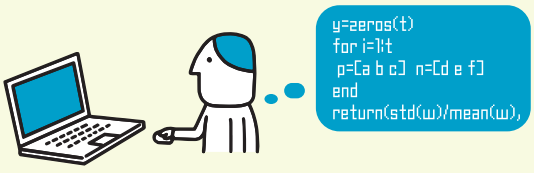
基礎から学ぶ Julia

基本文法からデータサイエンスまで

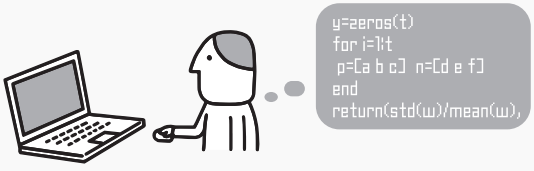


石井 一夫 著

SCC

基礎から学ぶ Julia

基本文法からデータサイエンスまで



石井 一夫 著

SCC

著者略歴

久留米大学バイオ統計センター 准教授。1995年 徳島大学大学院医学研究科博士課程修了後、東京大学医科学研究所、理化学研究所ゲノム科学総合研究センター、フランス国立遺伝子多型解析センター(CNG)、米国ノースウエスタン大学Feinberg医学部などを経て、2011年 東京農工大学農学府農学部特任教授、2017年より現職。博士(医学)、日本技術士会フェロー、APECエンジニア、IPEA国際エンジニア。2015年度情報処理学会優秀教育賞。

サンプルプログラム／サポート情報のご案内

下記の本書専用サポートページでは、サンプルプログラムのソースコード、正誤情報や、
本書に記載された項目等に関する補足情報などを、必要に応じて掲載します。

本書専用サポートページ
https://www.scc-kk.co.jp/scc-books/b-418.html

なお、サポートページの内容は必要に応じて随時更新されますので、ご注意ください。

はじめに

本書は、Juliaに初めて触れる方を想定したJuliaの入門書です。基本的にプログラミング言語の使用経験のない方を想定して書かれています。しかし、Juliaを使用するにあたってPythonやRなど他のプログラミング言語のライブラリやパッケージをインストールして、パスを通す必要があることも紹介していますので、そのような部分は、PythonやRなど他の言語の知識が多少あれば便利です。とはいえ、そのような知識がなくても読み通せるように配慮しています。必要に応じてPythonやRなどのセットアップ方法を他書などで参照されるといいと思います。また、C言語やWSL(Windows Subsystem for Linux)を用いた例も参考に一部示しています。実際には、データサイエンスを学ばれる方は、RかPythonを使用されることが多いですし、また、データサイエンス言語として、RやPythonに満足できずに、Juliaの使用を考える方が圧倒的に多いと思われますので、RやPythonを用いたJuliaの使用法の解説は、本書の読者として想定される一定の人たちのニーズに添うものと考えています。

また、データサイエンスを使用するには、線形代数や、解析学、数理統計学の知識が必要になってきます。本書でも、そのような分野の用語は出てきますが、そのような知識がなくても最後まで読み進められるように配慮しています。もちろん、必要に応じて線形代数や、解析学、数理統計学を他書などで学ばれるといいと考えています。

さて、Juliaというプログラミング言語ですが、PythonやRに比べて計算速度が速く、容易に習得できる次世代のデータサイエンス用のプログラミング言語として注目されています。大量のデータを計算処理するためにデータ分析では計算速度がどうしても必要になってきます。そのような場面では、速度の速い言語というのは選択肢に上がってきます。Juliaは習得が比較的容易であることから、プログラミングの入門としても適していると思われます。本書はその観点から、Juliaを取り上げ、解説を試みています。また、Juliaは、まだ開発が開始されてから時間がそれほど経過しておらず、日本語の解説書が少ないこと、その状況がJuliaの普及を妨げている可能性があることがあります。今後、Juliaの普及に伴って日本語ドキュメントの整備も進んでいくと思われますが、いまだ途上にあるため。そのきっかけになるような書籍が必要であると考えたからです。

本書は、全部で6章の構成としました。

1章は、Juliaの簡単な紹介とインストール方法の解説を試みています。

2章では、Juliaのプログラミング言語としての使用法を、初めての人にも分かるように配慮しながら解説しています。本書を一通りご覧になった方は、Juliaの公式ドキュメントJulia Documentationを参照してさらに理解を深めるといいでしょう。

3章〜5章は各論になり、統計(3章)、グラフィックス(4章)、機械学習・深層学習(5章)の各項目について、関連するモジュールや、パッケージの入門レベルの使用法を説明しています。いずれも、詳細は、それぞれのモジュールやパッケージの公式ドキュメントを参照してさらに理解を深めるといいと思います。関連の使用事例はウェブ上にもたくさん紹介されていますので、確認するといいでしょう。

6章は、その他の方法ということで、5章までで説明しきれなかったその他の事項、例えば IDEの使い方や、スクリプトの操作方法などを解説しています。

本書のコードを一通り試すことで、Juliaの使用法の大枠をつかめると思います。それを基に、よりプログラミング言語Juliaの理解を深めるために、より高度な書籍や情報にアクセスしていただければと思います。

Juliaの紹介は、第1章で述べていますが、筆者は次世代データサイエンス言語としてのJuliaの将来性に大きな期待を持っています。Julia本体や、関連モジュール、パッケージの日本語翻訳を含めなかなか進んでおらず、日本語ドキュメントの充実は課題が多いですが、今後、日本語情報が充実して、その普及が推進することを望んでいます。

本書の執筆に際し、ご協力いただいたスタッフ、家族およびJuliaやオープンソースの普及と活用推進に日々尽力されているすべての方々に感謝いたします。

久留米大学　石井一夫

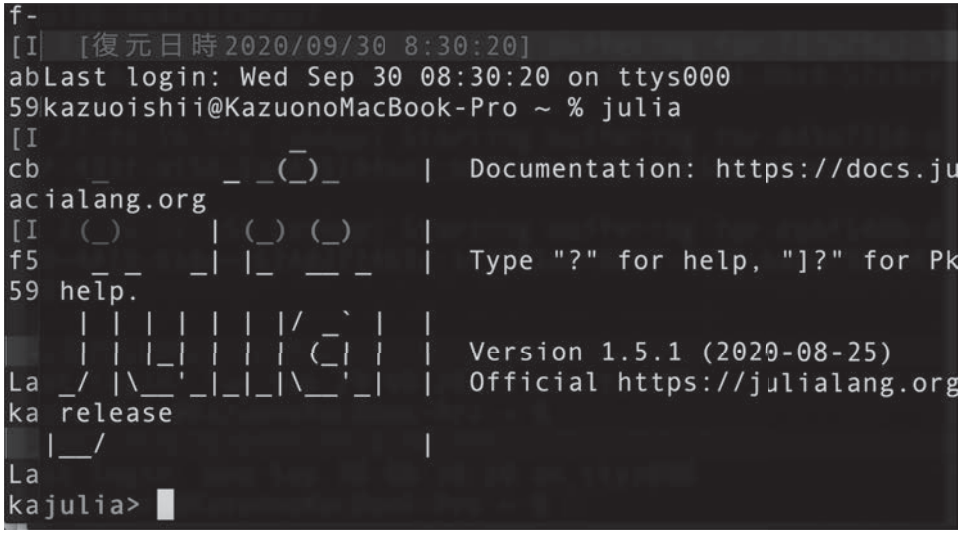
免責事項

本書の内容、コードなどは、正確を期しており、執筆時にテストを行っていますが、ソフトウェアアップデートが頻繁であるため、ソフトウェアの使用時に動作することを保証できません。また、その使用は使用者個人の責任の下で行っていただき、その使用により生じた結果に対し、筆者・出版社は一切責任を負うことはできません。

データ分析で重要なのは、スピードだ。
誰よりも速く、誰より先に、そして誰よりも大量に。

石井 一夫、久留米大学

*この書籍は、Julia Version 1.5.1 (2020-08-25) on macOS Catalina 10.15.で検証した。



目次

第1章 Juliaをはじめよう 1

1-1 Julia の特徴 2

1-2 Julia のインストール 7

1-3 Julia の起動と終了 17

1-4 Jupyter Notebook に Julia を追加するには 18

1-5 JupyterLab の起動 20

1-6 インタラクティブな操作環境 Pluto 22

第2章 プログラミング言語Julia 27

2-1 Julia の REPL 28

2-1-1 Julia モード 29

2-1-2 help モード 29

2-1-3 Shell モード 30

2-1-4 Search モード 30

2-1-5 Pkg モード 31

2-2 プリミティブデータ型と演算子 32

2-2-1 変数 32

2-2-2 整数と浮動小数点 34

2-2-3 任意精度計算 37

2-2-4 二項演算子 38

2-2-5 数学の方程式 40

2-2-6 ビット演算子 41

2-2-7 インクリメント演算子 42

2-2-8 比較演算子 42

2-2-9 文字列 44

2-2-10 特殊文字の入力 48

2-3 関数 50
2-3-1 関数の定義 50
2-3-2 無名関数 (Anonymous Function) 53
2-3-3 オプション引数 55
2-3-4 キーワード引数 55
2-3-5 制御フロー 56

2-4 Julia のデータ型 65
2-4-1 Julia のデータ型の概要 65
2-4-2 型の宣言 66
2-4-3 抽象型 68
2-4-4 プリミティブ型 70
2-4-5 複合型 71
2-4-6 Union 型 73
2-4-7 パラメータ型 74
2-4-8 データ型の演算 75

2-5 コレクション 77
2-5-1 タプル 77
2-5-2 名前付きタプル 79
2-5-3 配列とその操作 81
2-5-4 関数による操作 84
2-5-5 多次元配列 Array 型に関する操作 94
2-5-6 辞書 124
2-5-7 集合（Set） 125
2-5-8 部分配列（サブ配列） 127

2-6 モジュール 132

2-7 メタプログラミング 138

2-8 外部プログラムの呼び出し 146
2-8-1 C 言語の呼び出し 146
2-8-2 シェルのコマンドの呼び出し 147

2-9 パッケージ 149
2-9-1 パッケージの管理 149

第3章 統計解析 153

3-1 統計解析関連ライブラリとその使い方 154
3-1-1 線形代数（LinearAlgebra） 154
3-1-2 データフレーム（DataFrames） 172
3-1-3 統計パッケージ（Statistics） 194

3-2 Python の呼び出しと統計解析 198

3-3 R の呼び出しと統計解析 202

第4章 グラフィックス 207

4-1 Plots パッケージを用いたグラフィックス 209

4-2 グラフの修飾 211

4-3 さまざまなグラフ 216
4-3-1 散布図 217
4-3-2 棒グラフ 218
4-3-3 ヒストグラム 219
4-3-4 さまざまなバックエンド 220
4-3-5 複数グラフの描画 224
4-3-6 PyPlot パッケージを用いた描画 226
4-3-7 Gadfly パッケージを用いた描画 228

第5章 機械学習・深層学習 233

5-1 機械学習・深層学習関連パッケージとその使い方 234

5-2 Knet.jl の使用例 235

5-3 ScikitLearn.jl の使用例 240
5-3-1 iris データの準備 241

第6章　その他の補足 247

6-1 スクリプトの実行とファイルの入出力 248
6-1-1 スクリプトを用いたファイルの読み書き 248
6-1-2 テキストファイルの読み書き 251
6-1-3 エクセルファイルの読み書き 254
6-1-4 CSV ファイルの読み書き 257

6-2 IDE 259
6-2-1 Atom + Juno 259
6-2-2 VSCode（Microsoft Visual Studio Code） 260

索引 265

第1章

Juliaをはじめよう

Juliaは、データサイエンスに特化したプログラミング言語で、PythonやRなどのデータサイエンス指向の言語とともに非常に注目されています。特に、他の言語に比べて高速に動作することと、記述が非常に簡潔で書きやすいことが特徴です。Juliaは、a language that walks like Python, runs like C（Pythonのように書けて、Cのように動く言語）と表現されることがありますが、実際はそれ以上に強力な言語です。本章では、このようなJuliaの特徴を紹介し、その学習のはじめ方について述べていきます。

1-1 Julia の特徴

Juliaは、2009年に開発が始まり、2012年2月にオープンソースとして公表されました。Juliaは、従来のデータ分析言語の欠点を補完し、よりパワフルなデータ分析を可能にすることを目指した言語です。その設計思想は、MITの開発者(Viral Shah、Jeff Bezanson、Stefan Karpinski、Alan Edelman)の発表した"Why We Created Julia(なぜ僕らはJuliaを作ったか)"というドキュメントに、以下のようにまとめられています。

> Why We Created Julia(なぜ僕らはJuliaを作ったか)(Viral Shah, Jeff Bezanson, Stefan Karpinski, Alan Edelman; 2012年2月14日(火)):
>
> 一言で言えば、僕らは欲張りだからだ。
> 僕らはMATLABのパワーユーザーだ。何人かはLispハッカーだし、Python使いや、Ruby使いもいる。Perlハッカーだっている。髭が生える前からMathematicaを使っていた奴もいるし、いまだに髭が生えてない奴もいる。普通の人にはオススメしないくらい多くのグラフをR言語で描いてきたし、無人島に一つだけ持っていくプログラミング言語と言えば、それはC言語だ。
>
> これらの言語はどれも素晴らしいし、強力で大好きだ。でも、科学技術計算、機械学習、データ・マイニング、大規模な線形代数、分散・並行コンピューティング、といった僕らがやる仕事にはどれも、いくつかの仕事の局面で完璧に動作するけど、他の局面では使い物にならない。どれもトレード・オフなんだ。
>
> 僕らは欲張りだ。これでは十分ではない。もっといいものが欲しいんだ。
> https://julialang.org/blog/2012/02/why-we-created-julia/ より抜粋引用して翻訳。

データサイエンスでよく使われるプログラミング言語には、PythonやRがあります。Pythonは、汎用プログラミング言語ですが、科学技術計算や機械学習、深層学習用のライブラリが豊富に揃っており、グラフィックスも美しく描けるため、データサイエンスでは最も人気のある言語になっています。一方、Rは統計解析用のパッケージが豊富で、特に最新の統計解析技法を試すには優れています。商用のソフトでは、MATLABやMathematicaなどが人気です。どちらも非常に使いやすく高機能です。分野によっては、SASやStata、SPSSなども人気のある言語です。

科学技術計算においては、高速性能が重要となってきます。このため、FortranやCのような静的言語で書かれた高速なライブラリをPythonのような動的言語から呼び出す手法で、その性能を達成するようにしています。これらの静的言語は、記述モデルの抽象度が低く、複雑なアルゴリズムの実装には向いていない、事前のコンパイルが必要なためにいろいろな手法を試すには向いていない、という問題がありました。このため、静的言語で書かれた高速ライブラリを動的言語で呼び出す場合には、新たなアルゴリズムを実装するのは容易ではありません。

Juliaは、この問題を最新の計算機技術を用いて、解決を試みています。すなわち、動的言語の特徴である書きやすさ、試行のしやすさを維持したまま、静的言語と同程度の高速性能を得ようという試みです。端的に言えば「Pythonのように書け、Cのように動く(Walks like Python, runs like C)」を実現した言語と言えます。Juliaは、型推論を含む言語設計と、LLVMに基づくjust-in-time(JIT)コンパイラというキーテクノロジーによりこの試みを実現しています。

Juliaの開発者らは彼らの目指した言語の仕様について、以下のように述べています。

> Why We Created Julia(なぜ僕らはJuliaを作ったか)(続き)
>
> 僕らが欲しい言語は、自由なライセンスのオープンソースで、Cの速度とRubyの動的さがあって、Lispのような真のマクロを持つ同図象性の言語で、MATLABのように見慣れた数学の記述ができる言語だ。Pythonのように汎用的に使えて、Rのように統計処理が簡単にできて、Perlのように自然な文字列処理ができて、MATLABのように強力な線形代数計算もできて、シェルのように簡単にプログラムをつなぎ合わせたい。すごく簡単に習えて、超上級ハッカーも満足できて、インタラクティブに使えて、かつコンパイルできる言語が欲しい。
>
> (そういえば、Cに匹敵する実行速度が必要だってのは言ったっけ?)

1-1 Juliaの特徴

やたら注文が多いのは分かっているけど、Hadoopみたいな大規模分散コンピューティングもやりたい。もちろん、そのときにJavaとXMLで何キロバイトもの決まり文句を書いたり、バグを見つけるのに数百台ものマシンに分散した何ギガバイトものログファイルを読んだりはしたくない。幾層にも重なった複雑な構造の構築を押しつけられることなく、それでいて強力なマシンが欲しい。単純なスカラーのループを書いたら、1台のCPUのレジスタだけを使う機械語のコードがコンパイルできて欲しい。A*Bと書くだけで千の計算をそれぞれ千のマシンに分散して実行して、膨大な行列積を一気に計算したい。

型だって必要ないなら指定したくない。ポリモーフィックな関数が必要なときには、ジェネリックプログラミングを使ってアルゴリズムを一度だけ書いて、それをすべての型に使いたい。引数の型に基づいて多数のメソッド定義から最適なメソッドを選択してくれる多重ディスパッチ、共通の機能をまったく違った型にも提供できるようにしたい。これだけのパワフルでありながらも、シンプルですっきりした言語がいい。

これって、多くを望みすぎてるとは思わないよね?
https://julialang.org/blog/2012/02/why-we-created-julia/ より抜粋引用して翻訳。

Juliaは、MITライセンスに基づくオープンソースのプログラミング言語で、RやPythonの弱点を補うことができます。その特徴として挙げられるのが、(1)高速計算能力と(2)学びやすく書きやすいという二大特徴です。

(1)高速計算が可能である

一番の特徴は高速計算が可能であるということでしょう。RやPythonでは、計算速度が足りないという場面は、ビッグデータ分析ではよく経験します。特に、Rはビッグデータのインポートに時間がかかったり、メモリにデータが乗り切らないとか、といったことがよくあります。

Juliaの計算速度は、https://julialang.org/benchmarks/ に紹介されているベンチマークテストによると、機能にもよりますが、RやPython、MATLABやMathematicaの10倍から100倍で、FortranやJavaより速く、RustやGo言語、C言語に匹敵するパフォーマンスを出しています。

(2)学びやすく書きやすい

RやPython、MATLABやMathematicaを使ったことがある人には、とっつきやすく学びやすい言語です。比較的MATLABに似ているので、MATLABユーザーは特に移行しやすいです。MATLABにあるコードの多くが、そのまま動作します。数式をそのまま記述しても動作できたり、引数の組み合わせでいろいろな関数を動作させる多重ディスパッチという機能を持っていたり、記述形式が非常に柔軟です。動的型付け言語で変数の型をあまり気にせずに科学計算に集中できます。

さらに、以下に示すようないろいろな細かな特徴があります。

(3)グルー言語、シェル、マクロ、メタプログラミングのような高水準言語の機能を持つ

Perl、Python、Ruby、Lispいろいろな機能を持つコードが書きやすいスクリプト言語様の高水準言語になっています。

(4)科学技術計算のための機能や、ライブラリが豊富である

データサイエンス用言語としての科学技術計算用関数を多数揃えています。

(5)分散処理や並列計算が得意である

Juliaでは、標準ライブラリの一つとして提供されているモジュールで分散メモリ並列計算を実装できるなど、分散処理や並列計算が簡単にできるようになっています。

(6)CやPythonのライブラリを呼び出せる機能を持つ

現在発展途上であるため、独自のライブラリが少ないという欠点を補う意味で、他の言語のライブラリを呼び出して使うことができるようになっています。

1-1 Juliaの特徴

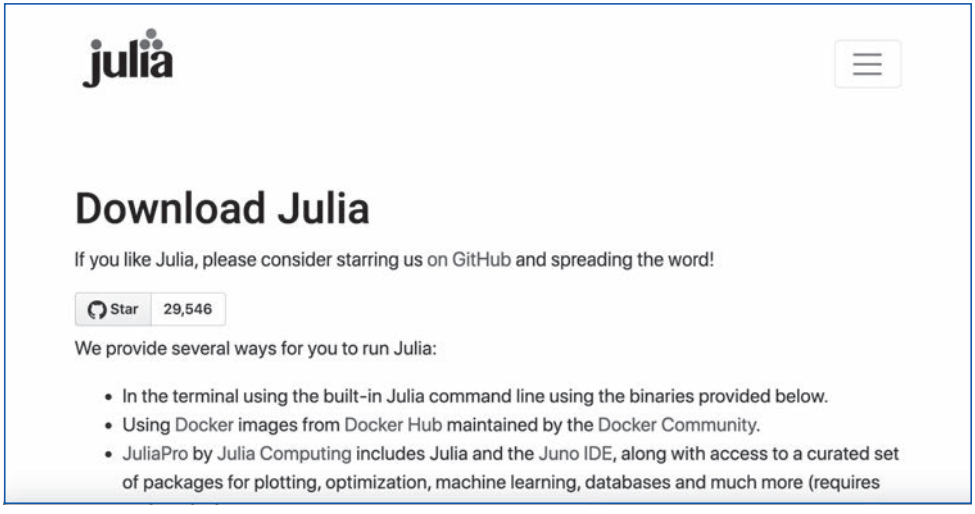
> Why We Created Julia（なぜ僕らはJuliaを作ったか）（続き）
> 僕らがごまかしようのないほど欲張りなのは分かってるけど、それでも全部欲しいんだ。2年半ほど前、この欲張りな言語を作り始めた。まだ完成してないけど、そろそろ1.0のリリースの時期だ。僕らが作った言語の名前はJulia。すでに僕らの無礼な要求に90%は応えてくれてるけど、もっときちんとした形にするために僕ら以外の要求も必要だ。だから、君がもし欲張りで理不尽でわがままなプログラマなら、ぜひこいつを試してもらいたいんだ。
> https://julialang.org/blog/2012/02/why-we-created-julia/ より抜粋引用して翻訳。

Julia 1.0は、2018年8月8日に、Juliaの発表から約6年を経てリリースされました。現在の最新バージョンは、バージョン1.5です。

1-2 Juliaのインストール

標準的なインストール方法を示しますが、インストール方法は頻繁にアップデートされるので、ウェブで最新の情報を入手して実行することをお勧めします。

以下のサイトにアクセスし、自分のOSにあったバイナリファイルをダウンロードします。
https://julialang.org/downloads/



本原稿執筆時の最新バージョンは、stable release: v1.5.1 (Aug 25, 2020)でした。

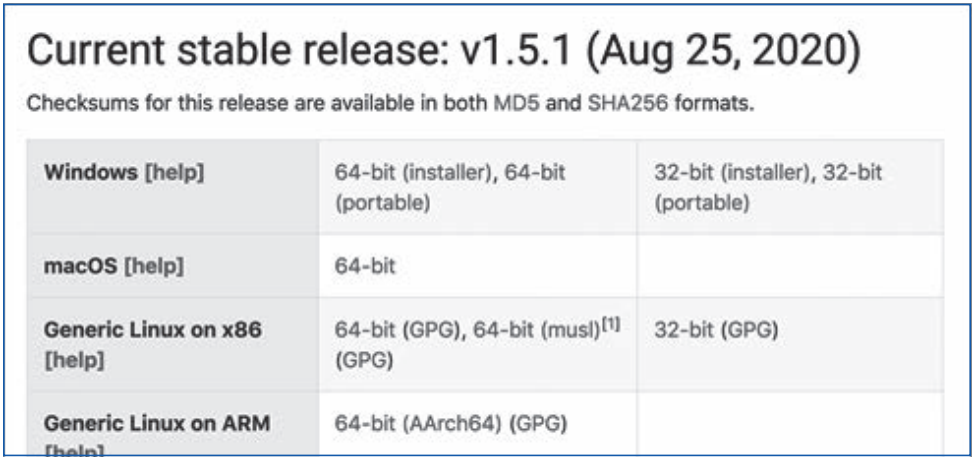
Windowsならば*.exeファイルを、macOSならば*.dmgファイルをダウンロードし、インストールノート(https://julialang.org/downloads/platform/#installation_notes)に従ってパスを通します。

補足:「*.exe」の「*」にはバージョンの名前が入ります。例えば「Julia-1.5.1-win64.exe」など。

1-2 Juliaのインストール

以下にWindows10の場合をご紹介します。

①https://julialang.org/downloads/にアクセスし、Windows Julia installerをダウンロードします(例えば執筆時の最新版は v1.5.1 (Aug 25, 2020)です)。



Current stable release: v1.5.1 (Aug 25, 2020)

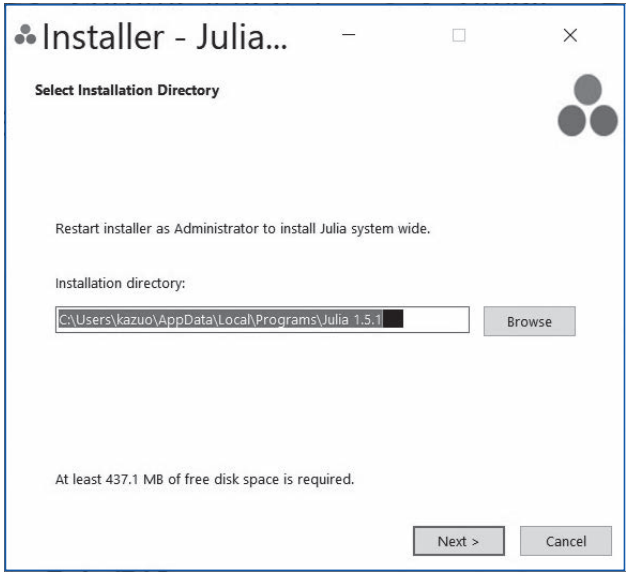
Checksums for this release are available in both MD5 and SHA256 formats.

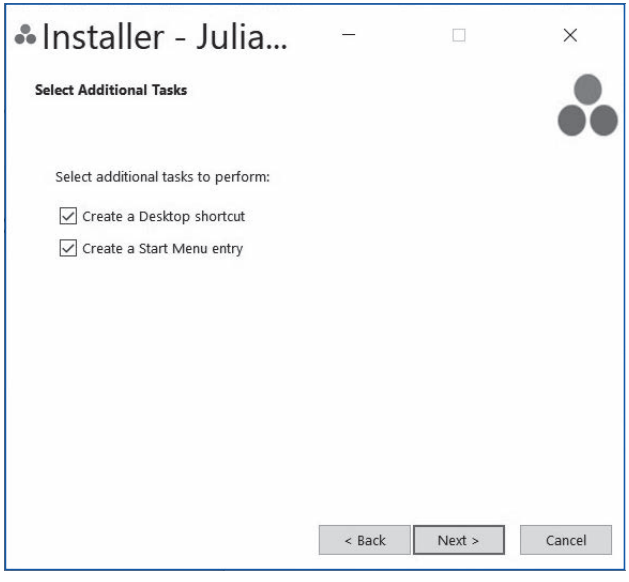
Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)
macOS [help]	64-bit	
Generic Linux on x86 [help]	64-bit (GPG), 64-bit (musl)[1] (GPG)	32-bit (GPG)
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	

②64-bit (installer)か、32-bit (installer)のいずれかをダウンロードします。64-bit (installer)は、64ビット版のみで、32-bit (installer)は、64ビット版と32ビット版の両方のパソコンで動作します。例えば64-bit (installer)をクリックすると「julia-1.5.1-win64.exe」の表示が出ますので、これを「保存」し、保存したファイルをクリックするとインストーラが開きます。

※Juliaのバージョンは頻繁にアップされます。バージョンにあわせて適宜ファイル名を読み替えてください。

③Juliaのインストール先は、デフォルトでは「C:¥Users¥(ユーザー名)¥AppData¥Local¥Programs¥Julia_1.5.1」が表示されます。「Next」を順次クリックしてインストールを進めます。

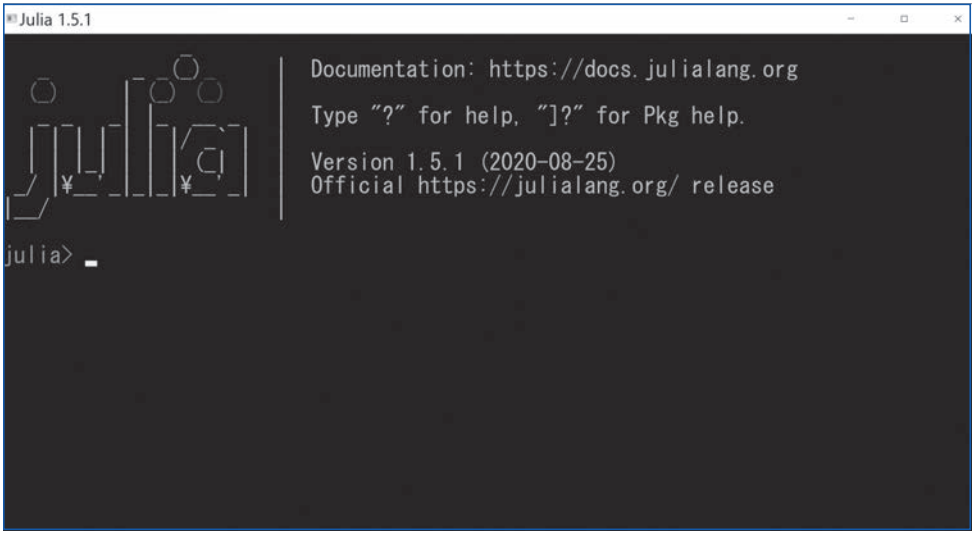




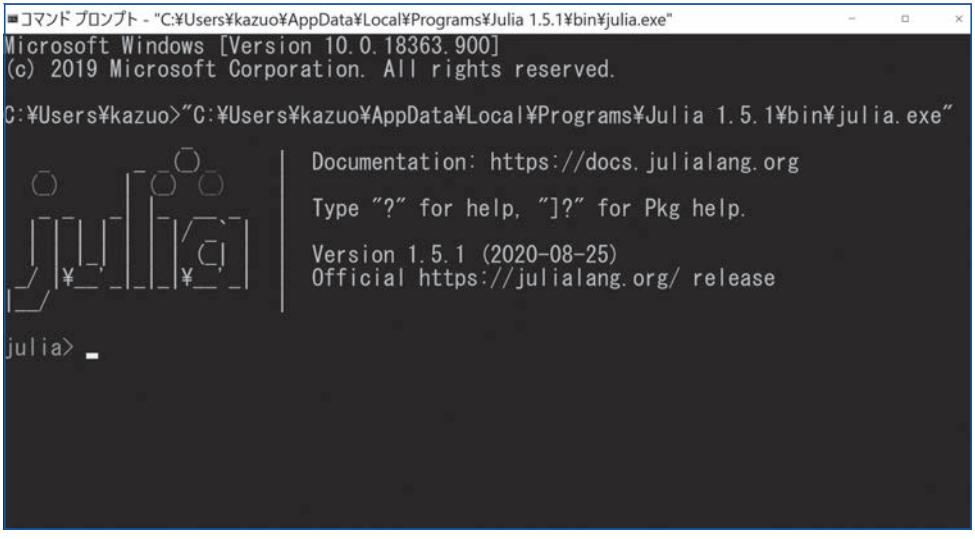
1-2 Juliaのインストール

④「Installation Successful」と表示されれば、インストールは終了です。

⑤ Windowsメニューから Julia_1.5.1 を選択すると、Juliaのコマンド入力端末が起動します。



⑥あるいは、Cortanaを用いて「コマンドプロンプト」ないし「Windows PowerShell」を検索して、JuliaのPATH「C:¥Users¥(ユーザー名)¥AppData¥Local¥Programs¥Julia 1.5.1¥bin¥julia.exe」をタイプし、EnterキーをクリックするとjuliaのREPL(対話的環境)が起動します。

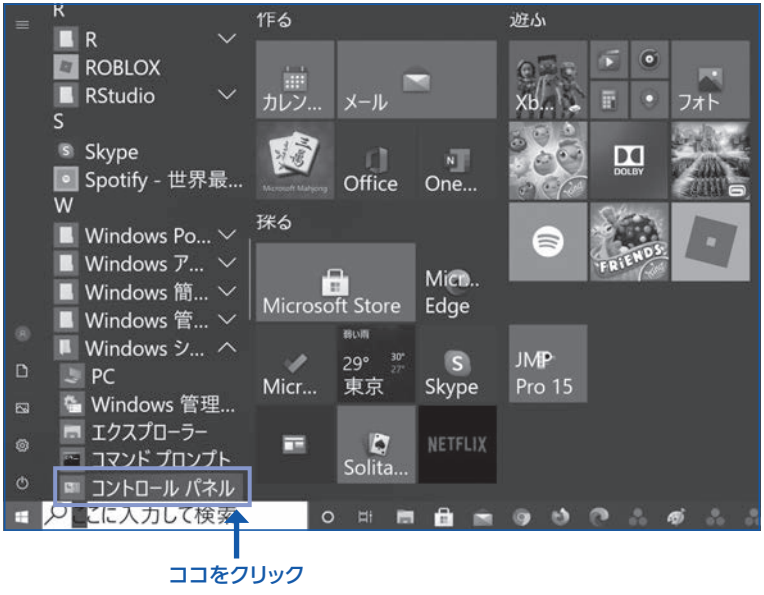


1-2 Juliaのインストール

⑦「コマンドプロンプト」ないし「Windows PowerShell」から「julia」とタイプするだけで、Juliaを起動できるようにするには、環境変数PATHを指定する必要があります。環境変数PATHの指定は以下の手順で行います。

i) デスクトップ左下にあるスタートメニューをクリックし、表示されたアプリの一覧の「Windowsシステムツール」をクリックします。

ii) 表示された中から「コントロールパネル」をクリックします。



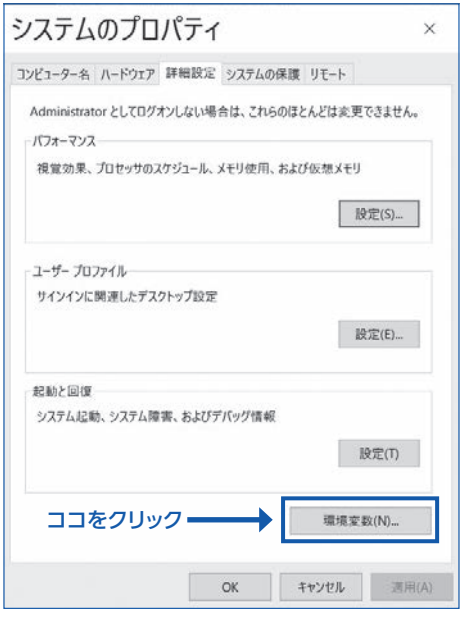
iii) 「コントロールパネル」が表示されたら「システムとセキュリティ」をクリックします。

iv）次の画面で「システム」をクリックします。

v）「システム」画面が表示されたら左側メニューの中の「システムの詳細設定」をクリックします。

1-2 Juliaのインストール

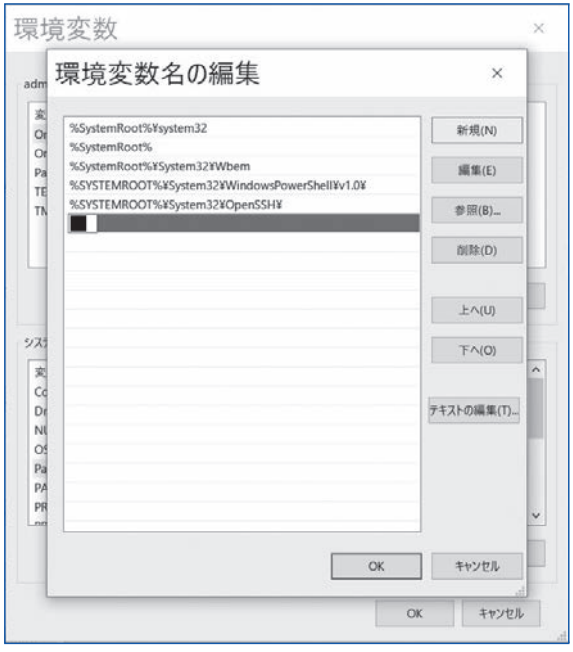
vi）「システムのプロパティ」画面が表示されたら「環境変数」をクリックします。



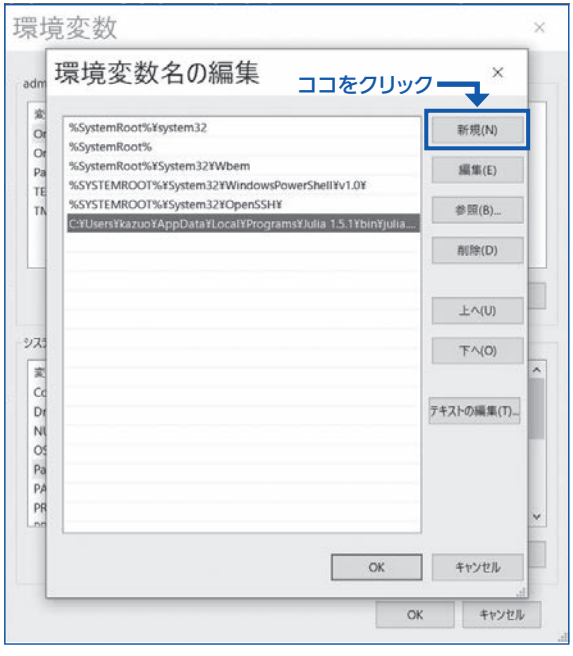
vii）「環境変数」の画面が表示されたら、この画面でPATHの設定を行います。

viii）システム環境変数の中で「変数」が「Path」と書かれたものを探します。見つかった場合には「Path」と書かれた箇所を一度クリックして選択してから「編集」ボタンをクリックします。

ix) 「環境変数名の編集」画面が表示されます。

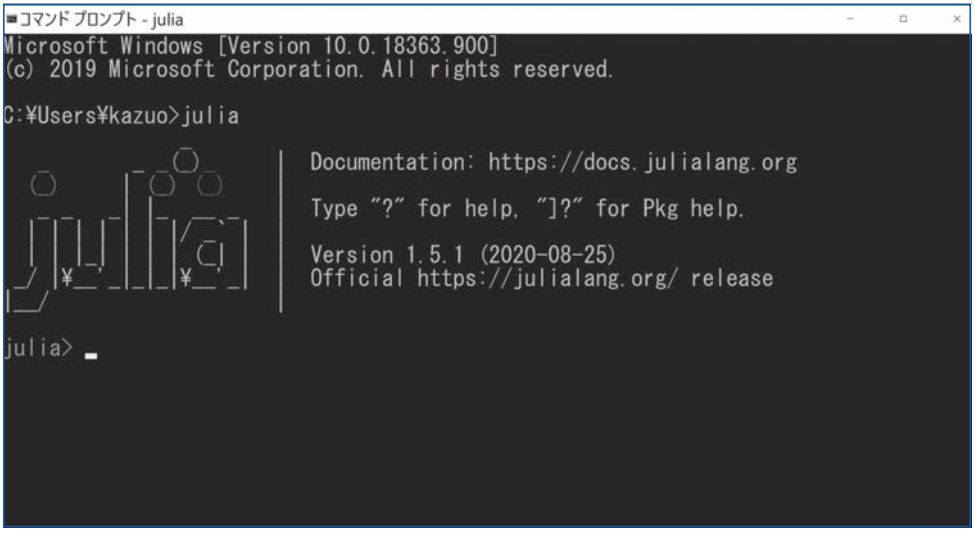


x) 右上の「新規」をクリックすると左側の一覧の最後に新しい項目を追加できるようになるので JuliaのPATHである「C:¥Users¥(ユーザー名)¥AppData¥Local¥Programs¥Julia 1.5.1¥bin」を入力し、入力が終わったら「OK」をクリックします。



1-2 Juliaのインストール

xi) これで「コマンドプロンプト」ないし「Windows PowerShell」から「julia」とタイプするだけで、Juliaを起動できるようになります。



Linuxの場合は、使用しているディストリビューションに合わせたパッケージ管理システムを利用してインストールします。例えばUbuntu の場合、端末で以下のようにタイプします：

```
sudo apt install julia
```

1-3 Julia の起動と終了

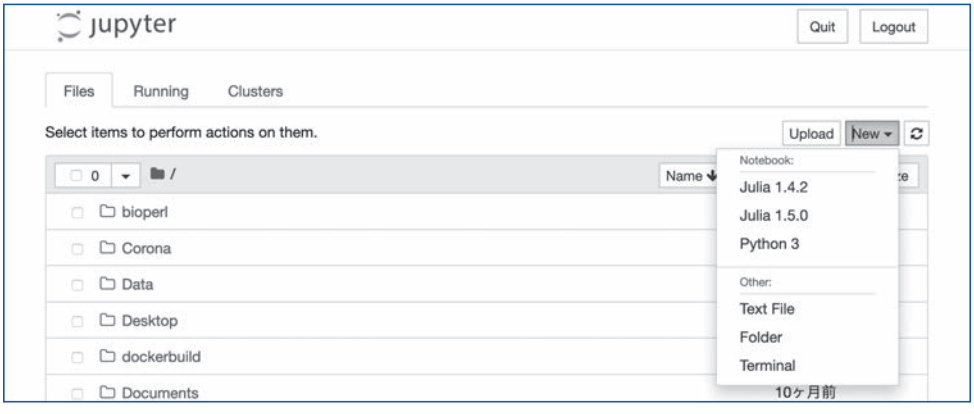
コマンドプロンプト(Windows)または、Terminal(macOSまたはLinux)で、juliaとタイプするとJuliaが起動します。起動するとJuliaのロゴが現れます。

Juliaを終了するには、Juliaモードで、exit() または quit() または Ctrl+d を入力します。

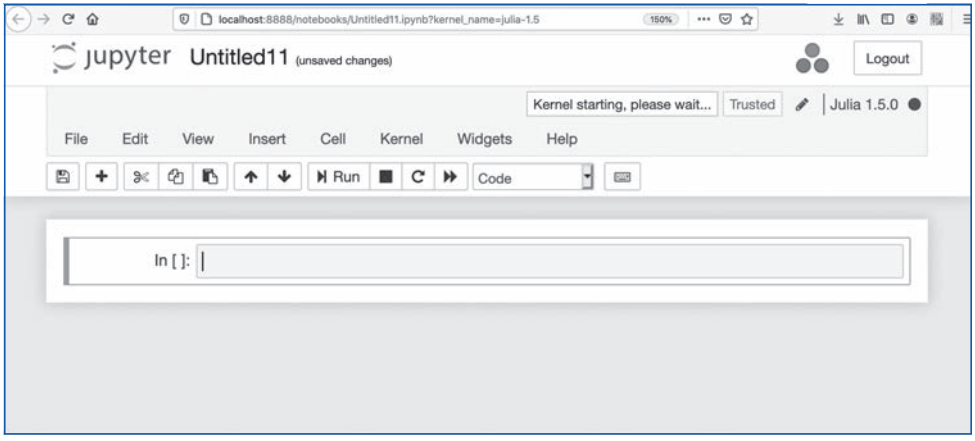
1-4 Jupyter Notebook に Julia を追加するには

Jupyter Notebookで Juliaを使用することもできます。その場合、以下のようにセットアップします。そのためにはAnacondaなどを用いてJupyter Notebookなどをあらかじめインストールしておく必要があります。Anacondaのインストール方法については、Webなどで確認してください。

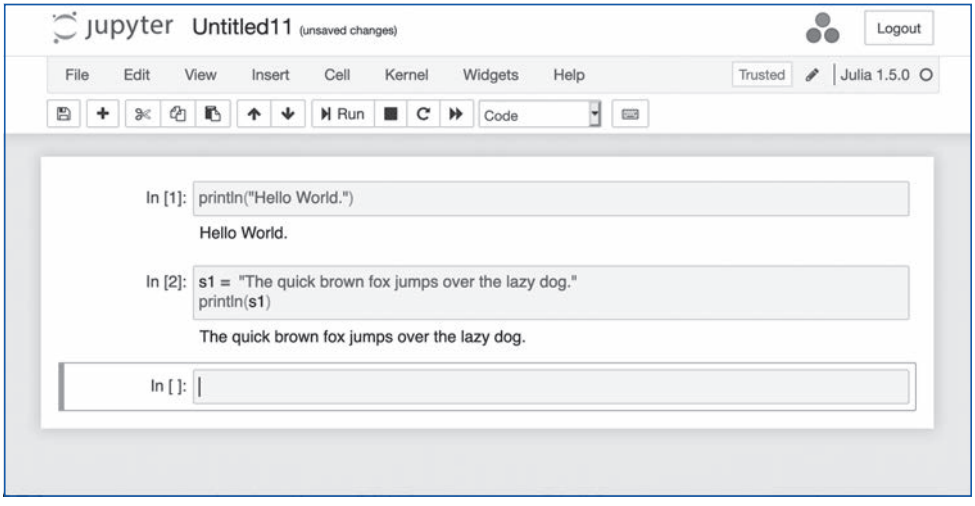
① コマンドプロンプト(Windows)または、Terminal(macOSまたはLinux)で、Juliaを起動し] をタイプして、入力します。
② 文字が青色に変化したら、add IJuliaとタイプして入力します。すると、IJuliaがインストールされます。
③ Ctrl + Cとタイプして、元のプロンプトの状態に戻れます。
④ Juliaを終了し、コマンドプロンプト(Windows)または、Terminal(macOSまたはLinux)で、jupyter notebookと入力して、Jupyter Notebookを起動します。
⑤ Newを選択すると、Juliaが追加されているので完了となります。



入力画面

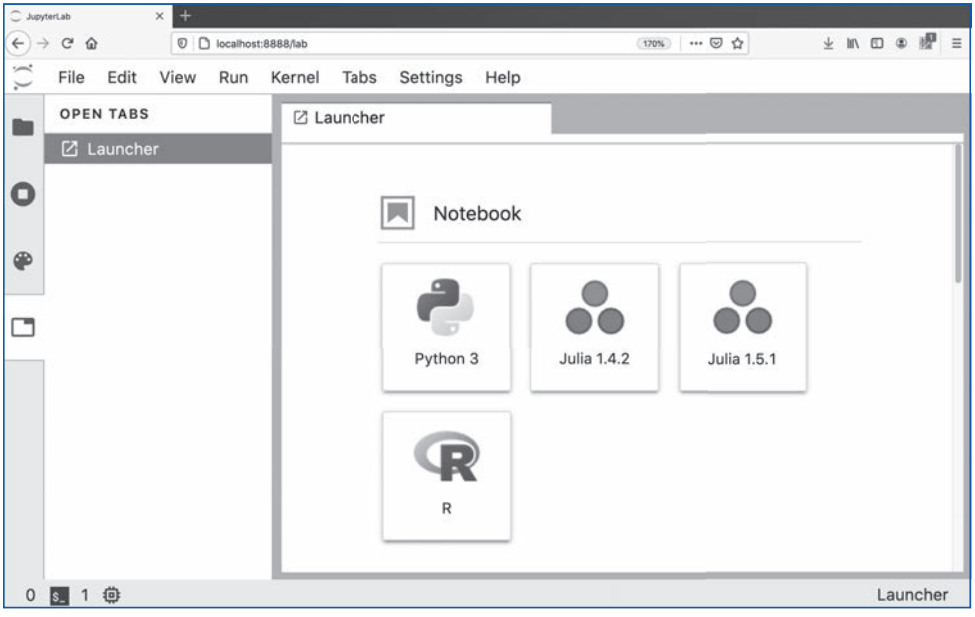

localhost:8888/notebooks/Untitled11.ipynb?kernel_name=julia-1.5
150%
jupyter Untitled11 (unsaved changes)
Logout
Kernel starting, please wait...
Trusted
Julia 1.5.0
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [ ]:

入力例


jupyter Untitled11 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.5.0
Run
Code
In [1]: println("Hello World.")
Hello World.
In [2]: s1 = "The quick brown fox jumps over the lazy dog."
println(s1)
The quick brown fox jumps over the lazy dog.
In [ ]:

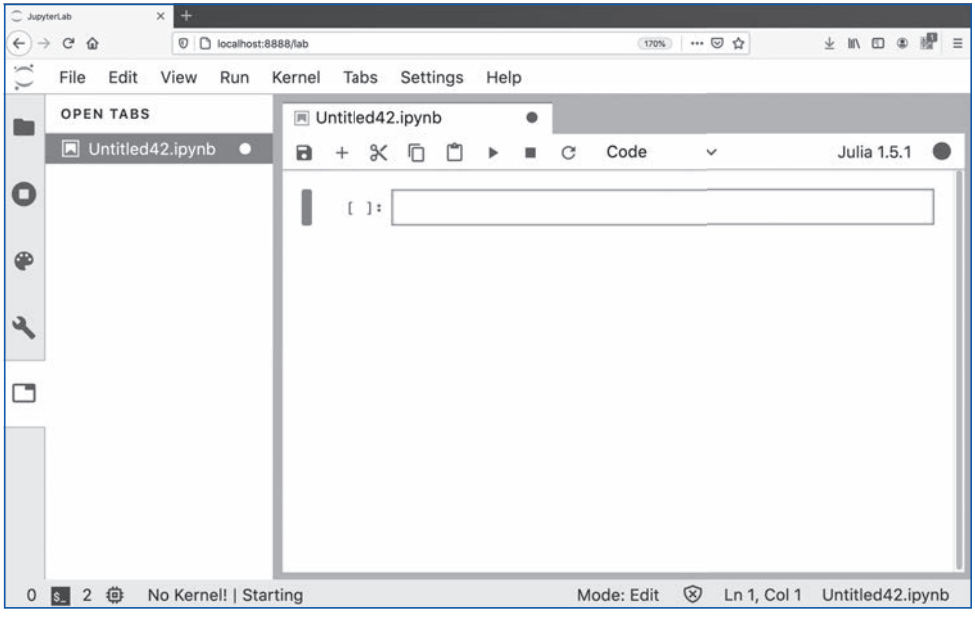
1-5 JupyterLabの起動

また、Anacondaをインストールした上で、これまでの設定を行っていれば、コマンドプロンプトや端末などでjupyter labと入力するとJupyterLabを起動することもできます。



JupyterLabの起動画面

Julia 1.5.1 をクリックすると以下のような入力画面が起動します。



JupyterLab の入力画面

1-6 インタラクティブな操作環境Pluto

さらに、JuliaCon2020では、Jupyter Notebookから派生したJulia用に特化した新規インタラクティブ環境Plutoが紹介されました。非常に使いやすくなっているので紹介します。

Plutoのインストール：以下のようにパッケージモードからadd PlutoでPlutをインストールします。

```
julia> ]
(@v1.5) pkg> add Pluto
```

Plutoのロード

```
julia> import Pluto
```

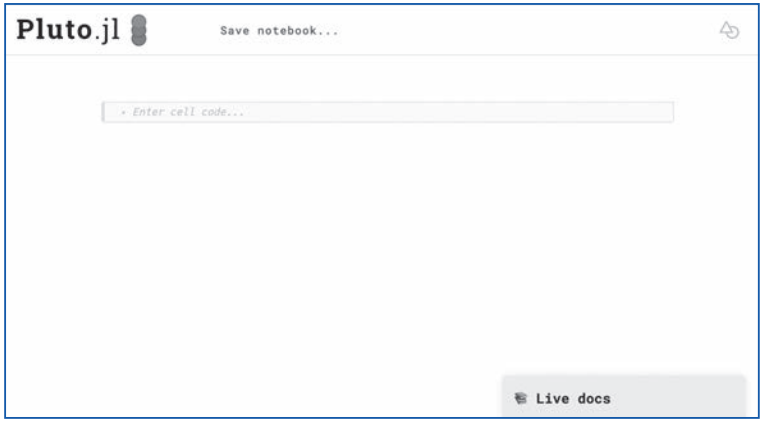
Plutoの起動：次に以下のコマンドで、Plutoを起動します。

```
julia> Pluto.run()
```

すると、以下のような画面がウェブブラウザから開きます。

Pluto の起動画面

「Create a new notebook」をクリックすると以下のような入力画面が起動します、これで、Jupyter Notebookと同じような使い方ができます。

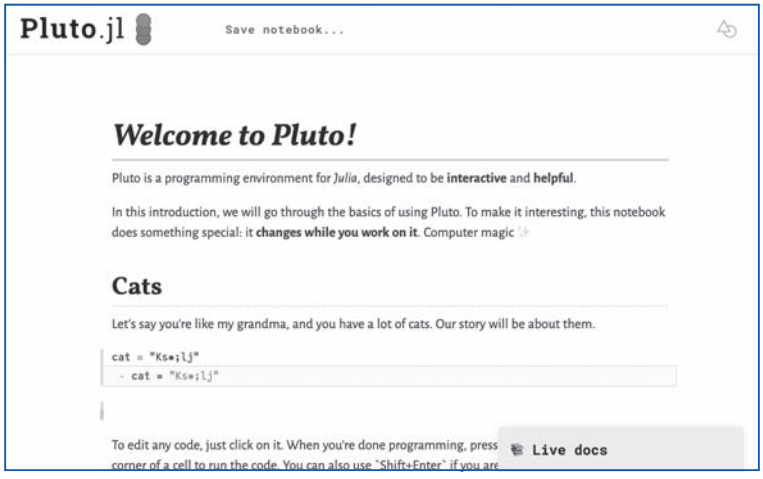


Pluto の新規入力画面

1-6 インタラクティブな操作環境Pluto

さらに、「Create a sample notebook」をクリックすると以下のようなサンプルコードや簡単なチュートリアルを見ることができます。以下に、例を示します。

Pluto のサンプルコード例の選択画面



Pluto のサンプルコードの画面例

参考文献および参考サイト

1) Joshi, Anshul『Juliaデータサイエンス―Juliaを使って自分でゼロから作るデータサイエンス世界の探索』石井一夫、岩中公紀、太田博三、大前奈月、兼松正人、古徳純一、菅野剛、高尾克也、中村和敬訳、NTS、2017年(原著2016年)
2) Bogumił Kamiński (著)、Przemysław Szufel (著)、中田 秀基 (翻訳)『Juliaプログラミングクックブック　言語仕様からデータ分析、機械学習、数値計算まで』、オライリージャパン、2019年
3) 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、2020年
4) Why We Created Julia
https://julialang.org/blog/2012/02/why-we-created-julia/ 2020年7月1日閲覧。
5) The Julia Programming Language
https://julialang.org/ 2020年8月28日閲覧。
6) J Bezanson, S Karpinski, VB Shah, A Edelman, Julia: A fast dynamic language for technical computing, arXiv:1209.5145 (2012)
https://arxiv.org/abs/1209.5145 2020年8月28日閲覧。
7) Download Julia
https://julialang.org/downloads/ 2020年8月28日閲覧。
8) https://julialang.org/downloads/ 2020年8月28日閲覧。
https://julia.mit.edu/
9) WindowsでJulia+Jupyter Notebookによる環境構築
http://blog.livedoor.jp/itengineerblog/archives/15992530.html 2020年8月28日閲覧。