotic
446 cell, 2012. URL https://commons.wikimedia.org/wiki/File:Eukaryote_DNA-en.svg.
- 448 [10] Thomas Shafee. Protein coding genes are transcribed to an mRNA in-
449 termediate, then translated to a functional protein. RNA-coding genes
450 are transcribed to a functional non-coding RNA, 2005. URL https://en.wikipedia.org/wiki/File:DNA_to_protein_or_ncRNA.svg.
- 452 [11] Prasad Patil, Roger D Peng, and Jeffrey T Leek. A visual tool for defin-
453 ing reproducibility and replicability. *Nature human behaviour*, 3(7):650–652,
454 2019.
- 455 [12] Roger D Peng. Reproducible research in computational science. *Science*,
456 334(6060):1226–1227, 2011.
- 457 [13] Roger D Peng and Stephanie C Hicks. Reproducible research: A retro-
458 spective. *Annual review of public health*, 42:79–93, 2021.
- 459 [14] Benjamin Haibe-Kains, GA Adam, Ahmed Hosny, Farnoosh Kho-
460 dakarami, MS Board, Levi Waldron, Bo Wang, Chris McIntosh, Anshul
461 Kundaje, Casey Greene, et al. The importance of transparency and repro-
462 ducibility in artificial intelligence research. 2020, 2020.
- 463 [15] Keith A Baggerly and Kevin R Coombes. Deriving chemosensitivity
464 from cell lines: Forensic bioinformatics and reproducible research in high-
465 throughput biology. *The Annals of Applied Statistics*, pages 1309–1334, 2009.
- 466 [16] Jeremy A Labrecque and Sonja A Swanson. Interpretation and poten-
467 tial biases of mendelian randomization estimates with time-varying ex-
468 posures. *American journal of epidemiology*, 188(1):231–238, 2019.

- 469 [17] Xuefen Fang. Using a coding standard to improve program quality. In
470 *Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on*, pages
471 73–78. IEEE, 2001.
- 472 [18] Yasset Perez-Riverol, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian
473 Uszkoreit, Felipe Leprevost, Christian Fufezan, Tobias Ternent, Stephen J
474 Eglen, Daniel SS Katz, et al. Ten simple rules for taking advantage of git
475 and github. *bioRxiv*, page 048744, 2016.
- 476 [19] Krzysztof J Gorgolewski and Russell Poldrack. A practical guide for
477 improving transparency and reproducibility in neuroimaging research.
478 *bioRxiv*, page 039354, 2016.
- 479 [20] Richèl J. C. Bilderbeek. BBBQ article issue 157. https://github.com/richelbilderbeek/bbq_article_issue_157, 2021. [Accessed 29-May-2022].
- 480 [21] Richèl J. C. Bilderbeek, Maksim V. Baranov, Geert van den Bogaart, and
481 Frans Bianchi. Transmembrane helices are an over-presented and evo-
482 lutionarily conserved source of major histocompatibility complex class
483 i and ii epitopes. *Frontiers in Immunology*, 12, 2022. ISSN 1664-3224.
484 doi: 10.3389/fimmu.2021.763044. URL <https://www.frontiersin.org/article/10.3389/fimmu.2021.763044>.
- 485 [22] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and
486 Vladimir Filkov. Quality and productivity outcomes relating to contin-
487 uous integration in github. In *Proceedings of the 2015 10th Joint Meeting on*
488 *Foundations of Software Engineering*, pages 805–816. ACM, 2015.
- 489 [23] Joseph R. Horgan, Saul London, and Michael R Lyu. Achieving software
490 quality with testing coverage measures. *Computer*, 27(9):60–69, 1994.

- 494 [24] Fabio Del Frate, Praerit Garg, Aditya P Mathur, and Alberto Pasquini. On
495 the correlation between code coverage and software reliability. In *Software*
496 *Reliability Engineering, 1995. Proceedings., Sixth International Symposium on,*
497 pages 124–132. IEEE, 1995.
- 498 [25] Rampal Etienne, Luis Lima Valente, Albert B Phillimore, Bart Haegeman,
499 Joshua Lambert, Pedro Neves, Shu Xie, Richèl Bilderbeek, Torsten Hauffe,
500 Giovanni Laudanno, et al. Daisie: Dynamical assembly of islands by spe-
501 ciation, immigration and extinction. 2020.
- 502 [26] Florian Privé, Julyan Arbel, and Bjarni J Vilhjálmsdóttir. Ldpred2: better,
503 faster, stronger. *Bioinformatics*, 36(22-23):5424–5431, 2020.
- 504 [27] Lifang Zhang, Yan Zheng, and Raimo Kantoa. A review of homomorphic
505 encryption and its applications. In *Proceedings of the 9th EAI International*
506 *Conference on Mobile Multimedia Communications*, pages 97–106, 2016.
- 507 [28] C-A Azencott. Machine learning and genomics: precision medicine versus
508 patient privacy. *Philosophical Transactions of the Royal Society A: Mathemati-*
509 *cal, Physical and Engineering Sciences*, 376(2128):20170350, 2018.

510 A Supplementary materials

511 A.1 Funding

512 This chapter has not been supported by funding.

"This chapter was written in spite of protestations
of the author's supervisor"

j)

(of neither ...)