Chapter

Data and Data Quality in Mathematics

Katja Berčič

Abstract

Pure mathematics is often viewed, even by its practitioners, as a discipline in which data play little or no role. Data, when acknowledged at all, are often seen as a byproduct of research rather than a research product in their own right. Yet databases and datasets are increasingly central to the way mathematicians formulate conjectures, test hypotheses, and explore complex structures. Unlike empirical data, data in mathematics often consist of exact values derived from symbolic definitions or computations and commonly describe highly structured objects such as graphs, elliptic curves, or manifolds. This combination of abstraction, precision, and low redundancy poses distinctive challenges for data quality, shifting the focus away from concerns like noise and bias toward correctness, completeness, consistency, and accessibility.

Keywords: mathematical knowledge management, digital mathematics libraries and repositories, computer-assisted mathematics, implementation challenges, data quality dimensions, mathematical data

1. Introduction

Data quality is increasingly central in science, with data integrity recognized as a prerequisite for accurate and reproducible data-driven studies. For scientific data, key quality principles include accuracy, completeness, reproducibility, understandability, interpretability, and transferability [1]. Although a comprehensive framework for scientific data quality has yet to emerge, its importance is growing, facilitated in part by initiatives such as the FAIR principles [2], which promote findability, accessibility, interoperability, and reusability.

In mathematics, however, discussions about data are rare, and those about data quality are rarer still. This reflects the traditional view that definitions, theorems, and proofs suffice for doing mathematics. Unlike in other sciences, validation rests on proof rather than on experiment or observation; accordingly, results rarely depend on data, which are far more likely to be used for exploration than for validation. Unlike raw data, which often have limited explanatory value, proofs are expected not only to certify truth but also to deepen understanding. Although the growing role of computers has softened this stance, the underlying philosophy persists: it remains a common claim that "mathematics rarely produces data and that

the few data available need no particular management" [3, p. 2]. More broadly, computer-assisted proofs have sometimes prompted debate, especially in high-profile cases such as the Four-Color Theorem (see, e.g., [4, Chapter 11] and [5, Chapter 4]) and Hales's proof of the Kepler conjecture (see [6] and [7, Chapter 13]).

It is, therefore, unsurprising that there are no widely established standards for data quality within the mathematical community. Repeating the search experiment of [3] shows that, at the time of writing, there are still no entries for mathematics in the RDA Metadata Standards Catalog.[1] The sometimes peculiar nature of mathematical data creates opportunities for higher quality but – when coupled with prevailing attitudes – also introduces additional challenges.

The history of table-making in mathematics is surprisingly long [8] – arguably stretching back to 2500 BCE. Mathematicians have also been among the early adopters of ideas related to data integrity and assurance, such as redundant computations, certificates of correctness, and formal verification. Moreover, attitudes toward data in mathematics are beginning to change, in part due to FAIR-related initiatives – such as the German National Research Data Infrastructure (NFDI) and its mathematics consortium MaRDI[2] – which have increased the visibility of mathematical data and heightened awareness of quality considerations.

Research data are often defined as "all digital and analog objects that are generated or handled in the process of doing research" [3, p. 40]. In mathematics, this broad definition encompasses artifacts as varied as paper publications, computational results, algorithms, code, software packages, literate programming environments (e.g., Jupyter, Mathematica), experimental and simulation data, descriptions of mathematical models, formalized mathematics, and collections of mathematical objects. By this definition, every mathematician who writes a paper also produces research data. Clearly, a discussion of data quality across all these categories lies far beyond the scope of this chapter.

For the purposes of this chapter, we focus on data in the narrower sense – principally databases and datasets centered on examples. In practice, drawing this boundary is challenging, as any editor of MathBases [9] (an index of mathematical databases) will attest. The line between collections of examples and encyclopedia-style resources is often blurred – consider, for instance, the DLMF [10], the NIST Digital Library of Mathematical Functions. Because formalization of mathematics offers opportunities for data quality in mathematics, we will also briefly consider data and libraries from formalized mathematics, while steering clear of code, mathematical models, and simulation data. The latter two, in particular, are more closely associated with applied mathematics. Even with this restriction, we will encounter a surprising diversity of data – in structure, intended use, methods of generation, and other characteristics. These differences necessarily shape the strategies available for assessing and improving quality.

To provide a foundation for a discussion of data quality, Section 2 provides an overview of mathematical data that falls within the scope of the chapter. Section 3 focuses on the dimensions of data quality as they relate to mathematical data and the specific challenges and opportunities they present in the mathematical setting. Section 4 describes the concrete strategies adopted in two popular mathematical databases to ensure data quality.

[1] https://rdamsc.bath.ac.uk/subject/Mathematics%20and%20statistics

[2] https://www.mardi4nfdi.de/

2. An overview of mathematical data

For the remainder of this chapter, we use the term "mathematical data" to mean the data within the scope outlined in the introduction. Such datasets are used to represent mathematical knowledge in a structured form or to serve as benchmarks for the performance of systems and algorithms. Because mathematical data are a niche topic – even within mathematics – and because they differ from the data one encounters elsewhere, an overview is necessary.

To ground the discussion, we begin with the Online Encyclopedia of Integer Sequences (OEIS) [11], which illustrates how large-scale, community-driven datasets can support both knowledge representation and exploratory research. We then briefly survey several other widely used mathematical databases to which we will refer throughout the chapter. After these examples, we step back to identify the higher-level properties that such collections exhibit. Finally, we devote a few pages to formalized mathematics and the data it entails, both to highlight how different such data can be and because of the growing interest in using formalization alongside collections of examples.

2.1 OEIS, the Online Encyclopedia of Integer Sequences

The OEIS [11, 12] is a database of knowledge about integer sequences, indexed by the sequences' initial terms. Researchers use it to identify known sequences or discover connections to prior work. As evidenced by the over 10,000 citations in the scientific literature, the OEIS has become an incredibly useful tool in the mathematical community.

The number of sequences in the database is over 380,000 at the time of writing in 2025 and grows by roughly 10,000 new entries each year. New sequences are added through a carefully moderated process. Contributors submit new sequences (or updates to existing ones), and a team of volunteer editors, comprising roughly 200 associate editors and about 30 editors-in-chief, reviews each submission for accuracy, completeness, and adherence to the OEIS style guidelines.[3] Beyond user submissions, the editors also gather information from external sources to keep the database up-to-date. They monitor journals and preprint servers (such as arXiv[4]) for mentions of integer sequences or OEIS identifiers, using tools like Google Scholar alerts to flag new papers. When a new paper or preprint references an OEIS sequence or contains a noteworthy sequence itself, the team updates the corresponding OEIS entry, adding the bibliographic reference or creates a new entry for a previously unlisted sequence.[5]

The success of the OEIS is unsurprising: it tackles the hard problem of determining whether a result – or an equivalent formulation – already appears in the literature. A researcher seeking a result involving an integer sequence can simply enter the initial terms into the OEIS search bar. Crucially, those initial terms are independent of the particular formulation used to state the result. Billey and Tenner [13] call this a "theorem fingerprint."

[3] https://oeis.org/community.html

[4] https://arxiv.org

[5] https://oeis.org/wiki/Works_Citing_OEIS

```
%I A007299
%S A007299 1,1,1,5,3,60,487
%N A007299 Hadamard matrices of order 4n.
%D A007299 M. Jones, The Catalan numbers, Amer. Math. Monthly,
  Vol. 256 (1939), pp. 1444-1578.
%K A007299 nonn,easy,more
%F A007299 a(n) = n^4 + 3*n.
%O A007299 1,4
%A A007299 Jane Smith (jsmith(AT)math.www.edu)
```

Listing 1.
A simplified example of the OEIS internal format. The flag `%I` *denotes the identification line,* `%S` *gives the beginning of the sequence,* `%N` *denotes the name of the sequence,* `%D` *provides detailed references,* `%K` *indicates keywords,* `%F` *specifies a formula,* `%O` *represents the offset (see [14]), and* `%A` *identifies the author, submitter, or other authority. Source [14].*

Each entry is stored in the OEIS internal format [14] (see **Listing 1** for an example), which prescribes the document-level structure.

Beyond the line flags, however, the format does not enforce a syntax for the content of lines; in particular, there is no standardized syntax for formulas or code. The formula syntax is usually predictable in practice, but it can be ambiguous (though clear to a mathematician). For example, `a*(x+y)` is often written as `a(x+y)`, which could also be read as the application of the function `a` to `x+y`. For further reading on parsing the OEIS, see, for example, [15].

Let us pause to note two observations. First, data entries can contain complex objects, such as mathematical formulae and code. Second, their syntax does not necessarily adhere to a single standard. Both features are common in mathematical data and can even be desirable: for example, in [16], integers are recorded either in standard decimal notation or – when it better reflects the mathematical content – in prime factorized form (e.g., `(2^18)(3^1)`).

2.2 LMFDB, the L-functions and modular forms database

The Langlands Program is a network of interconnected results and conjectures linking number theory, automorphic forms, and geometry, with the concept of an L-function at its core. L-functions also appear in mathematical physics and cryptography. Because the area is highly specialized, the LMFDB [17, 18] focuses on making these relationships explicit, thereby making the subject more accessible.

The LMFDB publicly launched in 2016 and has since grown into an open-source, collaborative project governed by an editorial board and backed by various research funding agencies. New datasets – such as one for abstract groups – continue to be incorporated into the LMFDB. Each object is presented on its own "homepage," which includes its defining invariants, links to related items, provenance information (including computation methods and reliability), and embedded "knowls" to explain specialized terms inline. The database is massive[6] – comprising billions of objects and terabytes of data – and supports browsing, querying, and export features that integrate with tools like SageMath, Magma, and PARI/GP. The efforts toward reliability, completeness, and provenance are summarized on a dedicated page.[7]

[6] https://www.lmfdb.org/api/stats

[7] https://www.lmfdb.org/rcs

The LMFDB illustrates how the correctness of data in mathematics can become complicated. There are several efficient algorithms used to produce data whose correctness depends on the Riemann hypothesis [18, Section 2.2]. While these algorithms have versions that do not depend on it, they are too slow to use for the computations. Such attribute values are marked with an asterisk in the LMFDB.[8]

2.3 HOG, House of Graphs, a database of interesting graphs

In graph theory, the Petersen graph (**Figure 1**) is a graph on 10 vertices with 15 edges. It frequently emerges as a counterexample to many conjectures, earning it near-legendary status among mathematicians for testing graph-theoretic statements. The House of Graphs [19] provides a searchable repository of graphs, based on the observation that, as the authors put it, "all graphs are interesting, but some graphs are more interesting than others." While "interesting" is never formally defined, it typically refers to graphs that have appeared as counterexamples, extremal cases, or have otherwise proven mathematically noteworthy. In addition to the database of interesting graphs, the House of Graphs also hosts a "graph meta-directory": for certain well-studied classes (e.g., snarks and fullerenes), it hosts complete enumerations subject to specified size constraints.

Cataloging all graphs up to a given size is both infeasible (see **Table 1** for the explosive growth in graph counts) and undesirable, since most graphs lack intrinsic mathematical interest – analogous to how it would not make sense for the OEIS to catalog all integer sequences. In this way, the House of Graphs provides a valuable resource for researchers testing new conjectures or investigating invariants.

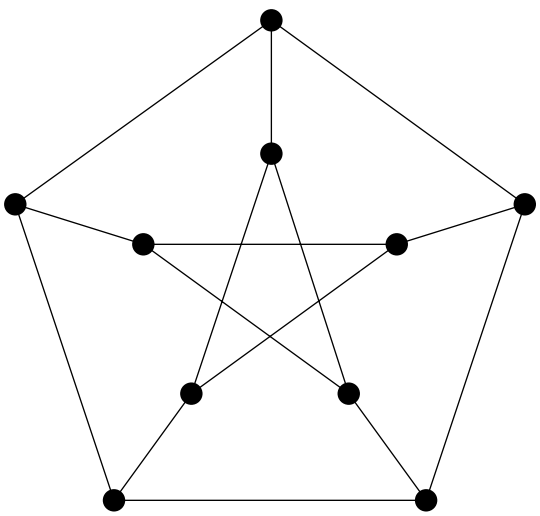

Figure 1.
The Petersen graph is notable as a counter-example in numerous graph-theoretic conjectures. This classic drawing illustrates its highly symmetric structure.

[8] See https://www.lmfdb.org/knowledge/show/nf.assuming_grh and https://www.lmfdb.org/NumberField/4.0.1086707041893.1 for an example

n	
1	1
2	2
3	4
4	11
5	34
6	156
7	1 044
8	12346
9	274668
10	12 005 168
11	1 018 997 864
12	165 091 172 592
13	50 502 031 367 952
14	29 054 155 657 235 488

Table 1.
The number of simple graphs on n unlabeled nodes (see also the OEIS sequence A0,00088 [20]), for n between 1 and 14, illustrating the combinatorial explosion.

Graphs and graph drawings can be exported to various formats readable by other software packages.

The House of Graphs welcomes community submissions under light moderation. It also incorporates extremal graphs identified by systems such as PHOEG [21] and the earlier GraPHedron [22]. These systems plot all (nonisomorphic) graphs of a given order against a chosen pair of (numeric) invariants, compute the convex hull of the plot (**Figure 2.**), and identify the graphs at the vertices of the hull.

2.4 Smallgrp, the (GAP) Small Groups Library

The Small Groups Library [23, 24] is a dataset in computational group theory, available in the computer algebra systems GAP and Magma. It provides complete, up-to-isomorphism catalogs of all groups of certain "small" orders – those whose size is below specific bounds or whose prime factorizations are constrained – thus allowing exhaustive listings by isomorphism type for each eligible order (see the GAP package manual for precise contents).

The package offers functions to retrieve a certain group of a given order, count groups of a given order in the library, and list them (including filtering). The group data in the library occupies only about 30 MB while containing over 400 million groups. This is only possible thanks to the efficient encoding of group families. An additional 47 MB is used by group-identification data. Although the data has been carefully cross-checked and is considered highly reliable, users are advised to validate results in critical scenarios.

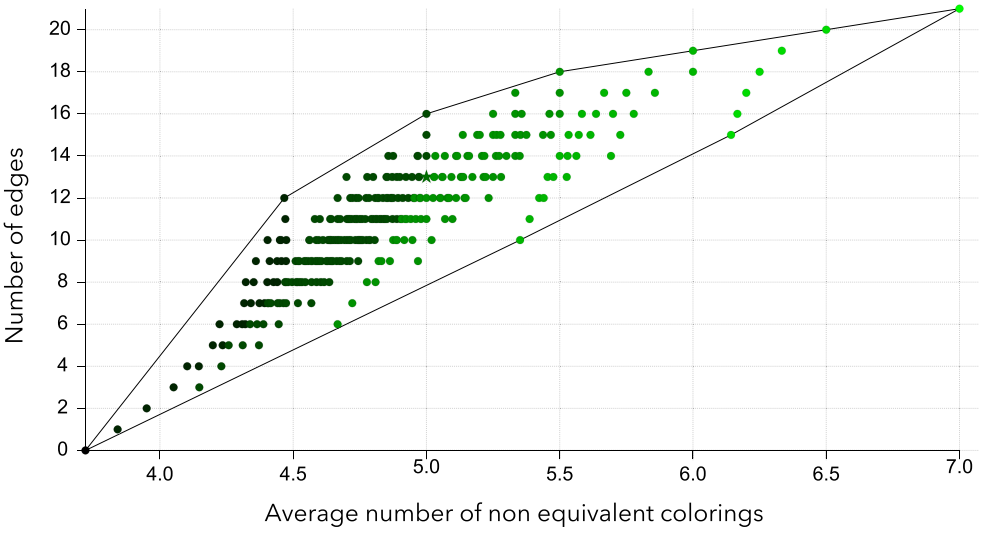

Figure 2.
The plot for graphs of order 7 shows the number of edges and the average number of nonequivalent colorings. The extremal graphs form the vertices of the convex hull. Source: https://phoeg.umons.ac.be/phoeg.

Much of the Small Groups Library's data is generated on demand. This design strikes a balance between storage efficiency and advanced search capabilities – tilting toward the former – while still enabling targeted queries. It also means the data is accessible only via the two computer algebra systems, rather than as a standalone downloadable dataset.

2.5 π-Base, a community database of topological counterexamples

The π-Base [25, 26] is a community-driven online database dedicated to cataloging topological spaces that serve as counterexamples in general topology. Founded in 2014 by James Dabbs and joined soon after by Steven Clontz, π-Base evolved as a digital realization of the book Counterexamples in Topology, aiming to make this key resource both searchable and interactive. Researchers and educators can explore the database by searching for spaces via their properties (i.e., "connected but not path-connected"). Users may also add new spaces, properties, and theorems; consistency checks are performed on entry into the database, ensuring that there are no contradictions introduced between properties of spaces and theorems. π-Base's data and software are openly maintained in repositories. The three atomic types of entries in π-Base are properties, objects and theorems. Each property or object is stored in a text file and has an ID, a name with aliases, references (in YAML format, **Listing 2**) and in some cases, a longer description (below the YAML block). Theorems (**Listing 3**) substantially reduce the need to enter space–property pairs manually: if an object satisfies certain Boolean properties, the truth value of additional properties follows. Using theorems in this way also decreases the probability of an error.

```
--
uid: P000001
name: "$T_0$"
aliases:
  - Kolmogorov
  - T0
refs:
  - doi: 10.1007/978-1-4612-6290-9
    name: Counterexamples in Topology
--
```

Listing 2.
Source for property P1 on π-Base. Note the use of a LaTeX formula in the name of the property. Source: https://github.com/pi-base/data/blob/main/properties/P000001.md.

```
---
uid: T000042
if:
  P000052: true
then:
  P000002: true
refs:
- doi: 10.1007/978-1-4612-6290-9
  name: Counterexamples in Topology
---
Asserted on Figure 9 of {{doi:10.1007/978-1-4612-6290-9}}.
```

Listing 3.
Source for the theorem T42 on π-Base. Source: https://github.com/pi-base/data/blob/main/theorems/T000042.md.

The π-Base model applies more broadly to domains where objects are described by Boolean properties – that is, they either satisfy a property or they do not. The Database of Ring Theory [27] follows a similar approach, whereas the House of Graphs would require extending the model to include valued properties (e.g., numeric invariants).

2.6 Collections of examples

People have used tables as calculation aids for over 4,000 years [8]. They have served purposes ranging from tabulating values of mathematical functions to summarizing empirical measurements. Collections of examples of mathematical structures – where each row represents one object and its invariants – stand in this tradition and are among the closest things to "typical" data in mathematics. We have seen several examples of collections of mathematical structures in the previous sections.

Such datasets help avoid recomputation, enable exploration, reveal patterns, inspire conjectures, and provide counterexamples. Once a conjecture crystallizes, attention often shifts to establishing a general proof. In practice, these collections curate (counter)examples, index or "fingerprint" theorems via simpler invariants, serve as knowledge references or algorithm benchmarks, and sometimes witness existence – where a single example suffices.

In collections of examples, most recorded values are expected to remain stable; changes usually reflect updates to auxiliary metadata (e.g., bibliographic references, code, or provenance notes). Corrections to computed values are rare.

2.6.1 Curated and generated collections

The OEIS, the House of Graphs, and π-Base are examples of databases in which entries are added for their interest to the target audience. In such cases, objects are often accompanied by links and references. In other settings, the purpose is better served by an enumeration – that is, generating all examples of a given type (often subject to a size constraint) – which enables exhaustive testing of hypotheses within that bound. Examples of such datasets are SmallGrp and most collections included in the LMFDB. Another such example is the enumeration of cubic arc-transitive graphs. Ronald Foster started collecting them in the 1930's [28]; the enumeration has since been extended several times[9] [29]. Clontz [26, p. ~5] likens curated databases to well-curated museums, featuring the most important or interesting examples from a domain, and enumeration databases to tools helping their users to find a needle in a haystack.

Computer-generated datasets have the potential for orders of magnitude more content. The decision on whether to include all objects of a certain type, perhaps subject to a size constraint, or to carefully curate the objects for inclusion, or to choose something in between, can also depend on the available computing and storage resources. Beyond enabling search and filtering, storing generated data often makes sense because the underlying computations are resource-intensive and time-consuming. For this reason, redundant recomputation (or recomputing with a newer algorithm) may be impractical.

It does not always make sense to enumerate all objects – perhaps most are not interesting, or their numbers grow explosively. The House of Graphs, in particular, does not fall neatly into either category, as its entries combine community contributions with automated generation of "extremal cases." It might not even be possible to obtain all objects of a given type, as is the case with topological spaces. In the case of π-Base, including more objects than necessary is not even desirable. Thus, in π-Base, every property, object and theorem is added manually by an expert.

2.6.2 On-demand generation of examples

When it is feasible to generate objects algorithmically, storing the entire output is not always desirable. If the main purpose is to support testing properties on selected subsets, on-demand generation often suffices. This philosophy underlies widely used graph-generation tools such as nauty and Traces [30]. nauty and Traces compute automorphism groups of graphs and digraphs and can produce canonical forms. The distribution also includes the `gtools` utilities; for example, `geng` generates nonisomorphic graphs very quickly. There are companion generators for bipartite graphs, digraphs, and multigraphs, as well as programs for manipulating files of graphs in a compact format.

[9] https://www.math.auckland.ac.nz/~conder/symmcubic10000list.txt

A related on-demand approach appears in the Small Groups Library, where certain families of groups are constructed rather than stored in full; this enables the enumeration to go further while keeping storage requirements manageable.

2.6.3 Quantitative aspects

While the OEIS is an example of a massively collaborative database, collections of examples can also be authored by a single author or a small group of authors (these are often not updated). Community databases, such as the LMFDB [17] and the House of Graphs [19], fall somewhere in between, with large numbers of contributors but not quite at the levels of the OEIS.

Some enumerations contain very few elements – even without explicit size constraints – and therefore do not resemble a dataset in the usual large-scale sense. The Platonic solids[10] (**Figure 3**) are a classic example of such a classification.

With increased computing power, enumerations have grown in size and complexity, and can contain over $3 \cdot 10^9$ objects [32] (while the LMFDB has billions of objects, it is a collection of enumerations of different, but related, objects).

Compiling such collections can demand very different resources, depending on the objects involved. Collection can require a lot of storage space (a single subgroup lattice of a large group can require several gigabytes), of computation time (the LMFDB team estimates that recreating the LMFDB from scratch would take roughly 3000 CPU-years), mathematical complexity, and community effort (e.g., the rich annotations for each sequence in the OEIS).

2.7 Formalized mathematics and data

Formalization of mathematics refers to expressing definitions, statements, and proofs in a precise language that a program can check or prove automatically [33]. While formal proofs require substantially more effort than informal ones, they can be checked with a computer, offering a higher level of assurance.

Proof assistants such as Rocq [34] (previously Coq), Isabelle [35], HOL Light [36], Mizar [37], and Lean [38] are computer systems that help users create, organize, and formally verify mathematical constructions and theorems; they also support writing programs and carrying out computations. Most proof assistants rely on small, trusted kernels – typically based on dependent type theory or higher-order logic – that validate each derivation, with many steps often automated by higher-level tactics and procedures. Several ambitious formalization efforts have been successfully completed, such as the Flyspeck Project [39], the Liquid Tensor

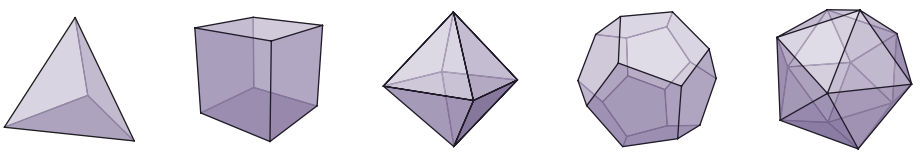

Figure 3.
Platonic solids: regular tetrahedron, cube, regular octahedron, dodecahedron, icosahedron; original image by the user drummyfish [31], licensed under CC0.

[10] https://en.wikipedia.org/wiki/Platonic_solid

Experiment [40], and the proof of the polynomial Freiman-Ruzsa conjecture [41], to mention just a few recent ones. Among these tools, Lean [42] has recently become the most popular in the mathematical community. Kevin Buzzard's *What is the point of computers? A question for pure mathematicians* [43] offers an approachable overview of the development and future of proof assistants in mathematics.

In contrast to interactive proof assistants, automated theorem provers (ATPs) apply decision procedures and search strategies to prove formulas in specific, restricted logical fragments automatically. ATPs can be remarkably powerful within their domains, but their expressivity is limited; one cannot, in general, formalize and develop a generic mathematical theory entirely within such a system. Proofs produced by ATPs can contain errors. However, when integrated with interactive theorem provers as "hammers" [44], ATPs are used to find easy subproofs and guide searches, while the interactive system checks the resulting proofs, ensuring correctness.

The proof assistant's kernel prevents declarations that do not type-check (or, equivalently, incorrect proofs), but it cannot prevent a user from introducing an inaccurate definition. A common safeguard is to state a specification theorem and prove that the definition satisfies it; it is not possible to prove that a bad definition satisfies the specification. Semantic errors, though rare, typically arise when a definition fails to capture the intended concept; these are mitigated by human review (in Mathlib, discussion on Zulip followed by GitHub review). Choosing definitions that are the most suitable for reasoning in the long run is inherently difficult.

It is natural to ask whether collections of examples of mathematical structures could benefit from formalization. Consider the OEIS from the previous section. In principle, certain mathematical properties of integer sequences could be verified automatically – especially if entries included references to formal proofs. On the one hand, standard computer algebra systems could check that the initial terms are consistent with the stated formulas. On the other hand, performing the same checks within a proof assistant would provide a higher level of assurance. By contrast, some attributes – such as bibliographic citations – still require human curation and editorial review.

We wrap up this section with a description of two very different libraries related to formalization. They are by far not the only ones; some other notable examples include the Isabelle libraries[11] and the Mizar Mathematical Library [45] for proof assistants, and SMT-LIB [46] for SMT solvers.

2.7.1 Mathlib

Mathlib [47] is a community-driven library of formalized mathematics for the Lean proof assistant. It serves as a foundation for both advanced mathematical research and verification projects[12] and – thanks to sustained community engineering [48] – has scaled to nearly two million lines of Lean code. Development is open on GitHub: contributions are submitted via peer-reviewed pull requests, and continuous-integration infrastructure enforces quality standards. The repository also contains tooling and infrastructure for Lean. Coverage spans areas such as algebra, analysis, topology, set theory, and number theory.

Beyond content, Mathlib leverages Lean's metaprogramming to provide powerful automation and custom tactics (e.g., the Aesop proof-search tactic [49]), as well as

[11] https://isabelle.systems/libraries.html

[12] https://lean-lang.org/use-cases/mathlib/

a documentation-generation pipeline and other community tools that make large-scale formalization feasible and maintainable. Official documentation and "Mathematics in Lean" tutorials lower the barrier to entry for researchers and students alike, and the Lean Zulip chat functions as the hub for design discussion and pre-review before GitHub pull requests.

2.7.2 TPTP, Thousands of Problems for Theorem Provers

While not exactly a library of formalized mathematics, the TPTP[13] [50, 51] is a library of thousands of logic problems for benchmarking ATPs. Created in the early 1990s to unify disparate collections, it aggregates problems from legacy libraries, publications, and community contributions across domains in logic, mathematics, computer science, engineering, and more. TPTP serves as a primary benchmark suite and is widely used in logic-based AI research. Known solutions are archived in the companion TSTP library.

Each file encodes a single problem in the standardized TPTP syntax, accompanied by rich metadata. The problems can be of mathematical or verification interest. Contributions are open and checked for syntactic correctness upon submission. Most metadata fields are optional – including `Result`, since a solution to a problem does not need to exist.

3. Data quality in collections of examples

Although mathematicians rarely frame their concerns explicitly in terms of "data quality," many common practices already reflect implicit attention to it. Yet aspects of quality are often neglected, largely because incentives are weak: authors seldom receive credit for work beyond what is needed to advance their immediate research goals. Consequently, practices that strengthen integrity and assurance are applied unevenly. Measures that ensure mathematical correctness are routine, whereas those that foster broader accessibility and reuse are less common. In particular, the "dark long tail" has been observed in mathematics as well [52, 53] and data often remain "in somebody's bottom-left drawer."

To ensure correctness, many collaborative databases employ some form of peer review – for example, pull-request reviews in Mathlib and editorial review in the OEIS. When data are generated algorithmically, the underlying methods are typically described in peer-reviewed publications and are often additionally tested by dataset authors. Community uptake also acts as a quality filter: the more widely data are used, the more likely errors are to be discovered and corrected, which, in turn, increases confidence. Some projects add redundancy – running independent implementations or alternative algorithms – either across the full corpus or on randomly sampled subsets when complexity makes full duplication infeasible.

Because datasets in pure mathematics are generally public, nonsensitive, and developed within communities with little incentive for malicious interference, data integrity centers on preventing unintentional errors rather than guarding against adversaries. Accordingly, the emphasis is less on security and more on correctness and consistency – much as research data quality focuses on robust, reproducible processing

[13] https://www.tptp.org

[1]. Given this overarching concern with correctness and the typically infrequent updates, the most critical phase for preserving integrity is the start of the data lifecycle.

In contrast, for instance, to the life sciences, mathematical results that have been proven true remain true indefinitely. For data, this means that the correctness of attributes with mathematical content does not degrade with time. Values of attributes with mathematical content change rarely, typically when errors are corrected. Timeliness is thus more relevant for attributes with bibliographic references, cross-links, code pointers, and provenance notes. In resources such as the OEIS, being up to date means synchronizing entries with the literature by incorporating new formulas and citations as they appear. Perfect currency is unattainable, but the OEIS mitigates this by monitoring journals and preprint servers (such as arXiv) and by relying on community submissions reviewed by editors; timestamps make (potential) staleness visible and facilitate updates.

In what follows, we examine how common dimensions of data quality apply in mathematics – completeness, accuracy, consistency, and accessibility [54] – omitting timeliness, which is less central in this context. We also comment on the dimensions discussed in [1] and [2].

3.1 Completeness

Completeness for mathematical datasets can be understood in (at least) two senses. We can ask whether the collection contains all the objects it is intended to. For curated resources that aim to gather "interesting" objects, enumeration-style completeness usually defies the purpose. The data should contain enough information to be helpful, though what precisely that means is difficult to articulate. For enumerations, completeness is well-defined: it means including all objects up to isomorphism, often within an explicit size bound. Completeness is typically established by a published proof. When a full proof is infeasible – often because eliminating duplicates entirely is impossible – projects specify the precise scope of coverage instead. As an aside, whether two descriptions define the same object can itself be a deep question: equality up to isomorphism, equivalence of presentations, or equality of expressions may be computationally hard or even an open problem in general. In some cases, a canonical labeling (such as [30]) can be used to simplify checking for duplicates (while computing the labeling is about as hard as comparing two graphs for isomorphism, it is much easier to compare two labelings).

3.2 Accuracy

Correctness, that recorded objects and their attributes are exactly what they claim to be, is central to mathematics, and thus to mathematical data. In the case of enumerations, correctness also requires completeness relative to the specification (e.g., "all objects up to size n") and no duplicates (when feasible). Mathematical correctness is established via proofs. Data are typically not accompanied by proofs for each object or attribute value, though results in the literature often underpin them. Because an analogous notion of correctness can be applied to data, it is useful to recall two views on proofs. A commonly held position is that in principle, every proof could be reduced to a sequence of small steps, each easily verified (though no mathematician would actually carry them out in practice) [55]. According to this view, when a proof fails, the error stems from not spelling out the steps sufficiently

and an unverifiable inference slipping through. By contrast, Lakatos [56] emphasizes proofs as research tools rather than final verdicts: when counterexamples arise, mathematicians repair arguments (strengthening lemmas), refine or revise definitions, restrict the domain, or generalize the claim so the counterexample no longer applies.

A higher degree of trust in correctness can be obtained with the help of formalization and computational certificates. Some algorithms can be adapted to return a witness (or certificate) – a short, easily verified proof that the output is correct [57] (thought the idea goes back at least to [58]). A simple example is integer factorization: it is much harder to factor 112909084933 than to check that $132241 \cdot 853813 = 112909084933$. Although such certificates are not yet common in mathematical datasets, they are a promising direction, especially in combination with formalization and formal verification. Verifying a certificate checker is typically simpler than verifying the original algorithm. See [59] for examples of some of the techniques that can be applied: proof-by-reflection, checking certificates, and using SAT solvers.

3.3 Consistency

Consistency, understood as uniformity in data formats, structure, and presentation, is a recurring challenge in mathematical data. For example, the lack of a standard syntax for formulas in the OEIS can make parsing or comparing entries difficult. Even when the mathematical meaning is clear, inconsistent formatting – such as variation in expression syntax or in how properties are labeled – can hinder both human readability and automated processing. Standardizing formats and metadata is difficult due to the wide diversity of mathematical objects and use cases. Even for seemingly simple structures like graphs, multiple representations exist, each suited to different purposes.

Beyond structural consistency, some databases also enforce consistency between values through internal mathematical checks. For example, the LMFDB verifies that related attributes agree according to known mathematical relationships, serving both to validate correctness and to detect potential data entry or computation errors. In π-Base, such consistency is enforced using theorems wherever applicable.

3.4 Accessibility

In the context of research data, accessibility refers to the ability of intended users to obtain data quickly and reliably from typical working environments, such as web browsers, scripts, and notebook systems. It also implies that data should be available whenever needed, ideally at no cost or under minimal access restrictions.

In pure mathematics, accessibility is not constrained by concerns about sensitive or personal data, since such data are essentially nonexistent in this domain. However, this does not mean that mathematical data are always easy to find or use. In practice, accessibility is often hindered by ad hoc hosting arrangements (or even lack thereof in the case of dark data), missing or ambiguous licensing, and a lack of controlled vocabularies facilitating search across datasets.

Mathematics has an important role to play in the broader scientific ecosystem: its results and datasets often underpin work in the physical, social, health, and life sciences. As such, “mathematics has a particular responsibility to science to preserve their results in a sustainable manner” [3, p. 2]. In response to the difficulty of

locating mathematical datasets, the MathBases [9] project was launched to increase the visibility of data resources in pure mathematics and to provide a centralized index of existing data.

3.5 An overview of challenges for FAIR and related dimensions

The quality dimensions discussed in [1] largely align with the FAIR principles – especially interoperability and reusability [2]. For mathematics, many of the corresponding challenges are surveyed in [3, 60], and recent work has begun to address them (e.g., [61, 62]).

A persistent obstacle to a standardized vocabulary for annotating mathematical data is the alignment problem (see Florian Rabe's overview in [63, Section 4.7]). Concepts across resources often overlap but differ in subtle ways – sometimes reflecting foundational choices – making it hard to determine a useful level of granularity. A related difficulty concerns representation: data formats can be far from the underlying mathematical meaning. Documenting codecs that link mathematical semantics to concrete encodings can help bridge this gap [63]. Compounding matters, accurate description often demands rich, technically sophisticated metadata (e.g., formulas and specialized notation), which raises the barrier to consistent annotation and reuse.

4. Practices in the LMFDB and the House of Graphs

In this section, we revisit LMFDB and the House of Graphs, this time reporting on how these datasets maintain data quality (primarily correctness). In practice, data maintainers in mathematics typically do not publish the details of how they ensure that their data is fit for purpose – this information had to be obtained through personal communication with the maintainers: David Roe of MIT for the LMFDB and Gauvain Devillez of Université de Mons for the House of Graphs. Both teams perform extensive checking of algorithm implementations, including against known results, and apply some checks to ensure internal consistency. However, both teams stress that while they aspire to apply such checks more broadly, the implementation would take a lot of effort for little academic recognition. These examples underscore both the complexity of the mathematical software ecosystem and the practical challenges of ensuring correctness in large-scale computational databases.

4.1 The LMFDB

Costa and Roe [64] document the LMFDB's migration from MongoDB to PostgreSQL, which reduced storage requirements, sped up queries, and improved data quality. The transition revealed typos and inconsistencies in the short field names encouraged by MongoDB's schema flexibility, which were corrected. It also enabled refactoring and faster internal consistency checks.

Beyond unit tests that accompany the code, the system runs additional data-quality checks. For classical modular forms, queries designed to return no rows are run after each update with the intention of detecting structural inconsistencies. Computationally intensive integrity tests are implemented in Python; when a full

pass is too expensive, they are applied to randomly sampled objects. In certain datasets, results are recomputed independently and compared: in Magma and GAP for finite groups, and Magma, SageMath, and Pari/GP for modular forms. Discrepancies can indicate bugs in external software or internal inconsistencies in the data. For computations involving modular forms, the LMFDB team has found errors in all three of the systems used.

While working on group-theoretic data, the LMFDB team uncovered approximately 30 bugs in Magma, including issues such as segmentation faults during group identification. A comparison of a new Magma implementation for computing automorphism-group sizes with the corresponding GAP code revealed discrepancies by a factor of 2 for some groups. These bugs have been fixed in both Magma and GAP, showing how identifying bugs can be a great benefit of enumeration projects.

4.2 The House of Graphs

To ensure data correctness, the House of Graphs relies on solvers (specialized algorithm implementations for computing graph-theoretic invariants) that have been extensively checked and are trusted in the community. For computationally inexpensive invariants, outputs from any new solver are cross-checked against existing results. For NP-hard invariants, only comparisons with known values are feasible; to date, no errors in invariant values have been reported. The team is also considering compiling curated benchmark sets – graphs known to be challenging for solvers – to target verification where it is most informative. The Lean-HoG project [65] aims to integrate the database with the proof assistant Lean, checking invariant values in the process.

5. Conclusions

Mathematical datasets typically represent exact, well-defined structures. Although they are not typically affected by some of the problems that can arise with empirical data sets, they are not immune to errors or other data quality problems. Their symbolic nature, precise definitions, and internal relationships make them particularly amenable to strategies for ensuring accuracy. However, the difficulty of aligning complex mathematical concepts and the lack of a standardized vocabulary can present challenges for accurately describing and annotating the data.

Could mathematical datasets serve as a testbed for data assurance? Their exactness makes errors easier to isolate, while their abstract content enables formal reasoning and internal consistency checks. These features may offer insights for building more rigorous validation workflows – even in domains where the data are noisier.

Realizing this potential is not just a technical issue. Improving data quality in mathematics takes significant, often invisible work. Without proper incentives or recognition, this work is easily overlooked and underfunded. Addressing these systemic challenges is essential for any effort to raise data quality in mathematics.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0024 and by the Slovenian Research and Innovation Agency (program no. P1-0294).

I would like to thank Andrej Bauer, Steven Clontz, Gauvain Devillez, Dimitri Leemans, Rob Lewis, Anja Petković Komel, Primož Potočnik, Florian Rabe, David Roe, Erik Štrumbelj, Ljupčo Todorovski, and Russ Woodroofe for their time, patience, and many productive discussions. Their thoughtful feedback and willingness to engage with my questions have been invaluable throughout the preparation of this chapter.

Author details

Katja Berčič[1,2]

1 Faculty of Mathematics and Physics, University of Ljubljana, Ljubljana, Slovenia

2 Institute of Mathematics, Physics, and Mechanics, Ljubljana, Slovenia

*Address all correspondence to: katja.bercic@fmf.uni-lj.si

IntechOpen

References

[1] Miller G, Spiegel E. Guidelines for Research Data Integrity (GRDI). Scientific Data. 2025;**12**(1):95

[2] Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016;**3**(1):160018

[3] Boege T, Fritze R, Görgen C, Hanselman J, Iglezakis D, Kastner L, et al. Research-Data management planning in the German Mathematical Community. European Mathematical Society Magazine. 2023;**130**:40–47

[4] Wilson R. Four Colors Suffice: How the Map Problem Was Solved. Revised Color ed. Princeton, NJ, USA: Princeton University Press; 2014

[5] MacKenzie D. Mechanizing Proof: Computing, Risk, and Trust. Cambridge, MA, USA: The MIT Press; 2001

[6] Morgan F. Kepler's Conjecture and Hales's Proof, A book review. Notices of the American Mathematical Society. 2005;**52**(1):1034

[7] Szpiro G. Kepler's Conjecture: How Some of the Greatest Minds in History Helped Solve One of the Oldest Math Problems in the World. Hoboken, N.J: John Wiley & Sons; 2003

[8] Campbell-Kelly M, Croarken M, Flood R, Robson E. The History of Mathematical Tables. Oxford, UK: Oxford University Press; 2003

[9] MathBases - Of, by, and for mathematical databases [Internet]; 2023. Available from: https://mathbases.org [Accessed: 2025-September-10]

[10] Olver FWJ, Olde Daalhuis AB, Lozier DW, Schneider BI, Boisvert RF, Clark CW, et al. NIST Digital Library of Mathematical Functions [Internet]. 2025. Available from: https://dlmf.nist.gov/Release 1.2.4 of 2025 March 15 [Accessed: 2025-July-27]

[11] The OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences. 2025. Available from: https://oeis.org/ [Accessed: 2025-July-27]

[12] Sloane NJA. "A Handbook of Integer Sequences" Fifty Years Later. The Mathematical Intelligencer. 2023;**45**(3):193–205

[13] Billey SC, Tenner BE. Fingerprint Databases for Theorems. Notices of the American Mathematical Society. 2013;**60**(8):1034

[14] The OEIS Foundation Inc . Internal Format Used in the On-Line Encyclopedia of Integer Sequences;. Available from: https://oeis.org/eishelp1.html [Accessed: 2025-July-27]

[15] Luzhnica E, Kohlhase M. Formula Semantification and Automated Relation Finding in the On-Line Encyclopedia for Integer Sequences. In Greuel GM, Koch T, Paule P, Sommese A, editors. Mathematical Software – ICMS 2016. Cham: Springer International Publishing; 2016. p. 467–475

[16] Wilson S, Potočnik P. A Census of edge-transitive tetravalent graphs. [Internet]. 2016. Available from: https://jan.ucc.nau.edu/swilson/C4FullSite/BigTable.html [Accessed: 2025-July-27]

[17] The LMFDB Collaboration. The L-functions and modular forms database. [Internet]. 2016. Available from: https://www.lmfdb.org [Accessed: 2025-July-27]

[18] Cremona J. The L-Functions and modular forms database project. Foundations of Computational Mathematics. 2016;**16**(6):1541–1553

[19] Coolsaet K, D'hondt S, Goedgebeur J. House of graphs 2.0: A Database of Interesting Graphs and More. Discrete Applied Mathematics. 2023;**325**:97–107. https://houseofgraphs.org

[20] The OEIS Foundation Inc. Number of simple graphs on nunlabeled nodes, entry A000108 in The On-Line Encyclopedia of Integer Sequences;. Available from: https://oeis.org/A000088 [Accessed: 2025-September-08]

[21] Devillez G, Hauweele P, Mélot H PHOEG Helps to Obtain Extremal Graphsch:32. In: Fortz B, Labbé M, editors. Operations Research Proceedings 2018. Cham: Springer International Publishing; 2019. p. 251–257

[22] Mélot H. Facet Defining Inequalities among Graph Invariants: The System GraPHedron. Discrete Applied Mathematics. 2008;**156**(10):1875–1891

[23] Besche HU, Eick B, O'Brien EA, Horn M. SmallGrp, The GAP Small Groups Library [Internet]; 2024. GAP Package. Available from: https://gap-packages.github.io/smallgrp/ [Accessed: 2025-July-27]

[24] Besche HU, Eick B, O'Brien EA. A Millennium Project: Constructing Small Groups. International Journal of Algebra and Computation. 2002;**12**:623–644

[25] The π -Base Community. π-Base, a community database of topological counterexamples [Internet]. 2014. Available from: https://topology.pi-base.org [Accessed: 2025-August-29]

[26] Clontz S Database-Driven Mathematical Inquiry and the π-Base Model for Small Semantic Databases. 2025. Submitted to LuCaNT; available at https://lucant.org/papers/2025/241114-Clontz.pdf [Accessed: 2025-September-08]

[27] Database of Ring Theory Community. Database of Ring Theory [Internet]. 2014. Available from: https://ringtheory.herokuapp.com [Accessed: 2025-September-08]

[28] Foster RM, Bouwer IZ. The Foster Census: R.M. Foster's Census of Connected Symmetric Trivalent Graphs. Winnipeg, Canada: Charles Babbage Research Centre; 1988

[29] Conder M, Dobcsányi P. Trivalent symmetric graphs on up to 768 vertices. Journal of Combinatorial Mathematics and Combinatorial Computing. 2002;**40**:41–63

[30] McKay BD, Piperno A. Practical Graph Isomorphism, II. Journal of Symbolic Computation. 2014;**60**:94–112

[31] Drummyfish. Platonic solids [Internet]; 2019. Available from: https://commons.m.wikimedia.org/wiki/File:Platonic_Solids_Transparent.svg [Accessed: 2025-July-27]

[32] Kohonen J. Generating Modular Lattices of up to 30 Elements. Order. 2019;**36**(3):423–435

[33] Harrison J. A Short Survey of Automated Reasoning. In Anai H,

Horimoto K, Kutsia T, editors. Algebraic Biology. Vol. **4545**. Berlin, Heidelberg: Springer Berlin Heidelberg; 2007. p. 334–349

[34] Rocq Prover [Internet]; 2025. Available from: https://rocq-prover.org [Accessed: 2025-July-27]

[35] Isabelle [Internet]; 2025. Available from: https://isabelle.in.tum.de [Accessed: 2025-September-11]

[36] Harrison J The HOL Light theorem prover [Internet]; 2009. Available from: https://www.cl.cam.ac.uk/~jrh13/hol-light/ [Accessed: 2025-July-27]

[37] Mizar [Internet]; 2025. Available from: https://www.mizar.org [Accessed: 2025-September-11]

[38] Lean [Internet]; 2025. Available from: https://leanprover-community.github.io [Accessed: 2025-September-11]

[39] Hales TC. Introduction to the Flyspeck Project. Schloss Dagstuhl – Wadern, Germany: Leibniz-Zentrum für Informatik; 2006

[40] Castelvecchi D. Mathematicians welcome computer-assisted proof in 'grand unification' theory. Nature. 2021;**595**(7865):18–19

[41] Gowers WT, Green B, Manners F, Tao T On a Conjecture of Marton; 2023. DOI: 10.48550/arXiv.2501.10823

[42] De Moura L, Kong S, Avigad J, Van Doorn F, von Raumer J The Lean theorem prover (system description). In: Automated Deduction-CADE-25: 25th International Conference on Automated Deduction. Springer; 2015. p. 378–388

[43] Buzzard K. What Is the Point of Computers? A Question for Pure Mathematicians. International Congress of Mathematicians. 2023;**2**:578–608

[44] Blanchette JC, Kaliszyk C, Paulson LC, Urban J. Hammering towards QED. Journal of Formalized Reasoning. 2016;**9**:101–148

[45] Mizar Mathematical Library [Internet]; 2018. Available from: https://www.mizar.org/library [Accessed: 2025-September-11]

[46] Barrett C, Fontaine P, Tinelli C The Satisfiability Modulo Theories Library (SMT-LIB) [Internet]. 2016. Available from: www.SMT-LIB.org [Accessed: 2025-July-27]

[47] The Mathlib Community. The Lean Mathematical Library. In: Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020). Association for Computing Machinery; 2020. p. 367–381

[48] Baanen A, Ballard MR, Commelin J, Ge Chen BG, Rothgang M, Testa D Growing Mathlib: Maintenance of a large scale mathematical library; International Conference on Intelligent Computer Mathematics. Cham: Springer Nature Switzerland.2025. (51–70). DOI: 10.48550/arXiv.2501.10823

[49] Limperg J, From AH AESOP: White-Box Best-First Proof Search for Lean. In: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs. Boston MA USA: ACM; 2023. p. 253–266

[50] Sutcliffe G. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0.

Journal of Automated Reasoning. 2017;**59**(4):483–502

[51] Sutcliffe G Stepping Stones in the TPTP World. In: Benzmüller C, Heule M, Schmidt R, editors. Proceedings of the 12th International Joint Conference on Automated Reasoning. No. 14739 in Lecture Notes in Artificial Intelligence; 2024. p. 30–50

[52] Hulek K, Müller F, Schubotz M, Teschke O. Mathematical research data – An analysis through zbMATH References. European Mathematical Society Magazine. 2019;**113**:54–57

[53] Heidorn PB. Shedding Light on the Dark Data in the Long Tail of Science. Library Trends. 2008;**57**(2):280–299

[54] Cichy C, Rass S. An Overview of Data Quality Frameworks. IEEE Access. 2019;**7**:24634–24648

[55] Geuvers H. Proof Assistants: History, Ideas and Future. Sadhana. 2009;**34**(1):3–25

[56] Lakatos I, Worrall J, Zahar E, editors. Proofs and Refutations: The Logic of Mathematical Discovery. 1st. Cambridge, UK: Cambridge University Press; 1976

[57] McConnell RM, Mehlhorn K, Näher S, Schweitzer P. Certifying Algorithms. Computer Science Review. 2011;**5**(2):119–161

[58] Blum M, Kanna S Designing programs that check their work. In: STOC'89: Proceedings of the twenty-first annual ACM symposium on Theory of computing; 1989. p. 86–97

[59] Bauer A, Berčič K, Devillez G, Taslak J. Incorporating a database of graphs into a proof assistant. In: Kohlhase A, Kovács L, editors. Intelligent Computer Mathematics. Cham: Springer Nature Switzerland; 2024. p. 146–162

[60] Conrad TOF, Ferrer E, Mietchen D, Pusch L, Stegmüller J, Schubotz M. Making mathematical research data FAIR: Pathways to improved data sharing. Scientific Data. 2024;**11**(1):676

[61] Bacher T, Garrote-López M, Görgen C, Neubert MJ Making Mathematical Online Resources FAIR: At the Example of Small Phylogenetic Trees; 2025. DOI: 10.48550/arXiv.2501.10823

[62] Berčič K, Koprivec F. Making the census of cubic vertex transitive graphs searchable and FAIR. In: Buzzard K, Kutsia T, editors. Intelligent Computer Mathematics. Vol. **13467**. Cham: Springer International Publishing; 2022. p. 323–328

[63] Berčič K, Kohlhase M, Rabe F. Towards a unified mathematical data infrastructure: Database and Interface generation. In: Kaliszyk C, Brady E, Kohlhase A, Sacerdoti Coen C, editors. Intelligent Computer Mathematics. Vol. **11617**. Cham: Springer International Publishing; 2019. p. 28–43

[64] Costa E, Roe D. Zen and the art of database maintenance. In: Balakrishnan JS, Elkies N, Hassett B, Poonen B, Sutherland AV, Voight J, editors. Arithmetic Geometry, Number Theory, and Computation. Cham: Springer International Publishing; 2021. p. 277–292

[65] Bauer A, Berčič K, Rabe F, Thiéry N, Taslak J. Automated

Mathematics: Integrating Proofs, Algorithms and Data (Dagstuhl Seminar 23401). Schloss Dagstuhl – Leibniz-Zentrum Für Informatik. 2024;**13**:10