

Julia をインストール

—定量的マクロ経済学の数値計算手法と応用—

山田 知明

明治大学

tyamada@meiji.ac.jp

2021 年 5 月 15 日@日本経済学会

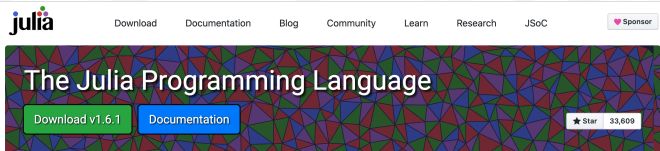


Julia を使う

- なぜ Julia?
 - フリーで使える
 - 高速：公式のベンチマーク
 - ただし、高速化させる方法 (コードの書き方) を理解する必要あり
 - ↑ 初心者は最初は気にする必要なし：後述
 - 近年、開発が盛ん
 - パッケージも増えてきた
 - チュートリアル等も充実：Youtube チャンネル
 - JuliaPro や Jupyter 等、開発環境周りも充実
- 注意：チュートリアルセッションでは Julia(と Fortran) を使っていますが、既に馴染んだ環境があるのであればそちらで OK
 - Matlab、Python、R など

Julia をインストール

- Julia をインストールする方法
 - 公式 HPから Download を選択して自分の環境 (Windows, Mac etc.) にあったバージョンを選ぶ
 - 色々書いてあるけど基本的に Current stable release で OK
 - スライド作成時の最新版は v.1.6.1 (April 23, 2021)
 - インストーラの指示通りにインストールすれば終了
 - フォルダ名やパスが長すぎるという警告が出るかも
 - その場合は適宜、Julia 用フォルダを別の場所に作る



Julia in a Nutshell

Fast

Julia was designed from the beginning for [high performance](#). Julia programs compile to efficient native code for [multiple platforms](#) via LLVM.

Dynamic

Julia is [dynamically typed](#), feels like a scripting language, and has good support for [interactive](#) use.

Reproducible

[Reproducible environments](#) make it possible to recreate the same Julia environment every time, across platforms, with [pre-built binaries](#).

Composable

Julia uses [multiple dispatch](#) as a paradigm, making it easy to express many object-oriented and [functional](#) programming patterns. The talk on the [Unreasonable Effectiveness of Multiple Dispatch](#) explains why it works so well.

General

Julia provides [asynchronous I/O](#), [metaprogramming](#), [debugging](#), [logging](#), [profiling](#), a [package manager](#), and more. One can build entire [Applications and Microservices](#) in Julia.

Open source

Julia is an open source project with over 1,000 contributors. It is made available under the [MIT license](#). The [source code](#) is available on GitHub.

[See Julia Code Examples](#)[Try Julia In Your Browser](#)

- 左上の Download を選択

バージョンを選択

Download Julia

If you like Julia, please consider starring us [on GitHub](#) and spreading the word!

Star 33,612

We provide several ways for you to run Julia:

- In the terminal using the built-in Julia command line using the binaries provided below.
- Using [Docker](#) images from [Docker Hub](#) maintained by the [Docker Community](#).

Please see [platform specific instructions](#) for further installation instructions and if you have trouble installing Julia. If the provided download files do not work for you, please [file an issue in the Julia project](#). Different OSes and architectures have varying [tiers of support](#), and are listed at the bottom of this page.

Note: Julia comes with a built-in package manager which downloads and installs packages from the Internet. In doing so, it necessarily reveals your public [IP address](#) to any server you connect to, and service providers may log your IP address. In Julia versions 1.5 and higher, by default the package manager connects to <https://pkg.julialang.org>, a free public service operated by the Julia project to serve open source package resources to Julia users. This service retains IP address logs for up to 30 days.

Current stable release: v1.6.1 (April 23, 2021)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

Windows [help]	64-bit (installer), 64-bit (portable)		32-bit (installer), 32-bit (portable)	
macOS [help]	64-bit			
Generic Linux on x86 [help]	64-bit (GPG), 64-bit (mus ^[1]) (GPG)		32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)		32-bit (ARMv7-a hard float) (GPG)	
Generic Linux on PowerPC [help]	64-bit (little endian) (GPG)			
Generic FreeBSD on x86 [help]	64-bit (GPG)			
Source	Tarball (GPG)	Tarball with dependencies (GPG)		GitHub

- 自分の環境にあったインストーラをダウンロード

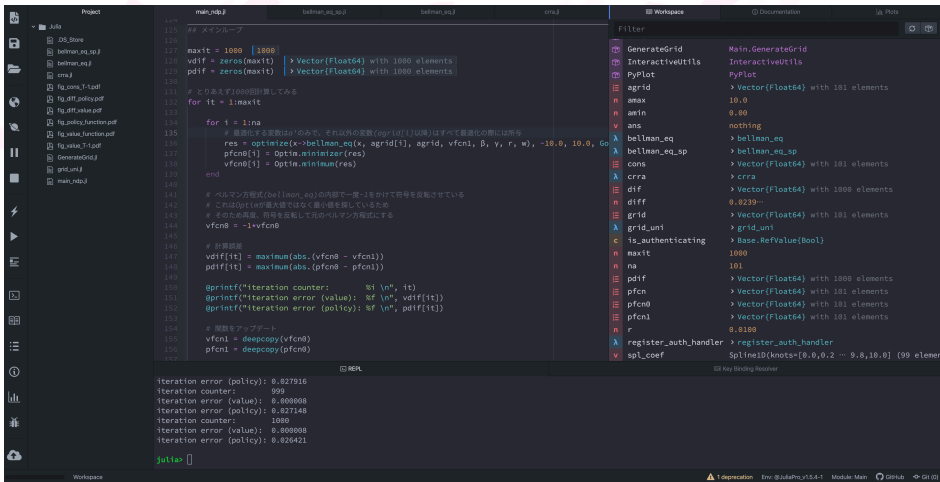
Julia をインストール (続き)

- Julia 本体をインストールしただけでは実用的ではない
- 追加が必要
 1. コードを書く際にシンタックスをハイライトしてくれるエディタ
Visual Studio Code、Sublime Text、Emacs 等
 2. 統合環境 (Integrated Development Environment: IDE)
 - JuliaPro を使う : 次で説明
 - VS Code を IDE 化することも可能
 3. パッケージも必要
 - 極値や根を探すためのパッケージ
 - データを扱うのであれば DataFrame や統計パッケージ
 - 図を描くためのパッケージ
 - ect.

JuliaPro をインストール

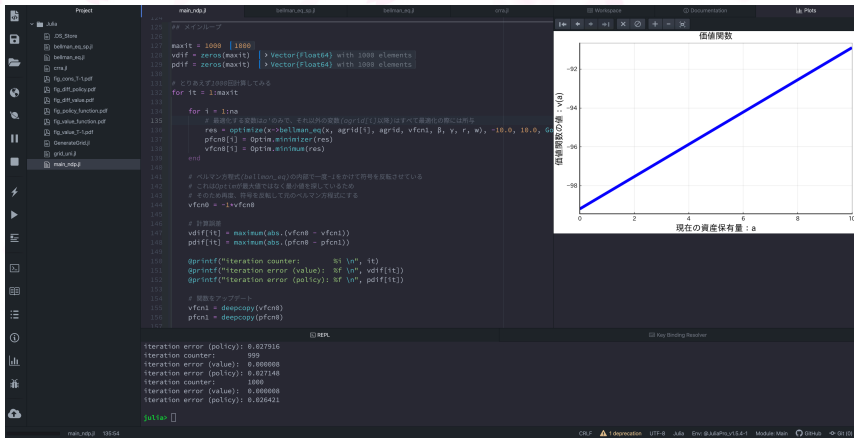
- Julia を Matlab ライクに使いたい
 - コーディングするスペース (もちろんシンタックスをハイライト)、ターミナル、変数を表示してくれるペイン、図の表示、デバッグ etc.
 - R だと RStudio に相当
- JuliaPro を使う
 - Pro って名前だけどこちらもフリー
 - ダウンロード
 - Julia エンジン は 内蔵
 - 前のスライドで説明した Julia をインストールする必要なし
 - ただし、バージョンが少し古い：特に問題なし

JuliaPro はこんな感じ (1)



- エディタ、REPL、Workspace(変数の中身が見られる)

JuliaPro はこんな感じ (2)



- 図も表示される

Jupyter をインストール

- Jupyter Notebook とは?
 - ブラウザ上で動く統合環境：Julia、Python、R など
 - もともと Python 用だったけど汎用的に拡張
- Julia で Jupyter Notebook を使うには?
 - IJulia というパッケージをインストール
 - パッケージのインストール方法は後述

Jupyter Notebook

jupyter Tutorial_JEA Last Checkpoint: 11時間前 (autosaved) ログアウト

ファイル 編集 表示 挿入 セル カーネル Widgets ヘルプ Not Trusted JuliaPro_v1.5.4-1 1.5.4

Run Markdown

3. カリブレーション

- モデルのパラメータを設定
- JuliaではUnicodeが使えるので α とか β を直接コードの中にかける
 - 数式に近い形でコーディングができる
 - `\alpha`と打ち込んでTabキーを押すと変換可能：使用環境によっては出来ないかも
- 行の最後にある`;`は実行結果を表示させないため
 - なくても結果には影響しない
 - Jupyterでを書かない場合、最後の実行の結果だけ表示される
- 1(整数)と1.0(浮動小数点)はコンピュータ上は別物なので注意
 - 今日の計算では問題ないが、Juliaの観点上、自分が扱っている変数の型(type)が何かを意識することが計算の高速化につながる
 - というか、意識しないで適当に換ってしまうと計算が遅くなる原因になる

```
In [42]:  $\beta$  = 0.99; # discount factor
 $\gamma$  = 2.0; # relative risk aversion
 $r$  = 0.01; # interest rate
 $w$  = 1.0; # wage
```

4. 状態変数 a の定義域を有限個の点の集まりとして表現する

- グリッド(grid)を生成

```
In [47]: amax = 1.0; # グリッド(資産)の最大値
amin = 0.0; # グリッド(資産)の最小値
na = 101; # グリッドの数
```

```
In [48]: # 上で読み込んだgrid_uniという関数を使って"等間隔"のグリッドを生成
agrid = grid_uni(amin, amax, na);
```

- Jupyter Lab というものもある

Julia にパッケージをインストール

- パッケージ：Julia に必要な機能を追加
 - 非線形方程式の根を計算したい
 - キレイな図を描きたい
 - 統計データを扱いたい
 - などなど
- チュートリアルセッションで使ったコードを動かすために必要なパッケージ
 - Dierckx
 - Interpolations
 - Optim
 - Printf
 - Plots

Packages をインストール

```
using Pkg
```

```
Pkg.add("Dierckx")
```

```
Pkg.add("IJulia")
```

```
Pkg.add("Interpolations")
```

```
Pkg.add("Optim")
```

```
Pkg.add("Plots")
```

- パッケージをインストールする方法
- REPL で] を押してパッケージモードにしてから add IJulia でも一緒

Julia を高速で動かすためのすごく大雑把な知識

- 注意：専門家ではないので不正確だったり間違っている箇所があるかもしれない点、ご了承ください
- 以下は大雑把なイメージです
- それでも、知っているのと知らないのとでは Julia を使っていく上で差が出るので一応、メモ書き程度にまとめました
- 今後、研究で Julia を使っていこうと思ったら公式マニュアルか本などを一通り目を通したほうが良いと思います
 - 「Matlab でこう書いていたのを Julia だとどう書くんだろう」みたいな対応関係の検索だけだと Julia の機能を十分に使いこなせない可能性あり

Julia を高速で動かすためのすごく大雑把な知識 (続き)

- 「Julia は Matlab や R より早い」と聞いていたのに体感できない
 - なぜ??
- C や Fortran は静的型付け (static typing) という仕組みで、変数の型 (整数とか浮動小数点) を事前に固定した上でコンパイル (機械語に翻訳) する
 - 事前に自分自身で変数の型を宣言する必要あり
 - コンピュータは型に沿った形で最適化
- Matlab や Python、R などは動的型付け (dynamic typing) という仕組みで変数の型にうるさくない
 - プログラミングの際に細かいことを気にしなくても良い
 - コンピュータはその都度変数を機械語に解釈するので遅い
 - 最適化が出来ない (しにくい?)

Julia を高速で動かすためのすごく大雑把な知識 (続き)

- Julia はどっち?
- 基本的に動的型付け
 - Matlab とか R みたいに変数の型を気にしないで書いて動く
 - ただし、その場合は計算速度が遅い可能性あり
- Just-in-time コンパイルと LLVM という仕組みで高速化
 - 細かいことは参考文献を読んでください
 - 重要: Julia は型を意識することで高速化できる
 - 他にも、「一つの長いコードを書かないで小分けに関数にまとめる」などの工夫が高速化のポイント
- Python や Matlab にも独自の高速化の方法がある
 - Cython、MEX など

Julia を高速で動かすためのすごく大雑把な知識 (続き)

- まとめ：Julia を本格的に研究で使うのであれば仕組みを理解する必要があるけど、まずは難しいことを考えずに、手を動かしてみるのが良いと思います！

参考文献：Web 検索ではなく本で読みたい場合

- 進藤裕之・佐藤建太『1 から始める Julia プログラミング』コロナ社
- Kaminski and Szufel 著・中田秀基訳『Julia プログラミング クックブック』オライリー・ジャパン
 - 中級者向け
 - 英語版もあるけど短くまとめた日本語版補遺がとても良い
- Lauwens, B. and A.B. Downey (2019): “Think Julia How to Think Like a Computer Scientist,” O'Reilly.
 - “Think Python”という有名な本の Julia 版
- Balbaert, I. and A.Salceanu (2019,May): “Learning Path Julia 1.0 Programming Complete Reference Guide,” PACT.
- Sengupta, A. (2019,June): “Julia High Performance Second Edition,” PACT.
 - 中級者向け