

R Markdown 入門 (Tokyo.R #91)

ill-identified

最終更新: 2021/4/19, 作成: 2021/4/12

概要

Tokyo.R #91 の『R Markdown 入門』の資料. これ単体でも自己完結的な Rmd 文書のサンプルになっている.

目次

この資料について	4
R Markdown とはなにか	4
1 初期設定	5
1.1 必要なソフト・パッケージ	5
2 始めの一步	9
2.1 最初の文書作成	9
3 Markdown: 本文の書き方	14
3.1 テキストの装飾	15
3.2 ブロック要素	16
3.3 ビジュアルエディタ	20
4 出力文書の変更	22
4.1 出力フォーマット	22
4.2 各種フォーマット	25
4.3 output フィールドの編集について	26
4.4 R Markdown テンプレート	27
5 コードチャンクを使いこなす	28
5.1 チャンクヘッダによるカスタマイズ	28
5.2 行内 R コード	29
5.3 図の掲載	30
5.4 表の掲載	32

6	スライドの作成	34
6.1	概論	34
6.2	xaringan (写輪眼) でできること	35
7	環境設定についての注釈	35
7.1	Linux (RStudio Cloud)	35
7.2	Mac	36
7.3	Windows 10	36
7.4	グラフィック関係	36
7.5	グラフィックのフォント	37
7.6	Python の対応状況	38
7.7	Julia の対応状況	38
7.8	それ以外の言語	38
8	本文の書き方に関する Tips	38
8.1	Markdown でハマりやすい場面	39
8.2	数式	40
8.3	相互参照	41
9	YAML メタデータについてより詳しく	44
9.1	YAML メタデータの基本構文	44
9.2	YAML メタデータでの設定	45
10	画像の出力	46
10.1	グラフィックデバイス	46
10.2	サイズの調整	48
11	PDF 出力の内幕	48
11.1	LaTeX の出力エンジン	49
11.2	R Markdown と LaTeX テンプレートの融合	50
11.3	サポート外の LaTeX エンジンを使用する	52
12	参考文献スタイルのカスタマイズ	52
12.1	参考文献設定の基本事項	52
12.2	default (CSL)	55
12.3	biblatex (BibLaTeX)	56
12.4	natbib (BibTeX)	58
13	その他	59
13.1	R Markdown 関係の役に立つパッケージ	59
13.2	YAML メタデータと出力フォーマットの設定方法について	59
13.3	どうしても rmdja を使いたくないが, PDF で (日本語文書を) 出力したいという人	60

13.4	文書のコンパイル時にエラーがでたら	60
13.5	ハマりがちな LaTeX エラー	61
13.6	<code>rmdja::beamer_presentation_ja</code> で Fira Sans がどうこうという警告がうざい	61
13.7	スクリプトで R Markdown をコンパイルする	62
13.8	R Markdown コンパイル処理の内容	62
13.9	HTML 版の印刷時に色が消える	63
13.10	Rpubs でハイパーリンクに失敗する	64
13.11	その他の関連するドキュメント	64
この文書作成時の環境		64
参考文献		65

この資料について



発表後、このページも自己完結的なチュートリアルとなるように加筆しました。これ以降は、入門者はスライドよりこちらのページに沿って学習すると良いと思います。

Tokyo.R #91 の補足資料として、スライドに書ききれなかった細かい環境設定等も記載した「完全版」です。よって想定読者は元スライドと同じです (= お使いの PC の操作, そして R および RStudio の最低限の操作はわかっているという前提です)。

入門レベルに対応した既存の資料は以下がありますが、ここではチュートリアル風の、よりステップバイステップな説明を意識しています。

- RStudio 公式のチュートリアル (英語)
- kazutan『R Markdown 入門』
- 高橋『再現可能性のすゝめ』

より発展的な多くの問題は、以下を読むことで解決できます。

- Xie et al. (a)『R Markdown クックブック』(翻訳版はこちら)
- Xie et al. (b) “*R Markdown: The Definitive Guide*” (英語)
- Xie (2020) “*bookdown: Authoring Books and Technical Documents with R Markdown*” (英語)
- Yihui 氏の knitr に関するドキュメント (翻訳) (コードチャンク関連)
- The Officeverse (英語) Word や パワーポイントへのエクスポートに関する総合的なドキュメントです。
- **rmdja** パッケージのマニュアル (実際には雑多な技術メモ, 上記どれにも書かれていない問題ならヒントが載っているかもしれません)

特に クックブックの 4 章は基本的な事項がまとまっていますが、日本語圏独特の (ややニッチな) 問題も含め、ここに再編集した内容を残しておきます。

rmdja パッケージの機能がひととおりまとまれば、そのうちパッケージのドキュメントとして再編集するでしょう。

R Markdown とはなにか

R Markdown とは、名前の通り、R と Markdown を統合したものです。Markdown とは、プレーンテキスト (いわゆるメモ帳) で、段落や箇条書きなどがあるリッチテキストを作るための構文です。HTML タグを知っている方は、そのシンプルなバージョンだと考えてください。

Markdown はプログラミング言語ではありませんが、R Markdown は Markdown に R プログラムを埋め込むことができます。よって、執筆中のレポートに、R で書かれた実験や数値計算、分析プログラムの最新の結果

を反映させることができます。例えば、これまで手動で Word や パワーポイントや LaTeX に貼り付けていた数値表やグラフをプログラムごと埋め込むことができます。

ファイル形式も半ば自動的に変換することができます。HTML, PDF, Word (DOCX), パワーポイント (PPTX) などへの変換機能があります。

1 初期設定

R Markdown の入門的な情報のあるページは冒頭に挙げたもの以外にもありますが、今回ここでは、「グラフやデータフレームを表の形で掲載する」といった、R Markdown の典型的な使用例をストレスなく行えるようになることを前提としています。

しかし、現状 R Markdown でこういったことをするのに必要な初期設定は OS など環境によって微妙に異なります (ネット上の設定に関する情報が微妙に食い違っているのもこれが大きな原因の 1 つです)。

そのため、ここで挙げる初期設定作業はやや複雑ですが、なるべく多くの OS でも動くように考慮したものを紹介しています。

1.1 必要なソフト・パッケージ

入門なので最初は細かい話を省きます。4.0.5 以降の R (起動直後のメッセージや `R.version` で分かります。) と、1.4 の RStudio 以降 (バージョンはヘルプや `RStudio.Version()` で分かります) がインストールされている前提です。

以下のリンクから、環境設定用のスクリプトをダウンロードし、`rmd-setup.R` に従ってください。github を使ったことがない方は、以下の画像の位置をクリックして、“Download Zip” を押せば関連ファイルをまとめてダウンロードできます。

ch 0 tags

Go to file

Add file ▾

↓ Code ▾

Update README.md

outputs

発表用スライド追加

RStudio settings

Update README.md

微修正

yesterday

outputs

yesterday

outputs

yesterday

outputs

yesterday

outputs

yesterday

 Clone

?

HTTPS SSH GitHub CLI

git@github.com:Gedevan-Aleksizde/tokyo



Use a password-protected SSH key.

 Download ZIP

<https://github.com/Gedevan-Aleksizde/tokyor-91-rmd>

なお、この文書も R Markdown で書かれており、上記に含まれている `addendum.Rmd` から作成されています。

該当スクリプトファイルの中身をここにも転載しておきます。R のコマンド以外にもやることはあるので、ただコピペして実行するのではなく、注意書きを読みながら実行してみてください。特に、Mac と Linux ユーザの方は外部ライブラリのインストールが必要になりやすいと思います。

RStudio Cloud はおそらく Ubuntu を使用してるので、お使いの方は「Ubuntu なら...」「Linux なら...」と書いてある指示に従ってください。

```
01 ##### (1) RStudio のバージョンについて #####
02 # 1.4.1103 は Windows 版では Python 使用時にエラーが発生します
03 # もし Python を使いたいなら新しいバージョンが出るまで待つか、
04 # 以下の daily build のどれかをインストールしてください
05 # https://dailies.rstudio.com/rstudio/oss/windows/
06
07 ##### (2) パッケージのインストール #####
08 # インストール済みであっても最新版にしておいてください
09 # ダイアログボックスでなにか言われたら NO!
10 install.packages(
11   c(
```

```

12     "tidyverse",
13     "remotes",
14     "rmarkdown",
15     "bookdown",
16     "officedown",
17     "citr",
18     "yamlthis",
19     "svglite",
20     "kableExtra"
21   )
22 )
23
24 ##### (3) ragg インストール #####
25 install.packages("ragg")
26 # ragg のインストール時, Mac/Linux では追加で外部ライブラリのインストールが要求されることがあり
  ↳ ます.
27 # その際は手動でインストールしてください. おそらくは以下のような操作になります.
28 #
29 # 例えば Ubuntu なら
30 # sudo apt install libharfbuzz-dev libfribidi-dev
31 #
32 # Mac なら homebrew でインストールします
33 # brew install harfbuzz fribidi
34
35 ##### (4) PDF 画像の準備 #####
36 # さらに PDF で画像を出力したい場合は, X11 と Cairo が必要です.
37 # Windows の場合, 以下が TRUE になっていることを確認してください
38 capabilities()[c("cairo")]
39 # Mac や Linux の場合は, 両方が TRUE になっていることを確認してください.
40 capabilities()[c("X11", "cairo")]
41
42 # Windows や多くの Linux 系はあまり気にしなくても良いですが,
43 # 最近の Mac はデフォルトで必要なプログラムが入っていないようです.
44 # Mac は以下の 2つをインストールすれば使えます (インストールには homebrew が必要です).
45 # ただし, xquartz のほうはうまく行かない例が報告されています.
46 # https://www.xquartz.org/ で dmg ファイルをダウンロードしてインストールすることも試してくだ
  ↳ さい.
47 #

```

```

48 # brew install cairo
49 # brew cask install xquartz
50
51 ##### (5) rmdja パッケージのインストール #####
52 # PDF は設定が複雑なので、私の作成した rmdja パッケージを使うことをお勧めします。
53 # このセッション時点では最新版は v0.4.5 です
54 remotes::install_github("Gedevan-Aleksizde/rmdja", upgrade = "never")
55
56 ##### (6) TeX のインストールします #####
57 # これはすでにインストールしている人、PDF 文書の作成を目的としていない人は不要です
58 # それなりに時間がかかるので注意してください
59 tinytex::install_tinytex()
60 tinytex::tlmgr_install("texlive-msg-translations")
61
62 ##### (7) 共通フォントのインストール (Linux のみ) #####
63 # 以降の説明を簡単にするため、Linux でのフォントを共通化します。
64 # これは Linux 系 OS をお使いの方のみ必要です。
65 # Linux 系 OS をお使いならば、Noto フォントをおすすめします。
66 # 例えば Ubuntu (RStudio Cloud も Ubuntu OS です) ならば以下でインストールできます
67 # sudo apt install fonts-noto-cjk fonts-noto-cjk-extra
68
69
70 # ---- ここで念の為 RStudio 再起動 ----
71
72 # PDF の閲覧は okular が便利です。MS Store で入手できます
73 # https://www.microsoft.com/ja-
74   ↪ jp/p/okular/9n41msq1wnm8?rtc=1&activetab=pivot:overviewtab
75 #
76 # Sumatra は軽量ですが、フォントの埋め込みを確認できません。
77
78 # ----- 以下は基本チュートリアル範囲ではあまり取り上げませんが、便利な拡張パッケージです
79
80 # Python 使いたい人へ
81 install.package("reticulate")
82
83 # Julia 使いたい人へ
84 install.package("JuliaCall")

```



```

85
86 # ragg が使えない /Linux 以外で PDF 形式の画像で文字化けを防ぎたい場合は以下を試してください.
87 remotes::install_github("Gedevan-Aleksizde/fontregisterer", upgrade = "never")
88
89 install.packages(c(
90   "xaringan",
91   "bookdownplus",
92   "blogdown",
93   "pagedown"
94 ))
95
96 # "word" 関連
97 install.packages(
98   c(
99     "officer", "rvgl", "openxlsx",
100    "ggplot2", "flextable", "xtable", "rgl", "stargazer",
101    "tikzDevice", "xml2", "broom"
102   )
103 )
104 # パワーポイントや Word にグラフをエクスポートする
105 remotes::install_github("tomwenseleers/export")
106 # Word の更新差分を考慮して編集できる (ただし現在開発停止中)
107 remotes::install_github("noamross/redoc")

```



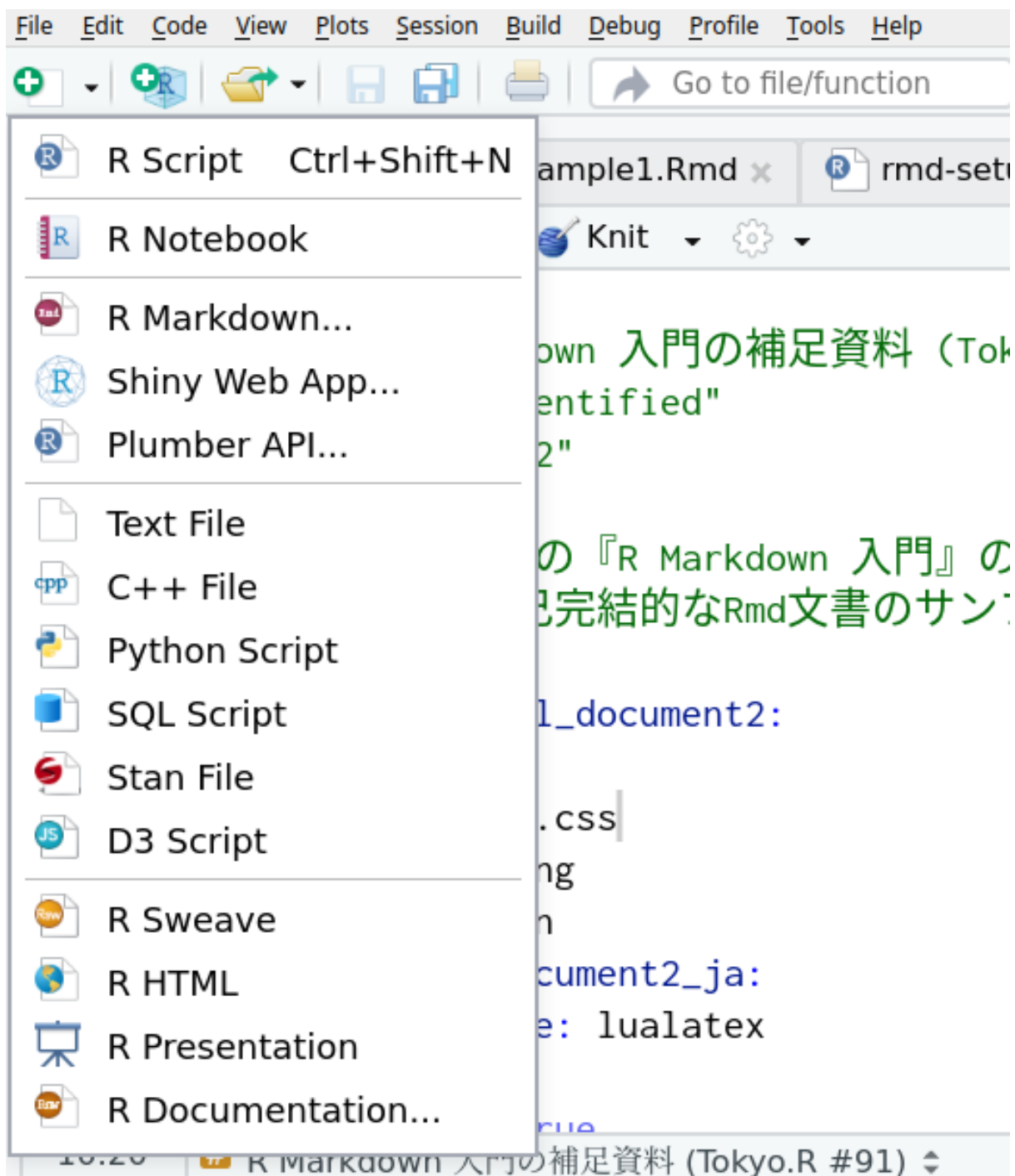
上記のインストール手順が難しくて分からないという方は、少なくとも R Markdown で最低限のチュートリアル操作をするのに必要な **rmarkdown** と **bookdown** と **tinytex** がインストールされ、かつ **tinytex** で TeX をインストールできていることを確認してください。ただし、この場合はチュートリアルで紹介する機能の一部が使えない可能性があります。

2 始めの一步

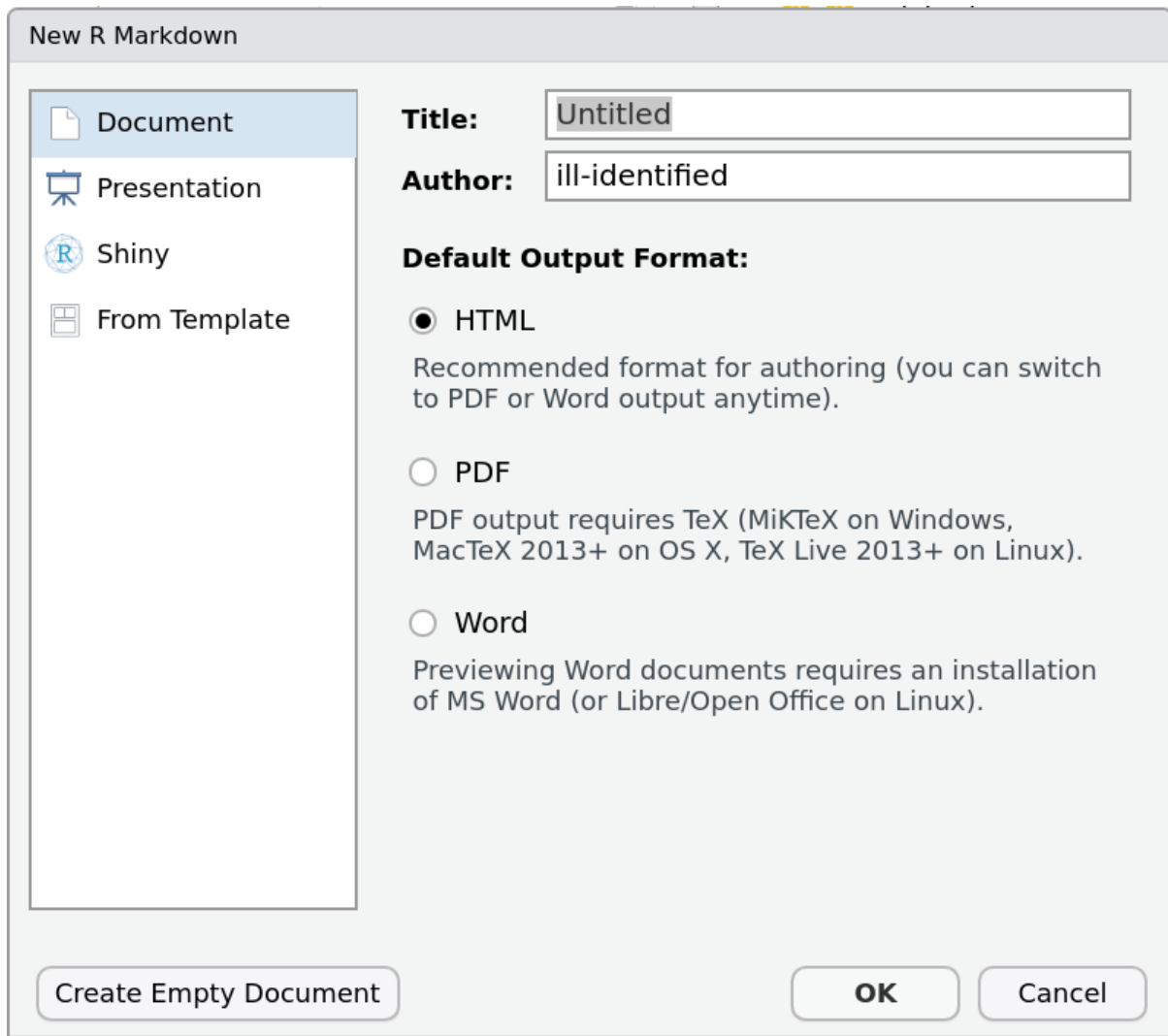
2.1 最初の文書作成

まずは文書を作成するまでをやってみましょう。

RStudio の左上のボタンを押し、R Markdown ファイルを新規作成します。「R Markdown...」がそれです。

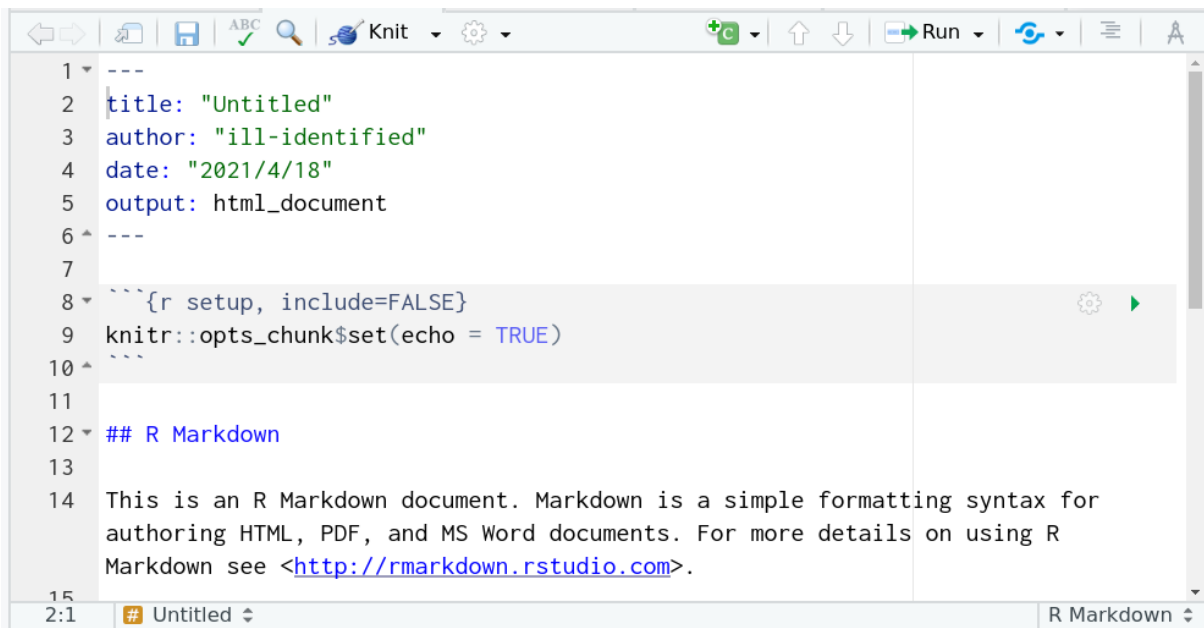


ウィンドウが現れます。デフォルトでは左側は“Documet,” 右側のラジオボタンは“HTML”を選択しているはずですが、このデフォルトの選択で決定してください。



新しい R Markdown ファイルが開かれます。すでに何か書かれているはずです。これは初心者のための出力サンプルです。これを適当な名前で保存してください。なお、R Markdown ファイルの拡張子は `.Rmd` です。

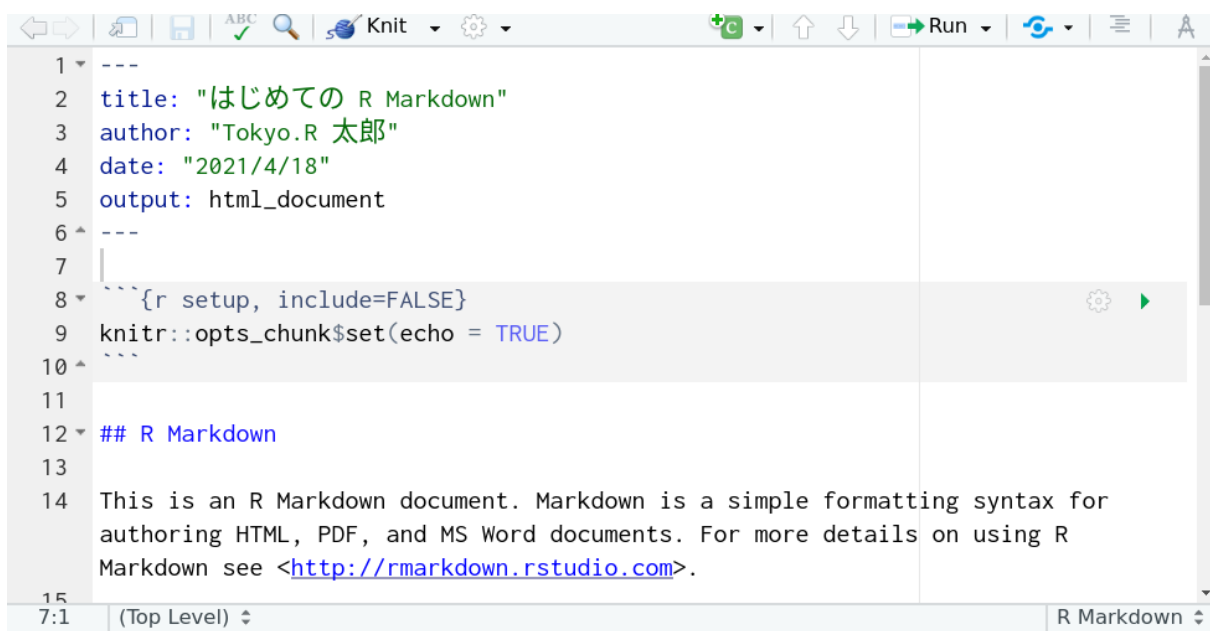
まずは文書のタイトルと著者を書き換えてみましょう。冒頭の `---`、`---` で囲まれた部分を書き換えてみましょう。`title:`、`author:` の右側の部分です。タイトルは “Untitled”，ユーザー名はおそらくあなたの PC アカウント名になっているでしょう。



```
1 ---
2 title: "Untitled"
3 author: "ill-identified"
4 date: "2021/4/18"
5 output: html_document
6 ---
7
8 {r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
    authoring HTML, PDF, and MS Word documents. For more details on using R
    Markdown see <http://rmarkdown.rstudio.com>.
15
2:1 # Untitled R Markdown
```

これを以下のように書き換えてみましょう (他のタイトルや名前にしてもいいです.)

```
title: "はじめての R Markdown"
author: "Tokyo.R 太郎"
```



```
1 ---
2 title: "はじめての R Markdown"
3 author: "Tokyo.R 太郎"
4 date: "2021/4/18"
5 output: html_document
6 ---
7
8 {r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
    authoring HTML, PDF, and MS Word documents. For more details on using R
    Markdown see <http://rmarkdown.rstudio.com>.
15
7:1 (Top Level) R Markdown
```



コロン (:) の後にはスペース 1 つ以上を空けることを忘れないでください

書き換えたら、上記を一旦保存します。名前は何でもいいです。拡張子は自動的に .Rmd となるはずです。

では、Rmd ファイルから文書を生成させます。この処理を**コンパイル**といいます。文書のコンパイルは、

RStudio のエディタの上にある、 ボタンを押します。

下のタブでなにやらログが流れた後、新しいウィンドウに生成された HTML 文書が生成されるはずです。また、同じフォルダに .Rmd の拡張子が .html となったファイルもできています。

はじめての R Markdown

Tokyo.R 太郎

2021/4/18

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed      dist
## Min.   : 4.0    Min.   : 2.00
## 1st Qu.:12.0    1st Qu.: 26.00
## Median :15.0    Median : 36.00
## Mean   :15.4    Mean   : 42.98
## 3rd Qu.:19.0    3rd Qu.: 56.00
## Max.   :25.0    Max.   :120.00
```



Including Plots

You can also embed plots, for example:



さて、これで初めての **R Markdown 文書の作成に成功しました!**



R Markdown では基本的に、 ボタンを押すタイミングで Rmd ファイルの内容が反映されます。^a よって、Rmd ファイルの中身を書き換えた場合、 ボタンを再度押さなければ内容が反映されません。

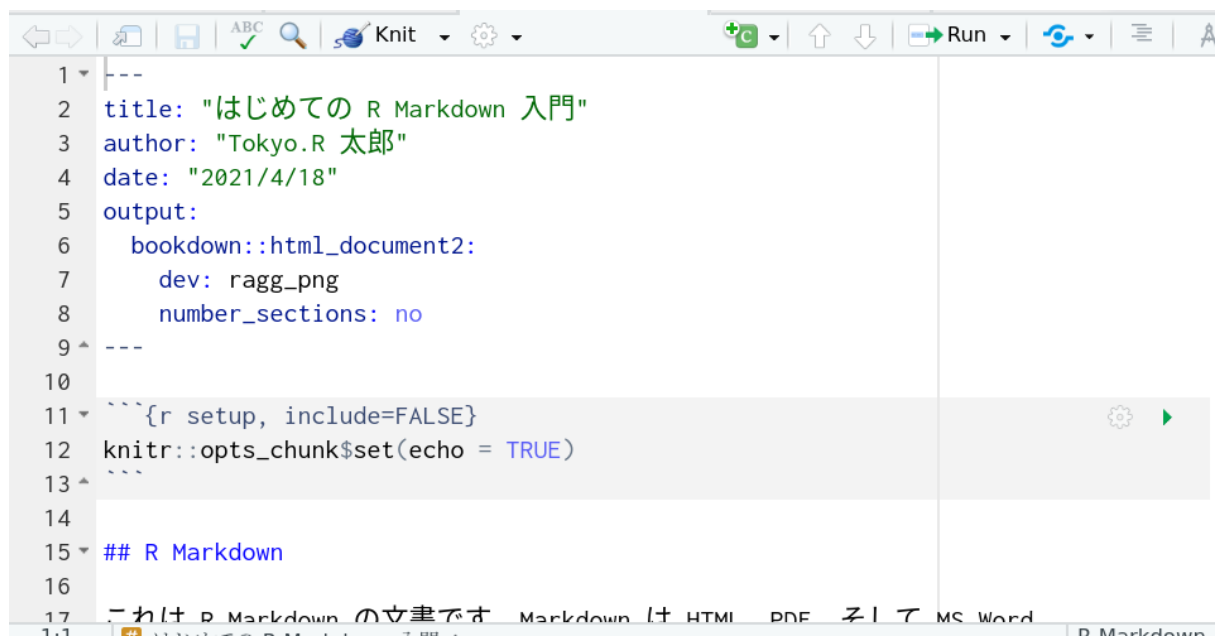
^a 例外は R Notebook です。

しかし、これだけでは R Markdown をレポート作成に活用できたとは言えません。というわけで、以降では基本的な使い方を紹介していきます。

3 Markdown: 本文の書き方



デフォルトのサンプル文は英語です。とっつきにくいのでこれを翻訳&微修正したバージョンを冒頭の資料に同梱しました。ファイル名は `example1.Rmd` です。以降はこれをコピーしたものを使用してください。



```
1 ---
2 title: "はじめての R Markdown 入門"
3 author: "Tokyo.R 太郎"
4 date: "2021/4/18"
5 output:
6   bookdown::html_document2:
7     dev: ragg_png
8     number_sections: no
9 ---
10
11 {r setup, include=FALSE}
12 knitr::opts_chunk$set(echo = TRUE)
13
14
15 ## R Markdown
16
17 これは R Markdown の文書です。Markdown は HTML、PDF、そして MS Word
```

まずは R Markdown 上で本文を書く方法です。--- 以下の部分は全て本文になります。

本文は Markdown と呼ばれる記法で書きます。これは HTML タグのようにプレーンテキストを装飾する記号ですが、**HTML タグよりシンプルで、見やすいはず**です。ふだんメモ帳で書くような構文になることを意識されています。

最近では、Markdown は R に限らず様々なサービスで使われています。たとえばチャットツールの Slack の文

字の装飾も Markdown で書くことができます。

以下に紹介する記法を, `example1.Rmd` 上でいろいろと試してみてください (ほとんどはデフォルトの `.Rmd` でも使えます)。

3.1 テキストの装飾

3.1.1 文字のスタイル変更

文字の一部のスタイルを変更します。HTML でいえば `` です。

文字の斜体 / イタリックによる強調は, アスタリスク (*) で文字列を囲みます。このように表示されます。そして太字は, アスタリスク 2 連続 (**) で囲みます。このように表示されます。この 2 つの記法は * だけでなく, アンダースコア _ でもできます。

また, プログラムの関数であることを強調したい場合はコード要素が良いでしょう。バッククオート (```, 日本語キーボードなら `shift + @` で入力できます) で囲みます。このように: `data.frame()`。なお, この方法でコードを書いても, 実行されるわけではありません。文書内でコードを実行する方法は, 後述のセクション 5 で紹介します。

3.1.2 ハイパーリンク・画像の貼り付け

ハイパーリンクは [テキスト](URL) で書くことができます。このように Rpubs のこのページへのリダイレクトができます。代替テキストが不要なら, URL をそのまま書き込んでもよいです: `https://rpubs.com/ktgrstsh/755893`。

まず, 画像ファイルの貼り付けは ! [代替テキスト] (画像のファイルパス) のような構文です。つまりハイパーリンクの先頭に ! が付いただけです。

! [Tokyo.R] (img/logo.png)



図1 Tokyo.R

サイズの細かい指定やグラフの貼り付けは, 後述するコードチャンクを使う必要があります。

3.1.3 任意の位置で改行

ほとんどのワードプロセッサは自動で行を折り返してくれますが、場合によっては手動で改行位置の調整が必要になります。しかし、Rmd ファイル上の改行は改行と認識されません。このような場合、**行末にスペース 2 つを追加**すると強制改行されます。

Rmd ファイルに以下のように書くと...

****行の末尾にスペースを 2 つ**追加すると改行できます。**

(ここで改行されているはずですが) このように

こうなるはずですが、

行の末尾にスペースを 2 つ追加すると改行できます。

(ここで改行されているはずですが) このように

スペースの有無はわかりにくいですが、実際に試してみればわかるはずですが、

また、空白行を 1 つ挟んでも改行されますが、正確にはこれは行を改めているのではなく**段落**を改めています。これはブロック要素といいます。

3.2 ブロック要素

文中の一部ではなく、独立した文のまとまりを定義することもできます。これらをまとめてブロック要素と呼びます (逆にさっきのようなものは「**インライン要素**」と呼びます)。ブロックとは段落 (パラグラフ) のようなものです。

もっとも基本的なブロック要素は、まさに**段落の区切り**です*¹。1 行以上の空白行をはさむと、それが段落の区切りとなります。それ以外のブロック要素も全て、改行がブロックの単位を決めることになります。この文章はこの次に空白行を挟んでいるので、段落が変わります。

ここからは新しい段落です。



ブロック要素の共通点として、**前後を空白の行で 1 行以上空けなければブロック要素として認識されません。**

3.2.1 見出しを書く

この資料のように、見出しを付けたい場合は 行の先頭に # を付け、その直後に**スペースを 1 つ空けて**書きます。

Rmd 上でこのように書きます。

*¹ HTML では、日本語文書のルールである「1 文字下げ」は用意されていないので、見た目の変化としてはすこし空白が広い程度のもとなります。

イン트로ダクション

本文

次のセクション

本文

を増やし, ##, ### と書くと見出しの階層が下がっていきます. これは HTML の <h2>, <h3>, ... と同様です.

3.2.2 箇条書き

箇条書きをしたい場合は, アスタリスク * を先頭に付けます. 強調と区別するため, これも直後にスペースを 1 つ空けて書きます.

Rmd 上でこう書いたとします.

- * 箇条書き
 - * 一段下げた箇条書き
- * 箇条書き

それがこう表示されます.

- 箇条書き
 - 一段下げた箇条書き
- 箇条書き

箇条書き記号は * の代わりにハイフン (-) やプラス (+) で書いても同様に使えます.

数字付きの箇条書きは, 1. のように数字の後にドット (.) を付けます.*² これは文字通り 1. と書き続けても勝手に数字をカウントしてくれるので便利です. 逆に, 1 以外の数字を書くことでカウントを操作することもできます.

Rmd 上でこう書いたとします.

1. その 1
 1. 一段下げたその 1
1. その 2
1. その 3

それがこう表示されます.

1. その 1

*² アルファベット順にすることもできますが, これは出力形式によっては機能しないことがあります.

1. 一段下げたその 1
2. その 2
3. その 3

3.2.3 コードブロック

インライン要素のコードの強調 ` は、長いコードの表示に不向きです。

```
data(mtcars)
fit <- lm(mpg ~., data = mpg)
summary(fit)
```

このように見づらく、1 行ごとにバッククオートや改行構文を書かねばなりません。複数行にわたるコードを書く場合は、インラインのコード強調ではなく、ブロック要素のコード強調、コードブロックを使います。コードブロックはバッククオート 3 連続 (```) で囲んだ部分です。この内部では改行もそのまま表示されます。

Rmd ファイル上では以下のように書きます。

```
```
data(mtcars)
fit <- lm(mpg ~., data = mpg)
summary(fit)
```
```

これが実際にはこう表示されます。

```
data(mtcars)
fit <- lm(mpg ~., data = mpg)
summary(fit)
```

なお、R コードの埋め込みは**この方法ではできません**。これはコードの表示のみです。中身を実行したい場合は、この次のコードチャンクのセクションを見てください

3.2.4 コードチャンク (第一歩)

コードブロックはコードを掲載するだけで、実行はできませんでした。実行する機能があるのは**コードチャンク**で、これが R Markdown を使う大きなメリットの 1 つです。

コードチャンクの書き方はコードブロックと似ていますが、R として実行することを意味するため、```` の後にさらに {r} を付けます。以下は example1.Rmd にも含まれているコードチャンクです。

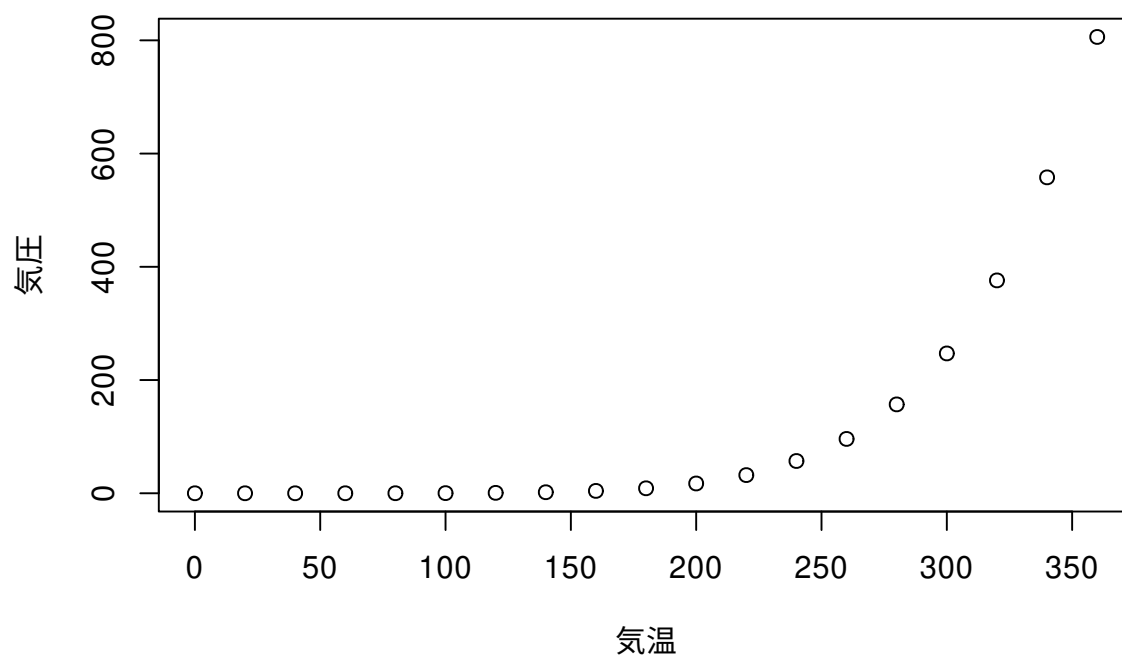
```
````{r}
summary(cars)
```
```

すると、以下のように表示されます。

```
01 summary(cars)
```

```
##      speed      dist
## Min.   : 4.0   Min.   :  2.00
## 1st Qu.:12.0   1st Qu.: 26.00
## Median :15.0   Median : 36.00
## Mean   :15.4   Mean    : 42.98
## 3rd Qu.:19.0   3rd Qu.: 56.00
## Max.   :25.0   Max.    :120.00
```

また、グラフも表示されます。



ふだん R で見ているものと全く同じものが出力できました。しかし、あなたが作りたいレポートは、**プログラム部分の表示が不要**だったり、**コードのみ**だったり、**作成したグラフや、データフレームをカッコよく整形した表**だったりするかもしれません。それらも**全て可能**です。コードチャンクでできることは多彩ですが、その説明は一旦、セクション5に後回しにします。

3.3 ビジュアルエディタ

Markdown はこのように便利ですが、プレーンテキストで書くのがどうしても苦手という場合は、**ビジュアル Markdown エディタ**という Word 風のエディタがあります。これは RStudio 1.4 以降で提供されるようになりました。

右上のコンパスのようなアイコン  をクリックしてください。



```
1 ---
2 title: "はじめての R Markdown 入門"
3 author: "Tokyo.R 太郎"
4 date: "2021/4/18"
5 output:
6   bookdown::html_document2:
7     dev: ragg_png
8     number_sections: no
9 ---
10
11 {r setup, include=FALSE}
12 knitr::opts_chunk$set(echo = TRUE)
13
14
15 ## R Markdown
16
17 これは R Markdown の文書です。Markdown は HTML, PDF, そして MS Word
18 文書を執筆するためのシンプルな構文です。R Markdown の詳細は <http://rmarkdown.rstudio.com>
19 をご覧ください。
20
21 **knit** ボタンをクリックすると、文書内の本文も埋め込んだ R
```

すると、以下のような Word 風の画面になります。フォントスタイルやいろいろな項目の画像の挿入がツールバーから行えるようになっています。

The screenshot displays the RStudio interface with two windows. The top window, titled 'example1.Rmd*', shows the R Markdown source code. It includes a YAML header with title, author, date, and output settings, followed by a code chunk for setup and a paragraph of Japanese text. The bottom window shows the R console with the output of the setup chunk, including the loading of the 'shiny' package.

Top Window: example1.Rmd*

```

---
title: "はじめての R Markdown 入門"
author: "Tokyo.R 太郎"
date: "2021/4/18"
output:
  bookdown::html_document2:
    dev: ragg_png
    number_sections: no
---

{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

# R Markdown
これは R Markdown の文書です. Markdown は HTML, PDF, そして

```

Bottom Window: Console

```

/ media/ks/DATA/User/Documents/Documents/developping/tokyor-91-rmd/
19 300 000.0000
> ?mtcars
> citr:::insert_citation()
Loading required package: shiny

```



Right Panel: R Markdown

R Markdown
グラフを含めよう

Insert Menu:

- Any... (Ctrl+/)
- Code Chunk
- Citation... (Ctrl+Shift+F8)
- Cross Reference
- Footnote (Ctrl+Shift+F7)
- Image... (Ctrl+Shift+I)
- Link... (Ctrl+K)
- Horizontal Rule
- Definition
- Inline Math
- Display Math
- Special Characters
- Paragraph
- Code Block... (Ctrl+Shift+\)
- Div...
- YAML Block
- Comment (Ctrl+Shift+C)

ただし、ここでは出力が HTML なので、RStudio 上の見た目と全く同じ文書が生成されるわけではないことに注意してください。他の媒体でも同様です。

ビジュアルエディタをやめて通常のエディタに戻したい場合は、コンパスのボタンを再度クリックしてください。右側に目次 (アウトライン) の表示が残ったままかもしれません。  の左隣にある  アイコンが、アウトラインの表示・非表示を切り替えるボタンです。Markdown の構文を覚えられてないうちは、これで構文の使い方を真似るとよいかもしれません。

これは個人的な意見ではありますが、Markdown 構文はとてもシンプルで効率的に書けますので、R Markdown をよく使う予定ならばなるべく早めに Markdown を覚え、ビジュアルエディタモードを卒業することをおすすめします。



ビジュアルエディタはファイルを自動で整形するため、通常のエディタと何度も切り替えると不具合を起こす可能性があります。1つのファイルに対して、どちらか1つのモードしか使わないことをお勧めします。

4 出力文書の変更

先ほどのチュートリアルでは HTML 文書を作成しました。しかしみなさんがほしいのは Word かもしれませんが、PDF かもしれません。rmarkdown パッケージは以下をサポートしています。

- HTML (より詳しく言うと、Bootstrap3 ベース)
- PDF (LaTeX を利用)
- MS Word (当然ですが DOCX 形式のみです)

さらに、プレゼン用スライドなど、体裁に応じたスタイル変更もできます。

この出力の変更は、再び冒頭の設定を書き換えることで実現できます。

4.1 出力フォーマット

デフォルトの文書では、冒頭に

```
title: "..."  
author: "..."  
date: "..."  
output: html_document
```

と書かれていました。また、今回提供している example1.Rmd には

```
output:  
  bookdown::html_document2:  
    dev: ragg_png  
    number_sections: no
```

と書かれています。この `output:` という項目が、文書の出力を決めるもっとも重要な部分です。ここに指定されているものを、**出力フォーマット関数**または単に**出力フォーマット**といいます。

rmarmkdown パッケージには、例えば以下のような出力フォーマットが用意されています。

- `html_document`: HTML 文書
- `html_notebook`: Notebook 文書, リアルタイムで更新される
- `pdf_document`: PDF 文書
- `word_document`: Word (DOCX) 文書
- `slidy_presentation`: HTML 形式のスライド
- `beamer_presentation`: PDF 形式のスライド

「出力フォーマット関数」という名前が示すように、実態は R の関数です。細かい仕様は関数のヘルプとして見ることができます。


では、試しに DOCX 形式にしてみましょう。Word がインストールされていなくても変換できますし、閲覧には Libre Office などが使えます*3。

`example1.Rmd` の冒頭の

```
output:
  bookdown::html_document2:
    dev: ragg_png
    number_sections: no
```

の `output:` 以下の行を、このように書き換えて保存します (複数行あります。消し忘れないように)。

```
output: word_document
```

そして  します。*.docx ファイルが作成されるはずです (お使いの環境によりませんが、今回は自動で開かれないかもしれません)。

なお、Word (`word_document`) や パワーポイント (`powerpoint_presentation`) などの Office 系出力フォーマットは、HTML や PDF と比べて細部のスタイルが崩れやすい傾向にあります*4。しかし PDF や HTML と違い DOCX や PPTX は編集用ファイルなので、編集が容易です。よって手動で調整する想定になっています。

開くとエクスポートされているはずです (以下は Libre Office で開いた場合です)。

*3 ただし、Libre Office では表示が完全でないことがあります

*4 これは内部で使用している Pandoc などの影響もあります。新しいバージョンの Pandoc を使用したり、パッケージの開発版を使用することで改善されることもあります。しかしそれは「入門チュートリアル」の範疇を超えるためここには書きません。

はじめての R Markdown 入門

Tokyo.R 太郎

2021/4/18

R Markdown

これは R Markdown の文書です。Markdown HTML, PDF, そして MS Word 文書を執筆するためのシンプルな構文です。R Markdown の詳細は <http://rmarkdown.rstudio.com> をご覧ください。

knit ボタンをクリックすると、文書内の本文も埋め込んだ R コードチャンクも全て含まれた文書が生成されます。R コードチャンクをこのように埋め込むことができます。

```
summary(cars)
```

```
##      speed      dist
## Min.   : 4.0    Min.   : 2.00
## 1st Qu.:12.0    1st Qu.: 26.00
## Median :15.0    Median : 36.00
## Mean   :15.4    Mean   : 42.98
## 3rd Qu.:19.0    3rd Qu.: 56.00
## Max.   :25.0    Max.   :120.00
```

グラフを含めよう

グラフを埋め込むこともできます。これが例です

では PDF はどうでしょうか。ここまでの流れから、以下のようにすれば良さそうです。

```
outout: pdf_document
```

しかし、実際にはなにかよくわからないエラーが表示され、うまくいかないか、うまく表示されないと思います。とりあえず出力したい場合は、

```
output: rmdja::pdf_document2_ja
```


表 1 主な出力フォーマット

| ファイル形式 | フォーマット関数名 |
|---------------|--|
| PDF | <code>'rmdja::pdf_document2_ja'</code> |
| HTML | <code>'bookdown::html_document2'</code> |
| Word (DOCX) | <code>'officedown::rdocx_document'</code> |
| スライド資料 (PDF) | <code>'rmdja::beamer_presentation_ja'</code> |
| スライド資料 (HTML) | <code>'xaringan::tsukuyomi'</code> |
| スライド資料 (PPTX) | <code>'officedown::rpptx_document'</code> |

とすれば動くでしょう。



今回の入門で初めて TeX をインストールした場合、PDF の初回コンパイルには時間がかかるかもしれません。

これは見ての通り **rmdja** パッケージを利用しています。もしインストールしていないのなら、YAML メタデータに次のような設定をすることで、**rmarkdown** のみで一応は日本語を表示することができます。

```
output:
  pdf_document:
    latex_engine: lualatex
documentclass: ltjsarticle
classoption: haranoaji
```

しかし、平文のみのテキストなど簡単な文書ならよいですが、いろいろ使っているうちに問題が出てくるとおもいます。日本語 PDF をうまく出力するにはいろいろな設定が必要になるためです。PDF 出力のカスタマイズはかなり複雑です。それはセクション 11 で紹介します。

スライドもフォーマットの書き換えで作成できますが、いくつか注意点があります。しかしそれは後のセクション 6 で紹介します。

4.2 各種フォーマット

他にもいろいろなパッケージがカスタマイズされた出力フォーマットを提供しています (表 1)。2021/4/17 時点では、以下がおすすめです。冒頭の環境設定ができていれば、必要パッケージは全て入っているはずです。

rmarkdown 本体の出力フォーマットが 1 つもありますが、それは次のような理由です。

- デフォルトのフォーマットに対して、**bookdown** の `...2` というフォーマットは相互参照機能が追加されている

- PDF 形式のフォーマットはどれも、日本語の文書として自然な形にするために必要な設定項目が多すぎる
- **officetdown** パッケージは、Word や PowerPoint のデフォルトの形式にスタイルの適用機能などが追加されている
- **xaringan** パッケージは、より便利な構文が追加されている

4.3 output フィールドの編集について

Rmd ファイルの冒頭の `---` で囲まれた部分は ^{ヤメル}YAML メタデータといい、文書の出力を決めるいろいろな設定項目があります。しかし、チュートリアル範囲ではほぼ `title`, `author`, `output` しか使いません。YAML メタデータでできることは多いですが、`output` が特に重要なので、基本チュートリアルではここだけ解説します。YAML メタデータの詳細はセクション 9 で独立して解説します。

`output` にはこのように出力フォーマットを指定できます。そして出力フォーマットは R の関数なので、さまざまなオプションを与えることができます。example1.Rmd の

```
output:
  bookdown::html_document2:
    dev: ragg_png
    number_sections: no
```

にある、`number_sections: no` というのは実は出力フォーマット `bookdown::html_document2()` に引数 `number_sections = FALSE` を与えているのと同じです。YAML メタデータの構文では、オプションを与える場合はこのように、以下のような書き方のルールがあります。

1. `output:` の直後に改行
2. インデントしてから出力フォーマット関数を書く
3. 出力フォーマット関数の後にも `:` を与える
4. さらに改行&インデントしてからオプション引数を書く
5. オプション引数も `=` ではなく `:` で代入する

特に、**YAML メタデータはインデントを揃えることが重要**です。R ではインデントをほとんど気にしないので、初心者はここでよく失敗します。



よって、複数のオプションを与える場合は例えば以下のように書き連ねることになります (以下で使われているオプションの意味はここでは解説しません。ヘルプなどを参照してください)。

```
output:
  bookdown::html_document2:
    number_sections: no
```

```
toc: no
code_folding: show
dev: ragg_png
```

さらに、複数のフォーマットも指定できます。

```
output:
  officedown::rdocx_document: default
  bookdown::html_document2:
    number_sections: no
    toc: no
    code_folding: show
    dev: ragg_png
```

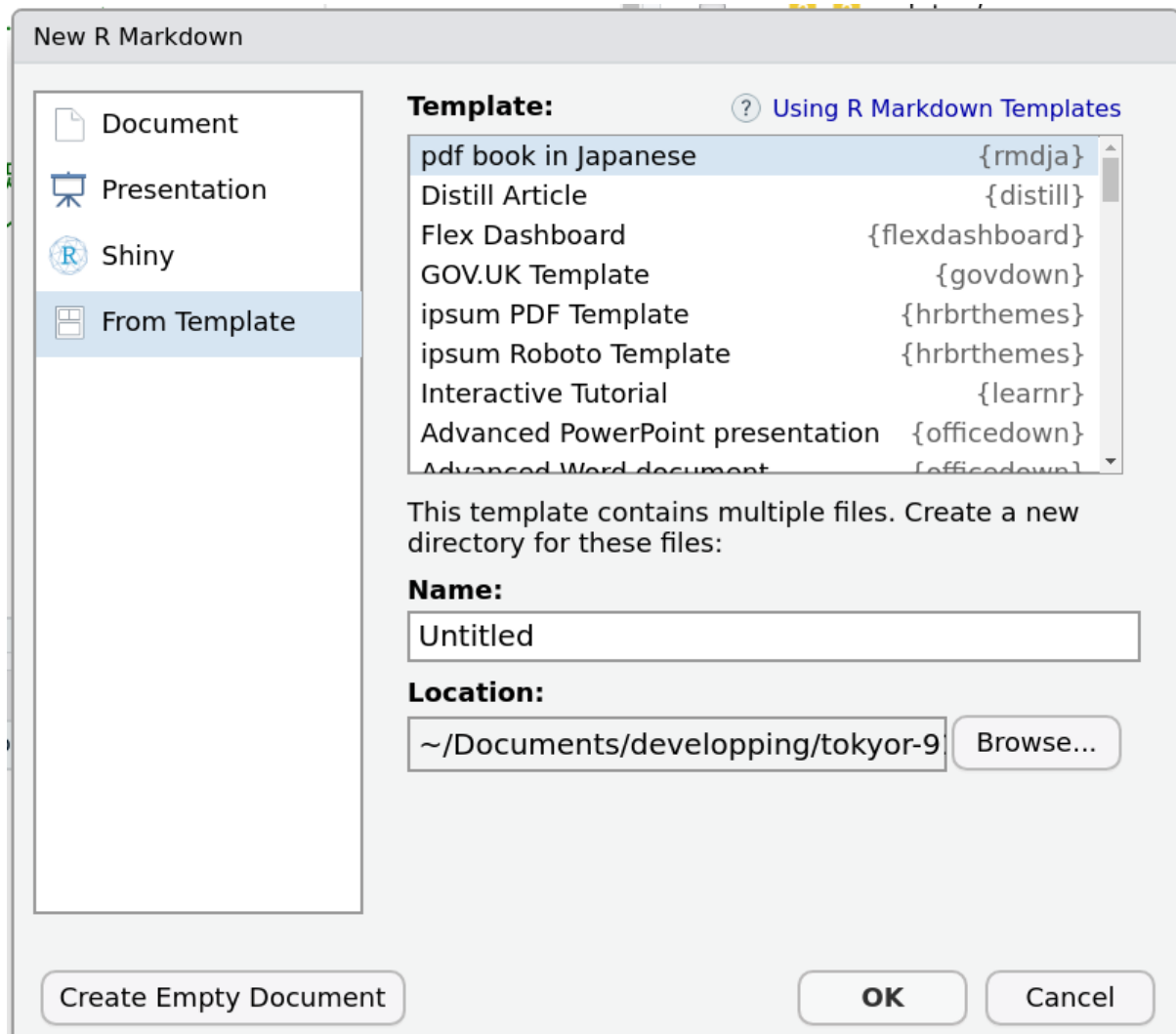
その場合、 で作成されるのは一番上のフォーマットのみです。 ボタンの横の三角形のボタンを押すことで、2 番目以降のフォーマットを選択できます。



R Markdown には便利な出力フォーマットが多数存在し、また上記のように複数のフォーマットを指定できますが、現状は 1 つの Rmd ファイルから HTML, PDF, DOCX など異なるファイルタイプでそれぞれ同じような出力を実現することはかなり困難を伴います。特に、Word は特殊です。熟練しないうちは、Rmd ファイル 1 つに対してどれか 1 種類だけを生成する用途に使うと良いでしょう。

4.4 R Markdown テンプレート

rmarkdown を含むいくつかのパッケージには、いろいろな項目を設定済みのテンプレートを提供しています。新規作成のとき、左側の “From Templates” を選ぶと、右側にテンプレートの名称と、提供元のパッケージ名が表示されます。



5 コードチャンクを使いこなす

セクション 3.2.4 ではコードチャンクの最低限の使い方のみ紹介しましたが、常にコードとテキストを出力するだけでは使いにくいと思います。そこで、より実用的な使い方を紹介します。

5.1 チャンクヘッダによるカスタマイズ

コードチャンクの先頭にある `{r}` は、**チャンクヘッダ**といい、括弧内にそのチャンクの設定をいろいろと書き加えることができます。

基本的な書き方は以下です。

```
{r LABEL, echo = F, eval = T, ...}
```

先頭の `r` は R 言語での実行を意味するため、必須です。次の LABEL はチャンクラベルといい、このチャンクの名前です。好きな文字列を書けます。しかし、チャンクの識別のために名前が重複しないようにしてください。ほとんどの場合は、後述する相互参照の時に使います。それ以外では省略してもかまいませんが、エラーが出た時にチャンクラベルを決めていると発生箇所が分かりやすいです。それ以降にチャンクオプションを追加できます。R の関数の引数と同様に、必ずカンマ (,) で区切ってください。R の関数の引数というのがポイントで、ここには R の引数と同じように文字列は引用符で囲む必要があったり、R のコードを与えたりすることもできます。

代表的なチャンクオプションの使い方に、以下があります。

- `echo = F`: コードを表示しない、スライドでは邪魔なのでよく非表示にします
- `eval = F`: コードを実行しない、コードだけ見せたい時に
- `fig.cap=""`: 図のキャプション
- `fig.align = "center"`: 図を中央揃えにする (出力フォーマットでも一括設定できることがあります)

例えばサンプルには以下のようなチャンクがあります。

```
```${r pressure, echo = F}
plot(pressure, xlab="気温", ylab="気圧")
```
```

この場合、`pressure` がラベルで、そしてコードは表示せず結果だけ表示したいので `echo = F` が指定されています。


使用可能なチャンクオプション一覧は knitr のドキュメントの翻訳 を参照して下さい。

5.2 行内 R コード

コード強調のように、ブロック要素ではなく行内に簡単な R コードを含めることができます。``r 1 + 1`` のように書くと、行内にコードの結果が出力されます。例えば、「`1 + 1 = 2`」

この最もよくある活用法は、文書の日付を自動更新することです。YAML メタデータの `date` は日付ですが、ここにも R コードを埋め込みます。

```
date: "`r Sys.Date()`"
```

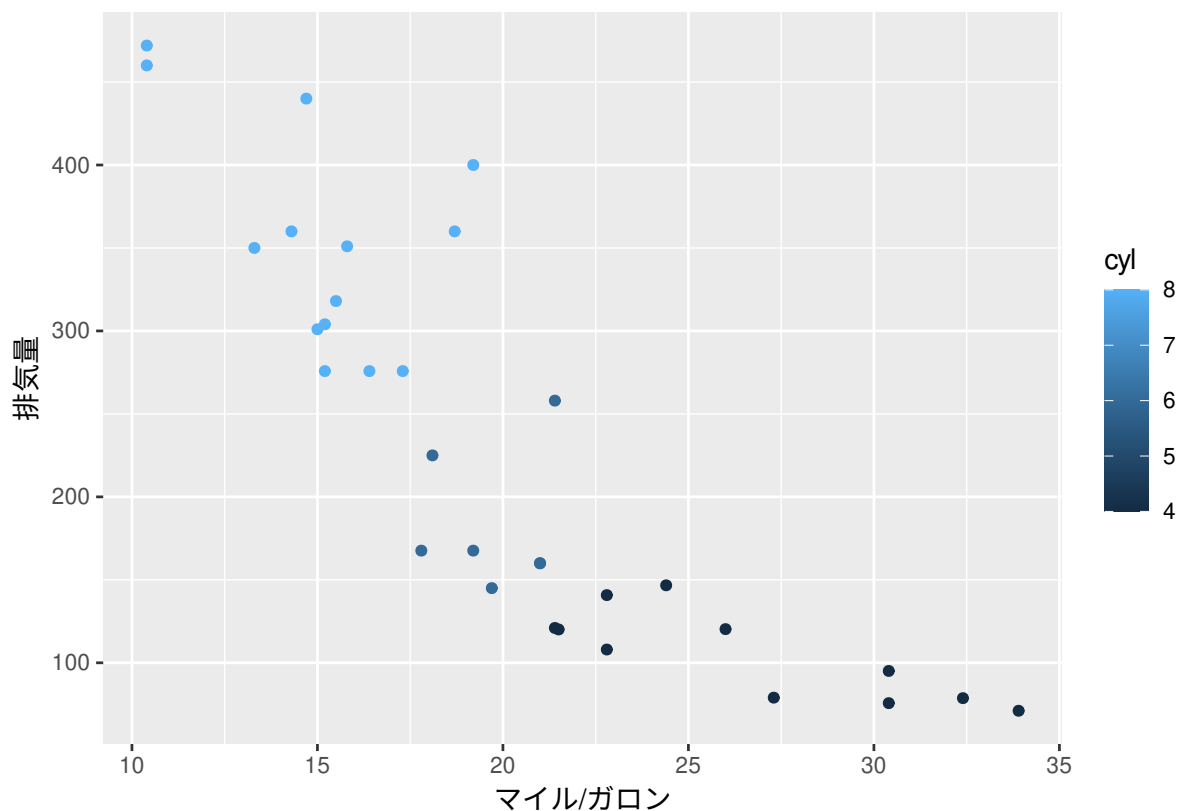
これで、日付が  したタイミングで自動更新されます。ただし、ここでは YAML の構文ルールの影響で、引用符で囲む必要があります。

5.3 図の掲載

図や表を, R Markdown で簡単に表示することができます. グラフは既に例 1 を 1 つ見せましたが, もう少し紹介します.

サンプルでもすでにグラフが表示されていました. 基本的にコードチャンク内で描かれたグラフはそのまま掲載されます. よって `ggplot2` のグラフも可能です.

```
01 data(mtcars)
02 ggplot(mtcars, aes(x = mpg, y = disp, color = cyl)) +
03   geom_point() +
04   labs(x = "マイル / ガロン", y = "排気量")
```



ところで, `example1.Rmd` を使っているなら, このグラフは日本語を含んでいるのに文字化けしていないはずです. この秘密は, YAML メタデータの `dev: ragg_png` にあります. これは `ragg::agg_png()` を使ってグラフ画像を貼り付けるオプションです. より正確に言うと, この YAML での指定は, 全てのチャンクオプションの `dev=` のデフォルトを `ragg::agg_png` にするということを意味します.

レポートや論文では図に「図 1: ナニナニのグラフ」と図のキャプションを書き、本文中で「図 1 を見よ」のように書くことが多いと思います。このような**相互参照**も可能です。図のキャプションは、チャンクオプションの `fig.cap = ""` に指定できます。

```
01 data(mtcars)
02 ggplot(mtcars, aes(x = mpg, y = disp, color = cyl)) +
03   geom_point() +
04   labs(x = "マイル / ガロン", y = "排気量")
```

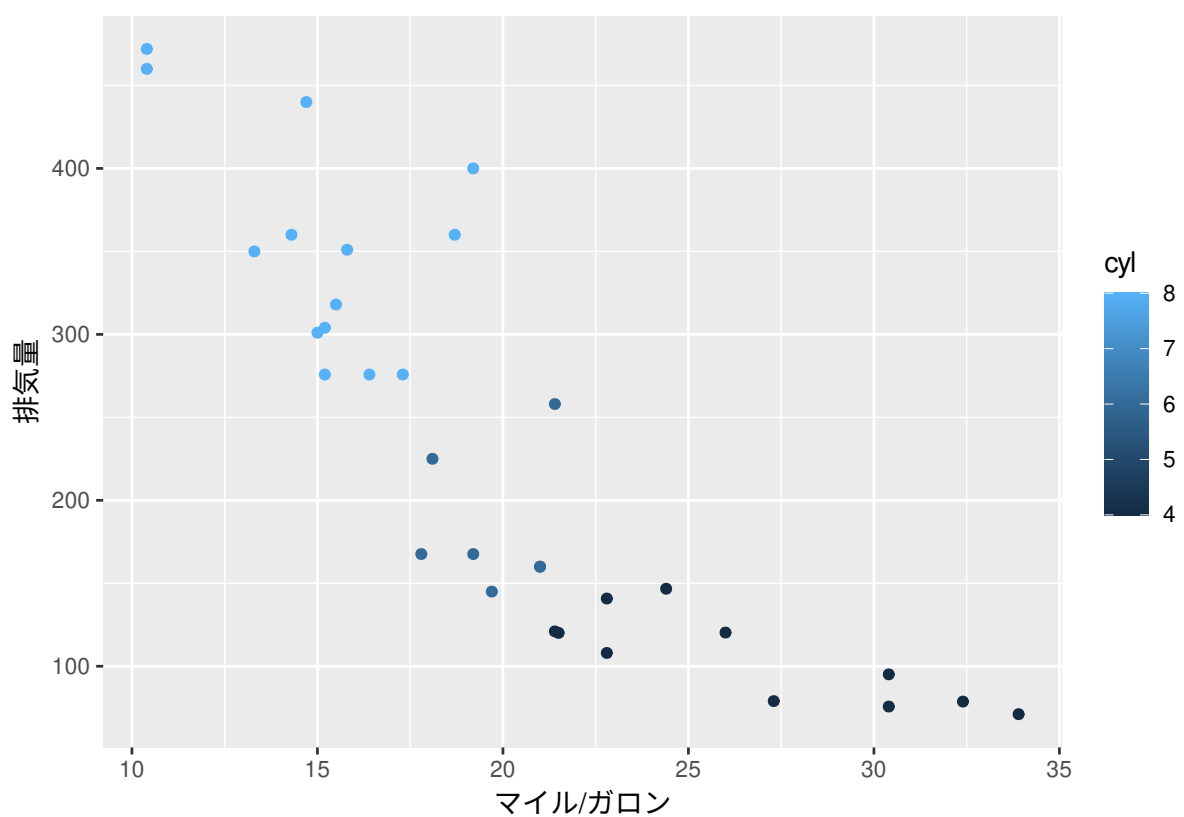


図 2 ggplot2 によるグラフ

Markdown 構文の画像貼り付けでは、キャプション表示や相互参照が使いません。コードチャンク内で `knitr::include_graphics()` を呼び出せば画像を貼り付ければ、これらが実現できます。

```
01 knitr::include_graphics("img/logo.png")
```



図3 Tokyo.R のロゴ

5.4 表の掲載

ここでは `mtcars` を使っていますが、表は基本的にデータフレーム形式にできるものならばなんでも表示できます。

```
01 data(mtcars)
02 mtcars <- head(mtcars[, 1:4])
03 mtcars
```

上記の結果は `officedown::rdocx_document` 試用時を除き、多くの場合で R コンソールと同様にテキストで表示されます。しかし HTML/PDF はいずれも罫線を引いて表を作ることができます。そのようにしたい場合は以下のように `kableExtra::kbl()` を使います (`knitr::kable()` と実質的にほぼ同じですが、少しだけ便利になっています.)。

```
01 data(mtcars)
02 mtcars <- head(mtcars[, 1:4])
03 kableExtra::kbl(mtcars, format = if (knitr::is_latex_output()) "latex" else "pipe",
  ↪ booktabs = T)
```


| | mpg | cyl | disp | hp |
|-------------------|------|-----|------|-----|
| Mazda RX4 | 21.0 | 6 | 160 | 110 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 |
| Datsun 710 | 22.8 | 4 | 108 | 93 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 |
| Valiant | 18.1 | 6 | 225 | 105 |

`format = if(knitr...` の長い引数は、デフォルトでは表の罫線が全く表示されないためこのように書いています。HTML なら CSS でも指定できますが、こちらのほうが簡単です。

表のみが必要ならば、ここでもチャンクオプション `echo = F` を使えます。

なお、PDF の場合は表がはみだすかもしれません。その場合は `kableExtra::kable_styling()` で自動縮小する機能を使うとよいでしょう (officedown 使用の場合はレイアウトが崩れることがあります)。以下がその例です*5

```
01 data(mtcars)
02 mtcars <- head(mtcars)
03 tab <- kableExtra::kbl(mtcars, format = if (knitr::is_latex_output()) "latex" else
    ↪ "pipe", booktabs = T)
04 kableExtra::kable_styling(tab, latex_options = c("scale_down", "HOLD_position"))
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|-------------------|------|-----|------|-----|------|-------|-------|----|----|------|------|
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| Valiant | 18.1 | 6 | 225 | 105 | 2.76 | 3.460 | 20.22 | 1 | 0 | 3 | 1 |

表のデザインを細かく指定したい場合は **kableExtra** が他にもいろいろな関数を提供しています。出力結果を表にするパッケージとして、**kableExtra** の他にも **broom**, **huxtable**, **flextable**, **stargazer**, **summarytable** パッケージなどが便利です。

*5 LaTeX では自動で位置調整がなされるため、さらに `Hold_position` というオプションで表の掲載位置を固定しています。



Word では、この方法は HTML や PDF と比べレイアウト調整がうまくいかないことが多いかもしれません。Word での表の出力にこだわりたい場合 Word での表出力に特化した **flextable** パッケージがあります。



R Markdown でよく使うと思われる操作は以上です。残りは補足やより発展的な内容です。

6 スライドの作成

6.1 概論

文書だけでなくスライドも作ることができます。有力なものとして、以下があります。

- HTML
 - `rmarkdown::slidy_presentation`
 - `xaringan::moon_reader`
- PDF
 - `rmdja::beamer_presentation_ja`
- PPTX
 - `officedown::rpptx_document`

スライドの場合、基本的に、セクションを表す `#` または `----` がスライドの区切りとみなされます。

しかし、残念ながら構文があまり統一されていないため、直感的ではありません。たとえば基本的なフォーマット `rmarkdown::slidy_presentation`

`beamer` は、`#` がセクションの区切りで、見出しのみの専用スタイルのスライドになり、`##` がスライドの区切りとみなされます。ほかはすべて `#` がスライドの区切りです。

つまり、`Beamer` は

```
# セクション区切り
```

```
## スライド 1
```

```
* ???
```

```
* !!!
```

```
## スライド 2
```

```
* ?!
```

で、それ以外のフォーマットはデフォルトで

```
# スライド 1
```

```
## スライド 1 の見出し
```

```
* ???
```

```
* !!!
```

```
## スライド 1 の見出しその 2
```

```
* ?!
```

となります。 `slide_level: 2` にすることで `beamer` と同じ感覚で書くことができますが、`xaringan` や `revealjs` はこれできません。

6.2 xaringan (写輪眼) でできること

- 本編では紹介できませんでしたが、HTML 形式のドキュメントでは `xaringan` パッケージも便利です。
- 個人的には `reveals` のスライドはブラウザバックが必要になるので好きではありません。

`xaringanthemer` パッケージは `xaringan` スタイルを拡張します。スタイルに合った `ggplot2` 用テーマも用意されていますが、英文を想定しているので使いづらいかもしれません (というかタセが強すぎる)。

ただし、Visual Editor モードでは編集できません (`xaringan` に限らず、`rmarkdown` 以外が提供しているパッケージは対応していないことがよくあります)。

7 環境設定についての注釈

7.1 Linux (RStudio Cloud)

Ubuntu 20.04 を想定しています。RStudio Cloud も Ubuntu OS のため、ほぼ同じだと思います。

- R (≥ 3.6)
- RStudio ($\geq 1.4.1103$)
- Noto フォント
 - Ubuntu 18.04 以降を日本語版インストーラから導入したならデフォルトで入っていますが、RStudio Cloud 等には入ってません。以下でインストールできます。

```
sudo apt install fonts-noto-cjk fonts-noto-cjk-extra
```

- 各種パッケージ
- TeX
 - TeX がなにかわからない人はとりあえず **tinytex** パッケージでインストールしてください
 - TeX Live をインストールしたい場合は **apt** でインストールしているとバージョンが数年遅れの場合があります. **tlmgr** からインストールを推奨します

7.2 Mac

申しわけないですが最新版 (Big Sur) は実機を持っていないので動作確認できません. この要件は Catalina で確認したものです.

- R (≥ 3.6)
- RStudio ($\geq 1.4.1103$)
- Homebrew
 - 以下の XQuartz 等いろいろな外部プログラムのインストールに必要
- XQuartz
 - グラフィック表示のためにあったほうが便利です
- 各種パッケージ
- TeX
 - TeX がなにかわからない人はとりあえず **tinytex** パッケージでインストールしてください
 - TeX Live (MacTeX) をインストール済みなら不要です

7.3 Windows 10

- R (≥ 3.6)
- RStudio ($\geq 1.4.1533$)
 - Python を使う予定がないならば, 安定版 1.4.1106 でも問題なし.
- TeX
 - TeX がなにかわからない人はとりあえず **tinytex** パッケージでインストールしてください
 - TeX Live をインストール済みなら不要です

7.4 グラフィック関係

R Markdown に必須ではありませんが, 良質な文書作成のためにグラフ関連のパッケージをインストールしておくことをお勧めします. もちろん R Markdown を使わなくとも, R のグラフ作成全般で有効に使うことができます.

以下は, 特に重要です.

- **ggplot2** (tidyverse パッケージ内に含まれているので個別インストールしなくてもかまいません)
- **ragg** v1.1.0 以降
- **fontregisterer**
- **svglite**

ragg をインストールする際 (正確には **ragg** が依存している **textshaping** パッケージ), Mac や Linux では以下のようなログが出るかもしれません. これは外部ライブラリ **harfbuzz**, **freetype2**, **fribidi** が不足しているということなので, 指示の通り, **apt**, **brew**, **rpm** コマンドなどでインストールしてください.

```
----- [ANTICONF] -----
Configuration failed to find the harfbuzz freetype2 fribidi library. Try installing:
* deb: libharfbuzz-dev libfribidi-dev (Debian, Ubuntu, etc)
* rpm: harfbuzz-devel fribidi-devel (Fedora, EPEL)
* csw: libharfbuzz_dev libfribidi_dev (Solaris)
* brew: harfbuzz fribidi (OSX)

If harfbuzz freetype2 fribidi is already installed, check that 'pkg-config' is in your
PATH and PKG_CONFIG_PATH contains a harfbuzz freetype2 fribidi.pc file. If pkg-config
is unavailable you can set INCLUDE_DIR and LIB_DIR manually via:
R CMD INSTALL --configure-vars='INCLUDE_DIR=... LIB_DIR=...'
```

ragg になぜそこまでこだわるかというと, (1) グラフに日本語を含めたときの文字化けの可能性を大幅に減らせることと, (2) Windows ではデフォルトのグラフィックデバイスよりも描画速度や品質が向上しているためです. これは R Markdown に限った話ではありませんが, R のグラフに日本語を含めようとするとよく文字化けします. **ragg** v1.1.0 以降では, フォントの指定がなくても, 自動で OS の標準フォントにフォールバックしてくれる機能が提供されています.*⁶ RStudio 1.4 からはプロットビューアでも使えるようになっていますので, R Markdown を使わなくともこちらは有効にしておくことをお勧めします.*⁷

ただし, SVG や PDF などのベクタ画像には対応していないため, その場合は **fontregisterer** や **svglite** が必要になるかもしれません.

7.5 グラフィックのフォント

OS ごとにデフォルトで入っているフォントが異なるため, 以下を想定しています.

- Linux: Noto
- Mac: ヒラギノ
- Windows: 游書体

ragg は最近の更新でフォールバックフォントが有効になりましたが, 以下の点に注意してください.

1. PDF でフォントを埋め込みたい場合は従来どおりフォントファミリ名の指定が必要

*⁶ 公式リリースノート: <https://www.tidyverse.org/blog/2021/02/modern-text-features/>.

*⁷ 日本語での解説: <https://uriibo.hatenablog.com/entry/2021/03/29/202756>.

2. Windows のフォールバックフォントは游書体ではなく MS フォントになっている

7.6 Python の対応状況

reticulate パッケージが必要です。Python 環境, OS と RStudio のバージョンの組み合わせによっては不具合が発生することがあります。

例えば以下を参考にしてください。

<https://ill-identified.hatenablog.com/entry/2021/02/22/233326>



reticulate パッケージは Python をインストールする関数も提供しているため, Python の事前インストールは必須ではありません。

Python の matplotlib でグラフを描画する場合, やはりフォントの設定がネックになります。特に PDF で埋め込みたい場合は, デバイスを明示的に変更し, かつフォントファミリの指定が必要になります。なおこの設定は R Markdown に限らず Python 全般で有効です。

基本的には `.matplotlibrc` の設定だけで完結しますが, 設定の変更が嫌ならば Python セッション中に設定を書き換えることもできます。

7.7 Julia の対応状況

JuliaCall パッケージによって Julia セッションを呼び出すことができます。ただし, まだ不安定なことが多いようです。

7.8 それ以外の言語

それ以外の言語は基本的にシステムコールとして呼び出すことになるため, チャンクをまたいで結果を保持することはできません。

それとは別にシンタックスハイライトができるので, コードの掲載には便利なはずです。

より詳しい話は『R Markdown クックブック』の 15 章を読んでください。

8 本文の書き方に関する Tips

ここでは, 「Markdown 構文そこそこ使うが, 引っかけやすいケース」の Tips を書いておきます。

8.1 Markdown でハマりやすい場面

8.1.1 箇条書きをエスケープ

箇条書きをエスケープしたければ, 以下のように行頭にバックスラッシュ + スペース.

\ (1) 箇条書き無効化

(1) 箇条書き無効化

または, 箇条書きの記号の直後にバックスラッシュ + スペースを挿入します

*\ 箇条書き無効化

* 箇条書き無効化

8.1.2 箇条書きのカウンタがうまくいかないとき

ネストした際に, 数字や abc でカウントする箇条書きのカウントがリセットされることがあります. これは Markdown の仕様で, スペース 4 つ以上のインデントでないとネストと認識されないためです. RStudio のインデントはスペース 2 つがデフォルトのため, R Markdown ではハマりやすいです.

失敗例

```
1. 1
1. 2
  * 2-1
  * 2-2
1. 3
1. 4
```

```
1. 1
2. 2
  • 2-1
  • 2-2
1. 3
2. 4
```

成功例

```

1. 1
1. 2
    * 2-1
    * 2-2
1. 3
1. 4

```

```

1. 1
2. 2
    • 2-1
    • 2-2
3. 3
4. 4

```

箇条書きにコードブロックなどのブロック要素を入れる場合も同様に 4 スペース=1 インデントで書く必要があります。

8.2 数式

LaTeX の数式を書くことができます。出力が HTML か PDF かで対応している記法が一部異なるので、ここではどちらでも使える方法を書きます。

まず、行内数式は $\$$ 記号で囲みます。 π は `\pi` となります。

独立行数式は $\$$ で囲みます。式 (8.2) を見てください (このように数式への参照もできます)。

```


$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \quad (\text{\#eq:norm})$$


```

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

`align` 環境を使いたい場合は、 $\$$ を書かずに直接 LaTeX コマンドを書いてください。

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{1}$$

8.2.1 脚注の連続

脚注^{~[ここに脚注]}の形式での脚注は連続して書けません。脚注を付けたい場所に^[^脚注 ID]と書いた後、適当な文の区切りのある場所で

^[^脚注 ID]: ここに脚注

というふうに本文を書きます^{*8*9}。

8.2.2 YouTube 動画の埋め込み

“R Markdown: The Definitive Guide” Chp.14.5 に書かれている以下の方法

![] (YouTube の URL)

は現在では**できません**。公式の「共有」->「埋め込み」からの `iframe` タグのコピーが必要です。ただし、そのままだとソースが埋め込まれてしまい正常に動作しません。コピーしたタグに、`data-external="1"` を追加するか、出力フォーマットに `self_contained: false` を設定します。後者は広範囲に影響があるため、意味を理解していないなら前者をおすすめします。

```
<iframe width="640" height="480" src="https://www.youtube.com/embed/lJIrF4YjHfQ" frameborder="0" allowfullscreen data-external="1"></iframe>
```

なお、PDF では動画を埋め込むことはできません。

8.3 相互参照

文献引用の相互参照は説明することが多いため、後の 12 節でまとめます。ここではそれ以外の、図表、数式セクションへの参照 (いまさっき「12 節」と表示してみせたもの) について言及します。

まず、相互参照は主に **bookdown** パッケージでサポートされており、**rmarkdown** のデフォルトのフォーマットでは利用できません。これは冒頭で **bookdown** に使用をすすめた理由の 1 つです。

例えばグラフを描きます。

```
01 ggplot(data.frame(x = c(-4, 4)), aes(x = x)) +  
02   geom_function(fun = dnorm, args = list(log = T)) +  
03   labs(y = expression("対数確率", log(p)))
```

先ほどのグラフは、R コードチャンクで描いています。そしてチャンクラベルは `plot-example` です。このとき、ラベルの頭に `fig:` を付けたものが図の参照 ID になります。よって、`\@ref(fig:plot-example)` で図

*8 ここに脚注その 1

*9 ここに脚注その 2

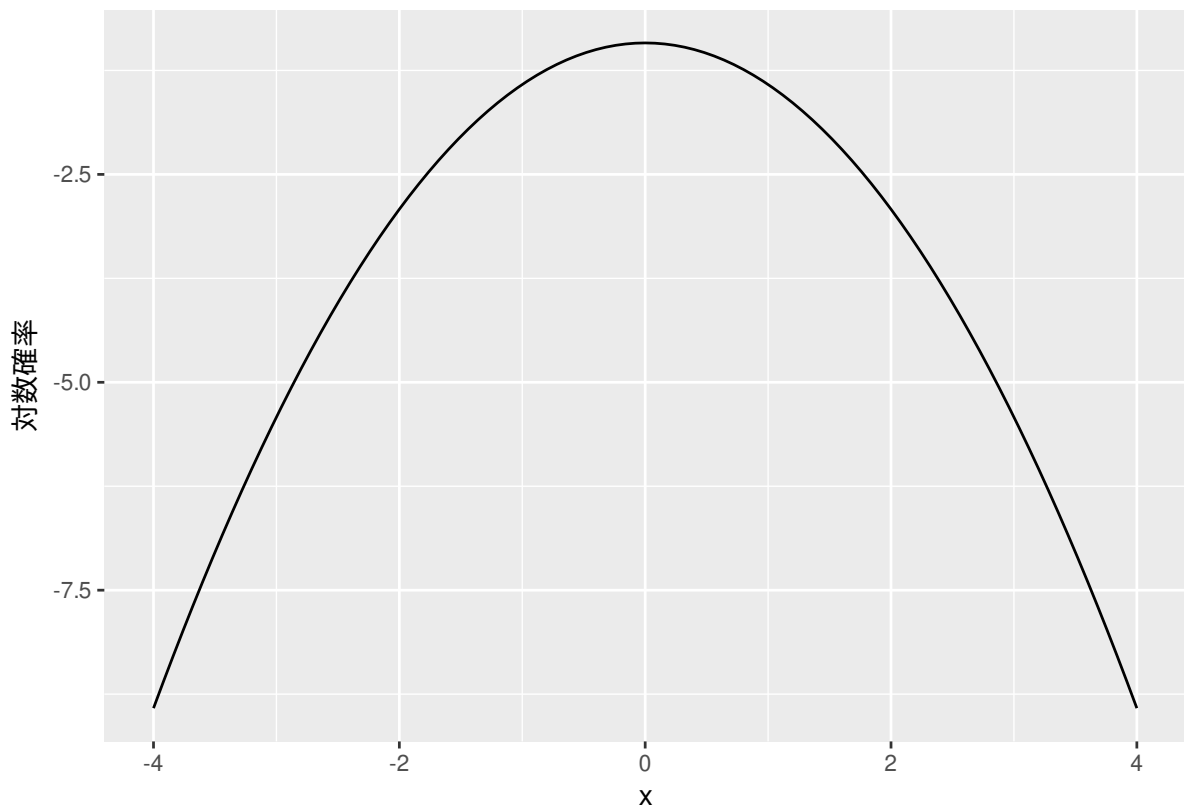


図4 標準正規分布

番号を自動で挿入できます (図 4). 「図: XXXX」のキャプション部分は, チャンクオプション `fig.cap=` で設定できます.

表も同様ですが, 表のタイトルを付ける場合は `knitr::kable()` や `kableExtra::kbl()` が必要になります (表 2). 表を参照する場合, `\@ref(tab:table-example)` のように, チャンクラベルに `tab:` を付ける必要があります.

```
01 data(mtcars)
02 knitr::kable(head(mtcars), caption = "表の例", booktabs = T)
```

kableExtra パッケージがあると, 表を装飾できます (表 3).

```
01 require(kableExtra)
02 mtcars[1:8, 1:8] %>%
03   kbl(caption = "kableExtra でスタイルを設定した表", booktabs = T) %>%
04   kable_paper(full_width = F) %>%
05   column_spec(2,
```

表 2 表の例

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

表 3 kableExtra でスタイルを設定した表

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1

```

06   color = spec_color(mtcars$mpg[1:8]),
07   link = "https://haozhu233.github.io/kableExtra/"
08 ) %>%
09 column_spec(6,
10   color = "white",
11   background = spec_color(mtcars$drat[1:8], end = 0.7),
12   popover = paste("am:", mtcars$am[1:8])
13 )

```

ただし、上記は Word の場合はうまく行かないかもしれません。Word に限って言えば **flextable** パッケージを使ったほうが良い出力を得やすいようです。

詳細は公式のドキュメントを参照してください。

https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html

8.3.1 より詳細な参考資料: Markdown 構文

Markdown 構文全般の説明は, Definitive Guide の 2.5 節などが網羅的です. Markdown にはいろいろな方言があるため, R Markdown 以外の Markdown 構文の解説は R Markdown では使えないことがあることに注意してください. 基本的に, R Markdown または Pandoc を想定した Markdown の解説を参照するとよいでしょう.

9 YAML メタデータについてより詳しく

9.1 YAML メタデータの基本構文

ここまで, タイトルや output しか設定していなかった YAML メタデータについて詳しく説明します.

Rmd ファイルの冒頭の ---- で囲まれた範囲は YAML メタデータといいます. この部分の構文は YAML という構文で書かれています. YAML はデータを記述するためのものですが, 例えば JSON と比べると Markdown に似ており視覚的に見やすいという利点があります. 典型的なものは

```
----
title: "タイトル"
output: html_document
----
```

のように, 常に **フィールド名: 値** という形で書かれます. さらに, インデントをとることでネストさせることもできます.

```
output:
  html_document: default
  pdf_document: default
```

これは出力フォーマットに html_document と pdf_document の 2 つを指定していることを意味します.

リスト (R のリストではなく, 複数の値を並べたもの, という意味) は 2 通りの書き方があります. 任意の数の値を与えられるフィールドに対して使います. 例えば著者が複数いる場合, 以下 2 つは同じです.

```
author: ["R 太郎", "R 花子"]
output: html_document
```

```
author:
- "R 太郎"
- "R 花子"
output: html_document
```

文書の概要を記入する `abstract` フィールドなどは、長いため 1 行に収めるのが難しいかもしれません。以下のように `|` または `>` を使えば複数行にわたるテキストが 1 つの値とみなされます。

```
abstract: |
  複数行に分けて書きたい場合は
  このようにバー（`|`）を使います
```

```
abstract: >
  複数行に分けて書きたい場合は
  このように不等号（`>`）も使えます
```

コロン (:) など構文で使われる特殊な記号を値に含めたい場合は、値全体を引用符で囲ってください。

ここまでは Rmd に限らない YAML メタデータの一般的なルールです。

技術的な補足: R Markdown の YAML メタデータは Pandoc の仕様を利用しています。

9.2 YAML メタデータでの設定

R Markdown では YAML メタデータのほとんどにデフォルト値が設定されているため、一切書かなくても出力できる事が多いです。デフォルトでは `output: html_document` なので、HTML 形式のレポートが出力されます。

しかしおそらく最低限の設定は必要でしょう。例えばセッションで使用したように、タイトル・著者・出力フォーマットの設定は最低限必要でしょう。

```
-----
title: "タイトル"
author: "氏名"
output: bookdown::html_document2
-----
```

R Markdown の YAML メタデータの設定で一番重要かつ、初心者にはわかりにくいのは `output` フィールドだ

と思われます。ここには R Markdown 出力フォーマットを指定します。出力フォーマットとは、R の関数です。デフォルトでは `rmarkdown::html_document` が使用されます。`html_document` のように **rmarkdown** パッケージに含まれている関数は、パッケージ名を書かなくてもよいですが、それ以外は上記のように **パッケージ名::** が必要になります。

スタイルを調整したい場合の多くは、`output:` 以下にさらに設定をネストして書くことになります。この項目は膨大で、かつフォーマットによって異なります。しかし、出力フォーマットは R の関数であり、出力フォーマットにネストできる項目が引数と同じであると分かれば簡単に調べることができます。つまり、(普通は) ヘルプページが用意されています。

```
-----
output:
  html_document:
    code_folding: hide # コードを折りたたんで表示する
    number_sections: yes # セクション番号を表示する
-----
```

なお、R の TRUE/FALSE に対応する値は、`true/false`, `yes/no` で置き換えることができます。

YAML の記入ミスがあった場合のエラーメッセージは少々わかりにくいです。

手動の記入に自信がないなら、**yamlthis** パッケージをインストールすると、RStudio の アドインから “Write New R Markdown or YAML File” を選んで設定用のダイアログボックスを開けます (図 5)。ただし、出力フォーマットの項目は **rmarkdown** で提供されているもののみ対応しています。

```
01 knitr::include_graphics("img/yamlthis.png")
```

現状、出力フォーマットで設定できる項目 (`output:` にネストされている項目) と、YAML メタデータのトップレベルに使用できる項目にはいくつか重複しているものがあり、注意が必要です。

10 画像の出力

10.1 グラフィックデバイス

すでに R に慣れている人はご存知と思いますが、R のグラフィックに日本語のタイトルやラベルを含む場合、環境によっては文字化けします。そしてその解決法が OS によって異なるため、情報の錯綜と混乱を生んできました^{*10}。

しかし、最近登場した **ragg** パッケージ v1.1.0 により、この問題は劇的に改善されました。

^{*10} この問題は <https://ill-identified.hatenablog.com/entry/2020/10/03/200618> に詳しく書きましたが、今は読まなくても大丈夫です。

New YAML

Cancel

Set up YAML

Author

Katagiri, Satoshi

Date

☒ Use system date

Title

無題

Subtitle

Output

html

Set html options

Export to

R Markdown ▼


Path

Untitled.Rmd

R Markdown Template

Browse...

Template (optional)

YAML

R Markdown Options

LaTeX Options

Citations

Parameterized Reports

Done

図 5 ymlthis の画面

1. Windows ではデフォルトの `png` デバイスより描画が美しい
2. 主要な OS で、特に指定がなくとも日本語フォントにフォールバックされる

という利点があります。よって文字化け問題に対して無用の労力をかける必要がなくなりました。今回の資料でも最初にインストールすることをお薦めしています。

R Markdown では多くの場合グラフィックデバイスのデフォルトが `png` なので、今回は明示的に `ragg_png` に指定しています。一方で、ベクタ画像が良いという場合は `svglite` パッケージによる `svglite` を指定することができます。

ただしこれは HTML/Word 出力の場合です。PDF の場合、画像も PDF 形式にできます。PDF ではフォントの変更はよりシビアなため、pdf ではなくより使いやすい `cairo_pdf` の指定をおすすめします。X11 や `cairo` を確認させたのはこれが理由です。

10.2 サイズの調整

R Markdown では画像サイズを決めるのに大きく分けて 2 種類のパラメータがあります。Markdown 記法は既存の画像ファイルを貼り付けるものでしたが、コードチャンクでは、内部でグラフを生成し、画像として保存、そして文書に貼り付ける、という動作をしています。よって、保存時のサイズと文書掲載時のサイズをそれぞれ決めるパラメータが存在します。

- `fig.width`, `fig.height` などは保存時のサイズで、単位はインチです。
- `out.height`, `out.width` などは掲載時のサイズで、単位は出力フォーマットに依存します。

しかし基本的には余白に対するパーセント表記で決めると簡単でしょう。

この使いにくいところは、例えばグラフ内の文字が出力時にどれくらいの大きさになるかがわかりにくい点です。自分でなにかルールを決めたり、オプションフックで自動調整する関数を書くという手があります^{*11}。

11 PDF 出力の内幕

R Markdown で PDF を出力する場合、`.md` ファイルを `pandoc` で `.tex` 形式に変換してから LaTeX で PDF を生成します (この作業をコンパイルといいます)。このとき、LaTeX プログラムの呼び出しを `tinytex` パッケージで制御しています。

まだ LaTeX をインストールして使用したことがない場合は、`tinytex` で簡単にインストールすることもできます。これはチュートリアル¹の環境設定にも書かれている、以下のコマンドで実行できます。2 行目は必ずしも必要ではありませんが、日本語ロケールだと毎回警告が出て邪魔なのでインストールしましょう。

^{*11} オプションフックは Yihui 氏の解説が参考になります。翻訳したものをご覧ください <https://gedevan-aleksizde.github.io/knitr-doc-ja/hooks.html#%E3%82%AA%E3%83%97%E3%82%B7%E3%83%A7%E3%83%B3%E3%83%95%E3%83%83%E3%82%AF>


```
01 tinytex::install_tinytex()
02 tinytex::tlmgr_install("texlive-msg-translations")
```

ただしこれは**必須ではありません**。既に LaTeX をインストールしていると**競合します**。注意してください。

LaTeX で PDF をコンパイルする場合、様々なプログラムを複数回呼び出す必要があります。それを制御するのが **tinytex** の役目の 1 つです。この役目は従来 latexmk というプログラムが担っていましたが、**tinytex** は latexmk の処理をエミュレートすることができます。その結果、エラー発生箇所の追跡が多少簡単になりました。

詳細は R Markdown クックブックの 1.2, 1.3 節を見てください

latexmk エミュレーションを無効にする必要のある場面として、後述する (u)pBibTeX を使用するというものがあります。

latexmk の使用法は以下が参考になります (必須ではありません)。

- <https://texwiki.texjp.org/?Latexmk>
- <http://www2.yukawa.kyoto-u.ac.jp/~koudai.sugimoto/dokuwiki/doku.php?id=latex:latexmk%E3%81%AE%E8%A8%AD%E5%AE%9A>

11.1 LaTeX の出力エンジン

R Markdown が使用できる LaTeX エンジンは基本的に以下の 3 つです。

1. **pdflatex** (pdfLaTeX)
2. **xelatex** (XeLaTeX)
3. **lualatex** (LuaLaTeX^{*12})
4. **tectonic**^{*13}

つまり、基本エンジンである LaTeX2e ひいては pLaTeX (pLaTeX2e) での使用は実は想定されていません^{*14}。(1) は欧文での処理のスタンダードになっています。これで日本語文書をコンパイルする場合、不可能ではありませんが難しい and 使用例が少ないため候補から外れます。最近は日本語文書を (2, 3) でコンパイルする例が増えています。(4) は最近登場し、Pandoc もサポートするようになったため、次にリリースされるバージョンからサポートされます。ただし、XeTeX ベースなので出力は基本的に **xelatex** と変わらないはずです。また、BibTeX/BibLaTeX, MakeIndex 等への対応も不十分なようです。よって通常は、(2, 3) のどちらかを使うことをおすすめします。

^{*12} 実際には LuaHBTeX

^{*13} **rmarkdown** >= 2.8 でサポート予定

^{*14} 英語圏の開発者は必ずしも日本語 LaTeX 事情を把握していないためです: <https://acetaminophen.hatenablog.com/entry/texadvent2020>

(2, 3) についてはそれぞれ特性が異なります。x_{el}at_{ex} はよりコンパイルが速いですが、和文の禁則処理の一部が甘いです。逆に l_{ua}lat_{ex} はより利用が活発なためか禁則処理がより厳格ですが、一方でより処理速度が遅くなっています。それ以外にも対応している文書クラスの設定が微妙に異なるなどの違いがあります。個人的な体感として、短い文書であれば速度はあまり気になりません。また、スライドであれば x_{el}at_{ex} の禁則処理の甘さも気になる場面が減ると思います。これも個人的な考えですが、普段は x_{el}at_{ex} を使い、完成版は時間のかかる l_{ua}lat_{ex} に切り替える、ということをしています*¹⁵。ただし、両者の互換性をよく理解していないとこのやり方はうまくいかないこともあります。

11.2 R Markdown と LaTeX テンプレートの融合

あなたが独自にカスタマイズした LaTeX のテンプレートを組み込んで、R Markdown で使用できれば便利でしょう。

LaTeX のテンプレートを R Markdown から操作する 3 つの主な方法があります。

1. YAML メタデータの設定項目 (Pandoc の機能)
2. Pandoc テンプレートの改造
3. プリアンブルに部分的に挿入する

まず、(1) は documentclass および classoption です。それぞれ、.tex ファイル冒頭の \documentclass[]{} の文書クラスとオプションに代入されます。article, report, book といったカスタマイズは、このオプションだけで十分でしょう。

R Markdown は、knit で処理した本文と YAML メタデータを Pandoc のテンプレートに代入します。ターミナルで `bash pandoc -D latex > latex.template.tex` を実行してみてください。Pandoc の LaTeX 用テンプレートファイルが出力されます。これはほぼ .tex ファイルですが、至るところに `$documentclass$` のような Pandoc マクロが含まれています。これは主に (1) の YAML メタデータで入力した項目が代入されます。このマクロに注意しつつあなたのテンプレートを移植することができます。デフォルトのテンプレートは日本語出力が十分に考慮されていないことの改善や、より視覚的に優れたデザインにするため、rmdja はデフォルトのものを改造した独自の LaTeX テンプレートを使用しています。詳細は『R Markdown Cookbook』6.2, 6.10 節や Pandoc の公式ドキュメント*¹⁶ を参考にしてください。

マクロも自分で新規に定義できます。rmdja ではフォントの選択に自由と利便性をもたせるため、jamainfont, jasansfont, jamonofont, jafontpriset という YAML メタデータのフィールドを新たに定義しています。

特定のパッケージを追加で読み込むとか、修正箇所がごくわずかなら、(3) の方法が簡単でしょう。例えば、kableExtra パッケージの機能を beamer で完全に使うには、追加のパッケージが必要と公式ドキュメント*¹⁷に書かれています*¹⁸。それは YAML メタデータに以下のように追記して実現できます。

*¹⁵ TeX Live 開発に関わっている日本人はなるべく LuaLaTeX を使うように主張しています。

*¹⁶ <https://pandoc.org/MANUAL.html#variables-for-latex>

*¹⁷ <https://haozhu233.github.io/kableExtra/>

*¹⁸ rmdja の beamer フォーマットでは、これらのパッケージの読み込みコマンドを内蔵済みです。

```
header-includes:
```

```
- \usepackage{booktabs}
- \usepackage{longtable}
- \usepackage{array}
- \usepackage{multirow}
- \usepackage{wrapfig}
- \usepackage{float}
- \usepackage{colortbl}
- \usepackage{pdflscape}
- \usepackage{tabu}
- \usepackage{threeparttable}
- \usepackage{threeparttablex}
- \usepackage[normalem]{ulem}
- \usepackage{makecell}
- \usepackage{xcolor}
```

ただし、パッケージのロードの順番によってエラーが起きることもあるので注意してください。例えば色を変えたりグラフィックをいじったりするタイプのパッケージは追加を要求される可能性が高い上に競合しやすいです。

なお、LaTeX パッケージを読み込むだけならば、出力フォーマットの引数 `extra_dependencies` が使えることもあります (『R Markdown クックブック』6.4 節)。

```
output:
```

```
  pdf_document:
```

```
    extra_dependencies:
```

```
      - booktabs
      - longtable
      ...
      - xcolor
```

別の方法として、PDF 系の出力フォーマットの多くには `includes` 引数があります。

```
output:
```

```
  pdf_document:
```

```
    includes:
```

```
      in_header: preamble.tex
      before_body: before-body.tex
```

```
after_body: after-body.tex
```

カスタマイズの多くはプリアンブルでなされるので、設定を記入したファイルを別に作り、`in_header: preamble.tex` のように指定します。 `before` は `document` 環境、つまり本文開始直後に挿入されます。よって洋書によくある冒頭のエピグラフや「誰々に捧げる」的な文章や、あるいはロゴマークを挿入するのに使えます。 `after_body` も本文の最後に挿入されます。PDF 版にのみ巻末索引や付録を付ける場合に使用しますが、これらは他の方法でも代替可能 (cf. `knitr::is_latex_output()` で条件出力させる) であることが多いのであまり使わないかもしれません。

`include-headers` と `includes` の違いは、前者はトップレベルの YAML メタデータなので PDF 以外のフォーマットにも入力されるということです。ほとんどの場合は無害ですが、もし HTML と PDF で異なる入力が必要な場合、`includes` を使うことになるでしょう。また、LaTeX とそれ以外のソースを別ファイルに分けることは保守性の改善につながるかもしれません。



フォーマットによっては、`in_header` が `header-includes` を上書きしてしまうことがあるので注意してください。

ただし、(u)pLaTeX など R Markdown が対応していない LaTeX エンジンに依存したテンプレートの場合、これだけでは「knit ボタンで一発コンパイル」ということはできません、残念ながら。それは次のセクションを参考にしてください。

11.3 サポート外の LaTeX エンジンを使用する

昔からある (u)pLaTeX などでのコンパイルを前提としたテンプレートを使う必要のある場面も多いと思います。これらを使うもっとも確実な方法は、`.tex` ファイルを出力して、手動でコンパイルすることです。現時点では、(u)pLaTeX で使えない設定が書かれていることが多いので、手動で書き換えることも必要になります。

現時点では、`rmja::beamer_presentation_jp` は XeLaTeX と LuaLaTeX のみの使用を前提としており、それ以外のサポートの予定はありません。プレゼン資料については体裁の厳格な制約がないものと考えているからです。

12 参考文献スタイルのカスタマイズ

12.1 参考文献設定の基本事項

参考文献リストの出力方法は以下の 4 通りの方法があります。

1. `default` (CSL)
2. `natbib` (BibTeX)
3. `biblatex` (BibLaTeX)

4. 手動で書き込む

(4) は文字通りなので省略します. (1 - 3) は文献データベースに登録したものを, 本文の引用されたものだけを自動的に取り出して一覧を出力します.

(1) は HTML, PDF, Word いずれにも使えます. (2, 3) は PDF のみです.

共通箇所は YAML で, 出力スタイルの設定と, `bibliography` フィールドで文献データベースファイルを指定することです.

```
output:
  XXX_output:
    citation_package: ENGINE
bibliography: REFS.bib
```

ENGINE には上記の 3 つのどれかを指定します. REFS.bib は文献データベースファイルです. 対応可能なファイル形式は RStudio の公式ページの一覧を確認してください.

https://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html

また, `default` の場合に限り, YAML メタデータ内に文献情報を書けるインライン参考文献リスト (`inline references`) の構文が使用できます. 例えばこのように書きます.

```
references:
- type: article-journal
  id: WatsonCrick1953
  author:
  - family: Watson
    given: J. D.
  - family: Crick
    given: F. H. C.
  issued:
    date-parts:
    - - 1953
      - 4
      - 25
  title: 'Molecular structure of nucleic acids: a structure for
    deoxyribose nucleic acid'
  title-short: Molecular structure of nucleic acids
  container-title: Nature
  volume: 171
```

```
issue: 4356
page: 737-738
DOI: 10.1038/171737a0
URL: https://www.nature.com/articles/171737a0
language: en-GB
```

しかし、基本的には .bib 形式で書いたほうが使いまわしやすいため、**.bib 形式のファイルを使うことを推奨します**。よって、以降の解説も .bib ファイルがあるという前提になっています。

特に PDF の出力で、BibTeX/BibLaTeX を使用する場合 .bib 形式の文献データベースファイルしか読み込むことができません。これまで LaTeX を使ったことがない、従って BibTeX も BibLaTeX も使ったことがない、という方は、まずはお持ちの文献リストを .bib ファイルに変換する方法を調べて下さい。

Zotero や Mendeley といった文献管理ソフトの多くには、.bib ファイルをエクスポートする機能があります（前者は better bibtex というプラグインも必要です^{*19}）また、CiNii や Google Scholar、あるいは国際学会・学会誌のレポジトリの多くには論文の書誌情報を .bib 形式でエクスポートする機能があるはずです。

文献引用の練習がしたい場合、`knitr::write_bib(x = c("パッケージ名"))` が便利です。これはパッケージ情報を .bib 形式で出力します。

文献データベースが設定できればあとは文中で引用するだけです。`@文献 ID` (Allaire et al. (2021)), または `[@文献 ID]` ((Xie, 2020)), `[-@文献 ID]` ((2019)) という Markdown 構文が使えます。さらに、セミコロン (;) で区切って複数文献 ID を同時に指定できます。一部の分野では引用が上付き文字^aのようになっているところもあると思います。そのようなケースは CSL 等文献スタイルファイルで制御すると想定されています。PDF 限定でなら、LaTeX コマンドも使えます。

文献 ID をいちいち覚えるのが嫌だという方は、`citr` パッケージによる RStudio アドインが便利です。“Insert Citation” を押してください^{*20}。Visual Markdown Editor でも “@ citation” というボタンを押すとリストが表示されます。こちらは Zotero 等の外部ソフトウェアとの連携や、DOI、PubMed ID などで文献検索もできます (図 6)。

```
01 knitr::include_graphics("img/visual-editor-citation.png")
```

参考文献リストのスタイルに拘らないのなら、設定は以上で十分です。以降の記述は、おそらくは体裁のルールが細かく決まっている、例えば、

- [1] (著者名) “タイトル (ここは__イタリック体で__),” (出版年), (雑誌名), ...
- (著者名) (出版年) 『タイトル (ここは太字で)』, (雑誌名), (巻数), ...

^{*19} Zotero の設定については以下を参考にしてください。 <https://ill-identified.hatenablog.com/entry/2019/03/05/195257>

^{*20} ただし、bib ファイルの更新を手動で行う必要や、bookdown のように文書を分割しているとうまく動作しなかったり、ややクセがあります。

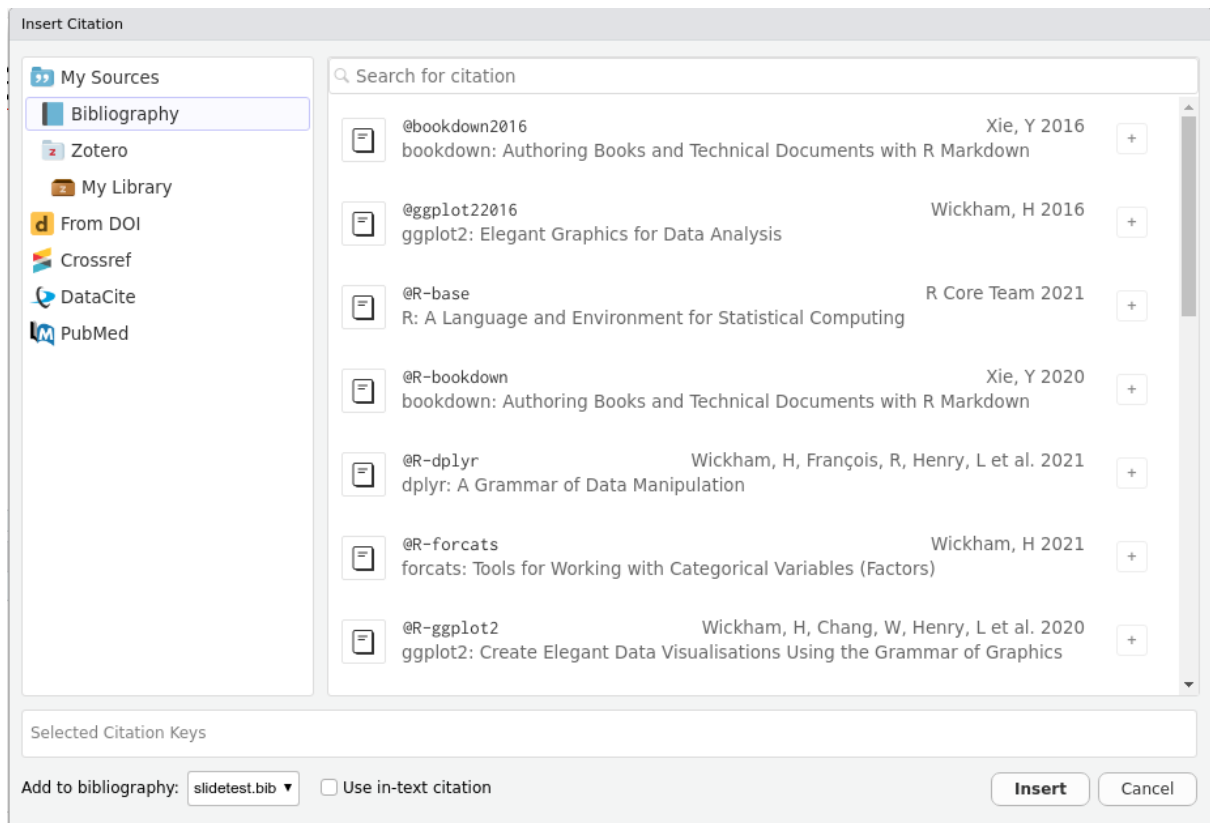


図 6 Visual Markdown Editor による引用リスト

といったふうに細かく決まっている学会・ジャーナルを想定した解説になります。国内の場合、おそらくこれに加えて和・欧文献で引用スタイルが個別に決められていることが多いと思いますので、そのことを念頭に置いた解説をします。

また、欧文であれば、**rticles** パッケージに既にご専門の分野のテンプレートが作られていないかも確認するとよいでしょう。

引用全般について『R Markdown クックブック』の 4.5 節が参考になります。

12.2 default (CSL)

CSL は Microsoft Word でも利用されている文献スタイルフォーマットです。よって、すでに Word をお使いの場合はこのスタイルを使い回すことができます。特にスタイルにこだわりがないのなら、CSL が最も簡単です。スタイルを指定する場合、以下のように CSL 形式で記述されたファイルを指定する必要があります。

```
output:
  XXX_output:
    citation_package: default
```

```
bibliography: REFS.bib
csl: my-style.csl
```

この my-style.csl に対応するファイルは、自分で用意する必要があります。例えば <https://www.zotero.org/styles> に IEEE や APA など欧文向けの様々なファイルが配布されています。

12.2.1 CSL のカスタマイズについて

CSL そのものは XML の構文で書かれおり、CSL Editor のようなエディタで作ることができ、Zotero によって共有することもできます。

CSL はローカライズには対応していますが、それはスタイル全体で同じ言語を使うことが前提です。多言語対応に関する開発者らの公式見解としては「文献単位のローカライズは未対応^{*21}」とのことです。CSL 開発者らは、この問題に対する代替案として Juris-M の使用を提案しています。

(英語ですが、該当パートのみすでに第三者により日本語訳されています)

<https://juris-m.readthedocs.io/en/latest/tutorial-ja.html>

ただし、Zotero の CSL リポジトリで “japanese” と検索すると、いくつかスタイルファイルがヒットします

<https://www.zotero.org/styles?q=japanese>

少し技術的な話: PDF の場合は、pandoc-citeproc という Pandoc フィルタを使って、文献情報を直接書き出します。

12.3 biblatex (BibLaTeX)

BibLaTeX は PDF 出力の場合のみ有効ですが、LaTeX マクロを使用できるため、CSL に比べてより複雑な組版処理ができます。典型的な YAML の記述は以下のようになります。

```
output:
  XXX_output:
    citation_package: biblatex
bibliography: REFS.bib
biblio-style: ieee
biblatexoptions: bibstyle=jauthoryear,citestyle=numeric,natbib=true
```

なお、HTML の場合のために csl フィールドも同時に書くことができます。

^{*21} “CSL doesn’t yet allow for per-item localization,” <https://citationstyles.org/authors/> さらに具体例として日本語ではよくそのようなスタイルを要求されることが名指しで挙げられています。

新たに登場した2つのフィールドはいずれも任意であり、必須ではありません。biblio-style は文献リストの体裁を決めるスタイルファイルを指定するフィールドです。デフォルトでは numeric か authoryear が使われます。biblatexoptions はスタイルに関するオプションです。例えば 上記の例では、文献リストのスタイル (bibstyle) と引用のスタイル (citestyle) をそれぞれ別のものに設定し、かつ natbib.sty の構文を使えるようにしています (natbib.sty については BibTeX の項目参照)。この場合、biblio-style の設定は文献・引用の両方のスタイルを一括して指定していますが、bibstyle, citestyle の設定によってそれぞれ上書きされています。その他にも多くのオプションが存在します。詳細は BibLaTeX のドキュメント等を確認してください。

BibLaTeX は後述の BibTeX の後継プログラムです。体裁の設定が LaTeX マクロで定義されているため、(多少) 設定が簡単になりました。BibLaTeX か BibTeX かでいうと、基本的にはこちらを使ったほうが便利ですが、日本語向けのテンプレートはまだ充実していないというのが現状です。

BiBLaTeX の機能としては、オプション指定のみでいろいろなテンプレートに切り替えられますが、それらは言語別の設定を考慮していません。特に姓名の順序は日本語とそれ以外で逆転して書くことが多いですが、これも想定されていません。そこで、rmdja パッケージではデフォルトで和欧を分けたスタイルテンプレート、jauthoryear.bbx をロードして使用するようにしました。デフォルトでスタイルを指定しない場合、自動でこのスタイルファイルがコピーされ、knit 時に使用されます。このファイルをそのまま使ったり、これを元に必要な体裁に改造したりしてください。

欧文限定ならば、BibLaTeX の主要フォーマットは Overleaf にサンプルが掲載されています。

https://ja.overleaf.com/learn/latex/Biblatex_bibliography_styles

文献データベースは .bib とほぼ同じものが使えますが、jauthoryear で適切に表示するには以下の2つのフィールドへの記入が必要です。

- a. language フィールド
- b. yomi または sortname フィールド

(a) は和・欧の分類のために必要で、それぞれ Japanese, English と入力してください (他の言語でもたぶん大丈夫です。保証はしません)。例えば BibTeX の jecon.bst は文字コードから言語を判別していましたが、現時点では jauthoryear にそのような機能はありません。

(b) は従来の BibTeX の yomi と同様に、五十音順を決定するための読みがなを記入するフィールドです。BiBLaTeX では一般には読みによる掲載順は sortname フィールドで指定しますが、jauthoryear では日本語圏での BibTeX との互換性を考え yomi フィールドでも代替できるようにしています。もちろん五十音順でなくてもよいのなら記入しなくてもかまいません。例えば Zotero ではその他欄に tex.yomi: ... と記入することでエクスポートにも反映されます

詳細は私の以前書いたブログ記事を参考にしてください。

<https://ill-identified.hatenablog.com/entry/2020/09/20/231335>

12.4 natbib (BibTeX)

BibTeX も PDF でのみ有効です。R Markdown では `natbib` という名称になっています。これは `natbib.sty` という LaTeX パッケージを使用しているからです。`natbib.sty` は [1] のような番号形式や、著者（年）形式などいろいろな表示形式をサポートしています。R Markdown では基本的に `natbib` と BibTeX がセットになっています。典型例としては以下のような YAML メタデータを書くことになります。

```
output:
  XXX_document:
    citation_package: natbib
bibliography: REFS.bib
biblio-style: ieee
```

BibLaTeX 同様に、HTML の場合のために `cs1` フィールドも同時に書くことができます。

`natbib.sty` は、例えば `\citet{引用 ID}` でテキスト形式引用、`\citep{引用 ID}` で括弧で囲んだ引用、`\citeyear{引用 ID}` で発表年のみ、あるいは `prettyref.sty` や `refstyle.sty` と併用することで図 XX のように書式付きの引用を行うことができます。^{*22} ただしこれらは LaTeX コマンドであるため、HTML では反映されません。PDF と HTML で出力を両立させたいなら、Markdown 構文の `@引用 ID`、`[@引用 ID]` を使用してください。^{*23}

BibTeX の場合、引用スタイルを決めるのは `natbib.sty` で、文献リストのスタイルは `.bst` ファイルによって決まります。

日本語圏では BibTeX を改造し日本語に対応させた `pBibTeX` または `upBibTeX` が最もよく利用されます。しかし、これらは LaTeX を制御する `tinytex` の `latexmk` エミュレーションが対応していませんし、`.bst` ファイルは `BibLaTeX` と互換性がありません。さらに、日本語を想定した `.bst` ファイル（典型例、`jipsj.bst`、`jplain.bst`、`jecon.bst`、などなど）の多くは `BibTeX` と互換性がなく、多くの場合は `BibTeX` の処理中にハングアップします。

よって、日本語を想定した `.bst` ファイルを使う場合、`natbib` でも `biblatex` でも動作しないことになります。そこで `latexmk` エミュレーションを無効にする必要がありますが、v.0.4.5 以降の `rmdja` が提供する PDF 系の出力フォーマットでは `natbib` を指定した場合に自動でこれが無効にされるため、**手動での手続きは不要です**。もし `natbib` にしつつ エミュレーションを有効にする必要がある場合^{*24} は、出力オプション `latex.emulation = T` を設定してください。

^{*22} 例えば Overleaf のページ https://www.overleaf.com/learn/latex/natbib_citation_styles に解説があります。日本語ですとこのページに出力例があります。 <http://otoguro.net/home/latex/bibtex/>

^{*23} 実際にはこれら 2 つは PDF 出力時に Pandoc によって内部でそれぞれ `\cite`、`\citep` へと変換されています。

^{*24} これはまれだと思います。本当に必要なのかよく確認して下さい。

このように、.bst ファイルを使う場合は処理がややこしいため、論文誌や学会がスタイルのルールを課していない場合は CSL や BibLaTeX にしたほうが簡単です。

<https://bookdown.org/yihui/rmarkdown/pdf-document.html>

13 その他

13.1 R Markdown 関係の役に立つパッケージ

R Markdown にはいろいろな派生パッケージや、レポート作成支援機能のあるパッケージが存在します。例えば多彩なテンプレートや出力フォーマットを提供してくれるパッケージがあります。CRAN に登録されていないもののみ、開発元のリンクを張っています。

- **bookdown**: 今回は相互参照機能のみ取り上げましたが、本来は書籍フォーマットを提供するものです
- **bookdownplus**: **bookdown** をさらに拡張して多彩な書籍フォーマットを容易。ただし日本語対応してるとは限りません
- **rticles**: 多くの分野の英語論文フォーマットをサポート
- **oxforddown**: Oxford の学位論文用...らしい
- **jpaRmd**: 国内の心理学論文フォーマットをサポート
- **blogdown**, **govdown**, **rmdformats**: Web サイト開発向け
- **pagedown**: CSS 組版
- **flexdashboard**: shiny ダッシュボードを R Markdown と統合することを目的としています
- **distill**, **tuftes**: 論文風の Web ページフォーマットを提供します
- **gm**: 楽譜の表示と再生ができるそうです
- **xaringan**: remark.js をベースとしたスライド作成用パッケージです。
- **officer**, **officedown**: デフォルトの Word/ パワーポイントフォーマットに機能を追加したものです。後者をインストールすれば事足ります。
- **redoc**: 共同作業者が Word しか使えないことを想定し、逆にインポートもできるように開発されたようですが、現在は中断しているようです。

作成支援系のパッケージには、RStudio アドインを提供するものがあります。

- **yamlthis**: YAML の設定をウィンドウで行えます。ただし、**rmarkdown** など基本パッケージのみの対応のようです。
- **citr**: 引用文献リストの検索と、参照 ID のコピーペーストが行えます。Zotero をお使いなら連携もできます。ただし、**bookdown** のような文書を分割するものとは相性が悪いかもしれません。
- **bookdown**: 語句引用ブロックの挿入など、いくつかのお役立ちショートカット機能があります。

13.2 YAML メタデータと出力フォーマットの設定方法について

- **includes-headers** は全てのフォーマットで読み込まれるため、HTML と LaTeX それぞれで設定が必要な場合は出力フォーマットの **includes**, **md_extensions**, **pandoc_args** などを使うことになります。

- YAML のトップレベルの設定は Pandoc に送られるが、この中には出力フォーマットと重複する項目がいくつかあります。

やや大雑把ですが、この問題の原因について解説した記事

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

これは Pandoc のシステムを使いまわしているのが原因です。現時点では未解決ですが、Pandoc の YAML によるインターフェイスを洗練するという動機で開発されている Quarto というものがあります。あくまで憶測ですが、このようなインターフェイスが R Markdown においても採用されるようになるかもしれません。

13.3 どうしても `rmdja` を使いたくないが、PDF で (日本語文書を) 出力したいという人

以下のような設定で、最低限ですが動きます。ただし細部のスタイルが不自然だったり、凝ったことをやろうとするとうまくいかないことがあります。

通常の文書 (論文、レポートなど)

```
output:
  pdf_document:
    latex_engine: lualatex
documentclass: ltjsarticle
classoption: haranoaji
```

プレゼン資料 (beamer)

```
output:
  beamer_presentation:
    latex_engine: lualatex
mainfont: Haranoaji Gothic
```

13.4 文書のコンパイル時にエラーがでたら

凝ったことをしようとすると、エラーが起こりやすくなります。シルバーバレットは存在しませんが、いくつか有効な方法があります。

まずはエラーがどこで発生しているのか確認してください。これには R Markdown の内部処理の順序も知っておくとよいでしょう (セクション 13.8)。おおむね、以下の処理のタイミングより前か後かを見極めるとほとんどのエラーの原因がわかります。

1. YAML メタデータの読み込み時のエラー

- YAML メタデータを書き間違えています
2. knitr 処理 (コードチャンクの実行) 時のエラー
 - つまり R のコードに問題があります
 3. Pandoc のエラー
 4. (PDF のみ) LaTeX のエラー
 - .tex のソースコードに問題があります.

出力フォーマットに `keep_md` や `keep_tex` というオプションがあれば、有効にしてください。前者はコードチャンクを実行し、変換した後の `md` ファイル、つまりプログラムを含まない Markdown だけのファイルを、後者は PDF を出力する場合の `.tex` ファイルを残します。これらは変換の中間処理に使うもので、最終的な出力には不要です。しかし最終的出力がおかしかったり、そもそもエラーで出力されなかった場合はこれらの中間ファイルに異常がないか確認するとよいでしょう。

13.5 ハマりがちな LaTeX エラー

13.5.1 Unable to read an entire line

Unable to read an entire line---bufsize=200000.

1 行あたりのバッファオーバーです。巨大な画像を貼り付けすぎか、R のコードミスで意味不明な長大文字列を出力してるとかを疑ってください。それ以外の理由なら、LaTeX の 1 行あたりのバッファサイズを変更してください。

13.5.2 Illegal parameter number in definition

```
! Illegal parameter number in definition of \iterate.
<to be read again>
      e
```

たぶん数式の相互参照がおかしいです。

ヒューマンエラーだけでなく、ビジュアルエディタを使っていると勝手に不自然な記法に変換されることがあるようです。

13.5.3 画像のエラー

SVG をそのまま貼り付けようとした。LaTeX は `.svg` をそのまま貼ることはできないので、`dev=svglite` にするとうまくいかない。pdf, png 等他のフォーマットに変換する必要があります。

13.6 rmdja::beamer_presentation_ja で Fira Sans がどうこうという警告がうざい

rmdja では metropolis テーマを想定しています。このテーマは

1. シンプルで従来の beamer テーマと比べ古臭くない (個人の感想です)

2. 比較的
3. 日本語対応が簡単

という理由で採用しています

たぶんこんな警告が出ます.

```
1: Package beamerthememetropolis Warning: Could not find Fira Sans fonts on input
line 95.
```

```
Package beamerthememetropolis Warning: Could not find Fira Mono fonts on input
line 95.
```

文字通り Fira Sans, Fira Mono フォントがインストールされていないときに発生する警告です. 基本的に無害ですが, 気になるならインストールしてください. Google Fonts で無料配布されています.

<https://fonts.google.com/specimen/Fira+Sans>

<https://fonts.google.com/specimen/Fira+Mono>

13.7 スクリプトで R Markdown をコンパイルする

R Studio で「Knit」ボタンを押すことでできる文書のコンパイルは, R のスクリプトでも可能です. 例えば:

```
01 rmarkdown::render("input.Rmd", output_format = rmarkdown::html_document(code_folding
   ↪ = "hide"))
```

逆に言えば, 「knit」ボタンは裏でこのコマンドを呼び出しているということです. 微妙に設定を変えた文書を複数作成したい場合は, これが役立つかもしれません.

13.8 R Markdown コンパイル処理の内容

`output_format` の各引数は次のような順番で評価されます. 各処理は関数として与えることに注意してください. つまり, いわゆる *generating function* です. 対話的なプログラミングに慣れていると, 各処理が実際に実行されるタイミングで環境がどうなっているかを考える脳の部分が衰えているかもしれませんが, がんばってください.

既存の出力フォーマットのなかにはデフォルトで処理が設定されているものもあるため, `pdf_document` など既存の出力フォーマットを改造する形で作成する際は注意が必要です. 単純に上書きしてしまうと意図したとおりにならないことがあります.

1. YAML メタデータの読み込み
2. `pre-knit`
 - `knit` コマンド実行前の `knit` のオプション設定ステップ

3. (knit 実行)
 - `knitr` で Rmd 内の設定を無視して書き換えることもできます
4. `post_knit`
5. `intermediates_generator`
6. `pre_processor`
 - YAML メタデータやソースファイルの場所などを引数に取る関数です
 - よって, Pandoc に渡す引数などを書き換えることができます
 - `rmdja` で日本語特有の環境に対応するための小細工も主にここで行っています
7. (Pandoc 実行)
8. `post_processor`
 - `pre_processor` と同じ引数を持つ関数を与えられます
9. `clean_supporting`
 - 中間ファイル削除ステップです
 - Rmd に必須の中間ファイルはデフォルトで削除されますが, ここまでの処理で一時ファイルを作成する処理を作った場合, ここで削除するのがスマートです
10. `on_exit`

詳細は公式ドキュメントまたは kazutan 氏の解説スライドをご覧ください.

http://kz-md.net/stat/tmp_box/intoTheRmarkdown.html#/

13.9 HTML 版の印刷時に色が消える

この問題は先日 R-wakalang で質問されたものです.

<https://r-wakalang.slack.com/archives/C06QP6NJ0/p1618105329127900>

これは, R Markdown の HTML 出力のスタイルの多くが Bootstrap というライブラリに依存しており, この設定が印刷時に背景色を無効にするようになっていきます. インクの節約の観点から, 嫌がらせ目的ではないと思いますが, 一方でブラウザの表示のまま印刷したい場合もあります.

CSS に `background-color: 色名 !important` のように設定すれば印刷時にも背景色が反映されますが, `formatter` など一部の作表パッケージはこのことを考慮していません. この問題は開発元にも報告されていますが, 現時点では解決していません^{*25}

よって, 現状は出力される HTML ファイルを手直しする必要があるかもしれません. CSS や Lua フィルタでなんとかするかもしれませんが, 私は Bootstrap の全容を把握していないのでちょっと自信がありません.

`rmarkdown` の表の標準は `knitr::kable()` なので, 現状はそれを拡張した `kableExtra` を使うのが無難だと思います.

他にも `flextable`, `huxtable`, `gt` といった R Markdown を想定したパッケージがあり, これらはより革新的

^{*25} <https://github.com/renkun-ken/formattable/issues/53>

な機能を搭載していることが多いですが、そのぶん開発途上だったり堅牢さに問題があることがあります。なお Yihui 氏の思想としては表の装飾にあまりこだわるべきではないということです。

13.10 Rpubs でハイパーリンクに失敗する

<https://support.rstudio.com/hc/en-us/articles/201105636-Using-external-links-in-RPubs>

[代替テキスト](URL){target="_blank"} のように書けば、ハイパーリンクに属性を追加できます。代替テキストなしの場合は <URL>{...} となります。

13.11 その他の関連するドキュメント

- 金・高橋『ドキュメント・プレゼンテーション生成』
- 江口『自然科学研究のための R 入門』*『今日からできる再現可能な論文執筆』

この文書作成時の環境

```
01 sessionInfo()

## R version 4.0.5 (2021-03-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.2 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
##
## locale:
##  [1] LC_CTYPE=ja_JP.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=ja_JP.UTF-8      LC_COLLATE=ja_JP.UTF-8
##  [5] LC_MONETARY=ja_JP.UTF-8  LC_MESSAGES=ja_JP.UTF-8
##  [7] LC_PAPER=ja_JP.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=ja_JP.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
```



```
## other attached packages:
## [1] kableExtra_1.3.4 officer_0.3.18 officedown_0.2.2 forcats_0.5.1
## [5] stringr_1.4.0 dplyr_1.0.5 purrr_0.3.4 readr_1.4.0
## [9] tidyr_1.1.3 tibble_3.1.1 ggplot2_3.3.3 tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
## [1] httr_1.4.2 viridisLite_0.4.0 jsonlite_1.7.2 R.utils_2.10.1
## [5] modelr_0.1.8 assertthat_0.2.1 cellranger_1.1.0 yaml_2.2.1
## [9] gdtools_0.2.3 pillar_1.6.0 backports_1.2.1 glue_1.4.2
## [13] uuid_0.1-4 digest_0.6.27 rvest_1.0.0 colorspace_2.0-0
## [17] htmltools_0.5.1.1 R.oo_1.24.0 pkgconfig_2.0.3 broom_0.7.6
## [21] haven_2.4.0 bookdown_0.22 webshot_0.5.2 scales_1.1.1
## [25] svglite_2.0.0 styler_1.4.1 generics_0.1.0 farver_2.1.0
## [29] ellipsis_0.3.1 cachem_1.0.4 withr_2.4.2 cli_2.4.0
## [33] magrittr_2.0.1 crayon_1.4.1 readxl_1.3.1 memoise_2.0.0
## [37] evaluate_0.14 R.methodsS3_1.8.1 fs_1.5.0 fansi_0.4.2
## [41] R.cache_0.14.0 xml2_1.3.2 tools_4.0.5 hms_1.0.0
## [45] lifecycle_1.0.0 munsell_0.5.0 reprex_2.0.0 zip_2.1.1
## [49] compiler_4.0.5 systemfonts_1.0.1 rlang_0.4.10 grid_4.0.5
## [53] rstudioapi_0.13 labeling_0.4.2 rmarkdown_2.7 gtable_0.3.0
## [57] DBI_1.1.1 rematch2_2.1.2 R6_2.5.0 rvg_0.2.5
## [61] lubridate_1.7.10 knitr_1.32.9 fastmap_1.1.0 utf8_1.2.1
## [65] rmdja_0.4.5 stringi_1.5.3 Rcpp_1.0.6 vctrs_0.3.7
## [69] dbplyr_2.1.1 tidyselect_1.1.0 xfun_0.22
```

参考文献

Allaire, JJ, Yihui Xie, Jonathan McPherson et al. (2021) *rmarkdown: Dynamic Documents for R*, retrieved from [here](#), R package version 2.7.

Aust, Frederik (2019) *citR: RStudio Add-in to Insert Markdown Citations*, retrieved from [here](#), R package version 0.3.2.

Xie, Yihui (2020) *bookdown: Authoring Books and Technical Documents with R Markdown*, retrieved from [here](#), R package version 0.21.

Xie, Yihui, Christophe Dervieux, and Emily Riederer *R Markdown Cookbook*, The R Series: Taylor and Francis, CRC Press, 1st edition, retrieved from [here](#), 翻訳版: <https://gedevan-aleksizde.github.io/rmarkdown-cookbook/>.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund *R Markdown: The Definitive Guide*: CRC Press, Taylor

and Francis Group, retrieved from *here*.

江口哲史 自然科学研究のための *R* 入門 in , Wonderful R, No. 4: 共立出版.

金明哲・高橋康介 ドキュメント・プレゼンテーション生成 in , Useful R, No. 9: 共立出版.

高橋康介 再現可能性のすゝめ *RStudio* によるデータ解析とレポート作成, Vol. 3 of Wonderful R: 共立出版.