

An overview of some tools for preparing documents containing mathematical text

John Nash and Prashanth Velayudhan

2026-01-09

Motivation

The preparation of scientific documents with mathematical content has always given their authors and printers difficulty in rendering the material to paper or screen. This article is an exercise to illustrate how several systems for preparing such documents compare in capability and ease of use at the beginning of 2026.

The text preparation systems to be compared are L^AT_EX(Lamport (1986)), Typst (Mädje (2022)), R Markdown (Xie, Allaire, and Grolemund (2018)), Quarto (Machlis (2022)) and Asciidoc (“AsciiDoc” (2013)). All the source materials for the tools we discuss here are plain text with markup tags. Such plain text is very well-suited to version control platforms like Git (<https://en.wikipedia.org/wiki/Git>) and Subversion (https://en.wikipedia.org/wiki/Apache_Subversion), where it is quite easy to view historical changes in a document or program code. Indeed **TexStudio** (<https://texstudio.org/>) integrates either of these version control approaches.

We note that bibliographic support is often of interest to users. Further, when workers come from different backgrounds, it can be important to be able to transfer material from one system’s format to that of another. There may also be details that depend on the context of the work that can or cannot be handled. We consider examples of such details, but make no claim to completeness.

A useful example

The Christmas counting song “The Twelve Days of Christmas” suggest the singer receives one gift – we shall unabashedly simply declare that the Partridge in a Pear Tree is a single gift – on the first of twelve days, then that gift plus two more on the second, and so forth. For our needs, we want to have a general formula for the total number of gifts received after n days. While the commonly known song uses 12 days, there are variants with other values of

n . The Faroe Islands use the inflationary value $n = 15$. In our exposition, we will rely on the Wikipedia reference “The Twelve Days of Christmas (Song)” (2025) as our authority. We will name our example **Partridges** for simplicity. This document serves as a **Quarto** version of this example.

Thus we seek a formula for $T(n)$, the total number of presents received after the n 'th day. This document is being prepared in Quarto and will serve as the base presentation illustrating the straightforward mathematical content, which we believe representative of the needs of a wide class of authors.

Single day number of presents

On a single day, the number of presents $S(n)$ is clearly the sum of an arithmetic progression (Equation [Arith-progrn](#)).

$$(Arith-progrn) \quad S(k) = \sum_{i=1}^k i$$

Clearly we now want to compute

$$T(n) = \sum_{k=1}^n S(k)$$

The formula for $S(k)$ is well known, and derived by noting that writing the sequence forwards and then backwards illustrates that twice the sum is $k * (k + 1)$, so we have

$$S(k) = \frac{k(k + 1)}{2}$$

We can use this so that we find

$$2T(n) = \sum_{k=1}^n (k^2 + k)$$

We therefore need

$$Q(n) = \sum_{k=1}^n k^2$$

To give our provisional expression as

$$T(n) = \frac{Q(n) + S(n)}{2}$$

$Q(n)$ is a well-known summation, often proved by mathematical induction,

$$Q(n) = \frac{n(2n+1)(n+1)}{6}$$

Thus we want:

$$\begin{aligned} T(n) &= \frac{1}{2} \left(\frac{n(2n+1)(n+1)}{6} + \frac{n(n+1)}{2} \right) \\ &= \frac{2n^3 + 3n^2 + n + 3n^2 + 3n}{12} \\ &= \frac{2n^3 + 6n^2 + 4n}{12} \\ &= \frac{n^3 + 3n^2 + 2n}{6} \end{aligned}$$

```
n <- 12
(n^3 + 3* n^2 + 2*n)/6
```

[1] 364

Table 1 shows the total number of gifts received as of each day. In the source code of our document, we entered this as static text, mostly numbers. However, we could have developed the table using a code chunk in any of a number of programming languages. Indeed, one of us (JN) worked with Yihui Xie of RStudio (now Posit) at the UseR! meeting in Los Angeles in 2014 to add a Fortran engine to Rmarkdown.

Day	Gifts
1	1
2	4
3	10
4	20
5	35
6	56
7	84
8	120
9	165

Day	Gifts
10	220
11	286
12	364

Table 1: Total gifts received as of each of the 12 days.

Complexity of input material (source code)

For workers wishing to prepare documents, a primary concern is to minimize the effort to get to a finished product. Thus the complexity of the text material they must enter is important.

\LaTeX , itself an attempt to streamline the use of Donald Knuth’s \TeX (Knuth (1984)), is now forty years old. It appeared at the juncture where traditional typesetting by professional workers using manual metal or photographic processes were giving way to computer-based typographic systems that end-users could employ themselves. As such, \TeX and subsequently \LaTeX have a lot of detail that the user must supply, even as Knuth had made huge strides in automating the typesetting of mathematical and similar material. Knuth, starting in 1977, needed to support \TeX with appropriate computer-defined fonts, as detailed in Knuth (1986).

\LaTeX accomplishes a lot of the tasks of formatting parts of documents, in particular the mathematical or other scientific material. Moreover, for tasks such as special headers or footers, or equation labelling, there are add-in packages. Such packages are declared in a preamble to the actual document material. Indeed, the structure of a \LaTeX file is of the form

```
\documentclass[a4paper]{article}
various \usepackage lines, along with some definitions of shortcut codes and similar definitions,
page specifications, etc.
\begin{document}
the user's essential material
\end{document}
```

In four decades of open source contributions, and with the personal eccentricities of many developers, there are frequently multiple packages purporting to offer similar capabilities. Unfortunately, it can be difficult to take features from more than one package. Our experience is that the packages appear to “fight” with each other, though we do not have a clear example where we can localize the trouble. Indeed, the user-contributed nature of \TeX packages does not lend itself to extensive documentation, in particular, documentation of limitations or edge-effects of the macro code. Thus users generally use an “error and retrial” approach until a satisfactory output is achieved. If other users follow our practise, they then archive the

example for future use in a copy and edit fashion. Working examples are, we believe, critical to effective advanced use of L^AT_EX.

Moreover, a central theme of our own L^AT_EXuse, illustrated by a biography published on archive.org and a dozen literary books on obooko.com, is a standardized template for the works. This means that an existing book is retitled, the material between the `\begin{document}` and `\end{document}` is deleted, and the preamble material is edited to adapt to the new work.

One of us contributes regularly to a weekly activity of the Stittsville Creative Writing Group, where writers are invited to read a short contribution. For the collections *Different Perspectives* (<https://www.obooko.com/free-short-story-collections/different-perspectives>) and *The Coffee Klatch Curmudgeon* (<https://www.obooko.com/free-short-story-collections/coffee-klatch-curmudgeon-nash>), the L^AT_EXdirective `\input{}` is used to incorporate a weekly text. The weekly contributions are almost entirely plain-text content, prefaced by a line with the title

```
\chapter*{Title of contribution}
```

and terminated by

```
\bigskip
\small
\noindent
J C Nash \copyright 2018-08-23
\normalfont
```

Given that this material is not mathematical, the only other L^AT_EXtag used in many contributions is `\dialog{}` which is user-defined and can be dropped into the document source code with a custom key-stroke in *TexMaker* (<https://www.xm1math.net/texmaker/>) or *TexStudio* software that is used to facilitate the entry and processing of documents. The `\dialog{}` tag is intended to supply the appropriate opening and closing quotation marks, which may be different for different targets for the resulting output document.

While we have described the use of sub-documents here for L^AT_EX, all the tools we discuss allow some form of inclusion of pieces of the content. Where sub-documents more or less stand alone they are very helpful in keeping a clean working structure. For L^AT_EX, this is especially helpful where the preamble is both detailed and demanding to establish and verify.

All of the systems discussed except *Typst* use L^AT_EXmarkup for entering mathematical material, though *AsciiDoc* requires what we consider to be a rather messy invocation of the L^AT_EXcode. Errors are avoided when using more than one system if the mathematical input is the same. There are helpful compact references to the L^AT_EXmath markup codes such as <https://tug.ctan.org/info/undergradmath/undergradmath.pdf>.

Rendering to output formats

When users want their material in display or printed form, the markup must be rendered. Generally this is carried out by command line programs, and editing and processing tools that provide a graphical user interface (GUI) typically invoke those programs, following up by calling display programs. Moreover, the interfaces quite often are able to handle errors by pointing to the (estimated) location of bugs in source material by highlighting material within the GUI editing module. For users, the quality of such interfaces can be measured in the convenience of the edit / render / debug cycle.

Typst has a web-based GUI, of which a version is free to use, though the command line “compiler” can be freely downloaded. The compiler is very fast to render documents, but the web-based interface may reduce the overall efficiency of use, especially if there are network issues. Moreover, the no-charge account has some limitations, though we have not made more than a cursory investigation.

Translation of input source between systems

We found that much of the material in our example could be simply copied and edited between our trial example documents. We judge the changes needed to be largely cosmetic, though they are tedious to carry out. It is likely some fairly simple programs or else AI tools could do this.

We note that *Rmarkdown* and *Quarto* offer to output intermediate L^AT_EXsource of documents. This allows for tweaking of formatting for specialized needs. Moreover, L^AT_EXblocks can generally be included inline in those processors to incorporate special parts of documents.

Typst offers input conversion of L^AT_EXsource via *MiTeX* (<https://typst.app/universe/package/mitex/>). We have not tried this facility.

Dynamic documents

Over the past decade and a half, it has become important to be able to update documents as new data becomes available, or to be able to demonstrate the input and output of a computer program along with the code. Both *Rmarkdown* and *Quarto* allow for the inclusion of code chunks with controls on whether the source and/or output is displayed. This is not unique to these systems, but *Typst* uses a special scripting approach, and *Asciidoc* uses ‘sys’ macro calls.

Examples of special details

We consider two particular tweaks to documents as examples of capabilities.

Non-standard equation labels

Consider that we may want to identify a particular equation, e.g., the formula for the sum of an arithmetic progression in our Partridges example, by a special label. Moreover, while the “numbering” is generally on the right hand side of a page, let us ask for it to be on the left.

We have been able to satisfy this demand to varying degrees in all the systems.

Customised single copy footer

When one of us was teaching in the Telfer School of the University of Ottawa, it was useful to be able to produce examination papers customized by date and the name of each student. Students were informed that each examination was unique, which reduced the value of trying to peek during the examination and saved the professor invigilation effort. (It also avoided the nuisance of an unidentified paper, and provided a default list of attendees through the verification of identity as papers were given out at the start of the examination.)

A fairly simple program (in BASIC) allowed Postscript to be modified to do this customization in the footer of each page. In 2010, modifications in a segment of the L^AT_EX preamble of a biography allowed each recipient to receive their own identified PDF copy. In all likelihood this identification could be removed from the PDF file with relatively simple programming.

We have not pursued how this might be carried out in other than L^AT_EX.

Observations and Recommendations

We note that L^AT_EX has had a remarkably good run as a tool for mathematical and scientific documents, as well as serving very well for general documents where the style and structure is relatively stable, as in the literary books one of us has written. It is still effective and efficient for such tasks. Moreover, it is used for the mathematical formatting within some of the newer typesetting systems.

L^AT_EX demands, however, a very heavy learning cost. Deviations from standard layout usually require a lot of investigation and experimentation to get a satisfactory solution to specialized needs. The system is such that one can always find a way to get to the desired result, but there is always a heavy time and effort cost to do so.

AsciiDoc possibly offers one of the nicer platforms for general documents, but our experience with mathematical material was less than comfortable.

Rmarkdown has proved remarkably capable. With the *Bookdown* (Xie (2016)) package it can be and has been used to prepare large documents.

Quarto

Both *Rmarkdown* and *Quarto* offer active inline code chunks in a variety of programming languages. We feel this approach is more attractive than that of *Typst* or *AsciiDoc*, but that could reflect our experience.

Typst offers quite simple syntax and fast rendering. Its math syntax will be a nuisance to those who already know that for LATEX, but for those starting from scratch, it is a worthwhile option unless one desires to include inline active code chunks.

Recommendations

If users have a well-established and efficient workflow where their efforts can focus on content rather than formatting, there is no strong argument for change. One of us has no intention of moving on from LATEX for literary work, since it has proved so efficient, even though the base documents are arcaneously detailed.

Similarly, if workers have a body of work in *Rmarkdown*, it is probably not worth moving to *Quarto* if there might be any nuisance glitches due to the content or YAML header differences.

If no active code chunks are needed, and users have not got any history with other tools, then *Typst* could serve well. Similarly, if there is a limited need for math text, *AsciiDoc* is a contender in such cases.

References

- “AsciiDoc.” 2013. <http://www.linuxlinks.com/article/20130921024111314/AsciiDoc.html>.
- Knuth, Donald E. 1984. *The TeXbook*. Addison-Wesley.
- . 1986. *The METAFONTbook*. Addison-Wesley.
- Lamport, L. 1986. *LATEX: A Document Preparation System*. Addison-Wesley Publishing Company. <https://books.google.ca/books?id=IgJUAAAAMAAJ>.
- Machlis, Sharon. 2022. “What is Quarto.” *Infoworld*, July.
- Mädje, Laurenz. 2022. “A Programmable Markup Language for Typesetting.” PhD thesis, 2022.
- “The Twelve Days of Christmas (Song).” 2025. *Wikipedia*. [https://en.wikipedia.org/w/index.php?title=The_Twelve_Days_of_Christmas_\(song\)&oldid=1326578965](https://en.wikipedia.org/w/index.php?title=The_Twelve_Days_of_Christmas_(song)&oldid=1326578965).
- Xie, Yihui. 2016. *bookdown: Authoring Books and Technical Documents with R Markdown*. The R Series. Boca Raton, Florida: Chapman & Hall/CRC.
- Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. <https://bookdown.org/yihui/rmarkdown>.