# Efficient Use of Julia

# Efficient Use of Julia

Xijiang Yu

A preview version
Norwegian University of Life Sciences
Ås, Norway

This book was set in TEX Gyre Pagella by the authors in LATEX.
Printed and bound in Norway.

Library of Congress Cataloging-in-Publication Data is available.

ISBN:

10 9 8 7 6 5 4 3 2 1

*This book is dedicated to my colleagues in NMBU, who have patiently listened my, sometime maybe too complicated, talks.*

# *Contents*

PART I SOME USEFUL TOOLS FOR JULIA

APPENDICES

# *Preface*

This book used a template from `https://github.com/sisl/tufte_algorithms_book`. It initially ran on a Ubuntu system. I setup a dedicated virtual Ubuntu machine for this template. Besides, I also installed below packages:

git, emacs, vim, texlive-extra-utils, texlive-latex-extra, python3-pip, pdf2svg, texlive-realscripts, texlive-xetex, texlive-fonts-extra, texlive-science, biblatex, texlive-latex-extra, rubber, texlive-bibtex-extra, texlive-pictures

Python package pygments was installed with pip. Package pygments_julia was installed with command

pip install –upgrade git+https://github.com/sisl/pygments-julia#egg=pygments_julia

This book template provides a starting point upon which authors may freely build to generate their own textbook entirely in LaTeX. We used this setup for Algorithms for Optimization, and have continued to refine it for a new textbook on decision making under uncertainty. The template allows for the direct compilation of a print-ready PDF, including support for figures, examples, and exercises.

This template is intended for textbook authors. Use of the template assumes prior expose to LaTeX. This template uses the `tufte-book` class to provide wide side margins.

Fundamental to our textbooks are the algorithms, which are all implemented in the Julia programming language. We have found the language to be ideal for specifying algorithms in human readable form. Algorithms can be typeset in algorithm blocks, tested, and used to generate figures. We use pygments and pythontex, including a custom pygments lexer and style.

I thank the template authors. Now I am starting to use it to write my own book, The efficient use of Julia (for animal breeders).

Xijiang Yu
Ås, Norway
Oct. 6, 2022

Ancillary material is available on the template's webpage:
`https://github.com/sisl/textbook_template`

# *Acknowledgments*

The authors wish to thank the many individuals who have provided valuable feedback on early drafts of our manuscript, including

# PART I:

# SOME USEFUL TOOLS FOR JULIA

# 1 *Introduction*

Welcome to the book template! This template is based on our previous book.[1]

## 1.1 *Using Julia*

In this section we provide some examples of how Julia algorithms can be typeset, tested, etc. We begin with a Julia algorithm, algorithm 1.1. We use our custom `algorithm` environment with a pythontex `juliaverbatim` environment inside to typeset the algorithm. A caption provides some additional information, and shows up in the margin.

```julia
function sample_function(x, a)
    if x > a
        return log(x)
    else
        return x + log(a) - a
    end
end
```

Algorithm 1.1. A sample function that takes in an evaluation scalar `x` and a scalar parameter `a`.

We can add a test for our algorithm in the source code. This test does not show up when you compile. You can run all tests by executing `jl runtests.jl`.

The code can be executed when creating figures. Figure 1.1 is a standard inline figure that executes Julia code using `PGFPlots.jl` to produce a TeXfile in the `fig` directory. This file is then compiled into the PDF.

[1] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.

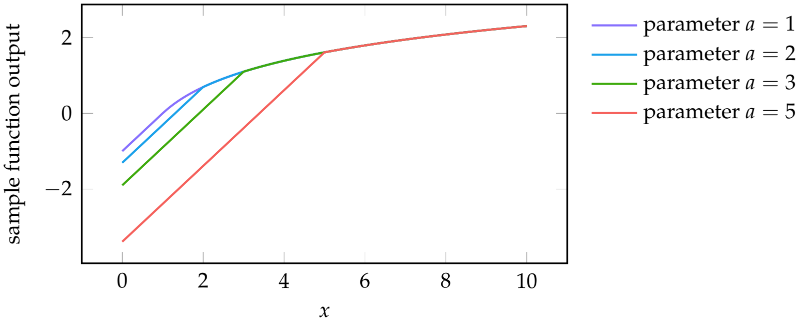

Figure 1.1. Curves obtained when using `sample_function` for several different values of `a`.

Of course, there is nothing stopping you from inserting tikzpictures directly, as shown in figure 1.2.

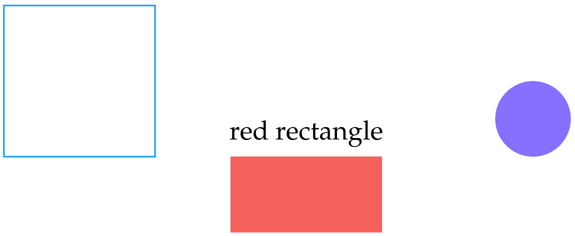

Figure 1.2. A figure made using a `tikzpicture` environment.

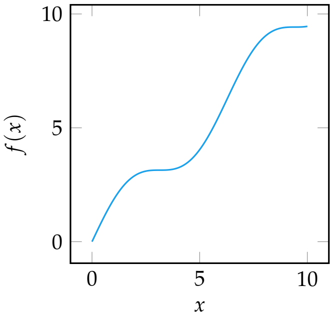

Figure 1.3. A `marginfigure` shows up in the margin.

Figures can also be placed in the margins, as with figure 1.4.

The `pythontex` package also supports the `juliaconsole` environment, as shown below. These environments act like the julia REPL, showing code as it is entered and executed.

```
julia> x = 5
5
julia> y = 7.5
7.5
julia> x^y
174692.81074217107
```

## 1.2 *Custom Environments*

We created two custom environments in addition to the `algorithm` environment when writing *Algorithms for Optimization*. The `example` environment creates an isolated section in a light-gray box with its own caption.

> An example of the `example` environment. Place anything you want here, including figures.

Example 1.1. An example `example`.

We also have `question` and `solution` environments, typically used at the end of each chapter. The solutions can automatically be placed at the end of the book by using `solutionappendix`.

**Exercise 1.1.** What is the meaning of life?

*Solution:* The meaning of life is 42.

## 1.3 *Math Notation*

*Algorithms for Optimization* uses lowercase characters with bold face for vectors, such as $\mathbf{x}$, and uppercase characters with bold face for matrices, such as $\mathbf{\Sigma}$. Scalars are lowercase with normal face, such as $\lambda$.

The contents of vectors and matrices can be typeset using `bmatrix`:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{1.1}$$

We often write out vectors horizontally in order to save space. In these cases, we use comma-separate entries:

$$\mathbf{x} = [x_1, x_2] \tag{1.2}$$

We construct an optimization problem as follows:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) \le \mathbf{0} \\ & h(\mathbf{x}) = \mathbf{0} \end{aligned} \tag{1.3}$$

See `tufte_algorithm_book.cls` for some additional math utilities, such as $\mathcal{N}(\mu, \nu)$ and $|\mathcal{A}|$.

## 1.4 Code Replacement

Sometimes the function you want typeset is almost, but not quite, the function that you want to also execute or test. For example, you may want to inject an extra line to store data into a global variable, or you may wish to replace a particular function with a specialized version.

For such cases we have a simple replacement feature built into the `pull_julia_code` script. The script searches for `"REPLACE \(.*\) \(.*\)"` in the lines between the beginning of the `algorithm` environment and the beginning of the `juliaverbatim` environment. It will then match the string between the first parentheses and replace it with the string between the second parentheses when exporting the code. The code block will still be typeset with exactly what is inside the environment — it is only the exported code that is changed.

This feature was intended to be used sparingly.

```julia
function replacement_example(x)
    return x^2
end
```

Algorithm 1.2. A sample typeset algorithm whose exported code differs from what is typeset. Here the `x^2` will be replaced with `sin(x)`.

## 1.5 Conclusion

That is more or less everything you need in order to get started writing your own textbook with the `tufte_algorithm_book` style. We hope you find it useful.

If you have any questions or find improvements, please create issues on our GitHub repo: `https://github.com/sisl/textbook_template`.

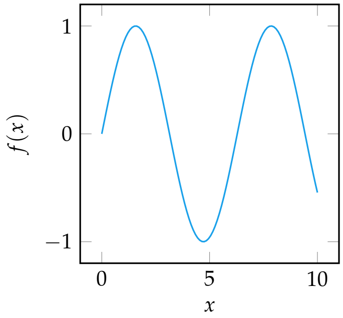

Figure 1.4. We find that when using algorithm 1.2, we actually get $f(x) = \sin(x)$ due to the replacement going on under the hood.

# APPENDICES

# A My Appendix Chapter

Oh yeah, and you can have chapters in the appendix.

# *References*

1. M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*. MIT Press, 2019.