

# 外国語教育研究者のための オープンサイエンス入門

## R Markdownを用いた実践編

寺井 雅人（愛知工科大学）

作成:2024-05-20, 最終更新(JST): 2024-08-06 11:15:44

お願い



- ・ハンズオン形式なので、R Studioを開いてご準備ください！
- ・どのタイミングでも質問していただいてOKです！

## / 自己紹介

### // 寺井 雅人（Masato Terai）

- ・所属：愛知工科大学（助教）
- ・研究：第二言語習得の語彙、知識の身体性
- ・R(Studio)歴：2018年～現在に至る
- ・Gmail: [terai.research@gmail.com](mailto:terai.research@gmail.com)
- ・X（旧Twitter）：[@uniquefreshman](https://twitter.com/uniquefreshman)
- ・プログラミング記事：[Qiita](https://qiita.com/terai)

## / はじめに

### // 研究成果や研究資料の公開・共有の必要性

#### /// 再現可能な研究のために、研究に使用したマテリアルの提出を推奨

- ・e.g., 実験で使用した刺激文、分析に使用した生データ、分析コード
- ・[Language LearningのHP](#)より

Shared Research Materials and Data Policy for Accepted Articles. [Language Learning](#) encourages accepted authors to

upload their data collection materials and/or data to the IRIS database (<http://www.iris-database.org>). IRIS is an online repository for data collection materials used for second language research. This includes data elicitation instruments such as [interview and observation schedules](#), [language tests](#), [pictures](#), [questionnaires](#), [software scripts](#), [URL links](#), [word lists](#), [pedagogical interventions](#), and so on. (...) The sharing of research instrumentation benefits the research community and helps authors and journals increase the visibility of their published research.

#### /// 正直「めんどくさい」し「時間がかかる」

## / 救世主！R Markdown！

### // Rで分析を行う場合、R Markdownが一番便利

- ・RとWordが合体したようなもの
- ・様々な媒体に簡単に出力できる
  - ・HTML、Word、Power Pointなどなど
  - ・例(1:00のあたり)

## / WSの内容

1. R Markdownを使って分析結果をまとめることができる
  - ・Markdown記法の基礎
2. R Markdownでまとめた結果を共有することができる
  - ・HTML、Word、Power Point形式での出力
  - ・R Pubs、Githubなどでの公開
3. 様々なパソコンで同じ分析環境を再現することができる
  - ・Rパッケージのバージョン管理


おまけ.

## / 1. R Markdownを使って分析結果をまとめることができる

## // R Studioを開いて作成する

### /// R Markdownの解体新書

- 【パート1】YAML（YAML Ain't Markup Language）ヘッダー：文章全体の体裁や情報を操作する
  - タイトル、サブタイトル
  - 作成者
  - 作成した日時、更新日時も設定可能
  - どのような形式で作成するか



**注意**

- YAMLヘッダーは、RでもMarkdownでもないプログラム言語で記述します。

- 【パート2】コードチャンク：Rのコードを記述するところ
- 【パート3】ドキュメントチャンク：[Markdownと呼ばれるプログラム言語](#)で記述するところ
  - 見出し、表、箇条書き、強調、斜体など、Wordのリボン部分にある機能をMarkdownで書く

### /// Knit を押して出力！

- 初期設定はHTMLファイル出力です

## // プロジェクト

- プロジェクト＝ディレクトリ
  - ファイルや操作履歴を保存できる
- プロジェクトを作成する利点
  - 研究ごとに分析に必要なファイルをまとめることができる

## // 実際に作ってみましょう

- デスクトップに新しいフォルダーを作成してください
  - 名前は、英数字のみがいいです（Rが関係しそうな場合、ファイル名、フォルダ名に日本語を使わない方が安心です）
  - 作り方を解説しているサイト（[私たちのR](#)）
- デスクトップのフォルダーを指定してプロジェクトを作成しましょう

## // ドキュメントチャンク：Markdown記法

## /// 覚えるのはマストではない。その都度調べてよく使うものを覚えていく

- Markdownなら生成AIはほぼ完璧に正解を教えてくれる
- 必要最低限で覚えておくとい記法**
  - 見出し → これはマスト！
    - #の数で指定。文字との間を半角あけるのを忘れない。
  - 箇条書き
    - `*`, `+`, `-` のいずれかを入れる。文字との間を半角あけるのを忘れない。
    - 半角スペースを2つ前（もしくはtab）に入れると、レベル2を作れる。さらに2ついれると、`..`
  - 強調
    - `*`で挟むと斜体
    - `**`で挟むとBold体
    - `***`で挟むとどうなるでしょう

## // Let's 実践

- 以下の文章をMarkdownを使って再現してください。

### / 名古屋飯といえば

#### /// ひつまぶし：Hitsumabushi

おすすめは以下のお店です。

- ひつまぶし花岡
  - 場所：栄

## // 答え合わせ



```
# 名古屋飯といえば
## ひつまぶし：*Hitsumabushi*
おすすめは以下のお店です。

- **ひつまぶし花岡**
- 場所：栄
```

## // 実は、Wordのように編集できます！

- Markdownで書かなくとも、VisualモードであればWordと似たようにできます。
- 以下で設定ができる
  - [Tools] → [Global Options...] → [R Markdown] → [Visual]
  - “Use visual editor by default for new documents”の項目に☑
  - “Soft-wrap R Markdown files”にも☑を入れると、右側にアウトラインが出ます
- 欠点として、少し動作が遅い。簡単なものはMarkdownで書く方が速い
- 表などはVisualモードがおすすめ

// コードチャンクの挿入

/// ショートカットキーが便利 : [Ctrl] + [Alt] + [I] (Windows) 、  
[Command] + [Option] + [I] (Mac)

- このコードの中はRです。Rで使う関数などを自由に指定できます。
- 以下のチャンク内でないと、動きません。

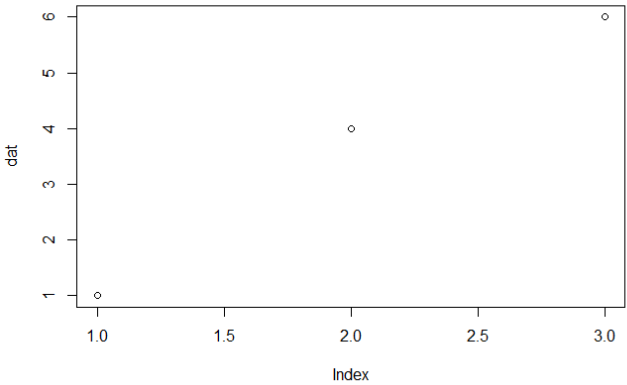
```
```{r}
```

```
dat <- c(1, 4, 6)

mean(dat)
```

```
## [1] 3.666667
```

```
plot(dat)
```



- {r}の中にもいろいろな指定ができます。
- コードを非表示にして結果だけを表示させたい（オープンサイエンスではないですが）

```
## [1] 3.666667
```

- チャンクのオプションは沢山あるので、その都度チートシートを参照するとよいです。
- [チートシート](#)

// Let's 実践

- 以下をドキュメントチャンクとコードチャンクを使って再現してください。

// 食費の合計

- 以下は、名古屋旅行で使った食費の合計である。
- 注! *hitsu*はひつまぶし、*miso*は味噌カツを表す。

```
hitsu <- 1300 * 2

miso <- 1000 * 2

total <- sum(hitsu, miso)
```

## // 答え合わせ



### ## 食費の合計

- 以下は、名古屋旅行で使った食費の合計である。
  - **\*\*注!\*\*** \*hitsu\*はひつまぶし、\*miso\*は味噌カツを表す

```
```{r}
hitsu <- 1300 * 2

miso <- 1000 * 2

total <- sum(hitsu, miso)
```
```

## / 2. R Markdownでまとめた結果を共有することができる

### // Let's 実践（一緒に）

- YAMLヘッダーのoutputを変更するだけ！

### /// Wordに出力

#### /// Before



```
title: "Untitled"
output: html_document
date: "2024-04-15"
```

#### /// After



```
title: "Untitled"
output: word_document
date: "2024-04-15"
```

- テンプレートを追加することも可能。テンプレートはきちんとレベル分けの設定などを行っておく必要あり（[設定の仕方](#)）。

### /// Power Pointに出力

- 図や表は必ず新しいページに表示されるなど、knitした後の修正が面倒。



```
title: "Untitled"
output: powerpoint_presentation
date: "2024-04-15"
```

### /// HTML slideに変更



```
title: "Untitled"
output: ioslides_presentation
date: "2024-04-15"
```

- #で定義するレベル分けでスライドの区切りが代わる
  - #でスライドのセクション見出し
  - ##で新しいページ
  - ###ページ内の太文字
- 枠にとらわれない！
  - 以下のチャンクを先頭に入れると、スクロール可能なスライドになる（チャンク内はrではなく、`=html`にする）。
  - 注意点として、タッチパッドや、マウスホイールでスクロールできますが、スクロールバーを掴んでスクロールはできません。



```
<style>
  slides > slide {
    overflow-x: auto !important;
    overflow-y: auto !important;
  }
</style>
```

### // HTML形式だと、簡単にウェブサイトができます

- リンクで他者に共有できます。
- 以下の二つは無料で利用できるが、無料版の場合、リンクを知る人だけが閲覧できるので注意

### /// RPubを使う

- コメント機能もあるので、「発表へのコメントは匿名でこちらへ」みたいのできそう。
- [使い方の解説](#)

/// Githubを使う

- Githubとは、コードのバージョン管理をするツール。共同編集可能。
- 今回は一番簡単な方法で実行します。しかし、Git(hub)をフルで使いこなせば、ファイルのバージョン管理、共同編集などが可能でより再現可能な資料作成に一步近づくと思います。R Studioとの連携も可能。
- [使い方の解説](#)

/ 3. 様々なパソコンで同じ分析環境を再現することができる

// パッケージのバージョン管理：renvパッケージ

- reproducible environments（再現性のある分析環境）の略
- RのプロジェクトごとにRの環境を作る。同じパソコンで、同じ名前でバージョンが異なるパッケージを使うことができる。
- 旧バージョンを試しにに使ってみたい場合が、今のバージョンも記録しておきたいときに！
- プロジェクトディレクトリを相手に共有できる！別のパソコンで分析したいときにも！
- Let's インストール

```
install.packages("renv")
```

- 使用する3つの関数
  1. `init` 関数：パッケージ管理の開始を宣言
  2. `snapshot` 関数：パッケージ情報の保存
  3. `restore` 関数：パッケージ情報の復元

// バージョン管理の開始

- 以下のコマンドを走らせると、プロジェクト内で使用しているパッケージを、RやRmdファイルなどから検出します。そして、そのバージョンの情報を `renv.lock` ファイルに保存します。

```
renv::init()
```

- 最初に実行すると、以下のメッセージが表示されます。

renv: Project Environments for R

Welcome to renv! It looks like this is your first time using renv. This is a one-time message, briefly describing some of renv's functions.

renv will write to files within the active project folder, including:

- A folder 'renv' in the project directory, and
- A lockfile called 'renv.lock' in the project directory.

In particular, projects using renv will normally use a private, per-project R library, in which new packages will be installed. This project library is isolated from other R libraries on your system.

In addition, renv will update files within your project directory, including:

- .gitignore
- .Rbuildignore
- .Rprofile

Finally, renv maintains a local cache of data on the filesystem, located at:

- "C:/Users/terai-masato/AppData/Local/R/cache/R/renv"

This path can be customized: please see the documentation in `?renv::paths`.

Please read the introduction vignette with `vignette("renv")` for more information. You can browse the package documentation online at <https://rstudio.github.io/renv/>. Do you want to proceed? [y/N]:

- yを押して進むと、開いているプロジェクトのあるディレクトリに **renv** というフォルダーが作成されます。その中に、3つのファイルと、1つのディレクトリが作成されています。また、 **renv.lock** というファイルも同じディレクトリに作成されます。新しく作成されたものは、すべて **renv** パッケージの利便に必要なので、削除しないでください。

```
- "C:/Users/terai-masato/AppData/Local/R/cache/R/renv" has been created
- Linking packages into the project library ... [33/33] Done!
- Resolving missing dependencies ...
# Installing packages -----
The following package(s) will be updated in the lockfile:

# CRAN -----
- base64enc      [* -> 0.1-3]
```

```
- bslib          [* -> 0.5.1]
- cachem         [* -> 1.0.8]
- cli            [* -> 3.6.1]
- digest         [* -> 0.6.33]
- ellipsis       [* -> 0.3.2]
- evaluate       [* -> 0.23]
- fastmap        [* -> 1.1.1]
- fontawesome    [* -> 0.5.2]
- fs             [* -> 1.6.3]
- glue           [* -> 1.6.2]
- highr          [* -> 0.10]
- htmltools      [* -> 0.5.7]
- jquerylib      [* -> 0.1.4]
- jsonlite       [* -> 1.8.7]
- knitr          [* -> 1.45]
- lifecycle      [* -> 1.0.4]
- magrittr       [* -> 2.0.3]
- memoise        [* -> 2.0.1]
- mime           [* -> 0.12]
- prettydoc      [* -> 0.4.1]
- R6             [* -> 2.5.1]
- rappdirs       [* -> 0.3.3]
- renv           [* -> 1.0.7]
- rlang          [* -> 1.1.2]
- rmarkdown      [* -> 2.25]
- sass           [* -> 0.4.7]
- stringi        [* -> 1.7.12]
- stringr        [* -> 1.5.0]
- tictoc         [* -> 1.2]
- tinytex        [* -> 0.48]
- vctr           [* -> 0.6.4]
- xfun           [* -> 0.41]
- yaml           [* -> 2.3.7]
```

The version of R recorded in the lockfile will be updated:

```
- R             [* -> 4.3.2]
```

- Lockfile written to "~/LET/LET\_Workshop\_2024/materials\_LETworkshop\_2024"

Restarting R session...

```
- Project '~/LET/LET_Workshop_2024/materials_LETworkshop_2024' loaded.
```

### /// パッケージ情報の保存

- パッケージの追加・更新・削除を行ったら、**snapshot** 関数を使って **renv.lock** ファイルを更新します。これを忘れると、バージョン情報が変更されないので、注意。
- 試しに、新しいパッケージをインストールしましょう。今回は、Rに関する様々な名言を含んでいる **fortunes** パッケージをインストールしましょう。興味のある方はこちらにリストがあります(R Fortunes: Collected Wisdom)。

```
install.packages("fortunes")
```

```
packageVersion("fortunes")
```

```
## [1] '1.5.4'
```

```
fortunes::fortune(which = 78)
```

```
##
## R is the lingua franca of statistical research. Work in all other languages
## should be discouraged.
##   -- Jan de Leeuw (as quoted by Matt Pocernich on R-help)
##       JSM 2003, San Francisco (August 2003)
```

```
fortunes::fortune(which = 386)
```

```
##
## If we put in a function into rstan that dropped chains, people would
##   -- Ben Goodrich (about (not) discarding selected chains from a stan
##       object)
##       Stan-users (December 2016)
```

- 今回の追加をrenv.lockファイルに追加しましょう。

```
renv::snapshot()
```

The following package(s) will be updated in the lockfile:

```
# CRAN -----
```

```
- fortunes    [* -> 1.5-4]
```

```
Do you want to proceed? [Y/n]:
```

- ファイルが更新される。

```
- Lockfile written to "~/LET/LET_Workshop_2024/materials_LETworkshop_26
```

- renvフォルダーの中の、library > R-あなたのバージョン > あなたのパソコン Platformを開くと、パッケージの名前と同じフォルダ (**fortunes**) が追加されています (五十音順になっているので見つけやすいです)。

### /// パッケージ情報の復元

- 記録しておくことで、1) パッケージを前のバージョンに戻したり、2) 他のパソコンにプロジェクトの分析環境を整えたりすることができる。
- パッケージの更新では、お馴染みの `install.packages` 関数や、`update.packages` 関数、`remove.packages` 関数を使うが、`renv` 関数で管理しているプロジェクト内では、これらの関数は `renv` 関数内のパッケージを呼び出している。特に、バージョンを指定したパッケージのインストールがしやすくなっている。
- 以下のように、**パッケージ名@バージョン名**で指定
  - 今回は、`fortunes` パッケージの古いバージョンをインストールする。
- 最初に、`remove.packages` で `fortunes` パッケージを削除。
- 注！ここで `renv::snapshot()` はやらない。やってしまうと、`renv.lock` から削除され、戻らなくなってしまう。

```
 install.packages("fortunes@1.4-0")
```

```
# Installing packages -----
- Installing fortunes ...      OK [linked from cache]
Successfully installed 1 package in 22 milliseconds.
```

```
The following loaded package(s) have been updated:
- fortunes
Restart your R session to use the new versions.
```

- restart (quit session) をして再度開く。
- **renv.lock** ファイルは確認すると、1.5-4のまま！

- しかし、パッケージのバージョンは1.4.0!


```
 packageVersion("fortunes")
```

- **バージョン1.4.0は2010年9月9日にリリースされている**
- 2016年に追加された386番目の名言はこのバージョンでは追加されていないので表示されない。

```
 fortunes::fortune(which = 78)
```

```
 fortunes::fortune(which = 386)
```

- 元に戻す場合は、`renv::restore()`

```
 renv::restore()
```

```
The following package(s) will be updated:
```

```
# CRAN -----
- fortunes    [1.4-0 -> 1.5-4]
```

```
Do you want to proceed? [Y/n]:
```

#### 注意



• パッケージによっては、手動でのダウンロードが必要な場合があります。その場合、エラーメッセージでどのパッケージのインストールができないのか表示されます。エラーメッセージに表示されているパッケージを見て、手動でそれをインストールします。

## // R自体のバージョン管理も可能です

### /// Windows

- [Tools] → [Global Options] → [R version]をクリックすると、過去にそのPCで使用していたRのバージョンが記録されている。

### /// Mac

- Windowsのようなセレクトボタンがない！



- [RSwitch](#)をダウンロードする必要あり

# / おまけ

## // R Markdownに含めておきたい情報

### /// 日付、日時

- いつ作成されたファイルで、更新はされたことあるのかを明記することが重要
  - デフォルトでは、ファイルの作成日を手動で切り替えないといけない。
- `r Sys.Date` : 2022-01-20



```
title: "Untitled"
author: "Masato Terai"
date: "`r Sys.Date()` in JST" # ``と""で囲うのを忘れない。
output: html_document
```

- `r Sys.time` : 2022-01-20 21:36:36



```
title: "Untitled"
author: "Masato Terai"
date: "`r format(Sys.time(), '%Y-%m-%d %X')`" # ``と""で囲うのを忘れない。
output: html_document
```

- 寺井おすすめ



```
title: "Untitled"
author: "Masato Terai"
date: "作成日:2024-05-20, 最終更新(JST): `r format(Sys.time(), '%Y-%m-%d %X')`"
output: html_document
```

### /// Rのバージョン & 使用したRのパッケージのバージョン

- Rのバージョンやパッケージのバージョン情報など



```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22621)
```

```
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=Japanese_Japan.utf8  LC_CTYPE=Japanese_Japan.utf8
## [3] LC_MONETARY=Japanese_Japan.utf8 LC_NUMERIC=C
## [5] LC_TIME=Japanese_Japan.utf8
##
## time zone: Asia/Tokyo
## tzcode source: internal
##
## attached base packages:
## [1] stats      graphics  grDevices datasets  utils      methods    base
##
## loaded via a namespace (and not attached):
## [1] assertthat_0.2.1 digest_0.6.33 R6_2.5.1 fastmap_1
## [5] xfun_0.41 cachem_1.1.0 knitr_1.45 htmltools_
## [9] rmarkdown_2.27 lifecycle_1.0.4 prettydoc_0.4.1 cli_3.6.1
## [13] fortunes_1.5-4 sass_0.4.9 jquerylib_0.1.4 renv_1.0.
## [17] compiler_4.3.2 highr_0.10 rstudioapi_0.16.0 tools_4.3
## [21] evaluate_0.23 bslib_0.8.0 klippy_0.0.0.9500 yaml_2.3
## [25] jsonlite_1.8.7 rlang_1.1.2 stringi_1.7.12
```

### /// パソコンのスペック

- CPUとコア数とRAM
  - CPU
  - コア数（論理コア）
    - 「論理コア」は「スレッド」「論理プロセッサ」、「仮想コア」とも呼ばれる
  - [CPUとコアについて](#)
- RAM (Random Access Memory)
  - ストレージ（SSDやHDD）ではなく、データを一時保存する場所



```
#install.packages("benchmarkme") # 入れていないかたは先にインストールしてください
benchmarkme::get_cpu()
```

```
## $vendor_id
## [1] "GenuineIntel"
##
## $model_name
## [1] "13th Gen Intel(R) Core(TM) i9-13900K"
```



```
##
## $no_of_cores
## [1] 32
```

```
benchmarkme::get_ram()
```

```
## 137 GB
```

### /// 処理にかかった時間

- 時間のかかる分析を行う人は明記する方が親切！
  - いつ終わるのか分からない見通しのつかない分析を実行するのは怖いので。コア数に加え処理にかかった時間が記載されていればおおよその見通しがつきます。

```
#install.packages("tictoc") # 入れていないかたは先にインストールしてください
library(tictoc)

tic() #測定開始

I <- NULL
for(i in 1:100000){
  I <- c(I, i)}

toc() #測定終了
```

```
## 5.48 sec elapsed
```

## // 関数の呼び出し方

1. `library(関数名)`：一番メジャー
2. `パッケージ名::関数名`：あまりメジャーじゃない
3. `require(関数名)`：`library()` とほとんど変わらないが、関数を読み込めなかったかどうかを論理値で返すことができる。関数を読み込めない（＝パッケージを入れていない）場合に、まずそのパッケージを読み込んで、処理に進ませるみたいな用途で使える。

```
if (require(パッケージ名) == FALSE) {
  install.packages(パッケージ名)
```

```
} else {
  #パッケージを利用してやろうとしていたことをここに書く
}
```

### /// 関数名が被ることがある（xxxはマスクされています）

- グラフの透明度を設定する `alpha` 関数（`ggplot2` パッケージ）と、信頼性係数を出す `alpha` 関数（`psych` パッケージ）
- エラーの原因になる場合あり。`::` で定義すると回避できる。

```
> library(ggplot2)
Need help getting started? Try the R Graphics
Cookbook: https://r-graphics.org
```

次のパッケージを付け加えます：‘`ggplot2`’

以下のオブジェクトは ‘`package:psych`’ からマスクされています：

`%+%, alpha`

### /// 関数とパッケージの対応は覚えにくい

- R Markdownのファイルの冒頭にこのようにまとめてある。確かに、その分析で使うパッケージが一目瞭然だが、個別の処理においてどのパッケージが読み込まれたのか分かりにくく、そのコードを参考に書き換えにくくなる。

```
library(dplyr)
library(stats)
```

- どのパッケージに属する関数なのか一目瞭然。ただしその都度パッケージ名を書くので、コードが長くなるという欠点も

```
data |>
  dplyr::select() |>
  stats::lm(formula = cyl ~ am)
```

## / まとめ

1. R Markdownでは、YAMLヘッダー、コードチャンク、ドキュメントチャンクを使って分析結果をまとめることができる。
  - 研究ごとにプロジェクトを作成しておくとうよい

2. 作成したR Markdownファイルは、Word、HTMLファイル、Power Pointなど様々な形式に出力できる。

- RPubやGithubを使えば、ウェブページとして共有できる

3. RのパッケージやR自体のバージョンは切り替えて使うことができる。

- 再現しようとする分析ファイルに記載されたバージョンに合わせて分析することでより再現性が高まる

## / 参考文献 & 資料

### // : 書籍、 : ウェブサイトの記事

-  「物理コア」と「論理コア」の違い
-  MacでのRstudio上でのR versionの切り替え方法
-  Rが生産性を高める データ分析ワークフロー効率化の実践
-  Rで関数の前に::をつけるのなんで？
-  R言語でパッケージから関数を呼び出す
-  Switching R versions in Windows
-  再現可能性のすすめ : RStudioによるデータ解析とレポート作成
-  私たちのR
-  YAMLについての基本知識
-  R Markdownでスライドを作成 (ioslides)
-  R Markdown クックブック
-  R Markdown Cheat Sheet