IOP Publishing

Computation in Science (Second Edition)
From concepts to practice
Konrad Hinsen

# Chapter 7

## Outlook: scientific knowledge in the digital age

The goal of scientific research is to discover new scientific knowledge, providing us with a better understanding of the world around us. In this final chapter, I will look at how computation and computing are changing the relation between scientists and scientific knowledge. The impact of these changes has become visible only recently, as it has taken several decades for computing to influence these very foundations of science. It is important for scientists to understand these changes, in order to take advantage of the opportunities they present and to take corrective action wherever they threaten to undermine the reliability and credibility of science.

For centuries, the nature of scientific knowledge had changed very little. It resided first and foremost in the heads of practicing scientists, in the form of factual, procedural, and conceptual knowledge. Factual knowledge consists of the kind of information you could store in tables or diagrams: the density of water, the names of the bones in the human body, the resolution of an instrument, etc. Procedural knowledge is about doing things. At the level of an individual, this can be using a microscope or finding the integral of a function. Examples at the collective level are developing a vaccine, or constructing a synchrotron. Conceptual knowledge consists of principles, classifications, theories, and other means that humans use to organize and reason about facts and actions. Conceptual knowledge relies on abstractions, as I explained in section 5.3, and its acquisition is an important part of what we call understanding.

Since the human brain has a limited capacity for remembering facts reliably, factual knowledge was the first to be recorded. Scientists have always stored detailed factual knowledge in lab notebooks and in reference works. Procedural knowledge soon followed, for example in the form of experimental protocols. Conceptual knowledge is different because we do not tend to forget concepts once we have firmly understood them. Conceptual knowledge is recorded in writing not so much as a memory aid, but for transmitting it to others, in the form of monographs, textbooks, and encyclopedias. A specific form of scientific document, the journal article, was

doi:10.1088/978-0-7503-3287-3ch7

developed for the communication of new discoveries, and usually combines all three forms of knowledge. All scientific knowledge stored in writing is called collectively the *scientific record*.

It is important to understand that the scientific record is a complement to but not a replacement for the knowledge in the heads of scientists, for two reasons. First, the scientific record preserves a trace of contributions and a database of established facts, but says little about the current state of our scientific understanding of the world. If you want to know current scientific consensus on a topic like climate change, you cannot just look it up in a library. You have to ask experts who have followed research on this topic for many years. Second, the contents of the scientific record are largely unintelligible to an untrained person. Interpreting recorded factual knowledge requires very specific conceptual knowledge. A table listing the density of water at different temperatures makes no sense to a person who does not understand the concept of a density, or the definition of temperature. Recorded scientific knowledge builds on more fundamental knowledge that the reader must already possess. General school education provides only a small part of it. All of today's scientists have received personal training from more experienced scientists to prepare them for consulting the scientific record and contributing to it.

Information technologies are currently revolutionizing all aspects of working with scientific knowledge, from its development via its distribution and preservation to its exploitation. In the following, I will briefly discuss the contribution that computation makes to this revolution, leaving aside the at least equally important changes in communication technology, which are profoundly changing how scientists collaborate in making new discoveries.

## 7.1 The scientific record goes digital

The introduction of computers and computer-controlled machines has changed the storage and retrieval of scientific knowledge in many ways. The most visible change is the transition from printed paper to digital files as the main medium. Books and printed journals are increasingly replaced by databases and Web sites. This has led to profound changes in the economics of scientific publishing. Both the cost of distribution and the cost of access to scientific knowledge have dropped dramatically, and the traditional roles of both publishers and libraries are losing importance. Digital data needs to be curated and preserved as well, but the corresponding roles remain to be defined. A major political struggle is currently going on between the well-established institutions, publishers and research organizations, while at the same time many scientists are experimenting with new technology for scientific communication. Judging from past revolutions in information technology, such as the invention of printing, we can hope to have a better infrastructure for managing scientific knowledge in the end, but in the meantime we will suffer accidental losses, such a Web sites disappearing and hyperlinks rotting.

Information collections that evolve over time mainly fall into one of two categories, which can be described by the metaphors of *streams* and *gardens* [1]. A stream is a timeline of contributions. Traditional examples are bank ledgers and

scientific journals, more recent forms are blogs and Twitter timelines. A garden is a resource that is continuously curated to remain up to date. Encyclopedias, both traditional paper editions and modern-day Wikipedia, are the best known examples. In the age of printed paper, gardens were expensive to maintain, and therefore there were only very few of them. In contrast, digital gardens in the form of a Wiki, a database, or an evolving piece of software can be maintained with cheap computing infrastructure, the main cost now being the work of the curators. Unfortunately, the social norms in academia have not yet adapted to this new economic situation. Scientists are judged by their contributions to streams, mainly in the form of journal articles, whereas participation in the curation of gardens is insufficiently appreciated. As a consequence, science is not yet profiting from digital gardens as much as it could.

A major advantage of digital data compared to printed paper is the possibility of automated processing at large scale. Data intensive fields such as bioinformatics could not even have existed before the transition to digital data. Data mining techniques have become commonplace in all domains of research, starting with everyday activities such as the use of Web search engines by scientists. The possibility to tie together information from many different sources is of course highly valuable for science, but it also puts a higher responsibility on each scientist for exercising critical judgment. Unreliable data and the occasional intentional misinformation are the most obvious issues. However, what might turn out to be the most serious problem is information stripped of its context by partially or fully automated processes. Re-using published datasets looks like an obvious gain in productivity, but transplanting them from their original scientific context to another one implies the risk of subtle mistakes, which in turn can undermine the public credibility of science.

## 7.2 Procedural knowledge turns into software

An important aspect of the digital revolution in science is that procedural knowledge —algorithms—can now be applied without human intervention. Before computers, every action, whether in computation or in doing experiments, was performed by a human. As I pointed out in section 5.3, following a complex sequence of steps requires abstractions, i.e. conceptual knowledge, and thus a minimum of understanding. Using machines, we can apply stored procedural knowledge without understanding it—in fact, we do this many times every day. We know roughly *what* our machines do, at the highest level of abstraction, but we usually do not understand *how* they do it, nor are we aware of many details that might well be relevant for a specific application. The only way to acquire a deep understanding of how computational models and methods work is to write computer programs that implement them—see section 1.2. Mere users of black-box tools written by others put themselves at risk of making serious mistakes. Statistical irreproducibility (see section 6.3) is one of the symptoms of this development.

The state of today's computing technology wraps digital scientific knowledge in another layer of opacity that could be avoided. In section 4.2.1, I explained the trade-off between performance and clarity in today's programming languages. Formulating algorithms in a way that humans can easily understand and work with requires languages that are more clarity-oriented than anything we have today. In fact, such languages should probably not be called 'programming' languages because their primary purpose would be the communication and preservation of procedural knowledge, rather than its application by a computer. On the other hand, many of today's large-scale simulations are feasible only due to efficient programs that are written in performance-oriented languages. As a consequence of this mismatch, much procedural knowledge of modern science exists only in the form of software that is efficient but unintelligible to its users. It even happens that software becomes unintelligible to its authors over the years, as incidental complexity accumulates (see section 5.5), although few authors will admit this openly. For the first time in the history of science, we have scientific knowledge that we can apply but which no scientist understands any more. A possible antidote could be a wider adoption of the principle of re-editable software (see section 6.7).

The use of programming languages as the only practical notation for scientific algorithms has been particular detrimental to computational models [2]. Models are primarily factual knowledge, stating that certain symbolic representations (equations, graphs, algorithms, etc) mimic the behavior of physical systems at some level of accuracy. What is commonly called a computational model is a model in which the symbolic representation takes the form of an algorithm. Computational scientists tend to focus on tools rather than on models, and thus on software implementing the computational aspect of a model, to the point of believing that the software *is* the model. The factual statements *about* these algorithms and their implementation, which give scientific meaning to the model, are easily neglected. Moreover, in a piece of software, the algorithms representing computational models melt together with other algorithms, such as data munging, user interfaces, or resource management, which often represent the major part of the code of any piece of scientific software. It thus becomes difficult to precisely identify a model. As a consequence, it also becomes very difficult to analyze a model or to compare competing models, even though this ought to be the focus of scientific work. Finally, computational models expressed as software can easily become victims of the complexification described in chapter 5, or get lost as a consequence of software collapse as discussed in section 6.7.

Plain factual data have also been infected by the opacity of scientific software. As I have explained in section 5.3.2, data have their its own abstractions which should be implemented as well-documented data formats based on well-designed data models. In reality, much scientific data are stored using undocumented file formats that are basically some program's internal data structures dumped to a file. The data can thus be used only using a particular program, making it as fragile as the program itself.

## 7.3 Machine learning: the fusion of factual and procedural knowledge

For a scientist of the pre-computer age, the distinction between factual and procedural knowledge was rather obvious, because the latter was associated with personal action. With the automated application of procedural knowledge by computers, the distinction becomes almost a technical detail. Consider a mathematical function such as the square root. There are well-known algorithms to compute it, but you can also make a table of the results and store it for later lookup[1]. The two approaches differ in the use of resources, but both give the desired result. As a user of mathematical software, you probably don't care how the results are obtained. However, if you want to understand the concept of a square root, the two representations provide very different and complementary perspectives. The table, or better yet a plot of its contents, shows in a direct way how the square root function behaves, whereas the algorithm illustrates its relation to other mathematical concepts.

Machine learning techniques introduce a third way of representing such input-to-output mappings. As I have explained in section 2.2.2, machine learning is based on very generic mathematical models with a large number of parameters that are fitted to a large training dataset. Once trained, a machine learning model is used exactly like an algorithm or a lookup table: they provide output when supplied suitable input. Another way to look at machine learning is as a method for partially converting factual knowledge into procedural knowledge, in much the same way as data compression techniques do.

The use of machine learning techniques in the acquisition and processing of scientific knowledge is very recent and so far best characterized as experimental. Its most optimistic proponents have already announced the end of the scientific method [3] because they believe that machine learning methods will extract from raw data everything one could possibly want to know about the world. At the other end of the spectrum, traditionally minded scientists consider machine learning as no more than sophisticated curve fitting. Both these extreme views will likely turn out to be wrong. The place that machine learning will occupy in the science of the 21st century depends on how useful its peculiar input-to-output mappings will be in reasoning about scientific questions. Current research on machine learning includes the interpretation of the parameters obtained by training, which would obviously be of interest in scientific applications. However, the mere fact that a given input-to-output mapping can be well represented by, say, a neural network of a specific architecture provides information about the system that is described by the mapping, and could possibly be exploited. In the long run, we can expect to see machine learning techniques developed specifically to create interpretable representations, in contrast to today's methods that focus on creating computational tools.

[1] I have discussed this partial equivalence between an algorithm and its result in section 2.2.4 as a way to measure the complexity of scientific models.

## 7.4 The time scales of scientific progress and computing

All knowledge has a finite lifetime. Even if information storage media could be preserved forever, the meaning of the information they contain is ultimately lost because the semantic context in which it was encoded cannot be recorded completely. Extreme examples are historical written documents that nobody can read today, because the languages and writing systems used at the time have been forgotten. Scientific knowledge is particularly vulnerable to becoming lost, because of the large amount of prior knowledge required to make sense of the scientific record.

Written human languages are the most stable semantic contexts we have: they change on a time scale of centuries to millennia. Scientific jargon and scientific notations are more short-lived. Journal articles written 100 years ago are already difficult to understand for today's scientists. The original writings of Galileo or Newton can be read only by scholars specialized in the history of science. The time scale on which original publications remain understandable is a few decades. This does not mean that knowledge is lost that rapidly. As the original writings become less and less clear, the aspects that are recognized as particularly important are constantly reformulated in review articles, monographs, and textbooks. This is why the insights of Galileo and Newton are still accessible to today's physicists.

The advent of computers has not changed the speed of scientific progress on a specific problem. Computers allow us to study more complex phenomena, and attack more questions in parallel, but the translation of individual scientific findings into robust insights relies on humans and still happens on the time scale of years to decades. However, computing technology evolves at a much faster pace. This creates a dilemma for scientific software: as part of the digital garden of scientific knowledge, it should advance at the pace of science, but as a computational tool, it must evolve at the pace of computing technology, as otherwise it becomes unusable (see section 6.7). This evolution is referred to as 'maintenance', a badly chosen metaphor because it suggests that software is subject to wear or decay. It is almost inevitable during maintenance to also change the scientific knowledge embedded in the code, intentionally or by accident. This is one reason why reproducibility, the subject of chapter 6, has become such an important subject in recent years.

The consequence of the different time scales on which scientific knowledge and computing technology evolve is that we are losing access to the original forms of digital scientific knowledge faster than it can be integrated into the reformulation process of science. For many computational studies performed during the last decades, it is already impossible to find the exact models and methods that were used. We are also losing data stored in formats that are defined by software that reads and writes them, if that software is not adequately maintained. To solve this problem, we will have to be more careful about how we store digital scientific knowledge, and in particular make an effort to isolate it from the rapid changes in computing technologies. This requires in particular defining data models and data formats that are independent of specific software packages, and use them to store

scientific knowledge in curated digital archives. In some disciplines, in particular the life sciences, this process has already been going on for a few decades.

## 7.5 The industrialization of science

Taking another step back in looking at the changes to scientific research that computation has already introduced or will likely introduce in the near future, they bear a strong resemblance to the transformations that the industrial revolution has caused in the ways we interact with the material world. In fact, computation provides the same kind of automation for information processing that industrialization enabled for processing matter. We can also observe first structural changes in scientific research that are similar to what happened in the early industrial age.

Since the beginnings of science, researchers have been working like craftspeople. Individuals define personal research projects and execute them using skills and competences they have acquired in a prior phase of apprenticeship to more experienced scientists. Bigger projects are realized through the collaboration of several individuals with different but overlapping skill sets. The findings resulting from a research project are considered personal achievements of their authors and associated with their names. The organizational structures of academia reflect this analogy very well: PhD students are apprentices, postdocs are journeymen, and tenured researchers are masters. Universities take the role of the medieval guilds, overseeing the practice of the crafts but not interfering with the day-to-day work of practitioners as long as it conforms to the established social norms.

Early industrial products were similar to the products of craftspeople they replaced, but due to automation they were cheaper and of more consistent, though not necessarily higher, quality. This is the stage that experimental scientific research has entered with high-throughput techniques, for example in sequencing genomes. The data analysis pipelines that bioinformaticians use to make sense of the resulting genome data can then be seen as the first-stage industrialization of theoretical science. The intellectual credit for the results of these automated procedures is attributed to the people who design the automation process, rather than to those who keep the machines running.

Increased productivity through automation was only the starting point of industrialization. What followed can be described as the emergence of increasingly complex organizational structures for the creation of increasingly complex artifacts, as for example computers. Collaborating craftspeople could never have produced such artifacts, because they lack the structure required to coordinate the large number of specialized experts involved in sophisticated technologies. Today's hierarchically organized companies coordinate the efforts of hundreds to thousands of people. But production is only one aspect of complex artifacts. They also need to be evaluated, sold, and maintained, and their impact on public goods such as the environment or public health requires regulation by public authorities. Industrial products are thus characterized by the many roles that people take in relation to them, with each role requiring specific competences. Designing and producing

computers, developing software, and using computers plus software are examples of such distinct roles and competences.

The transformation of scientific methods into black-box tools that I have outlined in section 7.2 can thus be seen as the second stage of the industrialization of scientific research. The roles of software developers and software users become distinct, following the lead of commodity software outside of science. The keyword that signals this stage of industrialization is 'reusable', implying the creation of products meant to be used by someone else than the original author. It is also increasingly applied to scientific datasets, as part of the FAIR (findable, accessible, interoperable, reusable) principles [4]. Datasets are thus also on the way to becoming industrial products.

Many of the problematic aspects of computation in science that I have mentioned in this book are symptoms of an incomplete transition to an industrial style of working. In the world of craftspeople, individual scientists are expected to understand the context in which data were collected and the analysis methods they apply to them. In a world of reusable but also more complex datasets and software-based methods, this is no longer possible, leading to symptoms such as statistical irreproducibility (see section 6.3). Likewise, craftspeople doing computational science have an intimate knowledge of their software tools, which are designed to be re-editable rather than reusable (see section 6.7). With complex reusable software, they cannot retain full mastery of their software installations, and suffer computational irreproducibility (see section 6.4).

There are of course important differences between the production of material goods and the creation of immaterial goods. For example, economies of scale play a major role in the former, but are absent in the latter. There is an even more important particularity in scientific research, whose goal is discovery. Discovery cannot be planned and thus cannot be organized on a large scale. For the foreseeable future, research is likely to be dominated by the work of craftspeople, with industry-like 'Big Science' remaining a complement. However, these craftspeople will use industrially produced tools (software) and components (datasets) for their work, much like a modern-day carpenter does. What remains to be worked out is the interfaces between industrial producers and the craftspeople working with their outputs.

One important aspect of these interfaces is where they are situated in knowledge space. An industrial product makes sense only if its users can operate with a much more limited knowledge of its characteristics than its producers. Industrial products must be specifically designed to be safe to use under such conditions. For example, a car whose driver can only drive safely if he or she is aware of the inner workings of the braking system is not acceptable. What this means for the designer is that the product's user interface must be composed of robust abstractions (see section 5.3). This is not the case today for most scientific software, and even less for published datasets. Safe use of these supposedly reusable items requires collaboration with their authors or with a community of power users.

The social aspects of defining the interfaces between industrial products and craftspeople also require further attention. We need auxiliary professions and

institutions that formulate best practices for the development, documentation, and use of industrially produced tools in science, and oversee their correct application. We must develop the analogues of user manuals, quality labels, expert evaluations, certified training, and safety regulations that have evolved at the interface of craft and industry elsewhere. Reproducibility, the subject of chapter 6, is already becoming a quality label, and certification agencies such as CASCaD [5] or CODECHECK [6] have started to offer expert services for attributing this label. Data management plans are an early example of regulation. As with quality labels and regulations in traditional industries, they do not promise perfection. Reproducible results can be wrong, and data managed according to best practices can contain mistakes. The goal is not unattainable perfection, but establishing trust in the work of others that one cannot verify oneself.

## 7.6 Preparing the future

In the early days of computing, in the 1950s and 1960s, technology was driven by the needs of scientific users, who were the most demanding clients at the time. In the following decades, computers have found their way into all aspects of our lives and the only computing technology that is still dominated by scientific applications is high-performance computing. In spite of the latter's high visibility, it represents a small fraction of the computing technology that scientists use for their daily work. With the exception of domain-specific scientific software, all of the technology that scientists use was developed outside of the scientific community and often for very different applications. As a consequence, scientists have started to consider computing technology as imposed from the outside. Few of today's computational scientists would even consider working with computer scientists or computer manufacturers on technological developments that better fit their needs.

All the problematic aspects that I have mentioned in this chapter can be traced back to the lack of technological developments that cover the specific requirements of scientific computing. Computer scientists do not develop better formal languages for scientific models because nobody asks for them. Programming languages are not tailor-made for scientific computing, except for high-performance languages, because scientists do not clearly state their needs. Reproducibility and long-term stability are not priorities in the design of computing systems, because scientists do not even envisage requesting them.

Computers have become so important for scientific research that computational scientists should care about their development with the same enthusiasm that their experimental colleagues show for the improvement of lab instruments. In other words, scientists must take a more active part in the development of computing technology again, at all levels from hardware via systems software and applications software to the management of scientific data. I hope that this book will contribute to this process by giving its readers sufficient background knowledge that they can formulate their requirements and discuss them with computer scientists, software engineers, and hardware designers. Ultimately, this will benefit everyone by leading to better science.

## 7.7 Further reading

The transformation of the concept of knowledge in the context of the information explosion caused by computers and the Internet is the subject of David Weinberger's *Too Big to Know* [7]. Ann Blair's similarly titled *Too much to know* [8] discusses the same problem in the historical context of the invention of the printing press.

The new communication technologies that were made possible by computers and the Internet are also likely to introduce profound changes into the process of doing scientific research. This topic is explored in detail by Michael Nielsen's *Reinventing Discovery* [9]. Another good use for new technologies is better explanation of scientific concepts and findings. The Web site 'Explorable Explanations' provides many examples, to which Bret Victor's 'Media for Thinking the Unthinkable' adds theoretical underpinnings.

An impressive example of publicly shared datasets explained through tutorials with embedded code is provided by the tutorials of the LIGO project on the observation of gravitational waves.

The use of computers for the generation or verification of mathematical proofs is the subject of an ongoing debate [10–12] about the status of computer-generated knowledge. A famous example is the proof of the four-color theorem [13].

My own contribution to improving the management of scientific knowledge in the context of computation is the development of a Digital Scientific Notation for physics and chemistry [14, 15] which is intended to permit the definition of computational models in journal articles and textbooks rather than exclusively in software.

## References


[1] Caulfield M 2015 The garden and the stream: a technopastoral https://hapgood.us/2015/10/17/the-garden-and-the-stream-a-technopastoral/

[2] Hinsen K 2014 Computational science: shifting the focus from tools to models [v2; ref status: indexed, http://f1000r.es/3p2] *F1000Research* **3** 101

[3] Anderson C 2008 The end of theory: the data deluge makes the scientific method obsolete *Wired*

[4] GO FAIR Initiative 2019 *FAIR Principles* https://web.archive.org/web/20191202184853/https://www.go-fair.org/fair-principles/.

[5] CASCaD - Certification Agency for Scientific Code & Data 2019 https://www.cascad.tech/.

[6] Eglen S and Nüst D 2020 *CODECHECK* https://codecheck.org.uk/.

[7] Weinberger D 2011 *Too Big to Know: Rethinking Knowledge Now that the Facts Aren't the Facts, Experts are Everywhere, and the Smartest Person in the Room is the Room* (New York: Basic Books)

[8] Blair A 2010 *Too Much to Know: Managing Scholarly Information before the Modern Age* (New Haven, CT: Yale University Press)

[9] Nielsen M 2013 *Reinventing Discovery: The New Era of Networked Science* reprint edn (Princeton, NJ: Princeton University Press)

[10] Thurston W P 1994 *On proof and Progress in Mathematics* (arXiv:math/9404236)

[11] Wolchover N 2013 In computers we trust? *Simons Foundation* https://simonsfoundation.org/features/science-news/in-computers-we-trust.
[12] Hartnett K 2015 *Univalent Foundations Redefines Mathematics* https://www.quantamagazine.org/univalent-foundations-redefines-mathematics-20150519/.
[13] Gonthier G 2008 Formal proof—the four-color theorem *Not. Am. Math. Soc.* 55 1382–93
[14] Hinsen K 2016 *Scientific Notations for the Digital Era* (arXiv:1605.02960 [physics])
[15] Hinsen K 2018 Verifiability in computer-aided research: The role of digital scientific notations at the human-computer interface *PeerJ Comput. Sci.* **4** e158