

bookdown + rmdja による多様なファイル形式の 日本語技術文書の作成

Katagiri, Satoshi

2020-09-19

目次

第 I 部 イントロダクション	7
序文	9
本稿の目標	11
既存のフォーマットとの違い	11
昔話あるいは既存資料との違い	12
想定読者	14
R Markdown ユーザ向けのメリット	14
第 II 部 最低限のチュートリアル	17
第 1 章 準備	19
1.1 インストール	19
第 III 部 R Markdown と Bookdown の基本機能	21
この部の概要	23
第 2 章 静的なコンテンツの作成	25
2.1 Markdown の基本構文	25
2.1.1 インラインでの書式変更	25
2.1.2 ブロック要素	26
2.2 Markdown を使った図表の挿入	27
2.2.1 コメントアウト	29
2.3 数式	29
2.4 カスタムブロック	30
2.5 脚注	31
第 3 章 動的なコンテンツの作成	33
3.1 プログラムチャンク	33
3.2 プログラムで数式を生成する	35
3.3 プログラムを使った図の挿入	36

3.4	それ以外のグラフィックデバイス	37
3.5	TODO: 図のレイアウト設定	39
3.6	R プログラムを使った表の装飾	41
第 4 章	相互参照と引用	45
4.1	相互参照	45
4.1.1	図表や式へのアンカーリンク	45
4.1.2	表への相互参照	45
4.1.3	章への相互参照	46
4.1.4	特殊な相互参照	46
4.2	文献引用	47
第 5 章	簡単なレイアウト変更	49
5.1	HTML	49
5.1.1	フォント変更	49
5.2	PDF	49
5.2.1	フォント変更	49
5.2.2	文書クラスの変更	50
第 6 章	rmdja による文書作成支援機能	51
6.0.1	クリエイティブ・コモンズの表記	51
6.0.2	ルビ表記	51
第 IV 部	製本と多様な形式への対応	53
第 7 章	製本方法の詳細	55
7.1	ファイル構成	55
7.1.1	_output.yml	55
7.1.2	_bookdown.yml	56
第 8 章	出力形式による表現の限界	59
8.1	HTML と PDF で処理を場合分けする	59
8.2	絵文字の出力	60
8.3	画像の保存形式	60
8.4	デフォルトの保存形式	60
第 9 章	Web アプレットの挿入	61
9.0.1	TODO: plotly	61
9.0.2	TODO: shiny	61
第 10 章	製本した文書を配布する	63
10.1	Wep ページのホスティング	63
10.2	TODO: 入稿するには	64
10.2.1	トンボの表示	64

10.2.2 フォントの埋め込み	64
第 V 部 応用	65
第 11 章 TODO: 文芸作品の作成のために	67
第 VI 部 デバッグ	69
第 12 章 製本時のエラーへの対処	73
12.1 エラーがどのタイミングで発生したかを特定する	73
12.2 YAML フロントマターを確認する	73
12.3 PDF の生成に失敗する場合	75
付録 A デフォルト値の自動調整	77
A.1 デフォルトのフォント	77
A.2 チャンクのデフォルト設定	78
付録 B PDF の組版に関する細かい話	79
B.1 画像の配置	79
B.2 取り消し線	79
B.3 TODO: しかし英文で書きたい場合	80
付録 C jecon.bst の紹介	81
参考文献	83

第Ⅰ部

イントロダクション

序文



注意: 絶賛作りかけ

長大な技術文書や良質な技術文書を作成するには手間がかかる。しかし時間をかけていい文書になるわけではない。無駄な手間を省き、効率よく快適に文書を作成するべきである。

たとえばこういう経験はないだろうか。

- プログラムの解説のため、外部サービスでシンタックスハイライトしてもらったテキストをコピペーストで貼り付ける
- グラフや図解を専用アプリケーションで作成し貼り付ける。修正のたびに貼り付け直す
- 図表に言及する際に「図 1」「表 2」と番号をタイプし、参照先へハイパーリンクを指定する
- 本文中で引用した参考文献のリストを巻末にコピペーストし、過不足がないか目視で確認
- $\sum_{k=1}^K \int_0^\infty f_k(x) dx$ などといった複雑な数式はプレーンテキストや HTML では表現できないため、画像を生成して貼り付ける
- 冒頭にかっこいいエピグラフを掲載したいので、1 時間かけて特別に枠やフォントを作成した
- 市販のワードプロセッサで作成した文書を渡したら、レイアウトが崩れて読めないと言われた

本稿は文書作成者をこのような様々なブルシットから解放するのが目的である。

R Markdown (`rmarkdown`) は、R プログラムを埋め込んだ動的なドキュメントから pandoc を利用して PDF や HTML 形式の文書を作成するパッケージであり、数式、図表の挿入、シンタックスハイライトされたプログラムなどを簡単な記述で掲載できる。名前の通り、その基本構文は Markdown である。よって、Markdown と R の知識が最低限あれば (R プログラムが必要ないなら Markdown だけでも) 文書を作成することができる。

`bookdown` パッケージは `rmarkdown` をもとに、ページ数の多い文書を作成し、配布する

ための機能を拡張したものである。しかし、PDF の出力に関しては欧文を前提としたフォーマットを使用しているため、日本語の適切な表示（組版やフォントの埋め込みなど）のできる文書を作成するには高いハードルが存在した。

本稿では、`bookdown` で日本語文書を作成する際の設定を容易にしたパッケージ `rmdja` を利用した日本語技術文書の作成方法を解説する。

本稿の目標

既存のフォーマットとの違い

もちろん、同様のことは既存のソフトウェアやサービスでも可能である。

たとえばはてなブログ、Qiita、といった既存のブログサービスには、Mathjax による数式レンダリングやプログラムをシンタックスハイライトして表示する機能が最初から用意されているものもある。しかしながら現状では以下のようない制約がある。

- 独自規格の構文が使いづらい、一部本来と違う構文で数式を書く必要がある、ページ内リンクが使えない、など。
- テキストエディタでしか書けない
- 数十ページ相当のテキストを投稿しようとしただけでエラーが発生する。

また、 \LaTeX (シンプルなテキストエディタでも、Overleaf や LyX といった強力なエディタでも)を普段使っている人間にとっては以下のような利点がある。R Markdown はそもそも PDF 出力時は \LaTeX に依存しているため、主な違いは操作の簡略化にある。

- 外部プログラムで作成した画像や計算結果をコピペーストせずそのまま貼り付けられる
- \LaTeX とほぼ同じ構文で数式を記入できる
- 主な設定は既に定義済みあり、本文は簡易な Markdown で書くことができる、よって「TeX は複雑でわかりづらい、時代遅れのシステム」といった私怨混じりの批判を跳ね返せる

Word を普段使っている人間にとっては以下のような利点がある^{*1}。

- 数十、数百ページの文書を書いてもクラッシュすることがあまりない
- 輪郭のはっきりしたベクタ画像を簡単に貼り付けられる
- 図表の配置や相互参照を手動で書く必要がない
- 読み手の環境に依存してレイアウトが崩れにくい PDF ファイルを出力できる

さらに、作成した文書は PDF 形式で出力することはもちろん、HTML 形式で様々なサイトで掲載でき^{*2}たり、電子書籍ファイルとしても出力可能である。このような多様な出

^{*1} ただし筆者は数年来 Word を使っていなかったため、これらのいくつかは既に改善されているかもしれない。

^{*2} bookdown 同様に R Markdown で作成した文書をブログ風のフォーマットで出力する blogdown パッケージというものも存在する。

力形式への対応しているソフトウェアはあまり例を見ない。

bookdown はこのように便利で、公式ドキュメントがとても充実しているにも関わらず、日本語に適したレイアウトの設定の煩雑さからあまり普及していない^{*3}。

- “Dynamic Documents for R · rmarkdown”
- “bookdown demo”
- “bookdown: Authoring Books and Technical Documents with R Markdown”
- “R Markdown Definitive Guide”
- “R Markdown Cookbook”^{*4}

基本的なことがらの多くは上記を読めば分かるのでここでは基本機能をダイジェストで伝えた上で、これらの資料に書いてない応用技を紹介する。YAML のオプションの意味についてはソースコードにコメントを書いた。以下、単に、**BKD* と書けば “bookdown: Authoring Books and Technical Documents with R Markdown” (Xie, 2020) を、RDG と書けば “R Markdown: The Definitive GUiide” (Xie et al., 2018) を、RCB と書けば “R Markdown Cookbook” (Xie et al., 2020) を指すことにする。

さらに、Python の jupyter は Python のコードチャunkとその結果を簡単に表示できる文書作成ツールである。出力オプションの少なさ（たとえば長大なコードもそのまま掲載されてしまう）や、IDE として見ても機能が少ないとからあまり使い勝手がよくなかったが、最近登場した **Jupyter book** はドキュメント生成能力を強化している。しかし R Markdown/bookdown ほど PDF 形式のことは考慮していないように見える。

昔話あるいは既存資料との違い

日本語コミュニティにおいて bookdown は以前から言及されていた。例えば 2016 年の kazutan 氏のスライド『R で本を作りたい』があり、同じく bookdown 製のデモページ、『Bookdown を用いた図表番号の自動付与と参照のテスト』がある^{*5}。

それ以降も R Markdown に関する情報を発信する人はいたが、大きく話題になることが少なかった。ユーザはいるものの、もっぱら HTML への出力用途に使い、PDF や同時出力に挑戦する人間はほとんど見られなかった。普及していない理由は情報の絶対的な少なさ（そして古さ）にある。本稿の目標はこのような状況を改善することにある。

もしすでに R Markdown や Bookdown に触れて、ネット上の情報を元に文書を作った人に対しては、本稿を読むことで以下のようないい處があるかもしれない

- IPA フォント以外のフォントを指定する方法

^{*3} 前例として bookdown で作成した文書を技術書展で配布している人が書いたブログが存在する：
<https://teastat.blogspot.com/2019/01/bookdown.html>

^{*4} 2020/10/19 に書籍としても発売されるらしい。

^{*5} ソースはこちら：https://kazutan.github.io/bookdown_test/hoge.html

- 表のスタイル, 画像の埋め込み方, 見出しのスタイル, といった基本的なレイアウトを見やすく変更する方法

なぜかネット上のこの手の情報では IPA フォントを使いたがる例が多い。10 年前ならいざ知らず, もうほとんどの主要 OS では IPA フォントはプリインストールされてないでこだわる理由はあまりないと思うのだが, どうだろうか? ノスタルジーに浸るのはもちろん自由だ, むしろファイルを持ってさえいれば IPA モナーフォントを埋め込むことも可能だろう。しかしフリーフォントならむしろカバレッジに優れる Noto フォントが便利だし, あるいは游書体とかヒラギノとか各 OS の基本フォントを使いたいだろう。フォントにこだわる人間なら, 欧文と和文で異なるフォントを使う混植がしたい人もいるだろう。rmdja では, 既に公開した pdf_presentation_ja フォーマットと同様に, bookdown でも混植できる^{*6}。

実際に変えようとすると, それでも結構ややこしいことがわかる。例えば以下のエントリ

<https://notchained.hatenablog.com/entry/2018/08/12/140637>

1. R Markdown では通常, L^AT_EX は tinytex でインストールしたものが呼び出される
2. そして tinytex は pL^AT_EX や pBIBT_EX などを想定していない。
3. その範囲で実行するには, X_HL^AT_EX または LuaL^AT_EX が必要
4. しかし, デフォルトでは文書クラスが日本語文書向けでないので, レイアウトがあまりよろしくない。具体的には禁則処理がおかしいとか, 見出しが英文風とか。

PDF でも日本語を表示する最低限の設定は, YAML フロントマターだけで行える。これは Atsy 氏が『R Markdown + XeLaTeX で日本語含め好きなフォントを使って PDF を出力する』で紹介しているものとだいたい同じ方法になる。

```
output:
  pdf_document:
    latex_engine: xelatex
  header-includes: |
    \setCJKmonofont{Noto Sans Mono CJK JP}
    \setCJKsansfont{Noto Sans CJK JP}
  documentclass: bxjsreport
  mainfont: Noto Serif
  sansfont: Noto Serif
  monofont: Noto Mono
  CJKmainfont: Noto Serif CJK JP
```

このままでも, とりあえず文字化けすることなく日本語を表示できる。しかし文書として整ったものにするのは難しい

^{*6} ただし, 現時点では HTML に対してフォントを細かく設定する機能, および HTML と PDF のフォントを一致させる機能はない。これは技術的というよりライセンス的に制約が多いからだ。

このままでは参考文献リストの表示も不自然なままである。だがこれ以上のカスタマイズは、Atusy 氏がやっているようにテンプレートを修正する必要がある。

なるべく選択肢は広げておきたいが、なんでもありではかえって余計なことをしがちである。よって既に作成した `beamer` フォーマットと同様に、`XeLATEX` および `LuaLATEX` のみの対応を想定している。

想定読者

既に紹介したように、R を普段使わない人間でも `bookdown` で同人技術書を執筆したという事例もある。よって非 R ユーザにもある程度配慮して書いておくが、あまりに細かい R の仕様説明などは行わない。良い参考書が既に数多く存在するからだ。よって非 R ユーザは、本稿にある使用例のうち、自分の書きたいものに使えそうなものをつまみ食いして使用することになるだろう。本稿ではそのような使い方をするのに最低限必要なセットアップの知識のみ記載する。

R Markdown ユーザ向けのメリット

R Markdown を使ったことのある人ならわかるろうが、HTML を作るのに納得の行く `output` の指定をしていたらこうなった

```
bookdown::gitbook:
  split_by: chapter
  dev: png
  dev-args:
    res: 200
  css:
    - '../../.css/style.css'
    - '../../.css/toc.css'
  keep_md: true
  config:
    toc:
      collapse: none
      before: |
        <li><a href="/">Top</a></li>
      after: |
        <li><a href="https://bookdown.org" target="_blank">Published with bookdown</a></li>
  toolbar:
    position: fixed
  edit : null
  download:
```

```

- pdf
- epub
- mobi

fontsettings:
  theme: white
  family: serif

sharing:
  github: true
  twitter: true
  facebook: true
  all: ['linkedin', 'vk', 'weibo', 'instapaper']

```

しかし rmdja ではこうだ.

```
rmdja::gitbook_ja:
  keep_md: true
```

PDF の場合も紹介しよう.

```

bookdown::pdf_book:
  toc_depth: 3
  toc_appendix: true
  toc_bib: true
  latex_engine: xelatex
  keep_tex: true
  keep_md: true
  citation_package: natbib
  pandoc_args:
    - '--top-level-division=chapter'
    - '--extract-media'
    - '.'
  template: XXXXX.tex.template'
  dev: "cairo_pdf"
  out_width: "100%"
  out_height: "100%"
  quote_footer: ["\\VA{", "}{}"]
  extra_dependencies: gentombow

```

こうなる

```
rmdja::pdf_book_ja:  
  keep_tex: true  
  keep_md: true  
  tombow: true
```

さらにチャンクオプションや場合によっては.tex ファイルのテンプレートすら調整する必要もあった。これらは和文と欧文の組版の違いに由来するものである。これらの煩雑な設定を内蔵し、かつフォントの設定など環境に応じて変更する必要のあるものある程度自動的に設定するようにしている。

第Ⅱ部

最低限のチュートリアル

第 1 章

準備

1.1 インストール

文書を生成するのに必要なものをインストールする。

このドキュメントは `rmdja` パッケージに含まれている。よってまずはこれをダウンロードしてほしい。

ただし, `bookdown` ver. 0.2 時点ではソースファイルのフォルダ配置していに不具合があるため [`^bookdown-subdir-error`], もし 0.2 以前のバージョンを使用しているなら, 更新するか, `github` から開発版をインストールする。同様に, `rmarkdown` も CRAN ではなく開発版をインストールしたほうがよい。

```
01 remotes::install_github("rstudio/rmarkdown")
02 remotes::install_github("rstudio/bookdown")
```

次に, 最低限のファイルやパッケージで動くほうのデモ用ディレクトリをコピーする

```
01 file.copy(system.file("resources/examples/bookdown-minimal", package = "rmdja"), "./", recursive = T)
```

[1] TRUE

`bookdown` の文書生成は従来の R Markdown と違い, RStudio の `knit` ボタンではできない。代わりに, 以下の 2 通りの方法がある。

1. `bookdown::render_book('index.Rmd', format = "bookdown::gitbook")`
などを呼び出す
2. RStudio の Build ペーンを使う

前者は, 以下のように実行する。順に HTML, PDF, epub を出力している

```

01 bookdown::render_book("index.Rmd", "rmdja::gitbook_ja")
02 bookdown::render_book("index.Rmd", "rmdja::pdf_book_ja")
03 bookdown::render_book("index.Rmd", "bookdown::epub_book")

```

コピーしたディレクトリ bookdown-minimal を設定する (図 1.1, 1.2).

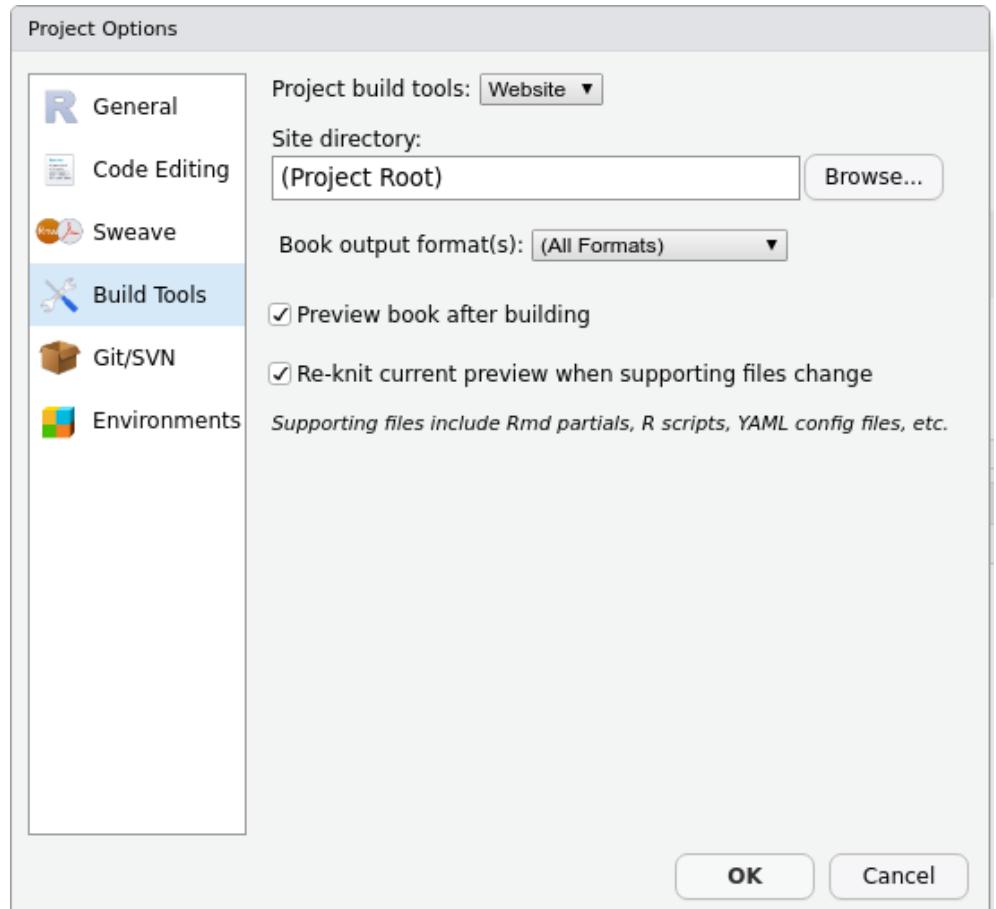


図1.1 Build ページの手動設定



図1.2 Build ページの手動設定

これで _book フォルダに出力がされる.

第 III 部

R Markdown と Bookdown の基 本機能

この部の概要

ここではまず, R Markdown の基本的な機能を紹介する. つまり bookdown 特有のものではなく, R Markdown 全般で使用できる機能も含めて紹介する. これ以降は自己言及的な説明が多いいため, この文書を生成しているソースコードと比較しながら確認することをおすすめする. ここで紹介する機能は BKD, RDG, RCB での記述に基づく. これら 3 つのドキュメントを読めば, ほとんどのことは可能になる — rmdja を作る理由になった LaTeX テンプレートの修正以外は — のだが, 本稿の重要な目的の 1 つは複数のファイル形式を両立することであるので, それができない書き方には触れないし, 技術文書の作成にあまり使わないような機能の動作確認はおこなわず, 技術文書作成で頻繁に使われ, 便利と思える機能のみ紹介する.

どちらにしろそのうちこれらを翻訳してくれる人が現れることだろう...たぶん.

第 2 章

静的なコンテンツの作成

日本語で書かれた資料でごく基本的なことについて、『R Markdown 入門』で一通り紹介されている。やや応用的なことも『R Markdown ユーザーのための Pandoc's Markdown』に書かれている。

また、既に作成している beamer の用例ファイルもどのようなことができるかの参考になるだろう。ただしこちらは PDF のみの出力を前提としているため、一部の機能は HTML で使うことができない。

まずは、単なるマークアップ、つまりプログラミングの複雑な処理を考えなくても良いタイプの構文を紹介する。それらの多くは単なる Markdown のものと同じである。

2.1 Markdown の基本構文

一応基本の Markdown の構文も挙げておく。詳細は (ref:BKDB)“Ch. 2.2 Markdown Syntax”を参照。

2.1.1 インラインでの書式変更

テキストの一部のみ書式を変える

アンダースコアで強調（イタリック）

`_underscore_`

underscore

**** 2 つで太字強調**

`** 太字強調 **`

太字強調

等幅フォント

`'bookdown' と 'rmdja'`

`bookdown` と `rmdja`

本文中に入力した URL は自動判別され、ハイパーリンクが付けられる。また、[テキスト](URL) という書式で、テキストに対してハイパーリンクを付けることができる。

URL は自動判別される: https://github.com/Gedevan-Aleksizde/my_latex_templates/tree/master/rmdja

[`rmdja` の github リポジトリ] (https://github.com/Gedevan-Aleksizde/my_latex_templates/tree/master/rmdja)

URL は自動判別される: https://github.com/Gedevan-Aleksizde/my_latex_templates/tree/master/rmdja

`rmdja` の github リポジトリ

2.1.2 ブロック要素

以降は行内では使えず、適切に表示するには前後に改行を挟む必要のあるタイプの構文である。

まず、引用ブロックを使えばかっこいいエピグラフを書き放題である。

```
> Нужны новые формы. Новые формы нужны, а если их нет, то лучше
>
>新しいフォーマットが必要なんですよ。新しいフォーマットが。それがないというなら、いっそ何
>
> `\\r tufte::quote_footer('--- A. チェーホフ『かもめ』')`
```

Нужны новые формы。Новые формы нужны, а если их нет, то лучше ничего не нужно。

新しいフォーマットが必要なんですよ。新しいフォーマットが。それがないというなら、いっそ何もないほうがいい。

— A. チェーホフ『かもめ』

`rmdja` では、HTML と PDF 両方で同様のデザインの枠で表示するようにしている。

Markdown では # は見出しを意味するが、`bookdown` にはさらにオプションが用意されている。

見出し名 {-} で、セクション番号のつかない見出しを用意できる。序文、章末の参考文

献、付録のセクションに使えるだろう。さらに、bookdown では # (PART) 見出し名で「部」の見出しを作ることができる。この見出しあはセクションの合間に挟まるが、選択することはできない。文書が長くなったときに、より大きな区切りを付けるのに役に立つだろう。さらに、# (APPENDIX) 見出し名 {-} で、以降の見出しの頭に「補遺 A, B, C, ...」と付番できる。

箇条書きは以下のように書ける。

```
* iris setosa
* iris versicolor
* iris virginica
```

- iris setosa
- iris versicolor
- iris virginica

1. iris setosa
2. iris versicolor
3. iris virginica

1. iris setosa
2. iris versicolor
3. iris virginica

インデントを使えばネストできる。

- 課長
 - 課長補佐
 - * 課長補佐代理
 - 課長補佐代理心得

2.2 Markdown を使った図表の挿入

markdown は表を記入することもできる。

Table: Markdown 記法の表

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2

4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

表2.1: Markdown 記法の表

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

画像ファイルも貼り付けられる。



図2.1 Johannes Gutenberg

しかし、キャプションを付けたり、表示位置やサイズを細かく調整したりするためには、後述するように **R** プログラムを経由して出力したほうが良い。

TODO: md 記法で画像貼り付けたときのサイズ統一

2.2.1 コメントアウト

HTML 式の `<!-- -->` でコメントアウトできる。コメントアウトされた箇所は生成ファイルでもコメントアウトされるのではなく、そもそも出力されなくなる。

2.3 数式

LaTeX 記法で数式を記述できる。HTML ならば Mathjax によってレンダリングされる。数式の記述ルールは少々ややこしい。これは現在の pandoc の仕様で HTML および LaTeX の規格で矛盾なく出力するためやむをえない措置である。

1. 改行をしない行内数式は \$ で囲む、または \(\,(,\,) で囲む。
2. 改行を伴う数式ブロックは \$\$ で囲む、または \[\,[,\,]\] で囲む。
3. align, equation 環境等を使う場合は、上記の記号を使わず、直接 LaTeX コマンド `\begin{align} ... \end{align}` を打ち込む。

```
\@ref(eq:binom) は二項分布の確率関数である
\begin{align}
f(k) &= {n \choose k} p^k (1-p)^{n-k} (\#eq:binom)
\end{align}
```

その出力は、以下のようになる。

(2.1) は二項分布の確率関数である

$$f(k) = {n \choose k} p^k (1-p)^{n-k} \quad (2.1)$$

Bookdown では従来の **R Markdown** できなかった数式への付番と、本文中での参照アンカーリンクの自動作成が可能となっている（詳細は [4.1 章](#)）。LaTeX にすでに慣れている読者に注意が必要だが、Bookdown 特有の制約として、付番したい場合は `\label{ID}` ではなく (`\#eq:ID`) を使う。また、PDF (LaTeX) と HTML (Mathjax) の仕様には

1. PDF では `align` は常に数式が付番され、`align*` 等はどうやっても付番されない
2. HTML では `align` でも `align*` であってもラベルを書かなければ付番されず、書けば付番される。

という違いがある。両者で同じ表示にこだわるのなら、付番を取り消す `\notag` を多用することになるだろう。

さらに、bookdown の機能として、LaTeX の「定理」「定義」「証明」などの環境に対応するものが提供されている（参考：BKD Ch. 2.2 [Markdown extensions by](#)

`bookdown`). これらの相互参照も可能である.

例: 以下に補題 2.1, 定理 2.1 を示す.

補題 2.1 (ボレル-カントリの補題). E_1, E_2, \dots をある確率空間の事象とする. これらの事象の確率の和が有限であるなら, それらが無限に多く起こる確率はゼロである. つまり,

$$\sum_{n=1}^{\infty} P(X_n) < \infty \Rightarrow P\left(\lim_{n \rightarrow \infty} \sup X_n\right) = 0,$$

$$\lim_{n \rightarrow \infty} \sup X_n = \bigcap_{n=1}^{\infty} \bigcup_{k \leq n} E_k$$

である.

Proof. 証明は読者の課題とする. □

定理 2.1 (無限の猿定理). 猿がほとんど確実にタイプライタの全てのキーを無限回叩くならば, ほとんど確実にテキストには任意の作品が含まれる.

Proof. 補題 2.1 より自明. □

2.4 カスタムブロック

数式のセクションの定理ブロックの応用で, 独自のブロックセクションを定義することができる. rmdja では BKD Ch. 2.7 Custom blocks で紹介されている例を予め使えるようにしている. それらは `type="..."` で指定できて, 以下の 5 種類がある.

- `rmdcaution`
- `rmdimportant`
- `rmdnote`
- `rmdtip`
- `rmdwarning`

である.



技術書によくある注意を喚起するブロック (`rmdcaution`).



技術書によくある注意を喚起するブロック (`rmdimportant`).



技術書によくある注意を喚起するブロック (`rmdnote`).



技術書によくある注意を喚起するブロック (rmdtip).



技術書によくある注意を喚起するブロック (rmdwarning).

このブロック内では Markdown の基本構文しか使えず、引用や相互参照などは使えない。これらをブロック内で使いたい場合は block の代わりに block2 と書く。ただしこちらは pandoc の機能のハックであるため、将来使えなくなる可能性もある。

2.5 脚注

脚注はインラインと、巻末に書く 2 通りがある。

ここにインラインで脚注 ^[脚注の本文]

ここにインラインで脚注^{*1}

本文は巻末に書く [^example-1][^example-2].

[^example-1]: 脚注の本文その 2

[^example-2]: 脚注の本文その 2

本文は巻末に書く ^{*2*3}.

ここにインラインで脚注 [^ 脚注の本文]

インラインで書くほうがシンプルに見えるが、この記法では間を空けずに連続して脚注を書くことができない。

このように書くと ^[脚注その 1]^[脚注その 2]上付きとして認識される

^{*1} 脚注の本文

^{*2} 脚注の本文その 2

^{*3} 脚注の本文その 2

第3章

動的なコンテンツの作成

3.1 プログラムチャンク

プログラムチャンクは、R Markdown 最大の特徴であり、R のソースコードや、その実行結果を Markdown に挿入できる。さらには **R** 以外の言語の動作も可能である。順番が前後してしまったが、定理などのカスタムブロックは本来はプログラムを入力するためのチャンクブロックであり、それを静的なテキストコンテンツの挿入に流用しているだけである。

以降は R で多くのユーザが頻繁に使うパッケージと、いくつかの技術文書作成に役に立つパッケージをインポートしている前提の説明とする。なお、rmarkdown、bookdown はチャンク内で特に読み込む必要がない。

```

01 pkgs <- installed.packages()
02 for (p in c("tidyverse", "ggthemes", "equatiomatic", "tufte", "kableExtra")) {
03   if (!p %in% pkgs) install.packages(p)
04 }
05 if (!"rmarkdown" %in% pkgs) remotes::install_github("rstudio/rmarkdown")
06 if (!"bookdown" %in% pkgs) remotes::install_github("rstudio/bookdown")
07 require(tidyverse)
```

```

Loading required package: tidyverse
-- Attaching packages -----
----- tidyverse 1.3.0 --
v ggplot2 3.3.2     v purrr    0.3.4
v tibble   3.0.3     v dplyr    1.0.2
v tidyverse 1.1.2     v stringr  1.4.0
v readr    1.3.1     vforcats  0.5.0
-- Conflicts -----
```

```
-- tidyverse_conflicts() --
x dplyr::filter()      masks stats::filter()
x dplyr::group_rows() masks kableExtra::group_rows()
x dplyr::lag()         masks stats::lag()
```

```
01 require(ggthemes)
```

Loading required package: ggthemes

```
01 require(equatiomatic)
```

Loading required package: equatiomatic

```
01 require(kableExtra)
```

このように、ログを掲載することもできる。これは再現性を重視する際に重宝するが、一方で単に画像などの主力だけを掲載したい場合もあるだろう。あるいは、プログラムの解説のため、プログラムは掲載するが実行しない、ということも必要になるかもしれない。プログラムと結果の表示/非表示はどちらも簡単に切り替え可能である。そのためには、チャンクオプションを指定する。

- echo: プログラムを掲載するかどうか
- message: プログラム実行結果の標準出力を掲載するかどうか
- warning: プログラム実行結果の警告を掲載するかどうか
- error: プログラム実行結果のエラーを掲載するかどうか
- eval: 文書作成時にプログラムを実行するかどうか
- include: 文書作成時にプログラムを実行し、かつ掲載しないかどうか
- results: 出力をいつもの R の出力風にするか ("markup"), 隠すか ("hide"), 出力を区切らずまとめるか ("hold"), テキストをそのまま出力するか ("asis")。最後はソースコードを動的に生成したい場合などに使う（後述）。

R の論理値は TRUE/FALSE または T/F と書く。

チャンクごとに個別に設定することも、デフォルトとして設定することもできる。前者の場合、

チャンクオプションは {} 内部に書く。r は R で実行するという意味である。

後者の場合、以下のようなプログラムでデフォルト値を上書きできる。

などと書く。なおこのチャンクは eval=F を設定することで、実行されることなくプログラムのみ掲載している（ただし、プログラムのみを掲載するなら、Markdown の機能でも可能である）。

```sh

```
echo Hello, Bookdown
```

```

{ } ブロック内の値にはさらに R プログラムを与えることができる。この使い方は後の章で解説する。

これらのオプションがあるおかげでプログラムとその結果の再現を説明したい場合はソースコードも表示させたり、回帰分析やシミュレーションの結果だけを掲載したい時は結果のみ表示したりできる。これが R Markdown の強みである。例えば Jupyter notebook/lab などは従来、コードセルと出力セルを自由に隠すことができなかった。

チャックに使用できる言語は R だけではない。つまり **Python** なども使用できる。以下で対応しているエンジンの一覧を表示できる。

```
[1] "awk"          "bash"         "coffee"        "gawk"        "groovy"
[6] "haskell"     "lein"         "mysql"        "node"        "octave"
[11] "perl"         "psql"         "Rscript"       "ruby"        "sas"
[16] "scala"        "sed"          "sh"           "stata"       "zsh"
[21] "highlight"   "Rcpp"         "tikz"         "dot"         "c"
[26] "cc"            "fortran"      "fortran95"    "asy"         "cat"
[31] "asis"          "stan"         "block"        "block2"      "js"
[36] "css"           "sql"          "go"           "python"     "julia"
[41] "sass"          "scss"         "theorem"      "lemma"      "corollary"
[46] "proposition"  "conjecture"  "definition"   "example"    "exercise"
[51] "proof"         "remark"      "solution"
```

また、新たにプログラムを追加することもできる。詳細は RDG Ch. 2.7 Other language engines を参考に。

TODO: 他の言語のプログラムを実行する際の注意点

3.2 プログラムで数式を生成する

プログラムチャックは、単にプログラムの計算結果を埋め込むだけでなく、静的なコンテンツを臨機応変に変更して出力させたり、あるいは手作業でやるには煩雑な加工処理を挟んでから表示させるのに役に立つ。

R のプログラムと組み合わせることで回帰分析の結果の数値をコピペすることなく数式で表示することができる。そのためには **equatiomatic** パッケージの `extract_eq()` を使う。

まずは、回帰係数を記号で表現するタイプ。LaTeX 数式をそのまま出力するため、チャックオプションに `results="asis"` を付ける必要があることに注意する。

$$\begin{aligned} Sepal.Length = & \alpha + \beta_1(Sepal.Width) + \beta_2(Petal.Length) + \beta_3(Petal.Width) + \\ & \beta_4(Species_{versicolor}) + \beta_5(Species_{virginica}) + \epsilon \end{aligned}$$

さらに `use_coef = T` で係数を推定結果の数値に置き換えた。

$$\begin{aligned} Sepal.Length = & 2.17 + 0.5(Sepal.Width) + 0.83(Petal.Length) - 0.32(Petal.Width) \\ & 0.72(Species_{versicolor}) - 1.02(Species_{virginica}) + \epsilon \end{aligned}$$

`equatiomatic` パッケージは現時点では `lm` `glm` に対応しており、`lmer` への対応も進めているようだ。

TODO: この書き方だと PDF で付番できない

3.3 プログラムを使った図の挿入

既に Markdown 記法による図表の挿入方法を紹介したが、プログラムチャックを介して画像を読み込み表示させることもできる。まずは、R のプログラムで既存の画像ファイルを表示させる方法。



図3.1 Johannes Gutenberg

もちろんのこと既存の画像だけでなく、データを読み込んでヒストグラムや散布図などを描いた結果を画像として掲載することもできる。

技術文書や学術論文では、画像の上か下に「図 1: XXXXX」のようなキャプションを付けることが多い。紙の書籍では絵本のように本文と図の順序を厳密に守るより、余白を作らないよう図の掲載位置を調整する必要があるからだ。

プログラムチャンクにはこのキャプションを入力するオプション `fig.cap` があるため, `plot()` 側でタイトルを付けないほうが良い. 例えば `ggplot2` パッケージの関数を使い以下のようなチャンクを書く^{*1}.

```
01 txt <- ````{r plot-sample, echo=T, fig.cap="`ggplot2` によるグラフ"}
02 data("diamonds")
03 diamonds <- diamonds[sample(1:NROW(diamonds), size =), ]
04 ggplot(diamonds, aes(x=carat, y=price, color=clarity)) +
05   geom_point() +
06   labs( x = "カラット数", y = "価格") + scale_color_pander(name = "クラリティ") +
07   theme_classic(base_family = "Noto Sans CJK JP") + theme(legend.position = "bottom")`````
08 cat(txt)
```

```
```{r plot-sample, echo=T, fig.cap="`ggplot2` によるグラフ"}
data("diamonds")
diamonds <- diamonds[sample(1:NROW(diamonds), size =),]
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) +
 geom_point() +
 labs(x = "カラット数", y = "価格") + scale_color_pander(name = "クラリティ") +
 theme_classic(base_family = "Noto Sans CJK JP") + theme(legend.position = "bottom")````
```

実際の表示は図 3.2 のようになる.

## 3.4 それ以外のグラフィックデバイス

LaTeX の `tikzDevice` を利用した作図が可能である. チャンクのエンジンを `tikz` に設定すれば, そのブロック内では `\begin{tikzpicture}` で始まる内容を記述できる. これは HTML でも出力される(図 3.3).

それ以外にも `knitr` が対応しているドローイングプログラムがある. 例えば DOT 言語(`dot`), Asymptote (`asy`), `node.js` や `D3.js` も使用できるが, HTML ベースのものは多くは PDF では表示できないことに注意.

なお, DOT 言語は `DiagrammeR` パッケージを経由して使うこともできる<sup>\*2</sup>(図: 3.5). グラフィカルモデルの記述などはこちらのほうが簡単かもしれない.

---

<sup>\*1</sup> なお, R ユーザーならば標準グラフィック関数である `plot()` 関数をご存知だろうが, 本稿では標準グラフィック関数の使用を推奨しない. 標準グラフィック関数のデバイスはもともと日本語フォントを想定しておらず, OS ごとに使用できるフォントも異なるためで, 品質維持のためには使用させない方針とした. 工夫すれば標準グラフィック関数でも日本語を適切に出力できるが, `ggplot2` を使用したほうが簡単であることが多いため, 標準グラフィック関数の解説書を作る以外では使うべきでない.

<sup>\*2</sup> <https://bookdown.org/yihui/rmarkdown-cookbook/diagrams.html>

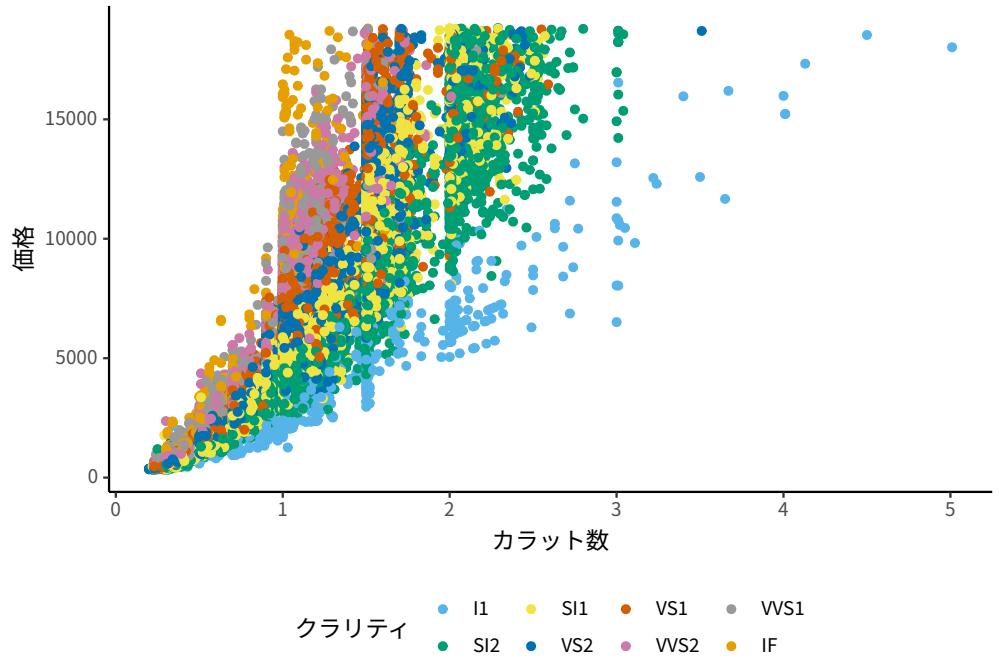


図3.2 ‘ggplot2’によるグラフ

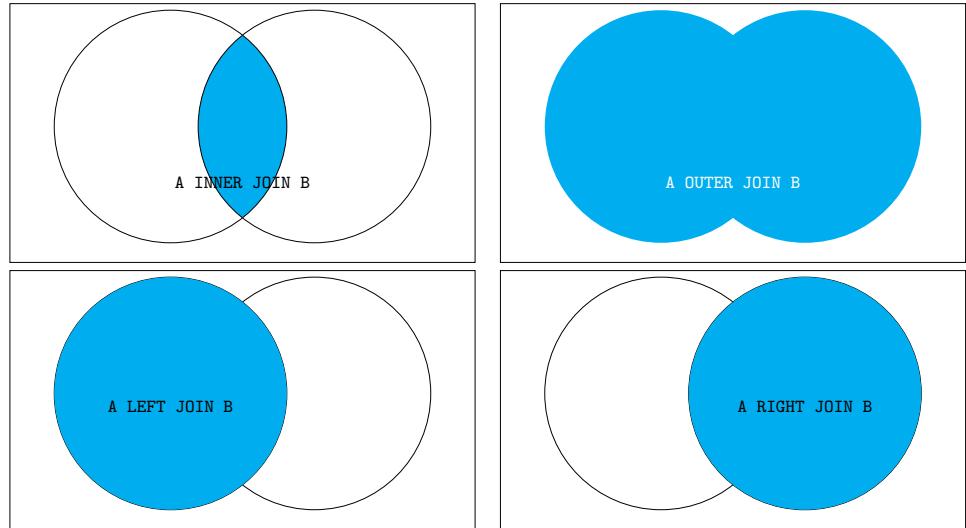


図3.3 tikz を利用した図の表示

```

01 DiagrammeR::grViz("digraph {
02 graph [layout = dot, rankdir = TB]
03
04 node [shape = rectangle]
05 rec1 [label = 'Step 1. 起床する']
06 rec2 [label = 'Step 2. コードを書く']
07 rec3 [label = 'Step 3. ???']"

```

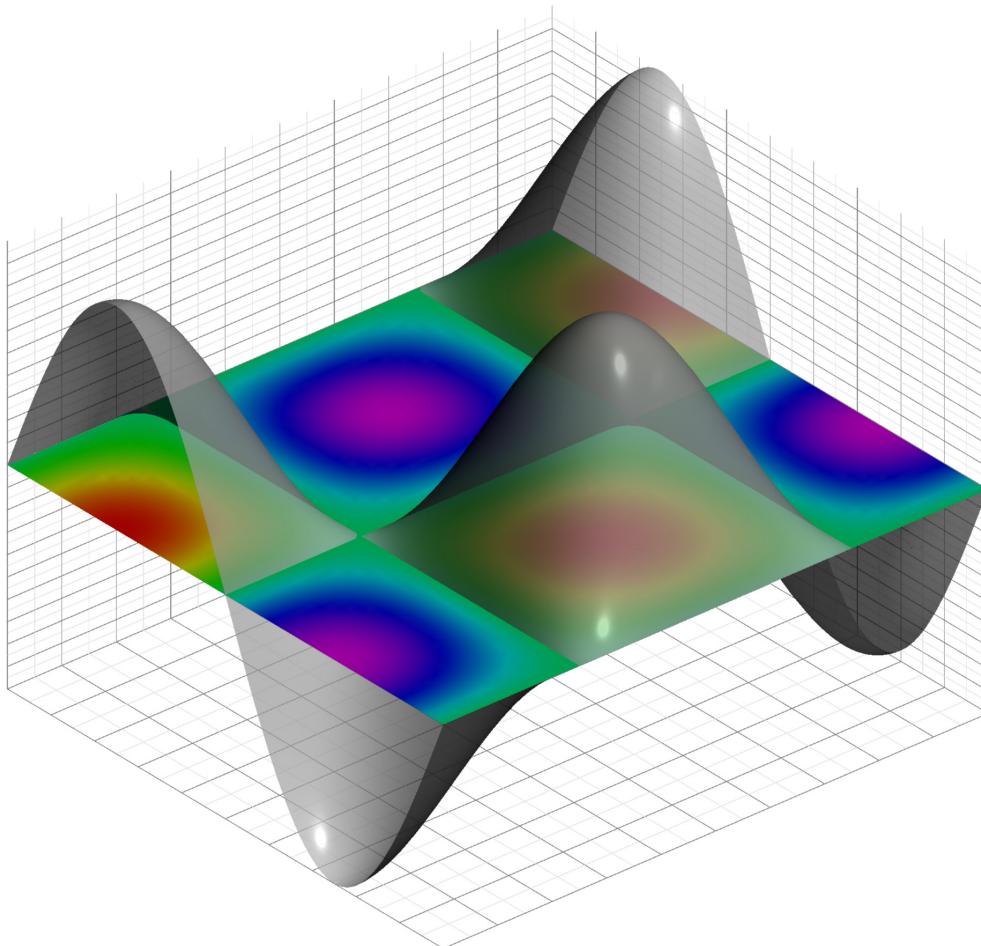


図3.4 Asymptoteによる画像(RCBと同じもの)

```
08 rec4 [label = 'Step 4. 給料をもらう']
09
10 # edge definitions with the node IDs
11 rec1 -> rec2 -> rec3 -> rec4
12 },"
13 height = 500
14)
```

### 3.5 TODO: 図のレイアウト設定

もしかすると技術文書ではなく文芸作品を作成したいユーザもいるかもしれない。そのような場合は別の章で解説する。

PDF ならばフロート設定のため、図が離れた位置に配置されることがある。そのため、「図 3.2」のような相互参照を使うと良いだろう。フロートを使うかどうかは、後のセクションで解説する TODO

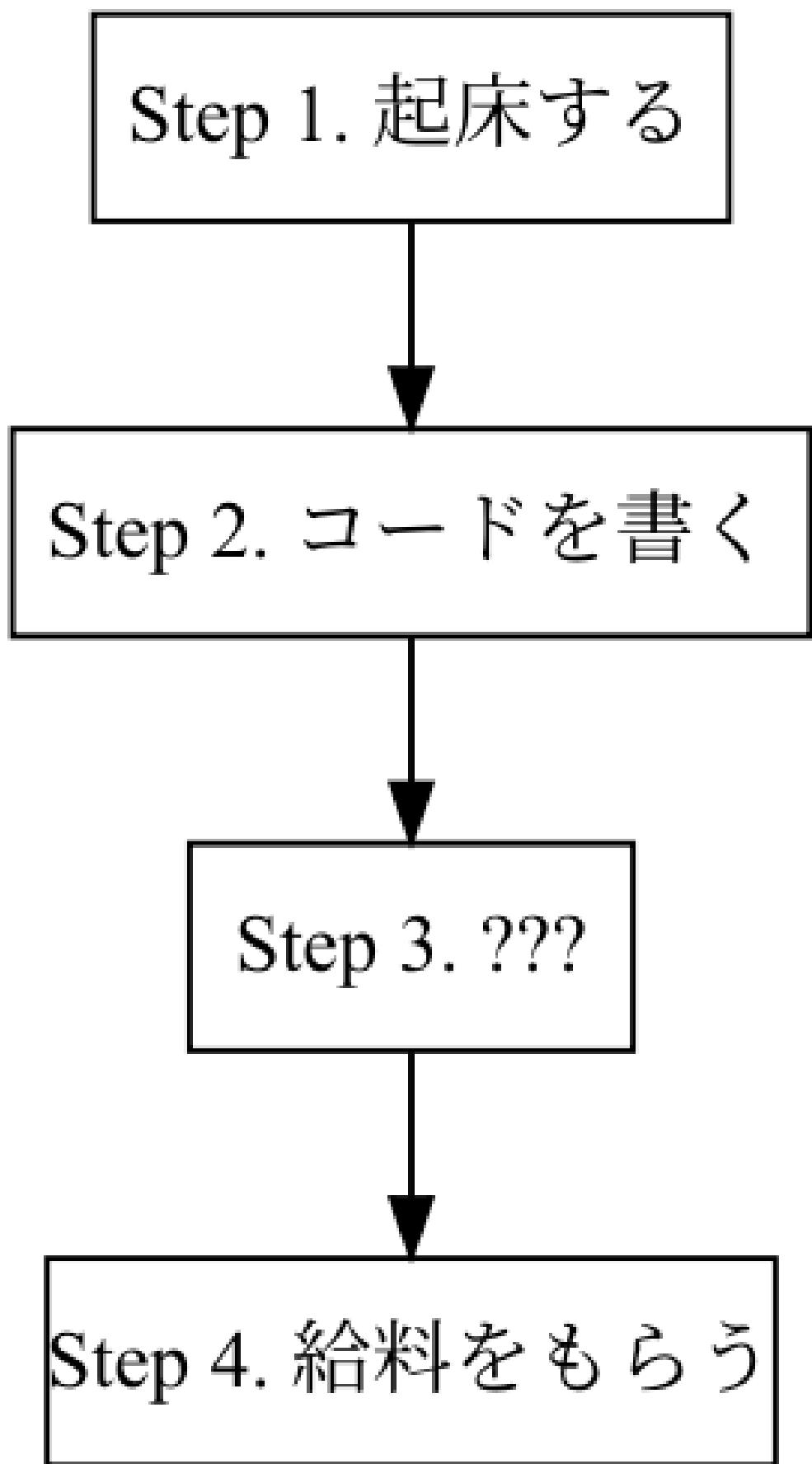


図3.5 DiagrammeRによるグラフィカルモデル (RCB, Ch. 4.15 より)

R のグラフィックデバイスを使っている限り、通常の R のコンソールと同じコードをチャンク内に書くだけで表示できる。

R のグラフィックデバイスではないとは、RGL や plotly など外部ライブラリに頼ったグラフ作成ツールのことである。判断できない人は、RStudio 上で実行して、“Plots” ページに表示されたら R のグラフィックデバイス、“Viewer” ページに表示されたらそうでない、で覚えていただきたい。後者を表示する方法は 9 章で後述する。R をこれまで使ったことがなく、それすらも何を言っているのか分からぬ、という場合は ggplot2 を使ってもらう。

最後の `fig.cap=""` がキャプションである。ただし、どうも日本語キャプションを書いたあとに他のチャンクオプションを指定するとエラーになるようだ。よって `fig.cap=` はオプションの末尾に書くべきである。また、`fig.cap=""` に数式や一部の特殊なテキストを直接入力することができない。この問題は相互参照について解説するセクション 4.1 で詳細を述べる。

`fig.cap` 以外のオプションはおそらく頻繁には変えないため、冒頭でまとめて設定したほうが楽だろう。

```
01 knitr::opts_chunk$set(
02 fig.align = "center",
03 fig.width = 6.5,
04 fig.height = 4.5,
05 out.width = "100%",
06 out.height = "100%"
07)
```

なお、これらは rmdja でのデフォルト値であるため、実際にこの値をあえて記述する必要はない。

ここで、`fig.width` と `out.width` の違いも述べておく

TODO

## 3.6 R プログラムを使った表の装飾

Markdown 記法を使った表記は既に紹介した。しかしこれは表の数値を全て手動で書かなければならぬ。もちろんこれも R 内のデータを手書きなどせずとも表示できるし、テーブルのデザインもある程度自由に設定できる。

R Markdown のデフォルトでは R のコンソールと同様にテキストとして出力されるが、bookdown では異なるデザインで表示されている。これは knitr, kableExtra パッケージなどで事後処理をかけることで見やすいデザインの表に変換しているからである。

表3.1 ‘knitr::kable()’ で出力された (PDF ではあまりかっこよくない) 表

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

この方法はシンプルで使いやすいが、R はテーブル状のデータ処理に長けているため、手動で数値を書くよりも簡単な方法がある。

```
01 data(iris)
02 kable(
03 head(iris, n = 10),
04 caption = "‘knitr::kable()’ で出力された (PDF ではあまりかっこよくない) 表"
05)
```

こちらは関数内にキャプションを書く必要があり、チャンクオプションに指定する方法はない(表3.1)。

```
01 data(iris)
02 kable(
03 head(iris, n = 10),
04 booktabs = T,
05 caption = "奇数行を強調し、PDF では booktabs を利用"
06) %>% row_spec(seq(1, 10), by = 2), background = "gray")
```

さらに、RCB の Ch. 10.1 The function `knitr::kable()` ではその他いろいろな書式設定を紹介している。

表3.2 奇数行を強調し, PDF では booktabs を利用

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa



## 第 4 章

# 相互参照と引用

### 4.1 相互参照

#### 4.1.1 図表や式へのアンカーリンク

図, 表, 式などに番号を自動で割り当て, さらにハイパーリンクを付加できる. `\@ref(ID)` を使う. 現状では `refstyle` や `prettyref` のように接頭語を自動で付けてくれないが, そのうちなんとかなるかもしれない.

`bookdown` の相互参照は, `LaTeX` の `prettyref.sty` のように, 接頭語: 参照 ID という記法になる. 参照 ID は通常, チャンク ID と同じである. 既に紹介したように, 数式参照の接頭語は `eq` で, 定理は `thm` である. 図表は `fig`, `tab`. その他の接頭語は BKD Ch. 2.2 Markdown extensions by bookdown を参考に.

#### 4.1.2 表への相互参照

Markdown 記法で表を書く場合, 以下のように `Table:` の直後にラベルを記入する (表 4.1).

**Table:** (\#tab:tab-md) Markdown 記法の表

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

表4.1: Markdown 記法の表

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

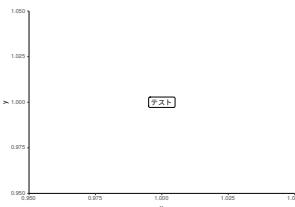
#### 4.1.3 章への相互参照

章見出しへの相互参照も可能である。これは Pandoc の機能を利用しているため、接頭辞は不要である。Pandoc の仕様により欧文であればタイトルがそのまま参照 ID となるが、非欧文の文字に対して適用されないため、参照したい章の見出しにの後にスペースを入れて {# 参照 ID} と書く必要がある。

#### 4.1.4 特殊な相互参照

チャンクオプションの `fig.cap` などに TeX 数式を書いても正しく表示できない。そのような場合は `ref` 参照を使う。`(ref:figcap1) \coloremoji{[]} $\sum \oint \mathfrak{A} \mathscr{B} \mathbb{C}$ \coloremoji{[]}` と書くと、図 4.1 のキャプションにも特殊な記号が使える。

なお、複数指定する場合は連続させず、改行で 1 行空けて宣言する必要がある。

図4.1  $\sum \oint \mathfrak{A} \mathscr{B} \mathbb{C}$ 

この参照は一度しか使えない。

PDF での表示では、図 4.1 のキャプションの外側が文字化けしていることだろう。これは絵文字出力に関する問題で、別のセクションで解説する。

これはかなり強力で、

1. 定義される前の行にも適用される
2. チャンクオプションだけでなく出力結果にも適用される

という仕様である。

TODO: 自己言及的な文章は書かないならこれくらいの認識でいいだろうが、より正確な話はどうするか

## 4.2 文献引用

YAML フロントマターの `bibliography:` に文献管理ファイル (`.bib`, `.json` 等) を指定することで、ファイルに含まれる文献への参照が可能になる。`@` 引用 ID で本文に引用を与えられ、文書に引用した文献の一覧が自動で生成される。また、`citr` パッケージにより、RStudio Addins に文献に対応する引用 ID を取り出して挿入する機能が追加される。

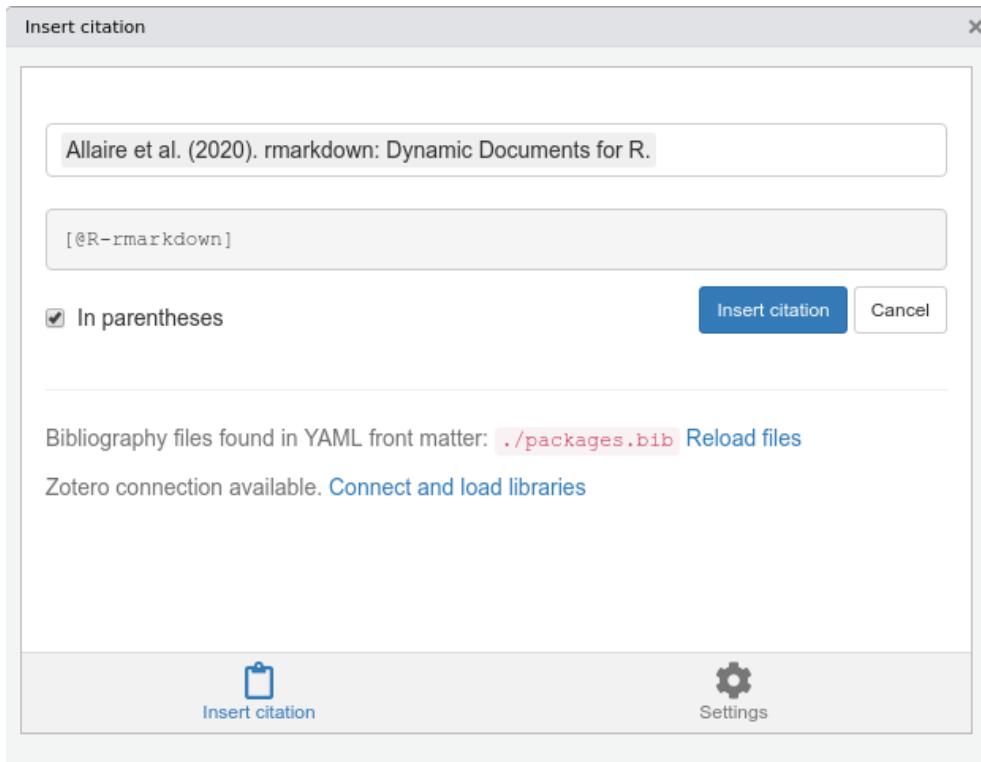


図4.2 `citr` パッケージの例

BibTeX, BibLaTeX、または pandoc-citeproc で処理される。これは `citation-package` または `_output.yml` の `citation_package` に依存する。HTML は基本的に pandoc-citeproc を使用した出力であり、影響が大きいのは PDF である。

- `default`: pandoc-citeproc を使用して参考文献を出力する。
- `biblatex`: biblatex を使用する。
- `natbib`: bibtex を使用し、本文中の参照には natbib.sty を使う。
  - `natbiboptions`: で `number`, `authoryear` などの natbib.sty のオプションを指定できる。

そのスタイルファイル (`.bst`, `.csl`) はそれぞれ `biblio-style:/csl:` で指定できる。

しかし国内論文雑誌などで規定された、日本語文献に対応した `.bst` ファイルを使って参考文献を記載したい場合には (u)pBBIITEX) が必要となるはずである。

これは R Markdown で日本語技術文書を作る際の特にアレな障害の 1 つである。PDF は BibTeX があるが、HTML では使えない。pandoc の出力に頼るしかない。pandoc は CSL という、Word でも使われている参考文献リストのフォーマットに対応しているが、機能が少ないため日本語文献の引用スタイルが崩れてしまう。

日本語対応 `.bst` ファイルを使いたい場合は少しトリッキーな操作が必要になる。`rmarkdown` は `tinytex` というパッケージでインストールされたスタンドアローンな処理系で PDF を生成している。冒頭のチャックで `options(tinytex.latexmk.emulation = F)` を指定することで、自分のマシンにインストールされている普段使っている処理系に処理させることができる。さらに `rmdja` では `natbib` を指定した場合に自動でカレントディレクトリに `.latexmkrc` をコピーするようにしている。しかしログが残らないなどデバッグしづらいところがあるため、このやり方はやや使いづらい。

TODO: この操作なしに (u)pBBIITEX) を使うには、たぶん `tinytex` と `rmarkdown` 両方の修正が必要。そこまで複雑ではないと思うのでそのうち修正してみたい。

なお、普段文献管理ソフトを使っていないが、数本程度の文献を引用リストに載せたい利用者は、`biblatex` の構文を利用して書くのがよいかもしれない。例えばここに書いてあるように。

<https://teastat.blogspot.com/2019/01/bookdown.html>

## 第 5 章

# 簡単なレイアウト変更

## 5.1 HTML

### 5.1.1 フォント変更

HTML は文字通り HTML で出力しているため, CSS の使い方次第でいくらでもデザインを変えることができる。

## 5.2 PDF

### 5.2.1 フォント変更

PDF を生成する場合, ver 0.3 時点ではデフォルトのフォントを OS に応じて変えていく。もし変更したい場合は YAML フロントマターの以下の項目を変更する

- `mainfont`: 欧文セリフフォント
- `sansfont`: 欧文サンセリフフォント
- `monofont`: 等幅フォント (コードの表示などに使用)
- `jfontpreset`: 和文フォントのプリセット
- `jmainfont`: 和文メインフォント (一般に明朝体を指定)
- `jsansfont`: 和文セリフフォント (一般にゴシック体を指定)
- `jmonofont`: 和文等幅フォント (コードの表示などに使用)

`jfontpreset` は `zxjafont` または `luatex-ja` によるプリセットで, 3 種類のフォントを一括指定できる。個別指定したフォントはこれを上書きする。特にこだわりがないなら一括指定で良いが, ソースコードを多く掲載する場合は `M+` や `Ricty` などのフォントを用意すると良いだろう。

さらに, それぞれの項目には `options` と接尾辞のついた項目が用意されている。フォントの相対サイズが合わず不格好な場合は

```
mainfont: Palatinno
mainfontoptions:
 - Scale=0.9
```

などと書いて調整できる。

インラインのフォント変更は TODO

### 5.2.2 文書クラスの変更

HTML は利用者側が見え方をある程度カスタマイズできる。かつて存在した Evernote Clearly やカスタム CSS を使って、そのぶん PDF は作成者側がよりレイアウトに注意を払うことになるだろう。本稿では文章の区切りを章立てにしている。しかし PDF 数十ページしかない文書を大きな文字サイズの見出しで区切るのは少しものものしい感じがする。YAML フロントマターを変更すれば、トップレベルの見出しを変更できる。

まず、今回は文書ということで書籍の組版をデフォルト設定にしている。もう少し小規模な文書ならば、レポートや論文記事形式のほうが良いかもしれない。例えば、以下のように指定する。

```
documentclass: bxjsreport
```

documentclass には LaTeX の文書クラスファイル (.cls) ならなんでも与えることができるが、~~X~~LaTeX または LuaLaTeX で日本語文書を作成することを想定しているため、BXjscls シリーズのクラスから選ぶことを推奨する<sup>\*1</sup>。よって、以下 3 種類の中から選ぶとよい。デフォルトは bxjsbook なので、これは明示的に指定する必要はない。

- bxjsbook
- bxjsreport
- bxjsarticle

このうち、bxjsbook がデフォルト設定となっている。

文書クラスとは別に、文書

その他、\_output.yml や \_bookdown.yml のコメントを参考に。

---

<sup>\*1</sup> <https://www.ctan.org/pkg/bxjscls>。但し、スライド用クラスである bxjsslide の使用は想定していない。

## 第 6 章

# rmdja による文書作成支援機能

### 6.0.1 クリエイティブ・コモンズの表記

Web 公開する文書ならばクリエイティブ・コモンズの表記をつけたいところだ。公式サイトで毎回発行するのは面倒なので表示する関数を用意した。ハイパーリンクも付けるようにしている。チャンクでは `results="asis"` オプションが必要になる。また、通常は `echo=F` を設定すべきだろう。冒頭の表記もこれで作成している。もちろんそれぞれの媒体に対応している。

文言の生成は未対応

### 6.0.2 ルビ表記

ルビはおそらく CJK 言語など一部の言語でしか使われていない（アラビア語とかペルシア語とかの補助記号は詳しく知らないが多分グリフとしてサポートされてるっぽいので無視）ため、ルビ表記も R Markdown ではサポートされていない。そこで簡単にルビを表示できる関数 `rmdja::ruby()` を用意した。インライン実行で使う。PDF での配置は `pxrubrica.sty` を利用したグループルビである。よって、1 字ごとに配置（モノルビ）にしたいとか、突出指定とか、細かいことは HTML タグや CSS や LaTeX コマンドを自分で書く。妥協案として、1 字ごとに呼び出す手もある。

グループルビの例: とある科学の超電磁砲, 皇帝ライハルト, 格館, シュタッヒバルムシュロス  
 シュワルツ・ランツェンレイター きれうりわり メキシコ  
 黒色槍騎兵, 喜連瓜破, MEXICO

分割して出力した例: 喜連瓜破, 黒色槍騎兵,

TODO: それ以外にも便利機能を少しづつ増やしていく予定



## 第 IV 部

製本と多様な形式への対応



## 第 7 章

# 製本方法の詳細

冒頭のチュートリアルで行った製本（ビルド）の仕組みをもう少し詳しく解説する。

`bookdown-demo` を念頭に置いた解説。`rmdja` も基本的に同じ。

- `index.Rmd`: デフォルトで最初に読み込まれる Rmd ファイル（名前を変える機能もあるが、現時点では不具合が起こりやすいのでおすすめしない）
- それ以外の Rmd ファイル: 連結して読み込むことが可能
- `_output.yml`: マルチメディア展開のための設定。PDF, HTML, EPUB それぞれの設定を書く
- `_bookdown.yml`: `bookdown` のレイアウト設定
- その他の設定ファイル: その他製本に必要なもの、画像ファイル、`.css` ファイル、`.bib` 等

`_output.yaml`, `_bookdown.yml` は `index.Rmd` のヘッダに書くこともできるが、長くなりすぎるので分割できる。`bookdown::render_book()` 関数は、ルートディレクトリのこれらを自動で読み込んでくれる。

### 7.1 ファイル構成

これらのファイルの中身を解説する。

#### 7.1.1 `_output.yml`

本来の YAML の `output:` 以下の記述をこの `_output.yml` ファイルに書くことができる。`output:` を複数書くと `rmarkdown::render_site()` やビルドツールでそれぞれの形式に一括作成してくれる。

```
output:
 bookdown::gitbook:
 lib_dir: assets
```

```
split_by: section
config:
 toolbar:
 position: static
bookdown::pdf_book:
 keep_tex: yes
bookdown::html_book:
 css: toc.css
documentclass: book
```

詳しくは BKD “[Ch. 3 Output Formats](#)” の章を.

### 7.1.2 \_bookdown.yml

\_bookdown.yml も index.Rmd の YAML ヘッダの bookdown: 以下に対応する内容を書くことができる. 例えばどの Rmd ファイルを読み込むかとか, LaTeX のときだけ, HTML のときだけ読み込むような設定も可能.

<https://ill-identified.hatenablog.com/entry/2020/09/05/202403>

詳しくは, BKD [Ch. 4.4 Configuration](#)

Build ページから文書をビルドするには, index.Rmd の YAML ヘッダに site: bookdown::bookdown\_site を書く必要がある. さらに, index.Rmd をプロジェクトディレクトリのルートに置いていない場合は, ツールバーの Build -> Configure Build Tools... から index.Rmd を置いているディレクトリを site ディレクトリとする設定が必要になる (図 7.1, 7.2).

または, bookdown::render\_book("index.Rmd", "rmdja::pdf\_book\_ja") などでも実行できるから, コマンドラインからも実行できる. 同時製本は rmarkdown::render\_site().

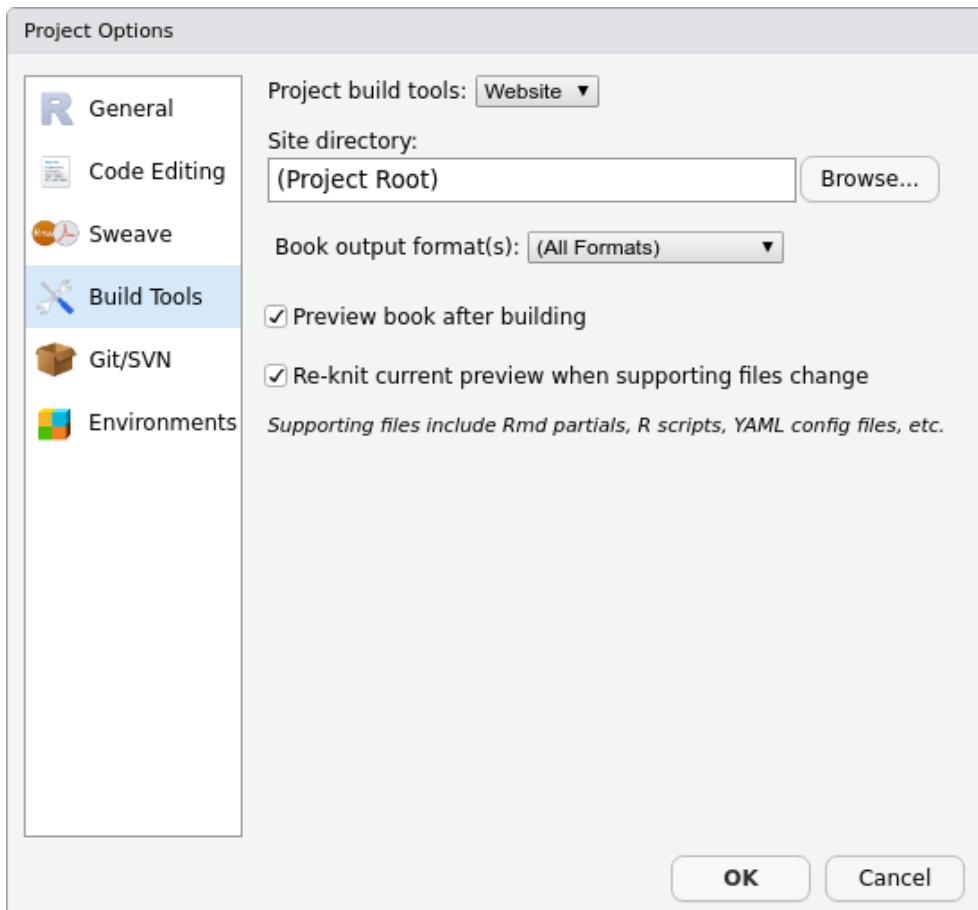


図7.1 Build ペーンの手動設定



図7.2 Build ペーンの手動設定



## 第 8 章

# 出力形式による表現の限界

### 8.1 HTML と PDF で処理を場合分けする

出力方法で言えば、HTML と PDF に大別できる。Rmd は HTML タグも LaTeX コマンドも受け付けるが、それぞれ HTML と PDF に変換する際にしか反映できない。よって、例えば複雑な図表を LaTeX コマンドでじかに Rmd ファイルに書いてしまった場合、HTML では表示されない。

紙媒体と電子媒体では表現できることに差がある。例えば紙はあらゆる環境で同じような見た目になるが、ハイパーリンクは付けられないし、一度出版してしまうと修正は容易ではない。PDF の見た目も読者の環境に依存しにくいが、やはり更新が容易ではない。

`bookdown` には既に印刷された本の中身を書き換えるする機能はないが、出力ごとに内容を変えることで、PDF にのみ更新履歴を表示することはできる。

`knitr::is_latex_output()`, `knitr::is_html_output()` などは、`knit` 時にどの媒体への変換処理なのかを判定するのに使える。`rmdja::ruby()` もこの機能を利用していりし、本文中の `LATEX` のロゴも HTML と PDF で使い分けている。

また、`_bookdown.yml` の設定、`rmd_files` は、媒体別に設定することができる。

```
rmd_files:
 html:
 - index.Rmd
 - html-only.Rmd
 latex:
 - index.Rmd
 - latex-only.Rmd
```

## 8.2 絵文字の出力

絵文字を HTML でも PDF でも出力したい場合, `\coloremoji{[]}` のように絵文字を囲む。ただし, RStudio のエディタは一部のマルチバイト文字の表示に対応していないので予期せぬ不具合に注意する。

現在の主要 Web ブラウザでは、特に設定せずとも Unicode 絵文字をカラー画像に置き換えて表示できるものが多い。しかし PDF 生成時には明示的にフォントを指定するか、画像に置き換える記述が必要である。その実現のため `bxcoloremoji` という LaTeX パッケージ<sup>\*1</sup> を利用する。このパッケージは CTAN に登録されていないため、別途インストールする必要がある。

## 8.3 画像の保存形式

技術文書での画像の多くはプロットなど単純な図形なので、写真などを掲載するのでない限り、PDF で出力する場合はプロット画像も PDF にするのが望ましい。JPG や PNG などのラスタ画像では拡大すると粗くなるが、PDF などのベクタ画像ならば拡大しても粗くならず、かつ単純な図形ならばファイルサイズも小さく済むことが多い。一方で HTML は通常 Web ブラウザで閲覧するため、PDF に対応していないことが多い。HTML でベクタ画像を掲載したい場合は **SVG 形式**で出力する。

R による SVG への出力は、従来組み込みの `SVG()` で行うことが多かったが、近年は新たなパッケージが出ている。有力なのは `svglite` と `rsvg` である。

<https://oku.edu.mie-u.ac.jp/~okumura/stat/svg.html>

`rsvg` のほうが高性能だが、`knitr` で対応しているのは `svglite` なので簡単に使いたいならこちらを推奨する。

## 8.4 デフォルトの保存形式

デフォルトでは、PDF は `cairo_pdf`、HTML では解像度を高めに設定した PNG を使用している。これは、件数の多い散布図など、ベクタ形式ではファイルサイズが大きくなりすぎる場合もありうるための判断である。

画像形式を変更したい場合は、チャンクオプションの `dev` で、オプションは `dev.args=list(...)` で変更できる。

<https://bookdown.org/yihui/rmarkdown-cookbook/graphical-device.html>

---

<sup>\*1</sup> <https://github.com/zr-tex8r/BXcoloremoji>

## 第 9 章

# Web アプレットの挿入

9.0.1 TODO: plotly

9.0.2 TODO: shiny



## 第 10 章

# 製本した文書を配布する

### 10.1 Wep ページのホスティング

HTML ファイルは様々な配布方法がある。もちろん自分でサーバを立てても良い。特に簡単なのは以下の 2 点である。

1. `github pages` を利用する
2. `bookdown.org` に投稿する

(1) の詳細は `github.com` の公式ドキュメントを見るのが一番良いだろう。

既に `bookdown` で作成した文書を公開している例は多数ある。例えば既に何度も言及した公式解説サイトはそれじたいが `bookdown` で作られているし, “R for Data Science” (Wickham and Grolemund, 2016)<sup>\*1</sup> は、内容の良さも含め一見に値する。また, “Hands-On Data Visualization: Interactive Storytelling from Spreadsheets to Code” (Dougherty and Ilyankou, forthcoming) という本<sup>\*2</sup>が来年出るらしい。そして面白いことにこれは R Google スプレッドシートとか R 以外の Web 上のサービスの利用法を紹介する文書である。

これらはいずれもソースコードまで公開されている。もちろんここでいうソースコードとは、本文中のプログラムだけでなく文書を生成する `Rmd` ファイルなども含める。

それ以外にも有名無名の多くのドキュメントが公開されているが、一方で日本語はまだままだ少ない。内容が豊富で、かつソースコードまで公開されている例として以下が見つかった。

- 『R で計量政治学入門<sup>\*3</sup>』
- 『AI レベルの倫理学<sup>\*4</sup>』

---

<sup>\*1</sup> <https://github.com/hadley/r4ds>

<sup>\*2</sup> ソース: <https://github.com/handsondataviz/book>

<sup>\*3</sup> ソース: [https://github.com/shohei-doi/quant\\_polisci](https://github.com/shohei-doi/quant_polisci)

<sup>\*4</sup> ソース: <https://github.com/MToyokura/Ethics-for-A-Level-Japanese>

bookdown の機能や見た目を確認することができる。さらに以下 2 つは私が作成したものである。

- ・『三國志で学ぶデータ分析 (Japan.R 2019)』<sup>\*5</sup> (Japan.R 2019 の資料)
- ・『経済学と反事実分析接觸篇 Economics and Counterfactual Analysis: A Contact』<sup>\*6</sup> (Tokyo.R 第 83 回の資料)

特に私の 2 作品は PDF のレイアウトにも注意を払っているが、当時は kazutan 氏作の `bookdown_ja_template` をさらに改良した kenjimyzk 氏のテンプレートを元にワンオフで作成したフォーマットを使用し、`rmdja` を使用していないため、あまりスマートでない書き方が見られる。

また、HTML 形式の文書には PDF など他のファイル形式のダウンロードリンクを設置することができる。これは `_bookdown.yml` で表示を指定できる。

## 10.2 TODO: 入稿するには

国内の印刷所で PDF 入稿する際のスタンダードは何だろうか？紙媒体でやったことがないので全くわからない。ver. 0.3 時点での対応を紹介する。

### 10.2.1 トンボの表示

`_output.yml` で

```
rmdja::pdf_book_ja:
 tombow:true
```

とすると PDF にトンボ (trimming mark) を表示する。これは `gentombow.sty` によるものである。しかし私はこの出力が適切なのか判断することができない。

### 10.2.2 フォントの埋め込み

少なくとも PDF ではフォントを埋め込みそこなったり、Type 3 フォントが設定されないようにしている。ただし Python 等を利用して描いた PDF は個別に設定が必要な場合があり、保証できない。

TODO: <https://teastat.blogspot.com/2019/01/bookdown.html> の記述のうち、まだ対応していないものがある。

---

<sup>\*5</sup> ソース: <https://github.com/Gedevan-Aleksizde/Japan.R2019>

<sup>\*6</sup> ソース: [https://github.com/Gedevan-Aleksizde/20190125\\_tokyor](https://github.com/Gedevan-Aleksizde/20190125_tokyor)

第 V 部

應用



## 第 11 章

# TODO: 文芸作品の作成のために

作家の京極夏彦は自分の作品を 1 ページごとに切り取っても作品として成立するようなレイアウトにこだわっているらしい。すでに説明したように技術文書や学術論文では図表の配置や改行などにあまりこだわりがない。しかし、不可能ではない。HTML では難しいが（不可能ではないが HTML でやるメリットが感じられないで対応する気がない）、PDF ではある程度のレイアウトの制御が可能である。

ただし、本当に厳格な JIS 準拠の組版にこだわるなら、LaTeX を直接編集しなければならない。



## 第 VI 部

### デバッグ



---

残念ながら、現状 bookdown は完全にプログラミング知識のないエンドユーザでも縦横無尽に使用できるかと言うと、まだまだ不安定でそのレベルには達していない。さらに悪いことに、rmarkdown および bookdown は knitr, pandoc, LaTeX といった様々なプログラムを継ぎ接ぎして実装されているため、R の知識だけではエラーが起った場合や、意図したとおりの出力が得られないときに原因が分かりにくくことがある。そこで、ここではエラーが出た際にどう対処するかのヒントを書いておく。



## 第 12 章

# 製本時のエラーへの対処

### 12.1 エラーがどのタイミングで発生したかを特定する

逆に言えば, Rmd ファイルを md ファイルに変換 (knitr による処理) するときにエラーが出たのか, md を各ファイルに変換 pandoc する際に起こったのかをまず特定するのが重要である. そのためには

1. `keep_md: true` を設定する
2. うまくいかないときはキャッシュを削除してから再実行する

という対処法がある. (1) は文字通り中間出力ファイルである .md を残すことを意味する. これが生成されないなら knitr でのエラーだと分かるし, 中身を見て不自然な内容になっているのなら Rmd の書き方が knitr に正しく評価されていないことがわかる.

キャッシュも私の経験上よくエラーの原因となっている. 以前に実行していたチャンクの結果が更新されていないせいで, knitr の処理の不整合を起こすことがある. \*\_files には出力に必要な画像ファイルが, \*\_cache にはチャンク実行結果のキャッシュが残っている. 後者は `knitr::opts_chunk$set(cache = T)` などでキャッシュを残す設定ができるので, F に設定した上でこれらのファイルを削除する.

処理に時間がかかるチャンクがあってキャッシュを作りたい場合は, 別途 rds や RData ファイルに結果を保存するという方法もある. しかもしもプログラムの再現性を重視する場合, この方法は望ましくないだろう. しかし残念ながら現状はこうするか, ひたすら長い時間を持つしかない.

TODO: <https://bookdown.org/yihui/rmarkdown-cookbook/cache.html>

### 12.2 YAML フロントマターを確認する

以前『[R] R Markdown の YAML ヘッダでハマったおまえのための記事』というブログ記事にも書いたように, YAML フロントマターは慣れないと書き間違えやすいのが現状である. もし自分で変更したのなら, 改めて確認すべきだろう. 特に, 製本直後にすぐ

に、心当たりのない R プログラム関係のエラーが出る場合、チャンクではなく YAML フロントマターの読み取りに失敗している可能性がある。

以下の **4 原則**を覚えておこう。以前は bookdown の話を想定してなかったので、さらに条文を 1 つ加えた。

1. `output:` 以下はフォーマット関数への引数
2. トップレベルのオプションは `pandoc` のオプション
3. タイプミスや位置間違えでも動いたり、動かなかったりする
4. `_output.yml` および `_bookdown.yml` を見る。

`output:` には `bookdown::gitbook` など、`bookdown` で提供されているフォーマット関数を指定しており、その配下に記入するのはフォーマット関数に与える引数である。よって、関数ヘルプを確認すれば有効な引数を知ることができる。しかし一方で、`...` が引数になっていることがあるので、タイプミスしてもエラーが出ないことがある。

また、YAML の構文でサポートされている配列は誤評価を引き起こすことがある。

```
output:
 bookdown::gitbook:
 toc_depth: 3
 toc: true
```

```
output:
 bookdown::gitbook:
 - toc_depth: 3
 - toc: true
```

上の例は正しい記法である。一方でハイフン `-` は YAML では配列を記述するために用意されている。下記の場合、キーワード引数ではなく位置引数のような扱いになるため、`toc` に対して `3` を代入することになり、エラーが発生する。逆に言えば、`-` を使う場合、キーワードを書かずに値だけを正しい順番で書けば機能する。

インデントしないトップレベルの引数は、基本的に `pandoc` に与える引数である。これ意味のない引数を与えててもエラーを返さないことが多いので、タイプミスに注意する。

しかし、フォーマット関数に `pandoc_args` という構文をサポートしていることや、フォーマット関数で `pandoc` の同名の引数を上書きする仕様のフォーマットもあるため、上記は絶対ではない。これが原因で、「`output:` 以下に書くべきものを間違えてトップレベルに書いたが、意図したとおりに機能した」あるいはその逆が発生することがある。また、`pandoc` の構文ではキーワードにハイフンを使うことができるが、フォーマットは R の関数でもあるため、ハイフンを使はず、アンダースコアで置き換えられる。この違いも書き間違えの原因になる。

## 12.3 PDF の生成に失敗する場合

それでもエラーが出る場合, 私の経験上ほとんどが生成した .tex ファイルをタイプセットする際にエラーが発生している. html との両立を考えると, どうしても pandoc が解釈できる構文に限界があるためである.

! LaTeX Error: XXXXX

とか

Error: LaTeX failed to XXXX

といったメッセージが表示されるのをすぐに分かる. さらに丁寧なことに, tinytex のデバッグ方法への[リンク](#)まで表示される

この場合最も重要なのは, 以下に尽きる.

1 options(tinytex.verbose = TRUE) を設定する 2 keep\_tex: true を設定する

これは keep\_md と同様に, 中間ファイルである .tex を残すことを意味する.

それでも解決しない場合, 改めてこのファイルを手動でタイプセットするのも 1 つの方法だ. もしうまくいったり, 異なるエラーが出るのなら, 環境の違いが問題かもしれない. そして upBibTeX を使うのなら, 後者が唯一のデバッグ方法だ.

ここでは rmdja の内部処理を解説する. knitr や rmarkdown の仕様に精通している, 自分で細かい設定をしたいユーザ向けの解説である.



## 付録 A

### デフォルト値の自動調整

R Markdown で日本語文書を作成する上での大きな障害の 1 つである、 YAML フロントマターの設定を改善している。 rmdja の文書フォーマットは YAML フロントマターのデフォルト値などを日本語文書に適したものに変更している。さらに、ユーザーを OS ごとのフォントの違いや煩雑で重複だらけの設定から解放するため、内部処理でも動的に設定変更している。もちろんこれらはユーザーによる YAML フロントマターやチャンクオプションの変更で上書きできる。

#### A.1 デフォルトのフォント

PDF 出力時のデフォルトフォントは、生成時に OS を判定して設定している。その設定は表 A.1 のようなルールである。

これらは Xe<sup>L</sup>AT<sub>E</sub>X ならば zxjafont, Lua<sup>L</sup>AT<sub>E</sub>X ならば luatex-ja で用意されているプリセットを使って設定している。使用 OS の判定は R の基本関数による。なお、Noto フォントを選んだのは Ubuntu 18 以降の日本語用フォントだからである。Ubuntu から派生した OS にはプリインストールされていることが多いようだが、Debian, Cent OS, Fedora 等にはおそらくプリインストールされていないので注意。現時点ではフォントが実際にインストールされているかを確認する機能はない。

フォントのプリセットを指定した場合、個別設定は無効になる。さらに、3 種類の和文フォントを全て設定していない場合もデフォルトのプリセットから選ばれる。

表A.1 OS/エンジン別のデフォルトフォント

engine	Linux	Mac	Windows (>= 8)	Windows (それ以前)
XeLaTeX	Noto	游書体	游書体	MS フォント
LuaLaTeX	Noto	ヒラギノ	游書体	MS フォント

## A.2 チャンクのデフォルト設定

デフォルトのグラフィックデバイスは、HTML では PNG、PDF では `cairo_pdf` している。R でよく描画するような単純な図形はベクタ画像が適しているが、件数のとても多いデータの散布図などはベクタ画像にするとファイルサイズが大きくなるため、そのような画像を適度に「劣化」させてファイルサイズを軽減してくれる `cairo_pdf` を標準としている。HTML に関しては、そもそもデフォルトの設定で PDF が表示できない Web ブラウザが多いことから、PNG をデフォルトにした。

また、`block`, `block2`, `asis` などのブロックを `echo=F` や `include=F` にするメリットはほぼないため、`knitr::opts_chunk$set(echo = F, include = F)` と一括設定してもこれらは `echo=T`, `include=T` のままである。変更したい場合は、チャンクごとに設定することで有効になる。

## 付録 B

# PDF の組版に関する細かい話

ここでは pandoc テンプレート等の設定を解説する。

3 種類の和文フォントを個別設定をした場合, X<sub>E</sub>L<sup>A</sup>T<sub>E</sub>X はフォールバックフォントを有効にしている。j\*\*\*\*fontoptions 以下に, FallBack=... というオプションでフォールバックフォントを指定すれば有効になる。

用紙サイズは, デフォルトは a4paper, B5 がよいなら b5paper オプションを class-options: に指定する。

PDF を印刷所に持ち込んだことがないため詳しいことはわからないが, ここで指摘されているようなトンボやノンブルは出力されるように作ってある(そしてここで紹介されているような LaTeX のコマンドの多くは rmdja では書く必要がなくなった)。

TODO: PART の扉ページにはまだノンブルが表示されない

<https://teastat.blogspot.com/2019/01/bookdown.html>

### B.1 画像の配置

現在, PDF で画像の配置を固定する方法について何も特別なものを用意していない。単純に自分は必要だとおもったことがないため。固定したい場合は R Markdown や Bookdown のドキュメントを参考にしてほしい。ただし, 通常は章や部をまたいで表示されることはない(はず)。

### B.2 取り消し線

LaTeX の各パッケージのバージョンによっては, 和文に取り消し線 (\sout) を与えるとタイプセット時にエラーが出ることがある。もともと ulem.sty は欧文を前提にしたものなので適当に妥協してほしい。

### B.3 TODO: しかし英文で書きたい場合

`rmdja` の機能を使いたいが、執筆は英語でしたいと言う場合は最低限以下のような設定変更が必要である。デフォルトは日本語用文書クラスのため、

`documentclass: book / report / article`

など欧文用文書クラスを指定する。

`rmdja` では和文フォントを参照するので、和文フォントの設定も手動で解除する必要がある。

を指定する。そして各種見出しも英文用に調整する。

TODO

## 付録 C

### jecon.bst の紹介

和文と欧文を使い分けたスタイルファイルとして, `jecon.bst` がある. `jecon.bst` の公式ではなく, 私がカスタマイズしたバージョンでも良い. こちらは本来よりも電子媒体としての利用を重視して,

- 参照 URL を表示せず, ハイパーリンクのみにする
- ArXiv ID の表示とハイパーリンク追加

といった変更をしている. 後者は, BibTeX エントリに以下のように `archivePrefix` に `arXiv` と言う値が入っていると, `eprint` の値が ArXiv ID として表示される. これは ArXiv から直接 `.bib` ファイルを取得したり, Zotero などでインポートすれば必ず入力される項目である.

```
archivePrefix = {arXiv},
eprint = {XXXX.YYYYYY},
...
```

TODO: 現在 `jecon.bst` の表示も少しおかしいので確認中.



# 参考文献

Dougherty, Jack and Ilya Ilyankou (forthcoming) "Hands-On Data Visualization Interactive Storytelling from Spreadsheets to Code," retrieved from [here](#).

Wickham, Hadley and Garrett Grolemund (2016) *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*, Sebastopol, CA: O'Reilly, first edition edition, retrieved from [here](#), (黒川利明・大橋真也訳,『Rで始めるデータサイエンス』, オライリー・ジャパン, 2017年,).

Xie, Yihui (2020) *Bookdown: Authoring Books and Technical Documents with r Markdown*: Chapman & Hall, retrieved from [here](#).

Xie, Yihui, JJ. Allaire, and Garrett Grolemund (2018) *R Markdown: The Definitive Guide*, Boca Raton, Florida: Chapman and Hall/CRC, retrieved from [here](#).

Xie, Yihui, Christophe Dervieux, and Emily Riederer (2020) *R Markdown Cookbook*, S.l.: CRC PRESS, retrieved from [here](#).