

**jamovi モジュールの作成方法**  
**－Moses 検定を例として－**

**How to make jamovi modules**

広島国際大学健康科学部心理学科 小野寺 孝義  
ONODERA Takayoshi

『HIU 健康科学ジャーナル』第2号 抜刷

2023年3月1日 発行

# jamovi モジュールの作成方法 -Moses 検定を例として- How to make jamovi modules

広島国際大学健康科学部心理学科 小野寺 孝義  
ONODERA Takayoshi

要旨：フリーの統計ソフト jamovi が広く利用されるようになってきた。jamovi は統計言語環境 R のスキンとして作動しており、開発はチームとして行われている。R は統計の専門家も多く利用しており、組み込まれている分析手法も数多い。jamovi には基本的な統計手法は網羅されているものの、R で利用できる統計手法が全て網羅されているわけではない。しかし、jamovi はモジュールを作成して組み込むことができるので R の統計手法をモジュール化すれば、それらを利用できる。また、自分で開発した統計手法を組み込むことも可能である。本稿では Moses 検定 (Moses Test of Extreme Reaction) を例として示しながら、jamovi モジュールの作成方法について解説する。

キーワード：jamovi, jamovi module, Statistical software, Moses Test of Extreme Reaction

## はじめに

統計ソフトウェアとして従来は SPSS が心理学や社会科学の分野ではよく利用されてきた。しかしながら、商用ソフトウェアで高額であるために予算が限られている場合には、気軽に利用することはできなかった。また、学生の教育用として利用するのも同様の理由で困難があった。SPSS のクローンをうたった GNU ライセンスの PSPP なども開発されてきたものの、分析手法が少なかったり、日本語対応に問題があるなど決め手とならなかった。そのような状況の中で jamovi が開発され、発表された。jamovi は統計言語環境 R のスキンとして動きながら、R の使いづらさを SPSS 風のユーザー・インターフェースで克服していた。そのため、SPSS 風のフリーソフトと紹介されることもある。利点としては jamovi の操作を覚えることで SPSS の操作もほぼできるようになることである。

しかし、jamovi は単なる SPSS のクローンではない。統計ソフトを世代で分類するなら、現在の SPSS は第2世代であるのに対して jamovi は第3世代、もしくは第4世代と分類できるだろう。第1世代の統計ソフトはコマンド入力ソフトである。IBM カードにパンチするにせよ、モニターから入力するにせよ、コマンドを入力して分析を行っていた。もちろん、コマンドは事前に修得しておかなければ分析は不可能であった。第2世代はグラフィカル・ユーザー・インターフェースの統計ソフトである。コマンド入力をする必要は無く、メニューから選択することで分析が行えた。第1世代がマニュアルと格闘する必要があったことと比べると、第2世代では、マニュアルなしにほぼ直感的な操作が可能になったのである。第3世代の定義については2つの異なる見方ができるかもしれない。統計ソフトの流行や影響力、広がりという歴史的観点でみる立場と統計ソフトの機能の観点でみる立場である。流行や影響力という歴史的視点でみれば R を第3世代とする考え方もできるだろう。ただ、R は本来、

コマンドを入力するものなので機能という観点で考えると第1世代と見なすこともできるかもしれない。これはRのユーザー・インターフェースが進化するというよりも、当時のグラフィカル・ユーザー・インターフェースからみると退化したものであり、またそれが初心者に対するRの敷居の高さにつながったと言える。歴史的観点でRを第3世代とすれば、コマンド入力からグラフィカル・ユーザー・インターフェース入力でRが使えるjamoviは第4世代ということになる。

しかし、機能という面で言えばjamoviはグラフィカル・ユーザー・インターフェースの第2世代を超えた第3世代ということになる。従って、jamoviは単なるSPSSのクローンではなく、次世代の統計ソフトということである。jamoviの開発者の1人であるJonathonはその定義<sup>1</sup>をjamoviフォーラムで示している。それによると第3世代の特徴は「interactive」と「stateful」ということになる。簡単に言えば、プログラムが今の状況や結果を保持し、入力に応じてインタラクティブに操作画面や出力結果を変更してくれるプログラムになっているということである。SPSSでは出力は分析を実行するごとに累積的に出力されるため出力が膨大になってしまう。しかし、jamoviでは出力を累積することもできるが、通常は同じ出力画面に結果だけを更新して示してくれる。統計分析は一度限りで終わることはまれで、現実には試行錯誤を繰り返すことが多い。外れ値の影響を知るために外れ値を含めた場合と含めない場合を分析したり、第3の変数の影響の有無を調べるためにある変数をモデルに加えたり外したり、あるいは因子分析で直交回帰と斜交回帰を比較したり、計算方法を変えてみることもあるだろう。このような時にjamoviでは出力が煩雑になることなく、シミュレーションのように即座に結果をみるのできるのである。このような即時性の利点は使用してみると直ちに実感できるものであり、第2世代との違いは明白となる。

jamoviにはこのような利点があるものの、現状ではRの分析が全て利用できるわけではない。jamoviで分析したいが、メニューに含まれていないためにRを使うしかないという人もいるだろう。また、自分が開発した統計分析が含まれていないのでjamoviに組み込みたいと考える人もいるかもしれない。jamoviではモジュールによってこのような場合に柔軟に対応できる。モジュールを作成して、jamoviの分析メニューに登録できるのである。

日頃、Rを利用して不自由がないという人であっても、モジュール作成の利点はある。それは教育場面である。グラフィカル・ユーザー・インターフェースで直感的操作が可能な統計ソフトであれば、ソフトのインストールやコマンドを教えるという負担が軽減し、統計ソフトが必要とされる統計学や社会調査、卒業研究の教育に集中できるからである。これは同時に学生の負担を減らすことでもある。分析までの環境作りやコマンドの試行錯誤は教育内容の本質ではないからである。

本稿ではjamoviに組み込まれていないMoses検定をモジュールとして作成する手順を示すことで、Rの分析手法をどのようにモジュール化するかを示す。

## 1 Moses 検定

Moses 検定(Moses Test of Extreme Reaction)<sup>2</sup>はノンパラメトリック検定の一種で2群の同等性を検定

<sup>1</sup> <https://forum.jamovi.org/viewtopic.php?p=2433>

<sup>2</sup> Mosesの外れ値反応、Mosesの外れ値処理(SPSS)と記されることもある。

する。母集団に関する前提を必要としない。データが同じ母集団から抽出されたものなら、それぞれの群で分布の広がりには差はないはずである。一方の群でのみ極端な値をとるケースが多いならば、両群は異なる母集団から抽出された可能性が高いことになる。このように Moses 検定は分布の幅を考慮するので、外れ値があると結果に大きな影響を受けることに注意が必要である。具体的な計算の詳細は小野寺・山本(2004)に詳しい。

SPSS では分析メニューのノンパラメトリック検定から Moses 検定が利用可能である。また、R では DescTools パッケージをインストールすることで Moses 検定を利用できる。しかしながら、jamovi では現在のところ Moses 検定を利用できるメニューが見当たらない。そこでモジュールを作成してみることとする。

## 2 モジュール作成環境の構築

### 2.1 jamovi と R の環境作成

最初に jamovi と R のインストールが必要になる。その方法については書籍、インターネットに情報が豊富なので、ここでは省略する。

### 2.2 jamovi のモジュールのインストール

jamovi のモジュール管理から jmv モジュールと jmvbaseR モジュールをインストールしておく。

### 2.3 jmvtools のインストール

モジュール作成のための jmvtools をインストールする。R のコンソール画面から以下を打ち込む。

```
install.packages('jmvtools',repos=c('https://repo.jamovi.org','https://cran.r-project.org'))
```

ここでエラーが出る場合には以下から最新版の jmvtools x.x.x.tar.gz をダウンロードする<sup>3</sup>。

```
https://repo.jamovi.org/src/contrib/
```

ダウンロードしたファイルは tar.gz という圧縮ファイルなので解凍ソフトを利用するか、Windows に付属の Windows powershell (Windows Terminal) から以下のコマンドを打ち込んで解凍する。

```
tar -xzf jmvtools_2.3.4.tar.gz
```

R コンソールから次のコマンドを入力すればインストールが終了する。

```
jmvtools::install()
```

jmvtools がインストールできたなら、次に jmvtools がインストールされている jamovi を認識できるかを確認する。ここでは本稿の執筆時点の最新版であるバージョン 2.3.18 がインストールされているものとして話を進めるが、適宜、バージョンに合わせて数字を変えて欲しい。「jamovi could not be found!」

<sup>3</sup> 執筆時点では jmvtools 2.3.4.tar.gz (2022-04-02)が最新版である。以下の説明では、この 2.3.4 版を例とするが、最新版の番号に置き換えて欲しい。

なお、node\_1.2.tar.gzもインストールしておくこと。

というエラーが出る場合には引用符閉じ記号が「'」となっているか確認するとよい。

```

jmvtools::check(home='C:\Program Files\jamovi 2.3.18.0') # 入力コマンド
# 以下は出力
jamovi compiler
jamovi 2.3.18 found at C:\Program Files\jamovi 2.3.18.0\bin\jamovi.exe
    
```

### 3 モジュールの作成 (create コマンドと addAnalysis コマンド)

R のコンソールから create コマンドで作成するモジュール名を入力する。ここでは Moses 検定なので MosesTest と入力する。「C:\Users\USER\Documents\」フォルダの下に「MosesTest」というフォルダが生成されることが確認できる。

```
jmvtools::create( 'MosesTest' )
```

フォルダの中にはさらに jamovi と R という名前のフォルダが2つできるが、中は空である。他に3つのファイルが作成される。

次に以下のコマンドを入力する。作業フォルダを「MosesTest」にして、具体的なモジュールファイルを作成するコマンドである。

```

setwd('MosesTest')
jmvtools::addAnalysis(name='MosesTest', title='Moses Test of Extreme Reaction')
    
```

この結果、MosesTest フォルダに新たに「build」と「inst」というフォルダが作成され、さらに空だった jamovi フォルダの中に「0000.yaml」「mosestest.a.yaml」「mosestest.r.yaml」「mosestest.u.yaml」という4つの拡張子 yaml ファイルが作成される。また、同じく空だった R フォルダの中には「mosestest.b.R」と「mosestest.h.R」という2つの拡張子 R ファイルが作成される。ここで自動的に作成された yaml ファイルでユーザー・インターフェースや出力を定義し、R ファイルで分析を定義していくことになる。

yaml ファイルの yaml とは「YAML Ain't Markup Language」の略で、HTML や TeX、XML のようにテキストでコマンドを埋め込むことで、コンピュータに指令を出す仕組みの規則であり、yaml は特にオブジェクトやデータ構造を記述することに優れているとされる。yaml で注意すべきことはインデントでまとまりを示すが、インデントにタブ文字が使えないことである。従ってタブでインデントを作るとエラーになる。半角空白で任意の字下げをして、揃えてまとまりを示す必要がある。

yaml ファイルのうち、モジュール作成で特に重要なのは以下の3つである。

- 1)xxxx.a.yaml : 分析定義ファイル。分析のメタ情報。引数やユーザー・インターフェースなど。
- 2)xxxx.u.yaml : ユーザー・インタフェース定義ファイル。分析定義ファイルから自動作成される<sup>4</sup>。
- 3)xxxx.r.yaml : 結果出力定義ファイル。結果の表示方法などの情報を記述する。

<sup>4</sup> より進んだユーザー・インターフェースを用いる場合は内容を手作業で変更する。

あとは yaml ファイルではなく、テキストファイルである xxxx.b.R ファイルに R のコードを記述することになる。

#### 4 モジュールのコンパイル (jmvtools::install()コマンド)

create コマンドと addAnalysis コマンドは一度だけ実行すればよい。だが、yaml ファイルや R ファイルはコーディングを行い、修正をかけるたびにコンパイルをして実行モジュールを更新する必要がある。それが jmvtools::install()コマンドである。ここではまだ何のプログラミングもしていないが、以下に R コンソールでコンパイルの指示を試みよう。jmvtools が jamovi を認識できることを確認してから setwd コマンドで MosesTest フォルダに作業場所を移して、コンパイルを行っている。

```
jmvtools::check(home='C:\Program Files\jamovi 2.3.18.0')
options(jamovi_home='C:\Program Files\jamovi 2.3.18.0')
setwd('MosesTest')
jmvtools::install()
```

以下が出力例の一部である。最終的に「jmo」という拡張子のファイルが作られたことがわかる。

```
jamovi compiler
jamovi 2.3.18 found at C:\Program Files\jamovi 2.3.18.0\bin\jamovi.exe
wrote: mosestest.h.R
. . .
wrote module: MosesTest_0.0.0.jmo
Installing MosesTest_0.0.0.jmo
```

#### 5 ユーザー・インターフェースの作成 (xxx.a.yaml ファイル)

図 1 のようにデフォルトのモジュール画面では従属変数 (Dependent Variable) と独立変数 (Grouping Variable) が用意されている。Moses 検定では 2 群を比較するので、それらはそのまま利用できる。一方で対立仮説 (Alternative hypothesis) のドロップリストや等分散の仮定 (Assume equal variance) のチェックボックスは不要である。ユーザー・インターフェースの定義は xxx.a.yaml ファイルなので、ここでは mosestest.a.yaml ファイルを開いてみる。



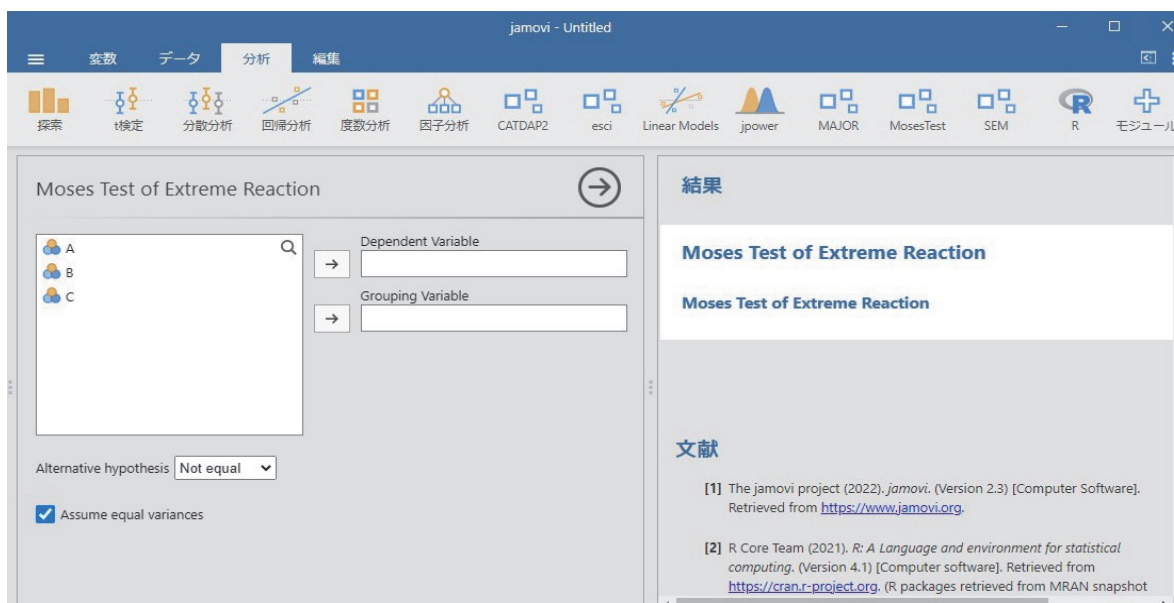


図1 デフォルトのモジュール画面

中に「-name: alt」や「-name: varEq」という行以下で定義されているので、行の先頭に「#」をつけてコメント行にするか、削除する。修正して保存後、再度「jmvtools::install()」コマンドを入力して、再コンパイルしてみる。不要な要素は消えているはずである。もちろん、必要であれば、チェックボックスなり、ドロップリストなり、ユーザー・インターフェースとして必要な要素をここで定義できる。

## 6 RにおけるMoses検定のチェック

ここではRのMoses検定をjamoviに組み込むので、Rにおけるコマンドと出力を調べておく必要がある。インターネットのRDocumentation<sup>5</sup>で調べてみるとコマンドはMosesTest(x,y)であり、統制群xと実験群yという2群を指定して比較分析ができる。また、パッケージとして「DescTools」のインストールが必要なことがわかる。Rのコンソール画面のメニューから「パッケージ」の「パッケージのインストール」を選択し、「Japan」、「DescTools」と順に選択していけばインストールは完了する。「パッケージの読み込み」で「DescTools」を選択すればMosesTestコマンドが使えるようになる。

ここで、MosesTestがどのような出力を出すかを知っておく必要がある。文献に当たっても良いが、簡単なのはRのコマンドmode()とnames()を使うことである。

以下はRコンソールでxの9個のケースとyの9個のケースを比較検定したコマンド例である。MosesTestの結果はresultsに出力され、その結果をmode()関数とnames()関数で調べている。

このコマンドの結果、mode()では"list"が表示され、出力結果はリスト型であること、names()では"statistic"、"p.value"、"method"、"alternative"、"data.name"が表示され、それぞれ「統計量」、「p値」、

<sup>5</sup> <https://www.rdocumentation.org/packages/DescTools/versions/0.99.46/topics/MosesTest>

```
x <- c(2, 3, 5, 7, 11, 13, 17, 19, 23)
y <- c(4, 6, 8, 9, 10, 12, 14, 15, 16)
results <- MosesTest(x, y)
mode(results)
names(results)
```

「分析名」、「対立仮説」、「データ名」に対応することがわかる。つまり、分析結果で出力されるのは、この5つということがわかる。

## 6 分析のコーディング

分析内容を指定するのはMosesTestフォルダの下にあるRフォルダに収められている「mosestest.b.R」である。現状、ファイルの中身はおおよそ、次のようになっているはずである。

```
MosesTestClass <- if (requireNamespace('jmvcore', quietly=TRUE)) R6::R6Class(
  "MosesTestClass",
  inherit = MosesTestBase,
  private = list(
    .run = function() {
      # `self$data` contains the data
      # `self$options` contains the options
      # `self$results` contains the results object (to populate)
    }
  )
)
```

「#」はコメント行であり、「self\$data」にはデータが入っており、「self\$options」にはオプションの指定が、「self\$results」には分析結果が入っていることを示している。分析のコードは「run=function(){」以下に記述する。ただ、その前にMosesTestはRのDescToolsに含まれているのでjamoviで実行する際に自動的にインストールされるように設定が必要になる。これは「C:\Users\USER\Documents\MosesTest」フォルダに作られた「DESCRIPTION」ファイルに記述する。ファイルの中にImportsという行があるので、カンマで区切り、「DescTools」を付け加えればよい。

```
Imports: jmvcore (>= 0.8.5), R6, DescTools
```

「mosestest.b.R」に戻り、次のように#(1)から#(6)の6行の記述を追加する。



```
MosesTestClass <- if (requireNamespace('jmvcore', quietly=TRUE)) R6::R6Class(
  "MosesTestClass",
  inherit = MosesTestBase,
  private = list(
    .run = function() {
      library(jmvcore) #(1)
      data <- jmvcore::naOmit(self$data) #(2)
      formula <- constructFormula(self$options$dep, self$options$group) #(3)
      formula <- as.formula (formula) #(4)
      results <- DescTools::MosesTest (formula, data) #(5)
      self$results$text$setContent (results) #(6)

      # `self$data` contains the data
      # `self$options` contains the options
      # `self$results` contains the results object (to populate)
    }
  )
}
```

「#」記号以下はコメントになるので不要だが、ここでは説明のために入れている。#(1)は `jmvcore` というライブラリを読み込んでいる。#(2)、#(3)で利用する `naOmit` や `constructFormula` 関数は `jmvcore` に含まれているため、ここで指定しておかないとエラーになる<sup>6</sup>。`constructFormula` 関数の書式は次の通りで、従属変数と独立変数をカンマで区切って指定すれば、「従属変数 ~ 独立変数」という R の式に変換してくれる。この式は従属変数が独立変数と線形式の関係であることを意味している。

```
constructFormula(従属変数, 独立変数) # → 従属変数 ~ 独立変数
```

#(2)では元のデータ `self$data` から欠損値をリストワイズで削除して、`data` に代入している。#(3)は変数間の関係を記述した式を `formula` にテキストとして代入している。次に#(4)の `as.formula` 関数で、このテキストが `formula` 属性を持っているものと定義して、再度、`formula` に代入している。#(5)では `Moses` 検定を行い、結果を `results` に代入している。

`MosesTest` の R の書式は `MosesTest(x, y)` と `MosesTest(formula, data, ...)` の 2 種類である。前者は `x` と `y` という 2 つのデータ比較で `t` 検定の対応のあるデータのように 2 つのデータが 2 つの変数として用意されている場合であり、後者は対応のない `t` 検定 (独立な標本の `t` 検定) のように 2 つの群を分けるグループ変数 (要因) が用意されている場合に当たる。ここでは後者の場合を採用している。

```
results <- MosesTest(formula, data) #(0)
results <- DescTools::MosesTest (formula, data) #(5)
```

<sup>6</sup> `formula <- paste(従属変数, '~', 独立変数)` のように式をそのままの形で指定もできるが、制御文字や空白文字の削除等、`constructFormula` を利用した方が問題が生じにくい。

ここで本来であれば#(0)のように、R の検定コマンドを指定すればよい。例えば、t 検定なら `t.test(formula, self$data)` で結果が正常に出力される。しかし、今回の MosesTest で#(0)のように指定して実行すると次のようなエラーになってしまう。

Error in MosesTestOptions\$new(dep = dep, group = group): argument "group" is missing, with no default

原因は `group` という名前が他のパッケージのものとバッティングしているためと考えられる。そこで MosesTest が DescTools のライブラリ内のものであることを明示するために「DescTools::」をつけている<sup>7</sup>。

## 7 モジュールの実行

「mosestest.b.R」のコーディングが済んだら、R のコンソールから「`jmvtools::install()`」を実行し、MosesTest フォルダ内にできた「MosesTest\_0.0.0.jmo」を jamovi にモジュールとして取り込む。具体的にはメニュー右の「モジュール」「モジュール管理」「外部モジュール」から上向きの矢印を押して、「MosesTest\_0.0.0.jmo」を指定すればよい。ここでは 2 値の独立変数と連続量の従属変数のデータを適当に分析しているが、図 2 の右に結果がテキスト出力されていることを確認できる。

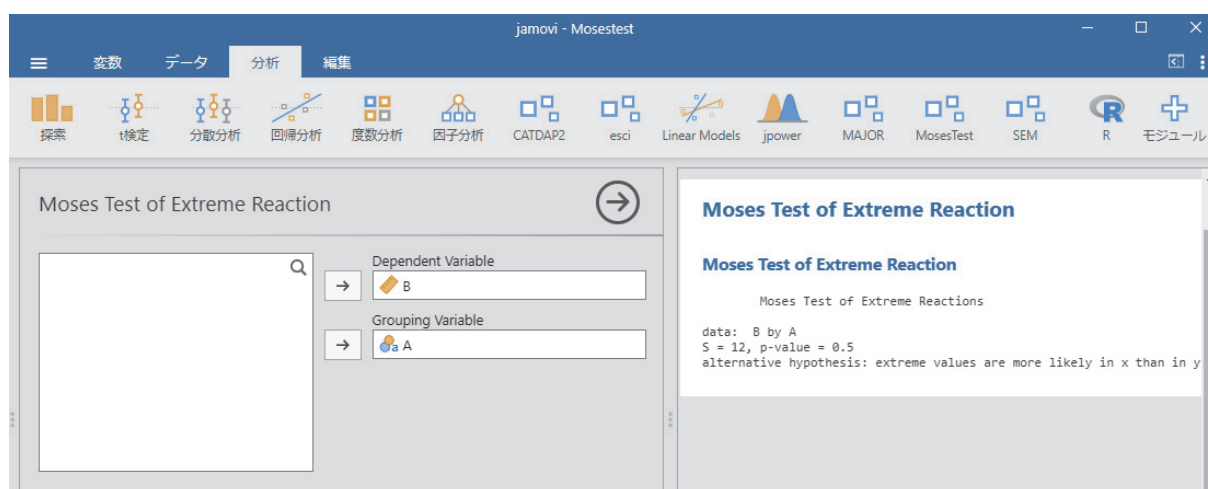


図 2 分析画面と出力画面

以上で jamovi モジュールが完成したことになる。

<sup>7</sup> このようなエラーが常に生じるわけではない。もし、モジュール作成の際にエラーに出会った場合にはこのように対処してみるとよいだろう。なお、ここでは `group` という名前がエラーの原因となっているが、別な名前が原因となることもある。

## 8 終わりに

ここでは最低限の機能と表示の jamovi モジュールを作成した。しかし、何らかのオプションを指定できるようにしたい、複数の従属変数を指定して一度に多くの分析をしたい、変数のデータ水準に応じて指定できるように変数指定をコントロールしたい、結果出力をテキストから表形式できれいに表示したい、グラフを出力したい、あるいは文献に自分の名前を示したいなど、さらに多くの改良をモジュールに加えたいと考えるかもしれない。それらについては残念ながら紙幅の関係で、ここでは解説できない。jamovi Developer's hub (<https://dev.jamovi.org/>)の Tutorial をみていただきたい。

ただ、モジュールの作成が、それほど難しくないと感じられたのではないだろうか。まずは自分が利用する分析手法を R から移植してみる簡易例として本稿が役に立てば幸いである。

## 引用・参考文献

- 馬場 真哉 (2020). R 言語ではじめるプログラミングとデータ分析 ソシム.
- Jonathon, L. (2019). Reply to a post of jRafi (<https://forum.jamovi.org/viewtopic.php?p=2433>)
- 間瀬 茂 (2007). R プログラミングマニュアル 数理工学社.
- 小野寺 孝義 (2022). jamovi モジュールの作成法 日本心理学会第 86 回大会チュートリアル配付追加資料 PDF
- 小野寺 孝義・山本 嘉一郎(編) (2004). SPSS 事典 -BASE 編- ナカニシヤ出版.
- R Core Team (2021). R: A Language and environment for statistical computing. (Version 4.1) [Computer software]. Retrieved from <https://cran.r-project.org>. (R packages retrieved from MRAN snapshot 2022-01-01).
- RDocumentation (2022). (<https://www.rdocumentation.org/>)
- The jamovi project (2022). jamovi. (Version 2.3) [Computer Software]. Retrieved from <https://www.jamovi.org>.
- Tutorial in dev.jamovi.org (<https://dev.jamovi.org/tuts0101-getting-started.html>)