Using *knitr* and LaTeX for literate laboratory notes

Boris Veytsman

1 Introduction

Many years ago I worked in a lab that hired a new student. His assignment seemed to be easy: to reproduce the results of another student and to expand on them. The work in question included mathematical modeling, computer simulations, and so on. To his dismay, the new student found out that the programs used undocumented libraries, the scripts had incomprehensible options, and the models had unstated assumptions. Deciphering all this turned out to be very difficult. While the original author was willing to help, he could not do much: the author started to forget the details of his research soon after graduation.

This experience had a profound influence on me. The question of whether I would be able to understand my own research in a decade or two became an obsession. Being a scientist, I asked myself how the problem was solved elsewhere. In experimental and applied sciences the research may cost millions of dollars. To preserve it, the researchers are required to keep detailed logs of their activity in the laboratory notes. The notes include the details of the experiments and their results, and also the hypotheses tested. There are courses [14] and books [6] about laboratory notes. It is fascinating to study laboratory notes of great scientists, for example, Linus Pauling. Pauling's notes comprise 46 notebooks spanning from 1922 to 1992, digitized by Special Collections & Archives of the Oregon State University [13]. His beautiful notes have a definite aesthetic value.

I would argue that the concept behind the practice of laboratory note keeping is somewhat akin to the concept of literate programming [8]. Knuth understood that code is just part of a programmer's output. The programmer's thoughts about these programs are even more important. Similarly, an important insight for science is that papers, preprints, presentations are not the research, but "an advertisement of the research" [15]. We must preserve the research itself [11, 15]. Laboratory notes are the means for this preservation.

Classic laboratory notes are physical notebooks like those of Linus Pauling. Unfortunately, this format has a number of flaws:

1. Physical notes are not searchable. While it is recommended to add a table of contents to a notebook [6, 14], it is time-consuming to keep it current, and has only limited value for a search.
2. Copying from physical notes is not easy. This is especially frustrating with code: retyping from the printouts pasted to the notebook pages is time-consuming and error-prone.
3. Physical notebooks are bulky.

Electronic notebooks may be searchable, allow easy copying and pasting, can be stored infinitely (if the proper backups are kept), and take miniscule space. On the other hand, physical notebooks are versatile, and one can write and doodle in them very quickly. It is difficult to match their convenience for the laboratory record keeping.

2 What should electronic laboratory notes store?

Electronic laboratory notes should allow the user to easily store a number of disparate items. As a theoretician, I put in my notes:

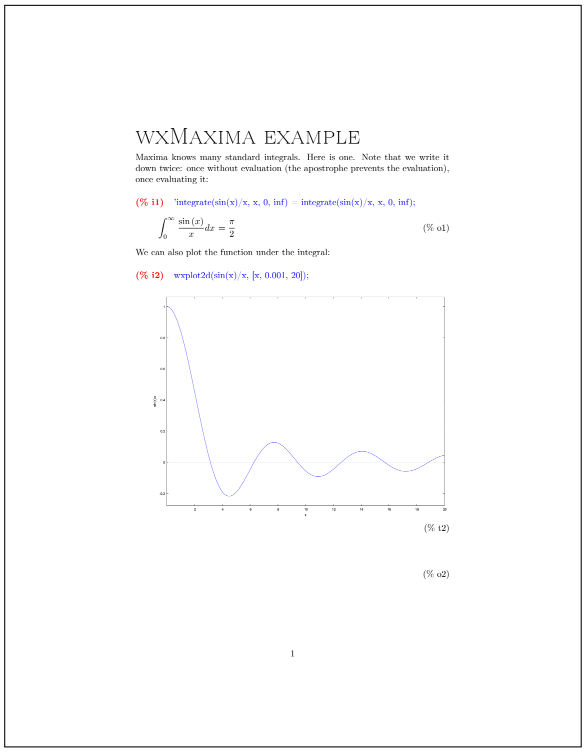
1. Prose. Sometimes short texts, sometimes longer paragraphs.
2. Equations, both inline and displayed.
3. Code snippets of various length.
4. Tables, sometimes produced automatically by code.
5. Plots, often produced automatically by code.
6. Sketches, doodles, diagrams, etc.
7. Bibliographies.

The standard LaTeX features like automatic numbering of objects, font changes, etc., may help to make the notes more readable and expressive.

3 Examples of notebook interfaces

Many commercial and free programs have so-called "notebook interfaces" for exploratory studies. These interfaces try to solve the same problem as laboratory notebooks: documentation of research. In this section we discuss two free programs. The first, *wxMaxima* [20], is a document-based interface for the famous computer algebra system *Maxima* [10]. The second, *Jupyter notebooks* [7], is intended "to support interactive data science and scientific computing". Jupyter notebooks were initially developed for Python programming, but have been extended to more than 40 computer languages.

Both these solutions are based on similar ideas, which are also used in many other notebook interfaces. They have a linear sequence of "cells" of different kinds. A text cell contains documentation. A code cell contains a program snippet. This cell could be "run": the program snippet is executed, and an output cell is added to the notebook. There are other types of cell to introduce metadata, section



WXMAXIMA EXAMPLE

Maxima knows many standard integrals. Here is one. Note that we write it down twice: once without evaluation (the apostrophe prevents the evaluation), once evaluating it:

(% i1) 'integrate(sin(x)/x, x, 0, inf) = integrate(sin(x)/x, x, 0, inf);

$$\int_0^\infty \frac{\sin(x)}{x} dx = \frac{\pi}{2} \qquad (\% \text{ o1})$$

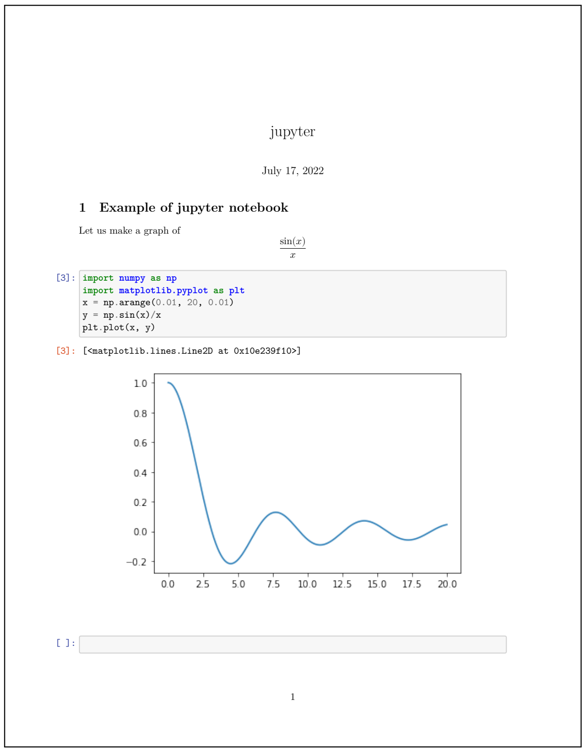
We can also plot the function under the integral:

(% i2) wxplot2d(sin(x)/x, [x, 0.001, 20]);

(% t2)

(% o2)

1



jupyter

July 17, 2022

1 Example of jupyter notebook

Let us make a graph of

$$\frac{\sin(x)}{x}$$

```
[3]: import numpy as np
     import matplotlib.pyplot as plt
     x = np.arange(0.01, 20, 0.01)
     y = np.sin(x)/x
     plt.plot(x, y)
```

```
[3]: [<matplotlib.lines.Line2D at 0x10e239f10>]
```

```
[ ]:
```

1

Figure 1: Examples of the output of wxMaxima and Jupyter notebooks

headers, etc. The text cells in Jupyter notebooks use the Markdown language [4], including LaTeX math syntax. These programs can export a document with the record of the session in either PDF or TeX format (Figure 1).

These notebooks are useful, and they are much better than no documentation at all. However, they do not satisfy many of the requirements stated in Section 2. We cannot easily number and reference objects. The output cells are not typically rendered as Markdown code, so we cannot typeset tables created by the code.

There are two reasons for these deficiencies. First, Markdown is not as expressible as LaTeX. There are extensions like *bookdown* [22] which alleviate this problem. However, as far as I know, most notebook interfaces do not support these extensions. Also, Markdown extensions make the language more LaTeX-like, which questions the whole premise of a simple typesetting language. Some readers may recall the famous phrase by Henry Spencer about the people who are condemned to reinvent Unix [19].

The second reason is more deep. The notebook interfaces are primarily records of the interaction between the user and the computer. The ideas of literate science, like the ideas of literate programming, suggest the centrality of the interaction between the user and other humans. Notebooks are basically code with text inserts. A literate interface should be the opposite: text with code inserts. This approach is discussed in the next session.

4 knitr-based notebooks

It is interesting that most notebook interfaces use TeX as the back end typesetting engine, even when Markdown is the front end. This leads to the idea of LaTeX as the laboratory notebook language. A set of TeX files is easily searchable with standard utilities such as `grep` and `find`, while PDF output provides readable documents. This approach satisfies almost all requirements listed in Section 2, with one exception: we want to add both snippets of programs *and* their output as tables and plots. While adding program code can be achieved within LaTeX (using, for example, the *listings* package [5]), the automatic addition of its output requires other means. The *knitr* package [21], based on the ideas of *Sweave* [9], can be used to create literate science [18].

A document in this case is a LaTeX file (with extension `.rnw`) that contains "chunks" of code. Initially, only R code was supported by knitr. Now knitr, like Jupyter notebooks, has been extended to other program languages, including Python. When processed by knitr, the chunks are typeset and their output is added to the document.

```
<<plot, dev='tikz', message=F>>=
library(tidyverse)
library(ggthemes)
theme_set(theme_bw())
data <-
    tibble(x=seq(0.01, 20, by=0.01)) %>%
    mutate(y=sin(x)/x)
ggplot(data) + geom_line(aes(x,y)) +
    xlab("$x$") +
    ylab("$\\sin(x)/x$")
@
```

Figure 2: knitr chunk (in R) for plotting $\sin(x)/x$

For example, consider the chunk in Figure 2. It programs a plot. When processed by knitr, both the typeset code and the plot are included in the TEX file (Figure 3). There are options to suppress the typeset code, change the graphics format, etc. [21]. For example, the chunk in Figure 2 uses the *tikz* format, so the plot has math typeset by TEX.

An interesting feature of *knitr* is the ability to re-render the output in TEX. This feature can be used to automatically produce tables. Consider, for example, the extrema of the function $f(x) = \sin(x)/x$. We can calculate them by solving numerically the equation $x\cos(x) - \sin(x) = 0$, obtained by differentiating $f(x)$. The R program in Figure 4 calculates the first six extrema at $x \geq 0$. It outputs six lines (for example: `5 & 14.07 & 0.07`). To typeset the table we output these lines as raw TEX code (with the chunk options `result='asis', echo=F`), and wrap it in a `tabular` environment. The result is shown in Table 1.

5 Problems with the knitr-based solution

The solution based on knitr is powerful. However, it has its own problems.

The first set of problems is related to the deficiencies of PDF format. This format is static by design. Sometimes we want to include movies, animation or interactive plots. It is possible to do

Number	x	$f(x)$
1	0	1
2	4.49	−0.22
3	7.73	0.13
4	10.9	−0.09
5	14.07	0.07
6	17.22	−0.06

Table 1: First six extrema of $f(x) = \sin(x)/x$ (see the code in Figure 4)

```
library(tidyverse)
library(ggthemes)
theme_set(theme_bw())
data <-
    tibble(x=seq(0.01, 20, by=0.01)) %>%
    mutate(y=sin(x)/x)
ggplot(data) + geom_line(aes(x,y)) +
    xlab("$x$") + ylab("$\\sin(x)/x$")
```

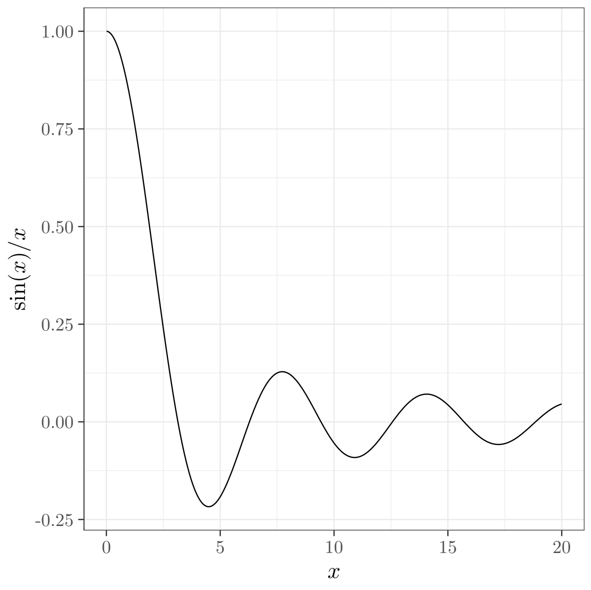


Figure 3: The typeset code (grayscaled for print) and plot produced by the chunk on Figure 2

```
find_extremum <- function(number) {
    result <-
        uniroot(
            function(x) {
                x*cos(x) - sin(x)
            },
            c((number-1)*pi,
              number*pi))
    x <- result$root
    f <- ifelse(x==0, 1,
                sin(x)/x)
    cat(number, "&", round(x, 2),
        "&", round(f, 2),
        "\\\\\n")

}
walk(1:6, find_extremum)
```

Figure 4: Code for calculation of extrema of $\sin(x)/x$. The results are shown in Table 1.

this using packages like *media9* [3], *animate* [2], and *FigPut* [1]. However, the recent debacle of Adobe Flash [12] makes one wary of PDF extensions.

Another problem is related to the speed of writing laboratory notes. I personally type prose *and* equations in LaTeX with the same speed I produce them. The same can be said about programming. However, doodling with a pen and paper is substantially faster than writing code in PSTricks [17] or Ti*k*Z [16]. Thus sketching may require different solutions: from scanning handwritten images to the use of special programs for fast doodling.

6 Conclusions

LaTeX allows the production of detailed laboratory notebooks, that can be easily read, searched and indexed. The addition of knitr helps to integrate the notebooks with the inclusion of typeset code and its output, such as plots, tables, etc.

This has been my preferred format of laboratory notebooks for several decades. It is quite versatile, reasonably fast and provides notes of archival quality.

References

[1] R. Fairman. *FigPut. Interactive Figures for LaTeX*, 2022. `ctan.org/pkg/figput`

[2] A. Grahn. *The animate package*, 2022. `ctan.org/pkg/animate`

[3] A. Grahn. *The media9 package, v1.24*, 2022. `ctan.org/pkg/media9`

[4] J. Gruber. *Markdown*, 2004. `daringfireball.net/projects/markdown/`

[5] C. Heinz, B. Noses, J. Hoffmann. *The Listings Package*, 2020. `ctan.org/pkg/listings`

[6] H.M. Kanare. *Writing the laboratory notebook.* American Chemical Society, Washington, D.C, 1985.

[7] T. Kluyver, B. Ragan-Kelley, et al. Jupyter notebooks—a publishing format for reproducible computational workflows. In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides, B. Schmidt, eds., pp. 87–90. IOS Press, 2016.

[8] D.E. Knuth. *Literate Programming.* No. 27 in CSLI Lecture Notes. Stanford, California, 1992.

[9] F. Leisch, R Core Team. *Sweave User Manual*, 2022. `stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf`

[10] Maxima. A computer algebra system, 2022. `maxima.sourceforge.io/`

[11] J.P. Mesirov. Accessible reproducible research. *Science* 327(5964):415–416, 2010. `10.1126/science.1179653`

[12] R.C. Moss. The rise and fall of Adobe Flash, 2020. `arstechnica.com/information-technology/2020/07/the-rise-and-fall-of-adobe-flash/`

[13] L. Pauling. Research notebooks, 1922–1972. Special Collections & Archives Research Center, Oregon State University Libraries. `scarc.library.oregonstate.edu/coll/pauling/rnb/`

[14] P. Ryan. *Keeping a Lab Notebook.* National Institutes of Health, Office of Intramural Training and Education, 2012. `www.training.nih.gov/assets/Lab_Notebook_508_(new).pdf`

[15] M. Schwab, N. Karrenbach, J. Claerbout. Making scientific computations reproducible. *Computing in Science & Engineering* 2(6):61–67, 2000. `10.1109/5992.881708`

[16] T. Tantau. *The TikZ and PGF Packages*, 2021. `ctan.org/pkg/pgf`

[17] T. Van Zandt, R. Niepraschk, H. Voß. *PSTricks. PostScript macros for Generic TeX*, 2007. `ctan.org/pkg/pstricks-base`

[18] B. Veytsman. Book review: Dynamic Documents with *R* and *knitr*, by Yihui Xie. *TUGboat* 35(1):115–119, 2014. `tug.org/TUGboat/tb35-1/tb109reviews-xie.pdf`

[19] Wikipedia contributors. Henry Spencer — Wikipedia, the free encyclopedia, 2022. `en.wikipedia.org/w/index.php?title=Henry_Spencer&oldid=1093428638`

[20] wxMaxima, 2022. `wxmaxima-developers.github.io/wxmaxima/`

[21] Y. Xie. *Dynamic Documents with R and knitr.* Chapman and Hall/CRC, Boca Raton; London; New York, second ed., 2015.

[22] Y. Xie. *bookdown: Authoring Books and Technical Documents with R Markdown.* Chapman and Hall/CRC, Boca Raton, Florida, 2016. ISBN 978-1138700109. `bookdown.org/yihui/bookdown`

◇ Boris Veytsman
Systems Biology School
George Mason University
Fairfax, VA 22030
`borisv (at) lk dot net`
`http://borisv.lk.net`