

Rmd から L^AT_EX 経由で言語学の論文・レポートを出力するには

beta ver. 2021/3/ 1 版

小川雅貴

2021/1/31 (最終更新: 2021/3/ 1)

概要

本資料では、統計プログラミング言語 R から言語学の論文・レポートを生成する方法を紹介する。これまで言語学の論文・レポートは、Microsoft Word や L^AT_EX から作成されてきた。しかし、Microsoft Word では、参考文献および図表・例文番号の自動付与と整形に加え、グロス付与・句構造木の描画のような言語学特有の記述も難しかった。また、L^AT_EX では、参考文献等の自動処理や言語学特有の記述は可能だったが、コマンドと本文（地の文）が混然一体となり見難く、かつプログラミングの慣熟までに高い敷居があった。そこで本資料では、より平易な文書生成プログラミングである RMarkdown を紹介する。これにより、参考文献および図表・例文番号の処理と言語学特有の記述の体系的自動化が可能になり、かつコマンドと地の文をより容易に識別しながら執筆できる。しかも、統計プログラミング結果も本文に直に・即時更新で取り込むこともできるようになる。本資料は、そうした技術を、言語学分野への応用を念頭に集約したものである。なお、現在の資料は、現在は RMarkdown から L^AT_EX を経由して PDF を出力する（但し L^AT_EX は極力使わない）ことに特化している。そのため近い将来、RMarkdown から docx/pptx ファイルを出力したり、beamer（スライド・ポスターを出力する L^AT_EX パッケージ）・revealjs（HTML ベースのスライド）を出力する方法は別途まとめた。

目次

1	とりあえずプログラミングで文書を作成してみよう	3
1.1	何をどうするの…?	3
1.2	R project（拡張子 .Rproj の設定）	4
2	RMarkdown ファイルの編集	6
2.1	本節以降の流れ	6
2.2	R パッケージのテンプレートから Rmd ファイルを作る	6
2.3	(R)Markdown の文法	6
2.4	*Office での操作に近い形で Rmd 編集をする	11
3	別ファイルの埋め込みと実行	12
3.1	別ファイル（子ファイル）の作成	12
4	本文中でのコード実行	13
4.1	本文中でのコード実行	13
4.2	地の文に小さな R チャンク（inline code）を埋め込む	14
5	図表番号等の自動参照	14
5.1	図表番号等の自動参照	14
6	文献の自動挿入と整形	14
6.1	文献の自動挿入	14

6.2	文献の自動整形	16
7	YAML セクションで論文の体裁を指定する	18
7.1	文書のメタ情報	18
7.2	ファイル冒頭の YAML でメタ情報を指定	18
7.3	さらに色々なメタ情報を入れるには	19
8	例文	23
8.1	pandoc の <code>example_lists</code> を使う	23
8.2	pandoc-ling を使う	24
9	本資料について	33
9.1	本資料のクリエイティブ・コモンズ・ライセンス	33

1 とりあえずプログラミングで文書を作成してみよう

プログラミングで文書を作成するとは、「編集するファイル（編集物）をプログラムに処理させて、成果物である PDF や docx/pptx ファイル等を得る」ことである。ここで、**編集物と成果物は別個のファイル**である。Microsoft Office（Word や PowerPoint）のように、編集物と成果物が同一となるのではない。

プログラムに処理させる際に、「この内容は成果物に反映させるが、この内容は考えがまとまっていないため成果物にはまだ入れない」というように内容を選択的に反映させることが可能である。論文・予稿のように、文書レイアウト規格が決まっている文書を生成する際は、規格を守ることをプログラミングで自動化させることで、規格を厳格に守れるようになるとともに、内容の充実により注力できるようになる。

文書生成プログラミングには \LaTeX や HTML など様々あるが、その中でも Markdown が最も覚えやすく始めやすいと考える。なぜならプログラムの文法が極めて単純であるためである。また、Markdown を使えば、1つの入力ファイル（Markdown ファイル）から、 \LaTeX を経由した）PDF 作成、HTML 作成、docx/pptx 作成など、複数種類の出力ファイルを生み出すことも容易である。

また、Markdown には様々な方言があるが、その中でも、統計プログラミング言語 R に根ざした RMarkdown が使いやすいと考える。なぜなら、第4節で述べる通り、R による計数データの処理が可能になるためである。しかも、RMarkdown は、R だけでなく、Python 等の他プログラムの実行も出来るように設計されている。さらに、RMarkdown での文書生成を支援するパッケージも多岐に渡って開発されており、非常に充実している。



1.1 何をどうするの…？

1. (RStudio で、RMarkdown をさらに書きやすくする設定を施す)
 - R project（ディレクトリ）の設定
2. 入力となる Rmarkdown ファイル（拡張子 .Rmd, .rmd）を、RStudio で編集
 - 本文の編集（第2.3節）
 - 地の文
 - コードブロック（プログラミングを行う部分；第4節）
 - 文書のメタ情報の編集：YAML セクションの編集
 - 著者名
 - タイトル
 - 日付
 - 出力形式
 - フォント設定など
3. RStudio で、Rmd ファイルから出力を生成する
 - これを **knit**（編む）という
 - だから RMarkdown の基礎パッケージに knitr がある
 - knit+r
4. 出力された成果物が表示される
 - このワークショップでは、docx/pptx と PDF を生成する
 - 編集自体は RStudio でなくても、VSCode・Atom など別のテキストエディタでも可能
 - RStudio の方が、出力を生成する時に便利
 - VSCode や Atom でも、追加パッケージを入れれば生成できる

1.2 R project (拡張子 .Rproj の設定)

R project

- R および RStudio が、作業ディレクトリを認識するためのタグのようなもの
 - 作業ディレクトリ (Working directory) :
現在編集中のファイルを格納しているフォルダ
- 作業ディレクトリに .Rproj ファイルが出来ると、その .Rproj ファイルをクリックするだけで、RStudio が起動し、同時に作業ディレクトリの内部構造も把握する
 - プログラミングで文書を作成するには、プログラムにファイルの在処を教える必要がある
 - その際、.Rproj ファイルがあると、R や RStudio にファイルの在処を教えるのが楽になる
- 文書生成に必要なもの (Rmd ファイル、画像ファイル) やデータ (csv, xlsx など) は、作業ディレクトリ内にあると良い
 - シンボリックリンクでも良い
 - * シンボリックリンク :
作業ディレクトリにないファイルの在処を、作業ディレクトリに記したリンク

1. 画面右上に  というボタンがあるので、そこをクリック
 - 初めて RStudio を使う場合には、(None) と書かれているかもしれない
2. 出てきたバナーの最初に「 New Project…」というボタンがあるので、そこをクリック

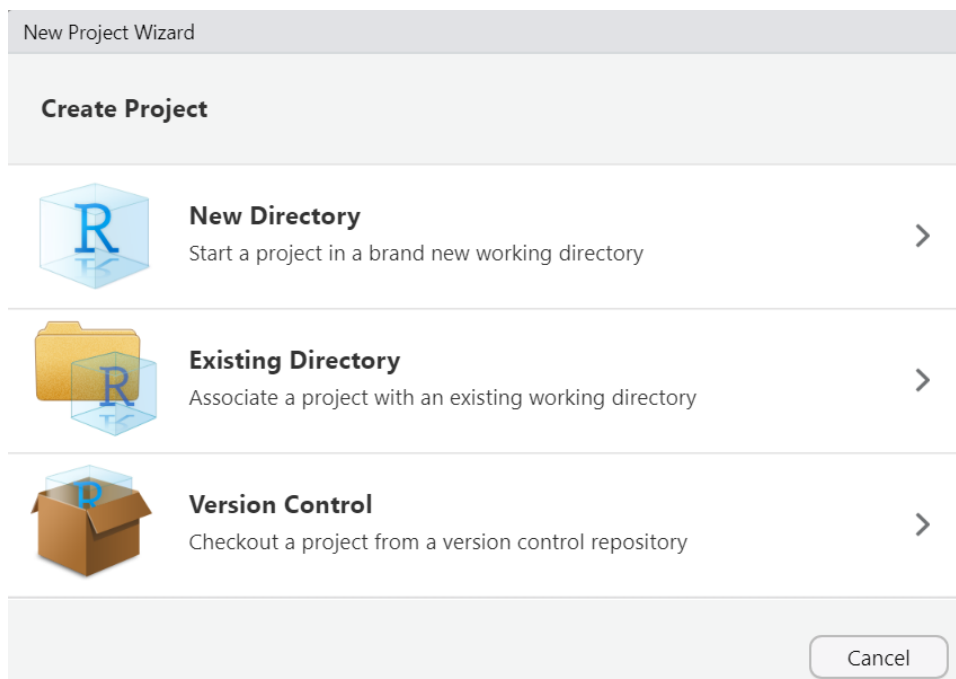


図1: Create Project

- ここでは、「New Directory」を選択
 - 新しいディレクトリ (フォルダ) が出来る
 - 同時に、その新しいディレクトリに紐づいた .Rproj ファイルができる
- 「Existing Directory」を選んでもよい
 - 既にあるディレクトリに .Rproj ファイルを作成する

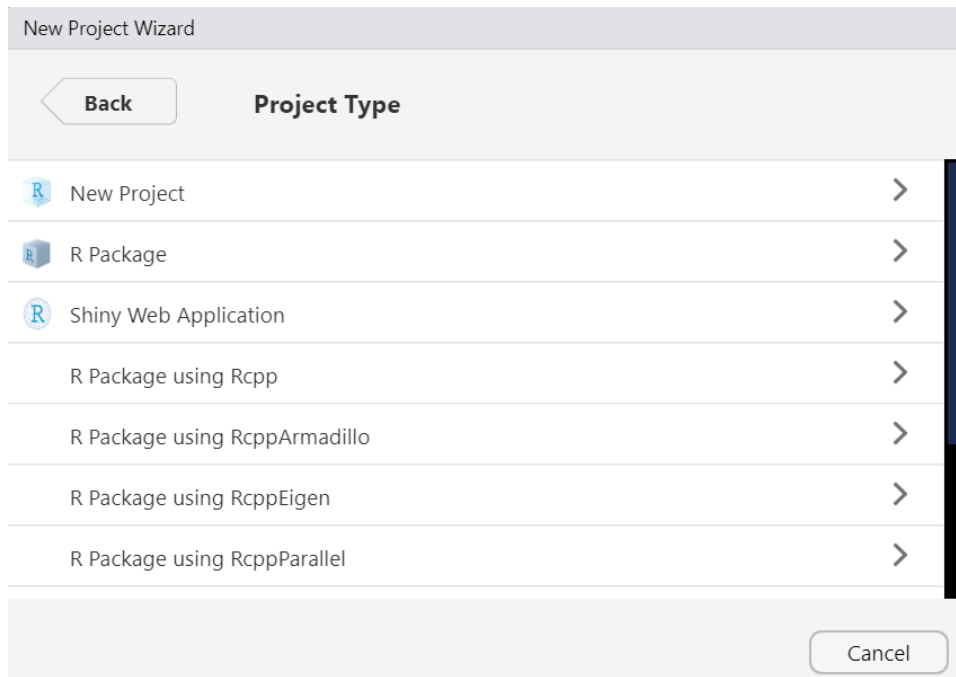


図2: Project Type

1. 先程の画面で「New Directory」を選択した場合、図2のような画面が出る
2. 「R New Project」を選択

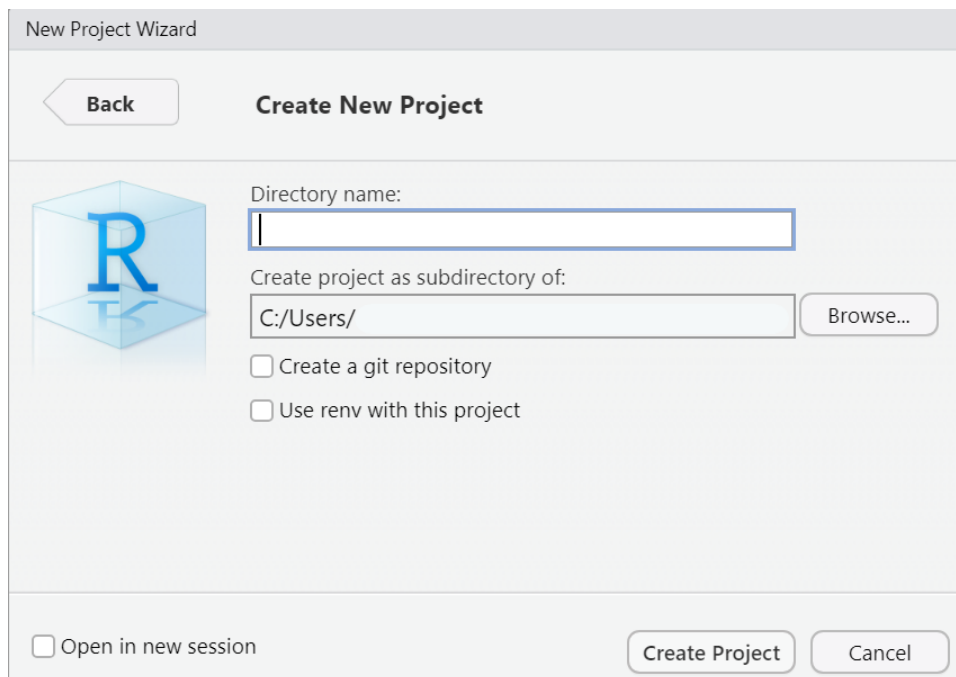


図3: Create New Project



1. 「Directory name」で新しいディレクトリの名称を設定
2. 「Create Project」をクリックすると、新しいディレクトリと `.Rproj` ファイルが出来、RStudio が再起動

2 RMarkdown ファイルの編集

2.1 本節以降の流れ

1. まず Rmd ファイルのテンプレートを RStudio で開き、文書生成 (knit) を体験
2. 次に Rmd の文法をかいつまんで紹介するので、テンプレートを編集していく
3. その後、「RMarkdown で文書を作ると可能になる 4 つのこと」を実践
 1. 別ファイルの埋め込みと実行
 2. 本文中でのコード実行
 3. 図表番号等の自動参照
 4. 文献の自動的挿入と整形

2.2 R パッケージのテンプレートから Rmd ファイルを作る

1. RStudio の画面の左上にある  ボタンを押す
2. 出てきたポップアップ上で、「 R Markdown…」ボタンを押す
3. 「Default Output Format」に PDF を指定
4. UntitledX (X は数字) というファイルが出てくる
5. 好きなファイル名で保存
 - ここでのファイルは編集ファイルのこと
 - 保存のショートカットキーは Ctrl/Cmd + S
6. 保存したファイルを開きながら、Ctrl/Cmd + Shift + K を押して文書生成
 - これが knit
7. (この場合) Word のファイルが出てくる
8. **Rmd** を編集する前に、**Word** を閉じる
 - docx/pptx を出力する時は、knit 毎に Word/PowerPoint を閉じる
 - 前の knit で出力した docx/pptx を閉じないと、knit できない

2.3 (R)Markdown の文法

(R)Markdown の文法の完全版は、下記の資料を参照するとよい

- Markdown の早見表
 - 文法自体が、早見表の 1/4 に収まる量
 - 日本語版 https://rstudio.com/wp-content/uploads/2016/11/Rmarkdown-cheatsheet-2.0_ja.pdf
 - 英語版 <https://rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf>
 - * 中国語版・韓国語版もある（上記より前のバージョンのものだが、情報はほぼ共通）
 - * 中国語版 <https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-chinese.pdf>
 - * 韓国語版 <https://github.com/rstudio/cheatsheets/raw/master/translations/korean/rmarkdown-cheatsheet-kr.pdf>
- Pandoc で使われる文法
https://rmarkdown.rstudio.com/authoring_pandoc_markdown.html
 - RMarkdown もほぼ共通

- RMarkdown と Pandoc Markdown を同一ファイル内で書いても OK
- ここでは、学術文書作成でよく使いそうなものをピックアップ
- このスライド以降で紹介するので、先ほど保存した Rmd ファイルを編集する
 - この段階では、先ほどの保存した Rmd ファイルの 7 行目以降なら、どこを編集してもよい
 - Markdown の練習サイトもある
 - * 自分のペースであれこれ試すなら
<https://daringfireball.net/projects/markdown/dingus>
 - * 練習問題を解きながら、ひとつひとつ覚えたいなら <https://commonmark.org/help/tutorial/>
- 使っていくうちに覚えていく
 - 覚えてからでないと使えないわけではないので、気楽に・気長に

2.3.1 地の文

これは地の文です。

1 行以上空行を入れると空けると改段落します。

(R)md ファイル上で行を詰めて書くと
改段落されません。

「編集中のファイル上では主節と従属節を分けて読みやすくしつつ、
成果物では主節の直後に従属節が続くように書く」といった使い方が出来ます。

強制改行するには

改行したい行末に半角スペース 2 つ ` ` かバックスラッシュと半角スペース 1 つ `\
` を書きます。

出力結果

これは地の文です。

1 行以上空行を入れると空けると改段落します。

(R)md ファイル上で行を詰めて書くと改段落されません。「編集中のファイル上では主節と従属節を分けて読みやすくしつつ、成果物では主節の直後に従属節が続くように書く」といった使い方が出来ます。

強制改行するには

改行したい行末に半角スペース 2 つ ` ` かバックスラッシュと半角スペース 1 つ `\
` を書きます。

2.3.2 Rmarkdown だけにメモを残すコメントアウト

- 原稿を書いている時には、「現段階では論文中からは除外しておきたいものの、次の稿では掲載するかもしれない文面」が生じうる
- docx ファイルを直に編集している場合
 - 除外したい文面は word の「コメント」機能を使い、コメント欄にコピペしておけるが…
 - docx ファイルでは、コメントの数が増えるほど、docx ファイルのサイズが肥大化し、最悪、Word プログラムの予期せぬ停止や docx ファイルの破損につながる恐れ
- Rmd で編集しその結果を別の最終ファイル（ここでは docx ファイル）として出力する場合
 - 「編集の際に生じたアイデアを Rmd には残しつつ、最終ファイルである docx には載せない」ことができる

- これがコメントアウト

- * R スクリプトでコードに関するメモをする際に、# を使い # 以降にメモを残すことがあるが、その Rmd 版

- プログラムの動作にほぼ影響しない

- * ファイルサイズも増えない

Rmd 上のある文面をコメントアウトする方法

- その文面 ... を <!--...--> で囲む
- RStudio のショートカットキーは Ctrl/Cmd + Shift + C(omment)
 - コメントアウトしたい行にカーソルを合わせてからこのショートカットキーを押す
 - その行を囲むように自動で <!--...--> が挿入される

例：夏目漱石の『草枕』の冒頭

原文

「山路を登りながら、こう考えた。智に働けば角が立つ。情に棹させば流される。意地を通せば窮屈だ。とにかく人の世は住みにくい。住みにくさが高じると、安い所へ引き越したくなる。どこへ越しても住みにくいと悟った時、詩が生れて、画が出来る。」

```
<!-- 山路を登りながら、こう考えた。 -->
智に働けば角が立つ。
情に棹させば流される。
<!-- 意地を通せば窮屈だ。 -->とにかく人の世は住みにくい。
<!--
住みにくさが高じると、安い所へ引き越したくなる。
どこへ越しても住みにくいと悟った時、詩が生れて、画が出来る。
-->
```

出力結果

智に働けば角が立つ。情に棹させば流される。とにかく人の世は住みにくい。

2.3.3 文字の強調（イタリック・太字）

The word *italicise* is italicised, **bold** is bold and ***bold-italic*** is bold italic.

日本語の文字の場合、* イタリック * にはならないが、** 太字 ** にはなる

- * で文字を強調
 - 左の例では、出力上、日本語の文字と * の間に半角スペースが入っているが、こうしたスペースは入れないこと
- RStudio で編集している場合、単語を選択してから Shift + : を押すと、単語が * で囲まれる
- (31 日に扱う方法で) PDF を出力する場合、日本語はイタリックにも斜体 (Oblique/Slanted) にもならない
 - イタリックデザインのフォントを使うか、 \LaTeX のコマンドを書けば可能
 - * そこまでして斜体にしなければならない状況は、狭義の学術文書生成では少ない？

- docx/pptx を出力する場合は, * による命令で可能

2.3.4 見出し

見出し 1 (「章」に相当)

これは, 見出し 1 の地の文です。これは, 見出し 1 の地の文です。

見出し 2 (「節」に相当)

これは, 見出し 1 の下の見出し 2 の地の文です。

見出し 3 (「小節」に相当)

これは, 見出し 1 の下の見出し 2 の下にある, 見出し 3 の地の文です。

見出し 1・2 個目 {#sec:ch-second}

第\@ref(sec:ch-second) 章より, 新しい章が始まります。

- # の数が少ないほど上位の見出し
- 見出し 6 (#####) まで設定可能
- 文書クラスとスライドで # の意味がやや異なる

	論文	スライド
#	章	セクション
##	節	スライドタイトル・##=1 枚
###	小節	ブロック見出し

- 見出しの参照も可能
 - 見出し側のタグ: {#...}
 - 例: {#name}
 - 本文での参照 (Rmd 記法):
 - \@ref(...)
 - 例: \@ref(name)
 - 図の参照などでも同様 (後述)

2.3.5 図の挿入

```
<!-- 基本形 -->
![図タイトル](path/to/figure.png)
```

```
<!-- 本文で図を参照する場合 -->
```

```
![(#fig:fig-ref) 図タイトル](path/to/figure.png)
```

図\@ref(fig:fig-ref) に、図参照の方法を示した。

```
<!-- 図のサイズを調整する場合 -->
```

```
![図タイトル](path/to/figure.png){width=75%}
```

```
<!-- サイズを調整した図を本文で参照する場合 -->
```

```
![(#fig:fig-ref2) 図タイトル](path/to/figure.png){width=50%}
```

図\@ref(fig:fig-ref2) も、図参照の方法である。

- () に、挿入したい図のパス（保存場所；アドレス）を書く
 - R Project を設定したディレクトリ内に画像ファイルがある場合、相対パス指定ができる
 - * 例：R Project を設定したディレクトリに下位ディレクトリ **figures** があり、さらにその中に **picture.png** がある時：
figures/picture.png
- 指定できる画像ファイル
 - docx/pptx を出力する場合、png と jpeg
 - PDF を出力する場合、これらに加え pdf も
- サイズは {height=80%} でも指定可能
 - 基本的には height か width のいずれかを設定
- 図を参照する際のタグ：(#fig:...)
 - 図タイトルを書く [] の中に書く

2.3.6 表の挿入

```
|      | 係数 | 標準誤差 | $p$ 値 |
|----:|:----:|:-----:|:-----:|
| 切片 | 2.75 | 0.63      | 0.04    |
| 傾き  | 0.82 | 1.44      | 0.1     |
```

: 表の例 (#tab:tab-ref)

表\@ref(tab:tab-ref) に解析結果を示した。

```
|      | 産出数 | % |
|----:|:----:|:-----:|
| 能動文 | 52 | `r round(52/(52+96)*100, 2)` |
| 受動文 | 96 | `r round(96/(52+96)*100, 2)` |
```

: 表の例 2 (#tab:tab-ref2)

産出文数は、表\@ref(tab:tab-ref2) の通りである。

- Rmd で表を書く方法は複数ある
 - セル内改行が出来る形式は、下記を参照のこと：
https://rmarkdown.rstudio.com/authoring_pandoc_markdown.html#Tables
- セル内にプログラミング言語のコードを入れて実行することもできる
 - 「表の例 2」での `r ...` の箇所
 - 実行結果は表2の通り

出力結果

表2: 表の例 2		
	産出数	%
能動文	52	35.14
受動文	96	64.86

2.4 *Office での操作に近い形で Rmd 編集をする

- 「(R)Markdown の文法覚えるのキツイ…」と思った方に朗報
- RStudio には「Office での操作に近い形で Rmd 編集をする」機能 Visual Markdown Editor がついている
 - RStudio 1.4 から始まった新しい機能

注意 !!

- 既に自分で書いた Rmd がある場合、その Rmd を Visual Markdown Editor で編集しない方がいい
 - Visual Markdown Editor が認識できる文法は、まだ限定的
 - Visual Markdown Editor が認識できない文法で書かれていた場合、その箇所が消失する可能性がある
 - Visual Markdown Editor に掛けようとする警告が出てくるので、本当に必要な操作か一旦考える
- 新規作成の Rmd を Visual Markdown Editor で書いていくのは問題ない
 - 但し、その場合も「Visual Markdown Editor から手書きに戻し、手書きで編集した後に、もう一度 Visual Markdown Editor に掛ける」ことは避けた方がよいかも

2.4.1 Visual Markdown Editor の始め方

1. Source ペインの右上に「A」のようなボタン（実際にはコンパスの図案）があるので、そこをクリック
2. Rmd ファイルを開けている画面が、Word を開けているような画面に変わる

2.4.2 Visual Markdown Editor の使い方

基本的に Word や PowerPoint のショートカットキーがそのまま使える（表3）

表3: Visual Markdown Editor で使えるショートカット

ショートカットキー	
地の文	Ctrl/Cmd + Alt + 0
イタリック体	Ctrl/Cmd + I
太字	Ctrl/Cmd + B
見出し	Ctrl/Cmd + Alt + 1 – Ctrl/Cmd + Alt + 6

3 別ファイルの埋め込みと実行

3.1 別ファイル（子ファイル）の作成

これから別途作成する Rmd ファイル 2 つを、それぞれ `child-English.Rmd`・`child-Spanish.Rmd` という名前で保存する

- この Rmd ファイルを、先ほど作った Rmd ファイルの中で参照し、最終的な docx ファイルにプログラミングの結果を出力させる
 - こうした親ファイルの中で参照する予定のファイルを、子ファイルと呼ぶ
- 子ファイルの保存先は、**（親ファイルのある）作業ディレクトリの中にする**
 - （親ファイルのある）作業ディレクトリの中であれば、その作業ディレクトリの下位ディレクトリの中でも良い

3.1.1 child-English.Rmd

```
(@example-English) English has numerous varieties all over the world and some linguists call them
↳ "World Englishes".
```

このように、`(@example-English)` には英語の例を載せました。
 地の文の行頭は、``(@...)`` で始めない方がよいと思います。
 なぜなら、``(@...)`` で始まる行は、常に「例文」として認識されてしまうためです。

3.1.2 child-Spanish.Rmd

```
(@example-Spanish) Cuando visiten España y países latinoamericanos,
podrán notar su diversidad lingüística.
```

なお、`(@example-Spanish)` の例のように、Rmd では、
 アクセント（`á・í` 等）・ティルデ（`ñ` 等）・ウムラウトのある文字（`ü` 等）を特殊な制御文字なしでそのまま入出力できます。
 また、``(@...)`` で始まる文を Rmd 内で改行する際は、改行後の文の冒頭に 4 つ半角スペースを入れます。

保存出来たら、親ファイルの中に R チャンクを書く

R チャンク

- R が実行できるコードブロック
 - バッククォーテーション 3 つ “” で囲む
- そのチャンクのオプション `{r ...}` として、子ファイル呼び出すコードを書く
- 例 1: `sub` ディレクトリにある `child-English.Rmd` を呼び出す場合

- このコマンド内では, `child="..."` で ... 内の Rmd ファイルを読み込んでいる

```
```{r English-example, child="sub/child-English.Rmd", eval=TRUE, include=TRUE}
```
```

- 例2: sub ディレクトリにある `child-English.Rmd` を呼び出していたが, やはり成果物に入れたくなくなった場合
 - このコマンド内では, `eval=FALSE, include=FALSE` で, 子 Rmd ファイルを無視している

```
```{r English-example2, child="sub/child-English.Rmd", eval=FALSE, include=FALSE}
```
```

- チャンク名を指定すると良い
 - 上記の例では, `{r ... , ***}` の ... に `English-example` や `English-example2`
 - 1つのチャンクにつき1つの名前

R チャンクのチャンクオプション

- 例: `include`
 - これにより, R チャンクの実行結果を, 最終的なファイルに掲載するかどうかを決められる
- 今回の場合, 子ファイルに記載されたコードの実行結果を docx ファイルに載せるかどうかを決める
 - そのため, `include=FALSE`

その他のチャンクオプションは, knitr や rmarkdown の開発者である謝益輝 (Yihui Xie) のウェブページ [Options - Chunk options and package options](#) を参照

チャンクオプションの設定

RStudio で Rmd ファイルを編集すれば, R チャンク右上の ⚙ ボタンを押した際に, 図4のような R チャンクのオプションを設定できる画面が現れる

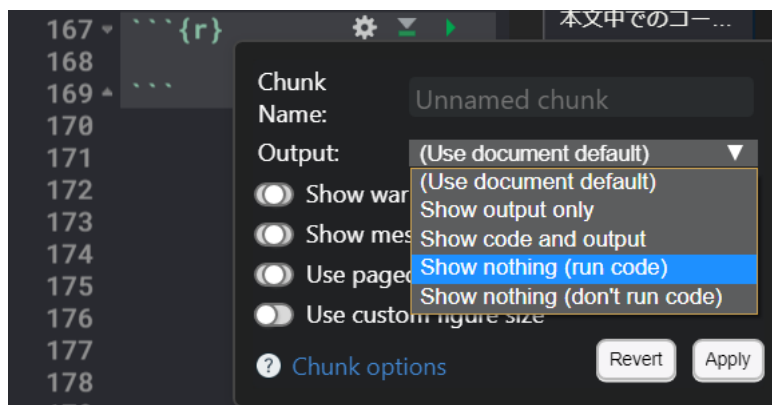


図4: RStudio で R チャンクのオプションを開く—*Show nothing* を選ぶと `include=FALSE` になる

R チャンクを RStudio の自動入力機能により挿入する

親ファイルの任意の行で, `Ctrl/Cmd + Alt + I(insert)` を押す

4 本文中でのコード実行

4.1 本文中でのコード実行

あるコードを Rmd ファイルの中で実行させる

1. R チャンクを Rmd ファイルの中を書く
2. さらに R チャンクの中にコードを書く
 - 例: `analysis-languageR.Rmd`

4.2 地の文に小さな R チャンク (inline code) を埋め込む

本文中で何らかの数値に言及する場合

- 例：コーパス調査で、能動文が 162 文、受動文が 23 件、自動詞文が 238 件見つかり、その総数を報告する場合
- 手計算の結果をコピーすると、打ち間違いが生じる恐れがある
 - 手計算だと、計算ミスが起きるかもしれない
 - 場合によっては、手計算さえできないような複雑な統計が必要かもしれない
- この結果を基にまた別の数値を報告する場合には、この数値自体が間違っているとミスが累積していく
 - 例：それぞれの文タイプの割合値を報告する場合

R が正確に値を計算できるのだから、それを本文にそのまま生かす

```
コーパス調査により、能動文が 162 文、受動文が 23 件、自動詞文が 238 件見つかった。
```

```
それぞれの割合は、順に`r round(162/(162+23+238)*100, 2)`%,
```

```
`r round(23/(162+23+238)*100, 2)`%,
```

```
`r round(238/(162+23+238)*100, 2)`% であった。
```

出力結果

コーパス調査により、能動文が 162 文、受動文が 23 件、自動詞文が 238 件見つかった。それぞれの割合は、順に 38.3%, 5.44%, 56.26% であった。

5 図表番号等の自動参照

5.1 図表番号等の自動参照

- 図表を本文中で参照する際は、`\@ref(...:****)` という記法を使う
- ... の箇所には、図であれば `fig`、表であれば `tab` というキーを入れる
- **** の箇所には、タグを書く
 - チャンクで図や表を作成した場合：
図や表を作成したチャンクのチャンク名（これがタグになる）
- 図表番号は、最終的に生成されるファイルでは常に自動的に付与される
 - 図や表の順番を入れ替えたり、図表を削除したりした場合には、knit の度に図表番号が自動的に更新される

6 文献の自動挿入と整形

6.1 文献の自動挿入

1. 文献の引用には、RMarkdown で利用可能な形式のファイルに文献情報を記す
 - [Bibliographies and Citations](#) を参照
 - https://rmarkdown.rstudio.com/authoring_bibliographies_and_citations.html
2. 文献ファイルを初めて作る場合には、テキストエディタだけで作れる `bib` ファイルがよいかも

6.1.1 bib ファイルの作り方

- 著者の姓・名ともに、イニシャルだけでなく全て入力するとよい
 - 参考文献欄で名前をイニシャルのみ表示するかどうかは、スタイル [第6.2節参照] が自動で判断
- 著者の姓名の書き方
 - 姓と名の間に半角スペースを置く
 - 姓と名の並べ方は次の通り
 1. 名 姓
 2. 姓, 名

- 複数著者の場合、著者名同士の区切りは `and`
- 基本的に、キーの中身は `{ }`（波括弧；curly brackets）で囲む
 - 但し、年号など数字は、括弧をつけてもつけなくてもよい
 - どのキーにも括弧をつけていた方が安心ではある
- 本 (`@book`) 以外の論文 (`@article`) やマニュアル (`@manual`)、予稿集 (`@inproceedings`) などを引用する際の文献情報の記入方法は、Patashnik (1988) や Wikipedia の該当ページを参照
 - 1 つの文献につき、`author`・`title` などのキーの種類が多く登録されている分には構わない
 - 例えば、`@book` では、`volume` キーは必須ではないが、書いてあっても構わない
 - どの種類の文献に対してどのキーを表示させるかは、スタイルが自動で判断する
 - 下記の `bib` ファイルの例では、次の種類の文献を入れた
- `bib` ファイルの名前は任意
- 文献情報に日本語表記が含まれる文献（マルチバイト文字で文献情報を記した文献；所謂「和文文献」）と、文献情報に日本語表記が含まれない文献（半角英数字だけで文献情報を記した文献；所謂「欧文文献」）は別の `bib` ファイルに入れた方がいいかもしれない
 - 前者は `cjk.bib` に記入し、後者は `non-cjk.bib` に記入する等

```
@book{PL-BDA2021,
  author      = {Bruno Nicenboim and Daniel Schad and Shravan Vasishth},
  title       = {An {I}ntroduction to {B}ayesian {D}ata {A}nalysis for {C}ognitive {S}cience},
  publisher    = {CRC Press},
  year        = {2021},
  url         = {https://vasishth.github.io/bayescogsci/book/},
  abstract     = {This book shows how to conduct Bayesian statistics with real data from
    ↳ psycholinguistic experiments. }
}
```

参照キー

- この `myref.bib` では、`PL-BDA2021` が *An Introduction to Bayesian Data Analysis for Cognitive Science* という本を論文中で参照する際のキーとなる
- 論文中で参考文献に言及するときには、このキーを使い、文献を `@` メンション
- 文中で著者に言及し出版年を括弧内に収める時は `@...`、著者名も出版年も括弧内に入れるときには `[@...]` と書く

`@PL-BDA2021` は、心理言語学で一般的な実験データを例に、ベイズ統計を用いた分析を行う方法を紐解いている。その本は、

```
↳ "This book is intended to be a relatively gentle introduction to carrying out Bayesian data
↳ analysis and cognitive modeling" [@PL-BDA2021]という一節から始まる。
```

出力結果

Nicenboim et al. (2021) は、心理言語学で一般的な実験データを例に、ベイズ統計を用いた分析を行う方法を紐解いている。その本は、“This book is intended to be a relatively gentle introduction to carrying out Bayesian data analysis and cognitive modeling” (Nicenboim et al. 2021) という一節から始まる。

文書末尾の参考文献

- こうして言及した参考文献に関しては、その文献情報が、最終的に出力されるファイルの末尾に自動で記載される
- その際、親ファイル冒頭の YAML セクションで `bibliography` キーを設定し、読み込む `bib` ファイルを指定する
 - 読み込む `bib` ファイルは複数でも可能

```
---
bibliography:
  - myref.bib
  - myref2.bib
---
```

6.2 文献の自動整形

YAML セクション（第7節参照）で、文献のスタイルを指定する。

<https://bookdown.org/yihui/rmarkdown/pdf-document.html#latex-packages-for-citations>

6.2.1 natbib を使う場合

1. output キー直下の bookdown::pdf_document2 の配下に citation_package: natbib を足す
2. bibliography キーに文献情報を記したファイルのファイル名（とそのパス）を書く
3. 文献のスタイルが決まっていれば、biblio-style にスタイルを書く
 - 所謂「人文系」なら、biblio-style: apalike を指定すると、APA 方式の「著者（年）」表示ができる
 - 但し、欧文文献のみ綺麗に出力される
 - 和文文献の処理は L^AT_EX の知識を要する…
4. natbib に追加のオプションを入れたい場合は、natbiboptions にオプションを書く
 - 発行年などの括弧を丸括弧にするには、natbiboptions: round

```
---
output:
  bookdown::pdf_document2:
    citation_package: natbib
bibliography: myref.bib
biblio-style: apalike
natbiboptions: round
---
```

6.2.2 biblatex を使う場合

1. output キー直下の bookdown::pdf_document2 の配下に citation_package: biblatex を足す
2. bibliography キーに文献情報を記したファイルのファイル名（とそのパス）を書く
3. 文献のスタイルが決まっていれば、biblio-style にスタイルを書く
 - 所謂「人文系」なら、biblio-style: apa6 を指定すると、APA 方式の「著者（年）」表示ができる
 - 但し、欧文文献のみ綺麗に出力される
 - 和文文献の処理は L^AT_EX の知識を要する…
4. biblatex に追加のオプションを入れたい場合は、biblatexoptions にオプションを書く
 - 発行年などの括弧を丸括弧にするには、natbiboptions: round

```
---
output:
  bookdown::pdf_document2:
    citation_package: biblatex
bibliography: myref.bib
biblatexoptions:
  - sortcites = true
  - sorting = nyt
  - backend = biber
---
```


6.2.3 和文文献をどうするか…はあ…

和文文献を処理するには、 \LaTeX の知識を要する。 RMarkdown 単体では完結しない問題であり、本稿執筆者自身がいまだに悩んでいる。本来であれば、簡単であっても説明を施せばよいのだが、執筆時点においては、それもままならない。そのため、ここに解決のカギとなる \LaTeX パッケージだけでも紹介する。

6.2.3.1 natbib ベースで文献を処理する jecon-bst <https://github.com/ShiroTakeda/jecon-bst>

6.2.3.2 biblatex ベースで文献を処理する biblatex-japanese <https://github.com/kmaed/biblatex-japanese>

7 YAML セクションで論文の体裁を指定する

7.1 文書のメタ情報

- 文書タイトル・サブタイトル
- 著者名
- 作成日時（更新日時）
- フォント
 - フォントサイズ
 - フォントファミリー・シリーズ
- 版型
 - サイズ
 - 余白
- 出力形式

7.2 ファイル冒頭の YAML でメタ情報を指定

YAML (YAML Ain't Markup Language)

- (R)Markdown ファイル等の冒頭に記される、データ文字列
- --- でチャンク（囲み）をつくり、そこに著者情報 `author`、文書名 `title` などの文書メタ情報を書く
- `output` の部分を変えることにより、`docx` ファイルだけでなく、`pptx` ファイルや `HTML` ファイル、(L^AT_EX 経由の) `PDF` ファイルを作成できる
- ここでは、まず、`date`・`author`・`title`・`output` の4つをおさえる
- 文書生成にあたり、キーの順番は無関係で、こうしたキーが全て揃う必要もない
 - 日付や題名がなくても十分文書生成は可能
 - * 但し、`output` の部分は指定が必要
 - * `output` 未指定では（デフォルトでは）`HTML` ファイルが生成
- `output` 以外のキーを指定する際は、" "（ダブルクォーテーションマーク）で中身を囲う

簡単な YAML の例

```
---
date: "2021-03-01"
author: "Your Name"
title: "officedown template"
output:
  officedown::rdocx_document:
    mapstyles:
      Normal: ['First Paragraph']
---
```

officedown で pptx ファイルを作るときの YAML 例

```
---
date: "2021-03-01"
author: "Your Name"
title: "Untitled"
output:
  officedown::rpptx_document
---
```

7.3 さらに色々なメタ情報を入れるには

論文を生成する際には、参考文献を入れる必要がある

Zotero・Mendeley・bibtexなどで作成した文献ファイルを記述すれば、その文献ファイルが参照され文書生成が実行される

```
---
bibliography: [bibs/your-bib-file.bib]
---
```

著者が複数いる場合には、下記例のように、**author:** まですで一旦改行し、（半角スペース 2 つ）と - と（半角スペース 1 つ）を置いてから著者名を書いていくと、複数人を併記可能

```
---
author:
  - 小川 雅貴
  - Carlos Luis Rafael Rivera
---
```

さらに、メタ情報に改行を含む場合は、下記の通り | を使う

```
---
title: |
  | 変なところに改行を入れて、なが~~~~~いタイ
トルを、つけてみました
```

出力形式ごとに設定できる YAML キーは異なる

出力形式ごとに設定できる YAML キーを網羅した一覧は下記から：

<https://cran.r-project.org/web/packages/ymlthis/vignettes/yaml-fieldguide.html>

7.3.1 本説明書の YAML 設定

```
title: |
  | 題名
  | 題名 2 行目
subtitle: " 副題"
author: ""
date: " 日付" #" 最終更新: 2021/3/ 1)"
abstract: |
  ここに要旨を書けます。

output:
  #bookdown::html_document2:
  #  pandoc_args:
  #    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua
  #    # - --lua-filter=configuration/box.lua
  #bookdown::word_document2:
  #  #reference_docx: configuration/docx/GengoKenkyu_template-Sep2020-JA.docx
  #  pandoc_args:
  #    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua
  #    # - --lua-filter=configuration/box.lua
```

```

bookdown::pdf_document2:
  latex_engine: xelatex
  keep_tex: TRUE
  toc: TRUE
  highlight: "espresso"
  includes:
    in_header:
      - configuration/latex/fig-tab-box.tex
      - configuration/latex/character-config.tex
      - configuration/latex/linguistic-sets.tex
      - configuration/latex/link-highlight.tex
      - configuration/latex/biblatex-japanese-config.tex
  pandoc_args:
    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua
  #citation_package: natbib
  citation_package: biblatex

###-----
###
### 参考文献に関する設定項目
###
###-----

# 文献情報が記されたファイルの指定
bibliography:
  - doc-creation/bibs/English.bib
  - doc-creation/bibs/Japanese.bib

# natbib のオプション
# biblatexoptions:

# biblatex のオプション
#biblatexoptions:
# - sortcites = true
# - sorting = nyt
# - backend = biber

# 参考文献のスタイルの設定 (LaTeX 経由で PDF を生成する場合)
#biblio-style: doc-creation/bibs/jecon-bst/jecon
#biblio-style: biblatex-japanese

# 参考文献の章・節の名前
biblio_title: " 参考文献"

###-----
###
### pandoc-ling の設定
###
###-----

# pandoc-ling で変換した例文を LaTeX でどのように出力するかの指定
latexPackage: gb4e

```

```

# pandoc-ling で変換した例文の番号をどのように出力するか指定
addChapterNumber: FALSE
restartAtChapter: FALSE

###-----
###
### LaTeX で PDF 論文・レポートを出力する際の設定項目
### 「必須」と特記がない場合は、原則任意設定
###
###-----

# 日本語で組版するなら
documentclass: bxjsarticle
classoption:
  ##-----
  ## pandoc オプションは必須
  ##-----
- pandoc
  ##-----
  ## 以降は任意
  ##-----
  # XeLaTeX を LaTeX エンジンにしている場合のフォント設定
  # 詳しくは、下記「zxjafont パッケージ」の第 2.1 節内にある「■多ウェイト用プリセット」を参照
  # http://mirrors.ctan.org/language/japanese/zxjafont/zxjafont.pdf#page=2
- jafont = haranoaji      # bxjsarticle のデフォルトのフォント
  #- jafont = yu-win10     # Windows 10 ユーザが游書体を使う場合
  #- jafont = hiragino-pron # ヒラギノ, Mac ユーザを中心にヒラギノを使いたい方はこちらを使う
  # 用紙サイズの指定
  # A4 以外のものを指定したい場合には、
  # 下記「zxjafont パッケージ」の第 2 節内にある「■用紙サイズ」を参照
  # http://mirrors.ctan.org/language/japanese/BX/bxjscsls/bxjscsls.pdf#page=11
- a4paper
  # 段落冒頭に括弧類が来た時の空きの補正方法
- everyparhook = compat
  # 長さの単位として、pTeX の和文用の単位を使いたい場合
- |
  ````{=latex}
 japaram = {units = true}
  ````

# 欧文組版の中に日本語を混ぜるなら
# documentclass: bxjsarticle とその classoption をコメントアウトし、
# その上で下記のコメントアウトを外す
#documentclass: article
#lang: jp # 将来 jp は ja に修正される 詳細は https://github.com/jgm/pandoc/pull/7050

###-----
###
### style ファイルなどが無い場合など自分で指定したい項目がある場合には、
### 以下を指定する
###
### LaTeX style ファイル (拡張子.sty) などの
### 学会で配布される documentclass およびその classoption がある場合には

```

```

### これより先の「LaTeXでPDF論文・レポートを出力する際の設定項目」は
### 設定しないことを勧める
### (学会の documentclass や classoption に指定された設定にならない恐れがある)
### (学会の documentclass や classoption を記したファイル本体を上書きするわけではない)
###
###-----

# PDFの余白の設定
# オプションの詳細は、下記の geometry パッケージ取り扱い説明書を参照
# http://mirrors.ctan.org/macros/latex/contrib/geometry/geometry.pdf#page=3
# footskip を除き、日本言語学会の予稿集の余白に準拠
# なお、2020年9月現在、日本言語学会学会誌『言語研究』の場合、
# footskip を除く下記全てを、25truemm に設定すればよい
geometry:
- left      = 20truemm
- right     = 20truemm
- top       = 20truemm
- bottom    = 30truemm
# 下部余白の上を基準点に、そこからどの位置にページ番号を載せるか
- footskip = 15truemm

# 段落間の空行を消し、段落冒頭を字下げする場合
indent: TRUE

# 論文本文の欧文フォントの設定
# ローマン体
mainfont: TeX Gyre Termes
mainfontoptions:
- Scale = 1

# 論文章節タイトルの欧文フォントの設定
# サンセリフ体
sansfont: TeX Gyre Heros
sansfontoptions:
- Scale = 1

# コードブロック用の欧文フォントの設定
# タイプライタ体
monofont: zcoN
monofontoptions:
- Scale = MatchLowercase
---

```

8 例文

プログラミングで言語学に関する文書を生成する大きな理由のひとつに、例文出力の容易さが挙げられる。例文番号は、文書生成 (knit) 時に自動で振られる。例文を本文中で参照する（例：「(1) は日本語の例であるが…」）際も、参照する例文と同じ番号が自動で出力される。さらに、例文の追加・削除・順序変更に応じ、毎 knit 時に例文番号が自動で変更される。

(R)Markdown で例文を表示させるには、第 8.1 節の「pandoc の `example_lists` を使う」方法と第 8.2 節の「pandoc-ling を使う」方法のどちらかを使えばよい。

同一の成果ファイルを出力する際は、両者を混ざって使うことはできない。なぜなら、`example_lists` の例文番号と `pandoc-ling` の例文番号が重複してしまうためである。例えば、この資料では、(1) と (1) は異なる例文を指しているが、番号が同一になってしまう。この問題の原因は、(1) を `example_lists` で、(1) を `pandoc-ling` で作成したことである。たとえ、一方のファイルでは `example_lists` だけで例文を書き、他方のファイルでは `pandoc-ling` だけで例文を書くというように、複数ファイルで 2 つの記法を使い分けていたとしても、それら複数ファイルから最終的に 1 つの文書を生成する場合には、例文番号が重複する。例えば、子ファイル 1 には `example_lists` だけで例文を記入し、子ファイル 2 には `pandoc-ling` だけで例文を記入し、これら子ファイル 1・2 を親ファイルで参照して 1 つの文書を生成をしようとする、例文番号が重複する。ただし、子ファイル 1 からは成果ファイル 1 を、子ファイル 2 からは成果ファイル 2 を、というように異なる成果ファイルを出力する場合は問題ない。

両者の比較は、表 4 の通りである。用途に合わせてどちらかの記法を選ぶと良い。

表4: `example_lists` と `pandoc-ling` の比較

| | <code>example_lists</code> | <code>pandoc-ling</code> |
|------------------------|----------------------------|--------------------------|
| 初期設定 | 不要 | 要 |
| 記法の難易度 | 極めて簡単 | 簡単 |
| 単一の例文 | 可能（極めて簡単） | 可能 |
| (1a) や (1b) のような入れ子の例文 | 不可能 | 可能 |
| グロス | 不可能 | 可能 |

8.1 pandoc の `example_lists` を使う

pandoc の `example_lists` (Numbered example lists)^{*1}を使えば、グロスなし・単一の例文を作ることができる。

書き方は次の通りである。

1. () 内に @ を入力し、@ 以降に半角英数字で例文へのラベルを書く
2. (@...) の後ろに半角スペースを 1 つ置いてから、実際の例文を書く

これで、番号付きの例文を出力することができる。

本文の中で例文に言及するときは、本文中に (@...) をそのまま書くだけで例文を参照できる。

```
(@good-ja-list) これは文法的に容認できる文です。

(@bad-ja-list) \* 文これ容認的では文法できる。

(@good-en-list) This is a grammatical sentence.

(@bad-en-list) \* This is an grammatical sentence.

日本語において
(@good-ja-list) は文法的であり、
(@bad-ja-list) は非文である。
同様に、英語において
(@good-en-list) は文法的だが、
(@bad-en-list) は非文である。
```

^{*1} <https://pandoc.org/MANUAL.html#numbered-example-lists>

出力結果

- (1) これは文法的に容認できる文です。
- (2) * 文これ容認的では文法できる。
- (3) This is a grammatical sentence.
- (4) * This is an grammatical sentence.

日本語において (1) は文法的であり、(2) は非文である。同様に、英語において (3) は文法的だが、(4) は非文である。

但し、現在の `example_lists` では、(1) のようには、1 つの番号の配下に複数の例文を置くことはできない。このような形式で例文を提示したい場合には、`pandoc-ling` が使える。

- (1) a. 現在の `example_lists` では、このように、1 つの番号の配下に複数の例文を置くことはできない。
- b. しかし、このような形式で例文を提示したい場合には、`pandoc-ling` が使える

8.2 pandoc-ling を使う

言語類型論学者で、コンピュータによる言語学研究支援環境を開発している [Michael Cysouw](#) が `pandoc-ling` というプログラム (lua フィルタ) を公開している。このプログラムにより、Markdown 記法で多様な形式の例文を \LaTeX にも docx にも出力することができる。

8.2.1 使うための準備

1. まずプログラムをダウンロードして作業ディレクトリ内に展開

<https://github.com/cysouw/pandoc-ling/releases>

2. YAML セクションに、次のような記述を追加

なお、以下の `configuration/docx/GengoKenkyu_template-Sep2020-JA.docx` や `configuration/pandoc-ling/pandoc-ling.lua` などでのファイル名やパス (ファイルが入っている [下位] ディレクトリへの宛先) は、実際に使用したいファイルおよびそのファイルが格納されているパスに変える。本テンプレートより文書を出力する際は、下記の設定のままで良い。

```
output:
### 論文を LaTeX 経由で PDF 化するなら
bookdown::pdf_document2:
  latex_engine: xelatex
  pandoc_args:
    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

### スライドを LaTeX 経由で PDF 化するなら
### 上の bookdown::pdf_document2 配下の
### YAML セクションを # でコメントアウトすること
bookdown::beamer_presentation2:
  latex_engine: xelatex
  slide_level: 2
  pandoc_args:
    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

latexPackage: gb4e
```

PDF 出力


```

output:
  bookdown::word_document2:
    reference_docx: configuration/docx/GengoKenkyu_template-Sep2020-JA.docx
    pandoc_args:
      - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

### docx を officedown を使って出力するなら
### 上の bookdown::word_document2 配下の
### YAML セクションを # でコメントアウトすること
officedown::rdocx_document:
  base_format: bookdown::word_document2
  reference_docx: configuration/docx/GengoKenkyu_template-Sep2020-JA.docx
  pandoc_args:
    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

latexPackage: gb4e

```

docx 出力

```

output:
  bookdown::powerpoint_presentation2:
    reference_doc: relative/path/to/pptx-styles-reference.pptx
    pandoc_args:
      - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

### pptx を officedown を使って出力するなら
### 上の bookdown::pptx_document2 配下の
### YAML セクションを # でコメントアウトすること
officedown::rpptx_document:
  base_format: bookdown::powerpoint_presentation2
  reference_doc: relative/path/to/pptx-styles-reference.pptx
  pandoc_args:
    - --lua-filter=configuration/pandoc-ling/pandoc-ling.lua

latexPackage: gb4e

```

pptx 出力

8.2.2 pandoc-ling での基本的な例文の入力法

本節以降で紹介する pandoc-ling での例文の入力方法は、概ね次の通りである。

1. 例文を書きたい箇所に、下記のように、: を 3 つ以上使い、囲いを作る

```

:::
:::

```

- : の数は、3 つ以上で、囲いの初めと終わりの : の数が揃っていれば、何個でも良い

2. 囲いの初めの ::: の後ろに、半角スペースを 1 つ置いて {.ex} と入力する

```

::: {.ex}
:::

```

3. 例文を参照するタグをつけるには、先ほどの **ex** の後ろに半角スペースを 1 つ置いて、**#** とタグを入力する

```
::: {.ex #this-is-an-example}
:::
```

4. 囲み内に、例文を書くと全体が完成する

```
::: {.ex #this-is-an-example}
これは例文です。
:::
```

8.2.3 1 行に収まる例文

第 8.2.2 節に示した通りそのままでも、1 行に収まる例文を記入できる。

容認度を示す際には、例文冒頭に ^ と容認度判断の記号（*、? など）を書いて、さらに半角スペースを 1 つ置けばよい。容認度判断の記号の扱いについては、第 8.2.7 節で詳述する。

```
::: {.ex #good-ja}
これは文法的に容認できる文です。
:::

::: {.ex #bad-ja}
^* 文これ容認的では文法できる。
:::

::: {.ex #good-en}
This is a grammatical sentence.
:::

::: {.ex #bad-en}
^* This is an grammatical sentence.
:::
```

日本語において
@good-ja は文法的であり、
@bad-ja は非文である。
同様に、英語において
@good-en は文法的だが、
@bad-en は非文である。

出力結果

- (2) これは文法的に容認できる文です。
- (3) * 文これ容認的では文法できる。
- (4) This is a grammatical sentence.
- (5) * This is an grammatical sentence.

日本語において (2) は文法的であり、(3) は非文である。同様に、英語において (4) は文法的だが、(5) は非文である。

8.2.4 ライプツィヒ・グロス付与規則

ライプツィヒ・グロス付与規則 (The Leipzig Glossing Rules)^{*2}に則った例文を提示する際には、次のようにする。

1. 囲い ::: の最初の側 (::: {*.ex #...*}) にある例文のタグ (*#...*) の後ろに、半角スペースを 1 つ置いて、*formatGloss=true* というオプションを書く

```
::: {.ex #this-is-an-example formatGloss=true}
:::
```

2. 囲いの間 (例文を書く箇所) に 4 行分 | を記入する

```
::: {.ex #this-is-an-example formatGloss=true}
|
|
|
|
:::
```

3. 2 行目の | の後ろに半角スペースを 1 つ置いてから、単語あるいは形態素ごとに分けられた例文を書く

```
::: {.ex #this-is-an-example formatGloss=true}
|
| kore=wa nihongo=no reibun=desu.
|
|
:::
```

4. 3 行目の | の後ろに半角スペースを 1 つ置いてから、単語あるいは形態素ごとの訳 (グロス) を書く
 - 形態素に対するグロスを大文字で記入すると、出力結果では、グロスがスモールキャピタル (SMALL CAPITAL) で表示される

```
::: {.ex #this-is-an-example formatGloss=true}
|
| kore=wa nihongo=no reibun=desu.
| This=TOP Japanese=DAT example.sentence=COP
|
:::
```

5. 4 行目の | の後ろに半角スペースを 1 つ置いてから、全体の通し訳を書く

```
::: {.ex #this-is-an-example formatGloss=true}
|
| kore=wa nihongo=no reibun=desu.
| This=TOP Japanese=DAT example.sentence=COP
| This is a Japanese example sentence.
:::
```

出力結果は次の通りとなる。

出力結果

(6) *kore=wa nihongo=no reibun=desu.*
This=TOP Japanese=DAT example.sentence=COP

^{*2} PDF 版は <https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf> に、Web 版は <https://www.eva.mpg.de/lingua/resources/glossing-rules.php> にある。

‘This is a Japanese example sentence.’

文法性・容認度判断を示す場合には、2行目の | の後ろに半角スペースを1つ置き、^と文法性・容認度判断を示す記号を書き、さらにその後ろに例文を書けばよい。

```
::: {.ex #basic-LGR-judgement formatGloss=true}  
|  
| ^* kore=desu nihongo=wa reibun=no.  
| This=COP Japanese=TOP example.sentence=DAT  
| This is a Japanese example sentence.  
:::
```

出力結果

(7) * *kore=desu nihongo=wa reibun=no.*
This=COP Japanese=TOP example.sentence=DAT
‘This is a Japanese example sentence.’

上述の例では、囲いの中の1行目の | の後ろには何も記入しなかった。だが、この1行目の後ろに、言語名や当該言語の文字表記体系における例文を記入することが可能である。

1行目の | の後に、言語名を書く例は、以下の通りである。

```
::: {.ex #LGR-lang-name formatGloss=true}  
| Japanese (Japonic; Japanese-Ryukyuan)  
| kore=wa nihongo=no reibun=desu.  
| This=TOP Japanese=DAT example.sentence=COP  
| This is a Japanese example sentence.  
:::
```

出力結果

(8) Japanese (Japonic; Japanese-Ryukyuan)

kore=wa nihongo=no reibun=desu.
This=TOP Japanese=DAT example.sentence=COP

‘This is a Japanese example sentence.’

1行目の | の後に、当該言語の文字表記体系における例文を書く例は、以下の通りである。

```
::: {.ex #LGR-orig-ex formatGloss=true}  
| これは日本語の例文です。  
| kore=wa nihongo=no reibun=desu.  
| This=TOP Japanese=DAT example.sentence=COP  
| This is a Japanese example sentence.  
:::
```

出力結果

(9) これは日本語の例文です。

kore=wa nihongo=no reibun=desu.

This=TOP Japanese=DAT example.sentence=COP

‘This is a Japanese example sentence.’

言語名も当該言語の文字表記体系における例文も両方提示したい場合には、前書き (preamble) の記法を使う。前書きの記法は、次の通りである。

1. 最初の囲い ::: 直後に、前書きの内容を書く
2. その前書きと例文（ここでは、1 行目の |）の間に空行を設ける

```
::: {.ex #LGR-preamble-orig-ex formatGloss=true}
Japanese (Japonic; Japanese-Ryukyuan)

| これは日本語の例文です。
| kore=wa nihongo=no reibun=desu.
| This=TOP Japanese=DAT example.sentence=COP
| This is a Japanese example sentence.
:::
```

出力結果

(10) Japanese (Japonic; Japanese-Ryukyuan)[| これは日本語の例文です。

kore=wa nihongo=no reibun=desu.

This=TOP Japanese=DAT example.sentence=COP

‘This is a Japanese example sentence.’

なお、前書きの記法は、第 8.2.3 節で見たような、グロスを使わない場合でも利用できる。

```
::: {.ex #preamble}
(おいしそうに見えないお菓子を食べている相手に向かって)

それ、ほんとうにおいしい？
:::
```

出力結果

(11) (おいしそうに見えないお菓子を食べている相手に向かって)

それ、ほんとうにおいしい？

8.2.5 複数行にまたがる長い例文

長い例文を書く、あるいは複数の文を 1 つの例文として提示する場合には、(R)Markdown の編集時点では改行できていると、自分がどのような例文を提示しようとしているのか把握しやすい。

例えば、次の入力で (12) のような出力結果を得られる。この (12) のような長さの例文は、論文の中ではあり得るが、その入力の編集はあまりしやすくないかもしれない。

```
::: {.ex #unreadable}
```

これは長い例文です。複数文そして複数行に渡って、長い例文を提示します。入力時点でこの長さだと、結構長いと感じませんか。感じませんか…。感じる人もいますか。いずれにしても、1 文で終わる例文ではありませんから、それなりに長いということは、同意いただけるかもしれません。いかがでしょうか。

```
:::
```

出力結果

- (12) これは長い例文です。複数文そして複数行に渡って、長い例文を提示します。入力時点でこの長さだと、結構長いと感じませんか。感じませんか…。感じる人もいますか。いずれにしても、1 文で終わる例文ではありませんから、それなりに長いということは、同意いただけるかもしれません。いかがでしょうか。

そうした場合には、`formatGloss=false` というオプションを追記するとよい。`formatGloss=false` は、例文のタグの後ろに半角スペースを 1 つ置いてから記入する。

```
::: {.ex #readable formatGloss=false}
```

これは長い例文です。
複数文そして複数行に渡って、長い例文を提示します。
入力時点でこの長さだと、結構長いと感じませんか。
感じませんか…。
感じる人もいますか。
いずれにしても、1 文で終わる例文ではありませんから、
それなりに長いということは、同意いただけるかもしれません。
いかがでしょうか。

```
:::
```

出力結果

- (13) これは長い例文です。複数文そして複数行に渡って、長い例文を提示します。入力時点でこの長さだと、結構長いと感じませんか。感じませんか…。感じる人もいますか。いずれにしても、1 文で終わる例文ではありませんから、それなりに長いということは、同意いただけるかもしれません。いかがでしょうか。

(12) と (13) は全く同一の例文であり出力結果も全く同一だが、その入力とは異なる。(12) の入力には一切改行が入れられなかった (改行を入れると、グロスとして認識されてしまう) が、(13) の入力には改行を入れられる。入力の際に改行を入れることで、例文全体がいくつの文で構成されているか把握しやすくなるとともに、個々の文に誤りがいないか確認することも容易になるだろう。

8.2.6 複数の例文が入れ子になっている例文

複数の例文が入れ子になっている例文の入力方法は、次の通りである。

1. 複数の例文を書き、例文ごとに改行する。
2. 個々の例文の前に、アルファベット 1 文字と . と半角スペースを置く

[開発者本人による例文](#) を参考に、実際に入力する。

```
::: {.ex #basic-nested}
```

a. 1 文目の例文である。

b. 2 文目が続く。

a. ``pandoc-ling`` がよしなに附番するため、アルファベットは何でもよい。そのため、入力では ``a.`` にしているが、出力では `'c.'` になっている。

e. 例文間の空行は無視される。
冒頭にアルファベット 1 文字と `.` と半角スペースがない文字列は、その直前の例文にそのまま続く。
:::

出力結果

- (14) a. 1 文目の例文である。
b. 2 文目が続く。
c. `pandoc-ling` がよしなに附番するため、アルファベットは何でもよい。そのため、入力では a. にしているが、出力では 'c.' になっている。
d. 例文間の空行は無視される。冒頭にアルファベット 1 文字と `.` と半角スペースがない文字列は、その直前の例文にそのまま続く。

(15) のように入れ子になっているグロス付きの例文を提示するには、下記のように、1 行目の | の直上の行にアルファベット 1 文字と `.` を入力する。例文全体の前書き (*Japanese (Japonic; Japanese-Ryukyuan)* の箇所) や入れ子になっている例文の前書き (「これは日本語の例文です。」「これです日本語は例文の。」) を入れるかは任意である。

```
::: {.ex #nested-LGR-preamble formatGloss=true}
Japanese (Japonic; Japanese-Ryukyuan)

g.
| これは日本語の例文です。
| kore=wa nihongo=no reibun=desu.
| This=TOP Japanese=DAT example.sentence=COP
| This is a Japanese example sentence.

b.
| これです日本語は例文の。
| ^* kore=desu nihongo=wa reibun=no.
| This=COP Japanese=TOP example.sentence=DAT
| This is a Japanese example sentence.
:::
```

出力結果

- (15) Japanese (Japonic; Japanese-Ryukyuan)
a. これは日本語の例文です。
kore=wa nihongo=no reibun=desu.
This=TOP Japanese=DAT example.sentence=COP
‘This is a Japanese example sentence.’
b. * これです日本語は例文の。
kore=desu nihongo=wa reibun=no.
This=COP Japanese=TOP example.sentence=DAT
‘This is a Japanese example sentence.’

8.2.7 文法性・容認度判断の記号について

文法性・容認度判断の記号には、任意のものを使用できる。記号だけでなく、単語で文法性・容認度判断を表現することも出来る。下記を参考に適切なものを使えばよい。

```
::: {.ex #grammaticality-judgement}
```

容認度判断の記号とその意味

- a. ^* 例文が非文であることを表すには、`*`を使う。
- a. ^? 例文が非文に近いことを表すには、`?`を使う。
- a. ^?? 例文がより非文に近いことを表すには、`??`を使う。
- a. ^% 例文の文法性判断に方言差・個人差があることを表すには、`%`を使う。
- a. ^# 他には、`#`が使える。
- a. ^& さらに、`&`も使える。
- a. ^*? 異なる記号を組み合わせてもよい。
- a. ^^Really?^ 単語で任意の判断を示すことも可能である。
- a. ^^*What?*^ 判断記号や単語をイタリックにしたり、ゴシック体にしたりすることもできる。
- a. ^^AreYouSure?^ 但し、半角スペースを含めることは、今のところ出来ない。
- a. ^^ちょ、待てよ^ しかも、任意の判断記号を使う際は、例文全体の体裁（空きの大きさ）が崩れ得るので、注意を要する。
- a. ここまで来ると、何をどう判断しているのかさっぱり分からないが、それでも、これは文法的な例文だと、私は信じている。

```
:::
```

出力結果

(16) 容認度判断の記号とその意味

- a. * 例文が非文であることを表すには、* を使う。
- b. ? 例文が非文に近いことを表すには、? を使う。
- c. ?? 例文がより非文に近いことを表すには、?? を使う。
- d. % 例文の文法性判断に方言差・個人差があることを表すには、% を使う。
- e. # 他には、# が使える。
- f. & さらに、& も使える。
- g. *? 異なる記号を組み合わせてもよい。
- h. Really? 単語で任意の判断を示すことも可能である。
- i. What? 判断記号や単語をイタリックにしたり、ゴシック体にしたりすることもできる。
- j. AreYouSure? 但し、半角スペースを含めることは、今のところ出来ない。
- k. ちょ、待てよ しかも、任意の判断記号を使う際は、例文全体の体裁（空きの大きさ）が崩れ得るので、注意を要する。
- l. ここまで来ると、何を判断しているのかさっぱり分からないが、それでも、これは文法的な例文だと、私は信じている。

9 本資料について

9.1 本資料のクリエイティブ・コモンズ・ライセンス

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

表示 - 非営利 - 継承 4.0 国際 (CC BY-NC-SA 4.0)

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

「参考文献」欄の例

Nicenboim, Bruno, Daniel Schad, and Shravan Vasishth (2021). *An Introduction to Bayesian Data Analysis for Cognitive Science*. CRC Press. URL: <https://vasishth.github.io/bayescogsci/book/>.

Patashnik, Oren (1988). *BibTeXing*. URL: <https://ftp.kddilabs.jp/CTAN/biblio/bibtex/base/btxdoc.pdf>.