

Development of J-Assist, a statistical analysis support tool for Julia

Hiroki Inoue¹, Hachiro Uchiyama²

¹Department of health informatics, Niigata University of Health and Welfare, Niigata, Japan

²Center for General and Liberal Education, Doshisha University, Kyoto, Japan

Keywords: Julia, statistical analysis, Jupyter Notebook, automatic execution

Received: 8 July 2020 / Accepted: 5 February 2021

Abstract

Julia is a new programming language that was released in 2012. Similar to Python, in the future, Julia has the potential for use in numerical analysis applications. In this study, we developed a free software named J-Assist that automatically executes statistical analysis operations in Julia version 1.4, which could introduce and popularize Julia among researchers and clinicians in the field of health and medical welfare. J-Assist provides a simple graphical user interface and is easy to install. Through this study, we demonstrate that it is possible to configure a system that automatically executes calculations and scripts in Julia. We compare J-Assist with EZR and Jupyter Notebook. J-Assist can be considered a simpler version of EZR. However, the execution languages used in J-Assist and EZR are different. J-Assist is intended for users who are new to Julia and programming and want to use Julia for quick statistical analysis; by contrast, Jupyter Notebook is suitable for users who want to develop complex programs and perform other complex analyses in Julia. This study also discusses the limitations of the current J-Assist software and functions that might be added to it in the future to expand its applicability for statistical analysis.

Introduction

Julia is a new programming language with its version 0.1 having been released only in 2012 [1]; the features of this new programming language include simple syntax and faster code execution compared with existing programming languages [2]. The current version of Julia, namely Julia 1.0, was released in 2018 [3]. A backward compatibility issue was resolved in this latest release of Julia; therefore, an increase in the number of Julia developers is expected. Another advantage of Julia is that it is mostly free to use under its MIT license [4].

Furthermore, because Julia is free for general use, it promotes collaboration between researchers in a project by allowing unfettered sharing of numerical analysis processes and results without the need for purchasing or licensing software that could incur significant financial costs, which might be unsuitable in educational settings.

Moreover, Julia is being continuously developed through the GitHub Julia repositories [5] and has the potential to be a programming language for general use alongside Python, which is an existing programming language that supports a wide range of numerical calculations and analyses.

In a manner similar to Python, in the future, it

Corresponding author: Hiroki Inoue

Department of health informatics, Faculty of Healthcare Management, Niigata University of Health and Welfare, 1398 Shimami-cho, Kitaku, Niigata 950-3198, Japan

TEL/FAX: +81-25-257-4532, E-mail: inoue@nuhw.ac.jp

is expected that Julia will become the language of choice in the field of health and medical welfare, wherein statistical analyses are commonly used for research.

Furthermore, EZR is a free program that is based on R, which is yet another well-known, free statistical analysis programming language [6]; with EZR, users can perform statistical processing in R via mouse operations only. EZR had been cited in 180 research articles in English as of November 2015 [7] and this number grew to over 3000 articles by November 2019 [8], which demonstrates that the existence of supporting software based on a programming language correspondingly increases the overall use of the programming language.

Thus, there is considerable potential for the use of Julia by enabling users who are not familiar with the language to still easily perform statistical analysis operations using it by developing an analysis software tool, which could popularize Julia with researchers and clinicians in the field of health and medical welfare; in addition, it could provide them with a new research method. To the best of our knowledge, there is no automated assistive software for Julia that is simple and easy to learn and use for users who are unfamiliar with Julia. Therefore, in this study, we developed a free and novel tool called J-Assist that can automatically write and execute scripts in Julia for statistical analyses.

Materials and Methods

The J-Assist application developed in this study automatically generates, activates, and executes a Julia statistical analysis script based on selections made by users from a menu.

I. Software design of J-Assist application

1. Fundamental required specifications

(1) Purpose of software

It is used to make Julia easier to use by simplifying the typing of Julia code on the user terminal

and automatically executing calculations and scripts in Julia. The code to be executed is used for performing statistical analyses that are often used by health and medical welfare users.

(2) Target users

The application is intended for busy clinicians, students, and researchers who are unfamiliar with the Julia language but wish to use it to perform routine calculations and statistical analyses without the need for typing at the cost of independent programming work.

(3) Target operating system

The user can visually determine the dataset used for the calculation from the data in the table. Instead of using J-Assist to directly edit the user data, Microsoft Excel is used; hence, the target OS used is Windows 10.

(4) Hardware requirements

The PC running J-Assist must be running Julia 1.x and Microsoft .Net Framework 4.7.

(5) Aim

J-Assist avoids the need to type Julia code on the user terminal and, instead, lets Julia automatically execute calculations and scripts through GUI operations.

2. Input and output

The design related to inputs and outputs is discussed in the “Requested functional specifications” subsection below.

(1) Input

(1-i) The imported dataset is read from the Excel worksheet.

(1-ii) The user selects the following: Excel file to be imported, range of datasets to be imported from Excel, analysis type to be edited, and variables to be selected.

(2) Data processing

Julia language command statements for statistical analysis are added to the imported dataset.

(3) Output

(3-i) The edited Julia language code is written directly to the command line (Julia’s built-in

Read-Eval-Print-Loop (REPL)).

(3-ii) At the same time, the edited Julia code is saved as a text file.

(3-iii) By using the J-Assist output, Julia reads the edited Julia code written directly on the command line and performs the calculations. Thus, J-Assist enables Julia to read the code and to perform calculations automatically. This specification is essential.

3. Requested functional specifications

The following functional specifications were considered based on the above-described fundamentally required specifications and the inputs and outputs while developing the J-Assist application:

The order of the items in the functional specification was kept as close as possible to the procedure followed when the user operates J-Assist.

(1) Installation

The installation procedure of J-Assist on the PC should be easy.

(2) Setup

No setup is required for Julia itself. Because setting environment variables is often cumbersome, we avoided this complication by not requiring users to set the environment variables during the setup process.

(3) Data range selection

The imported data files are in Microsoft Excel (.xlsx file) format. The use of this format enables users to select the range of data for analysis directly from an Excel worksheet.

The imported data and its range can be confirmed from the application window. Data from Excel files can be imported into J-Assist while the Excel files are still open.

(4) Selection of analysis and variables

(4-i) Simple user interface design

We designed a minimalistic interface screen for J-Assist to simplify operations and avoid occupying too much space on the computer screen.

(4-ii) Graphical User Interface (GUI) operation

J-Assist is based on a GUI consisting of a menu with analysis methods using which users can make selections based on their requirements.

(4-iii) Mouse-based operation is promoted

Mouse-based operations are promoted in J-Assist to reduce the amount of typing required from users.

Specification (4-iii) should be implemented to avoid the need for users to perform complicated operations.

(5) Automatic generation and execution of statistical analysis scripts

This is described above in points (3-i) and (3-iii) of the “Input and output” subsection.

(6) Saving generated scripts

This is described in point (3-ii) of the “Input and output” subsection.

(7) Details of Julia

(7-i) Direct use of Julia source code

In developing our J-Assist application, we neither modified nor reused the Julia source code to avoid infringing some restrictions of the MIT license.

(7-ii) Available Julia functions

To ensure reliable performance, the J-Assist application uses the statistical analysis functions already implemented in Julia. Moreover, we selected only those functions that have been developed completely and for which stable releases have been made.

(8) Other

Julia can also access and use Python and R scripts. However, it is better to directly use Python or R for analysis in the case of scripts written in Python and R, respectively. Thus, support for other languages is not included as a specification for J-Assist.

II. Ethical considerations

No ethics review was conducted because this study did not involve human or animal subjects.

III. Measurement of calculation time

The time required to calculate commands with and without J-Assist was compared, and measurements were made to determine the extent to which J-Assist reduces the typing effort of the user. The Windows 10 “Alarms & Clock” application was used for time measurements.

Results

1. Overview of the developed program

J-Assist was developed using Microsoft Visual Basic .NET 2019. The installation package for J-Assist consists of nine files and is approximately 1.5 MB in size. Its performance was verified on a PC running the Microsoft Windows 10 operating system and Microsoft Excel (Office 365). Julia version 1.4.0 was used to check the operation of J-Assist.

2. Implementation of J-Assist application

The implementation of the functions of the J-Assist application is explained below in keeping with points (1)– (7) of the “Requested functional specifications” subsection above. It was explained according to the actual user operation procedure to the extent possible.

(1) Installation

The J-Assist application can be used for free if the user agrees to the terms and conditions displayed during the installation. J-Assist can be downloaded from the UMIN (University Hospital Medical Information Network) website (see footnote below). J-Assist can be easily installed using the Microsoft Installer. However, Julia must be downloaded from the official page for the Julia language (<https://julialang.org/>) and installed before using J-Assist.

(2) Setup

J-Assist does not require the user to set up Julia’s environmental variables. Furthermore, the user is required to input and register the full path

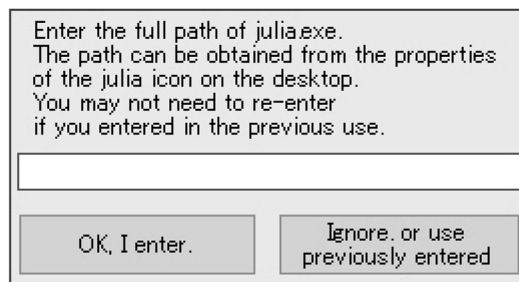


Figure 1. Dialog box to input the full path for Julia.exe.

The full path for Julia.exe must be inputted and registered when J-Assist is used for the first time. However, a different full path for Julia.exe can be inputted and registered later, if required. The full path of Julia.exe can be obtained by right-clicking on the Julia desktop shortcut icon, selecting “Properties” from the context menu, copying the “Target” from the “Shortcut” tab, and pasting it into the dialog box.

of Julia.exe only the first time the J-Assist application is used (Figure 1).

Users can easily obtain the full path of Julia.exe from its desktop shortcut and copy and paste it into J-Assist (Figure 1). The full path of Julia.exe can be modified later, if required.

Furthermore, the Julia packages necessary for statistical analysis that must be installed are already included in Julia itself.

(3) Selection of data range

Figure 2 shows a flowchart of the operation of J-Assist, Excel, and Julia. As shown in this figure, the data ranges are selected and imported from Excel. The data range selection is performed before “Selection of analysis and variables” (see Figure 2).

Figure 3 shows the main operation interface of the application. At the top of the window, buttons are available to select and open an Excel file, input a range of data to be imported from the open Excel file, and import the selected range of data

Footnote: <https://upload.umin.ac.jp/fileshare/registrant.cgi>

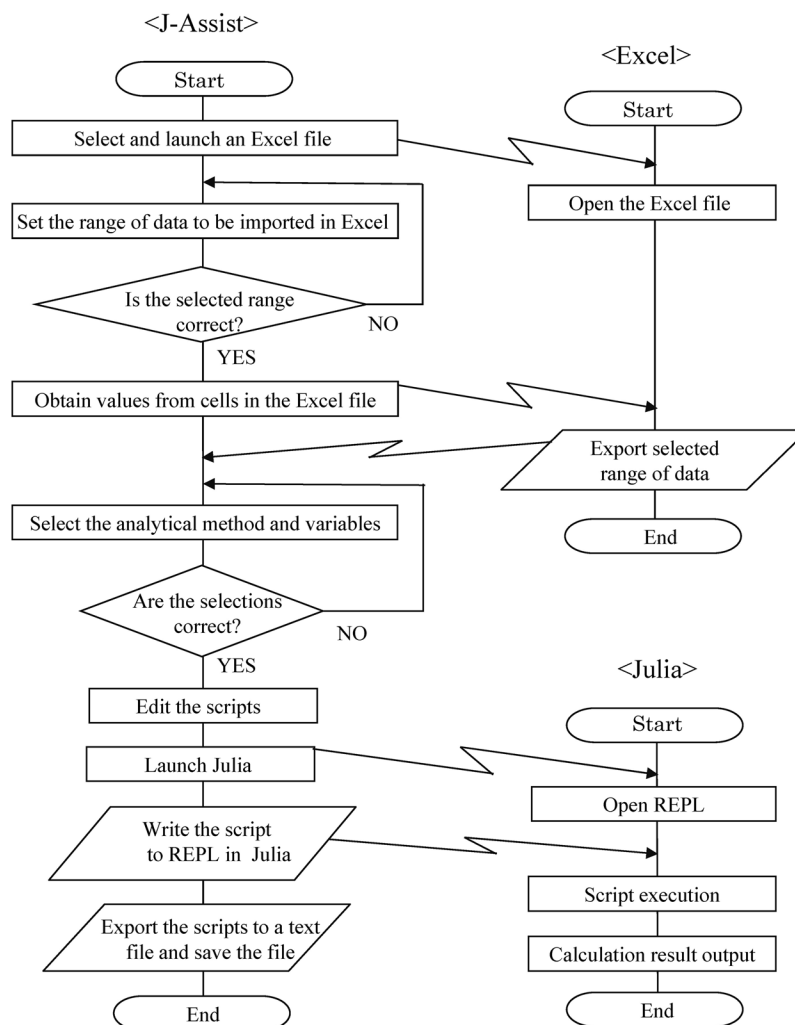


Figure 2. Flowchart of the implementation and operation of J-Assist, Excel, and Julia.

into J-Assist.

Imported data files are in Microsoft Excel format. The procedure for importing the data file and selecting the data range is shown in (3-i) to (3-vi).

(3-i) First, the user can select the Excel file with the data to be imported using the Open Excel file button. For example, as shown in Figure 4, the user selected test.xlsx to import data into J-Assist. It should be noted that the data in the test.xlsx file were only dummy data used herein for explanation.

(3-ii) The selected file test.xlsx was then opened; subsequently, the user selected the range of data

to be imported from the opened Excel file. Herein, the data for variables under columns A1 through D1 in the Excel worksheet were imported for the data range of A2:D10 (Figure 5).

Accordingly, the user inputted the required range of data on the Range of Data screen that opened after clicking on the Range button (Figure 6).

(3-iii) Data selected in the Excel file must be processed first to allow those data to be suitably imported into the application (Figure 5).

In particular, the selected range must be edited to ensure that there are no missing or empty val-

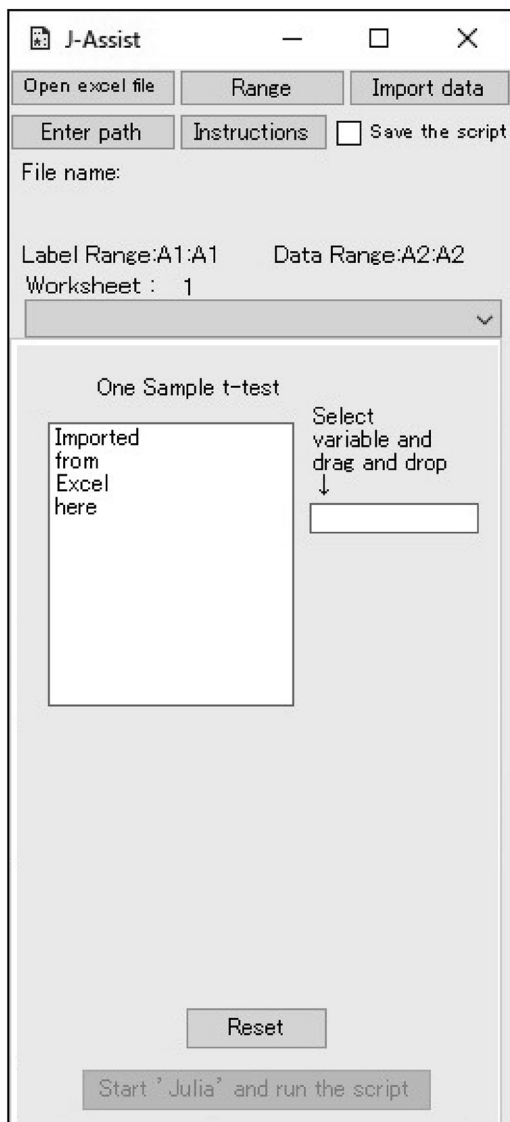


Figure 3. Main operation interface of the J-Assist program.

Select and open the Excel file to import data using the Open Excel file button. Set the range of the imported data from the Excel file by clicking on the Range button. Import the selected data range into J-Assist using the Import data button. Register the full path of the Julia.exe file using the Enter Path button. The file for instructions can be accessed using the Instructions button. The generated script can be saved to the PC by checking the “Save the Script” checkbox. Upon startup, a One Sample t-test screen is displayed.

ues. If there are missing values, “missing” must be written in the relevant cells of the Excel file beforehand. Addressing the issue of missing or empty values in the dataset is a Julia spec-based constraint. It is important for users to address the issue of missing or empty values in datasets before conducting statistical analyses using J-Assist, such as calculating correlation coefficients or conducting paired t-tests with these datasets because Julia can still process and analyze datasets with missing data using the skipmissing or dropmissing functions [9], which might lead to differences in actual and calculated results.

Furthermore, data with characters such as #, (, [, =, or <, which are used as tokens in Julia programming statements, should be avoided because errors might occur during script generation in J-Assist if these tokens are present in the data.

Moreover, duplicate names for column headings should be avoided when creating the Excel file to be imported because multiple column vectors with the same name within the selected range might cause data import errors.

(3-iv) After inputting the range of data to be selected on the Range of Data screen and clicking the Import data button, the required data are then imported into J-Assist.

(3-v) The name of the selected Excel file and the imported data range are displayed on the interface (Figure 7). Range of the imported data is shown on the application window.

(3-vi) Users can work on the data imported into J-Assist directly in the open Excel file; in addition, J-Assist will work even if the file is closed.

(4) Selection of analysis and variables

“3. Required functional specifications (4)” are all implemented as follows. J-Assist has a simple user interface design (Figure 3).

As shown in Figure 3, the required statistical analysis method can be selected from a dropdown list of available methods by using the mouse (Figure 8). In the example shown here, the user selects Multiple Linear Regression as the statistical

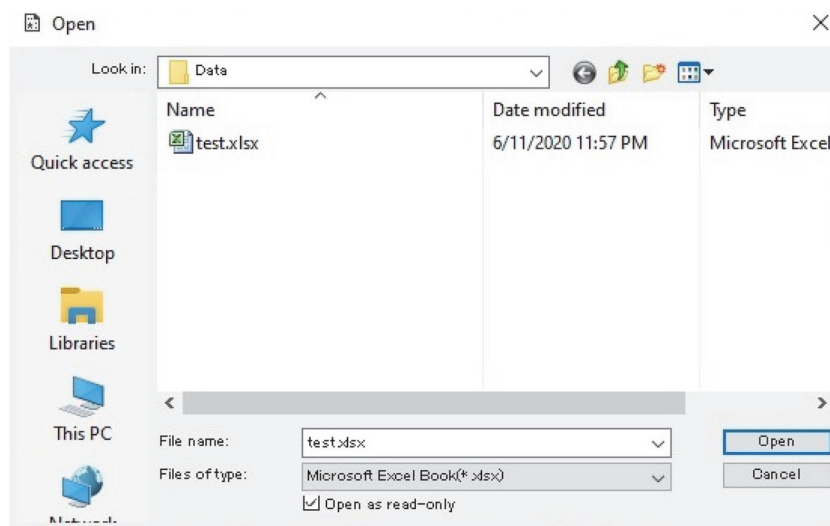


Figure 4. File selection screen.

The file selection screen opens up when the Open Excel file button is clicked.

	A	B	C	D	E	F
1	John	Jane	Doe	Smith	BS	HR
2	1	14	5	12	85	69
3	3	3	2	19	77	60
4	4	23	3	15	91	72
5	6	34	23	13	97	65
6	7	21	8	22	71	66
7	6	5	3	34	96	71
8	3	16	1	20	85	75
9	4	2	12	10	78	81
10	2	7	5	36	89	78
11	2	11	1	13	94	76
12	6	4	1	42	79	63
13						
14						

Figure 5. Range of data to import into J-Assist. In test.xlsx, the different variables are represented as columns A1 through D1 (dotted line) while the data range is A2:D10 (solid line).

analysis method.

Then, below the dropdown list, the variables from imported data are displayed. Users can select the variables they want to include in the analyses from these displayed variable names. As shown in the example in Figure 9, the “Objective” and “Explanatory” variables are selected using the drag-and-drop functionality in J-Assist on the Multiple Linear Regression screen that opens up

after selecting the method for analysis. Variable selections can be changed by clicking on the Reset button.

(5) Automatic generation and execution of statistical analysis scripts

Finally, clicking on the “Start ‘Julia’ and run the script” button leads to the automatic generation of the script, activation of Julia, and execution of the script in Julia. The output of the execution for our example is shown in Figure 10.

After importing the data, analysis methods and variables along with other settings can be modified to create and execute new scripts in Julia.

(6) Saving generated scripts

If the “Save the Script” box is checked beforehand, J-Assist will save the generated script as a text file. Figure 11 shows an example of a generated and saved script.

(7) Available Julia functions

Table 1 lists the statistical analysis methods that are available in J-Assist. Based on the specifications described above, we implemented functions for statistical analysis in J-Assist using already developed Julia packages, and excluding those that are still in development.

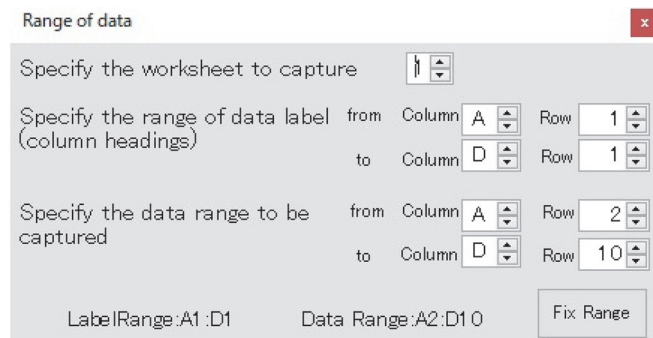


Figure 6. Range of Data screen.

The Range of Data screen can be opened using the Range button and the range can be selected via mouse clicks.

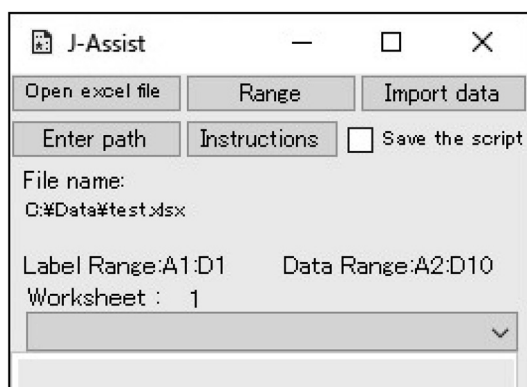


Figure 7. Display after data is imported to J-Assist.

The name of the imported Excel file and selected data range are displayed on the J-Assist screen.

In particular, we selected functions from the packages that had stable releases and were already incorporated into Julia [10][11][12].

Except for the Anderson–Darling test and the Kolmogorov–Smirnov test wherein probability distribution names need to be typed, there is no need to type any text while executing other statistical tests in J-Assist.

3. Measurement of calculation time

Time A from clicking the “Import data” button to the end of calculations in Julia was approxi-

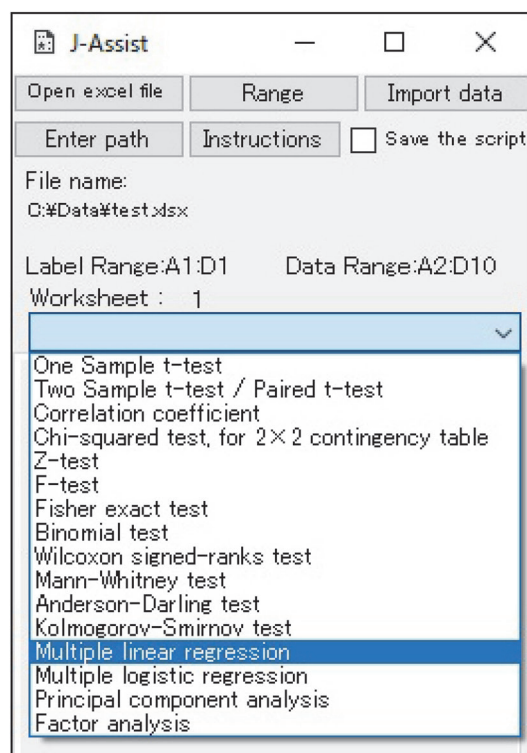


Figure 8. Analysis method menu.

In this example, Multiple Linear Regression is selected as the analysis method.

mately 25 seconds. Time B to complete the calculation was approximately 14 seconds when the generated and saved script in Figure 11 was directly pasted into the Julia REPL and executed.

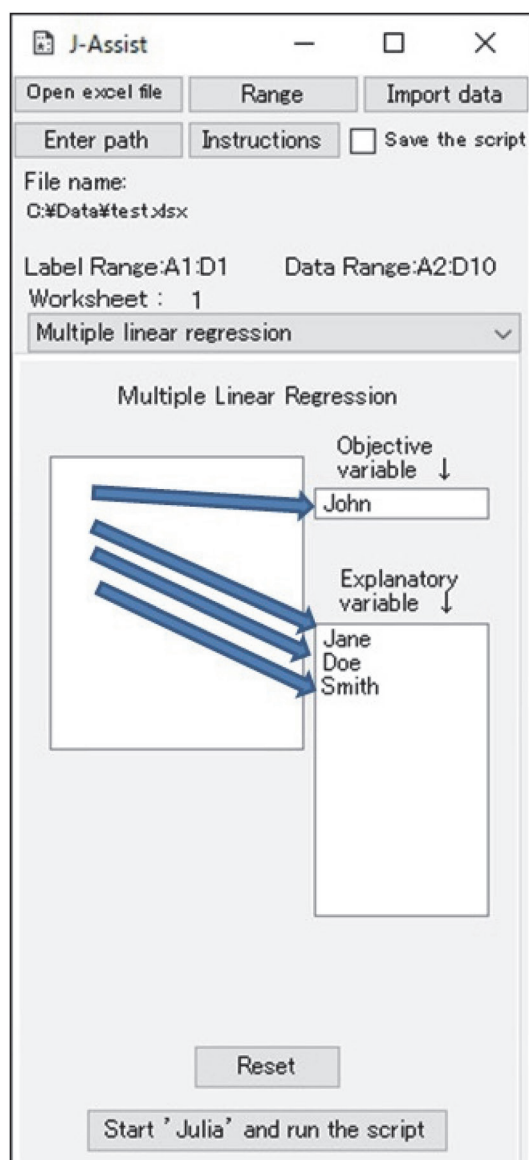


Figure 9. Variable selections after choosing the statistical analysis method
Variable selections are made through the drag-and-drop feature implemented in J-Assist.

Therefore, it took $A - B = 25 - 14 =$ approximately 11 seconds for J-Assist to command Julia to execute this calculation to Julia.

Discussion

Through the use of the J-Assist application de-

veloped in this study, it was demonstrated that it is possible to configure a system to automatically execute the code and scripts in the current version of the new programming language Julia (version 1.4).

The practical time it took J-Assist to command Julia to execute the calculation was approximately 11 seconds in this test. The time of 11 seconds is considered to be well worth the time and effort required for the user to type the same program statement. However, this time difference is not measured mechanically and is affected by the skill of the user's mouse operation.

We believe that the existence of a simple tool to perform statistical analyses such as J-Assist is beneficial for introducing and popularizing Julia with researchers and clinicians in the field of health and medical welfare. Here, we compare the properties and functions of J-Assist with those of other programs to demonstrate its features.

1. Comparison with existing software that assists statistical analysis through menu operations

EZR is a common support tool based on the R programming language. Both EZR and J-Assist have an easy-to-use interface. EZR has several available functions for statistical analysis using R.

Furthermore, with EZR, it is possible to directly edit the data in data files after they have been imported into the tool. In fact, the development of J-Assist was inspired by EZR; although the language of execution used in both tools differs, at present, J-Assist can be considered as a simpler version of EZR.

However, J-Assist will not necessarily evolve into a software similar in functionality and scope to EZR because that will depend on the growth and popularity of Julia in the future. For example, Julia is well-suited for machine learning application and might be used for artificial intelligence (AI)-based developments in the field of health and medical welfare [13].

Thus, the long-term development of J-Assist,

```

C:\WINDOWS\system32\cmd.exe - C:\Users\HInoue\AppData\Local\Programs\Julia\Julia-1...
julia> lm(@formula(John~Jane+Doe+Smith),data)
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Array{Float64,1}},GLM.DensePredChol{Float64,LinearAlgebra.Cholesky{Float64,Array{Float64,2}}}},Array{Float64,2}}

John ~ 1 + Jane + Doe + Smith

Coefficients:

```

	Estimate	Std. Error	t value	Pr(> t)	Lower 95%	Upper 95%
(Intercept)	0.923344	2.50746	0.368239	0.7278	-5.52228	7.36897
Jane	0.0506726	0.0826484	0.613111	0.5666	-0.161782	0.263127
Doe	0.12341	0.131855	0.93595	0.3923	-0.215535	0.462354
Smith	0.075715	0.0875898	0.864427	0.4269	-0.149442	0.300872

```

julia>

```

Figure 10. Output results for Multiple Linear Regression analysis performed in Julia

```

Historyofscriptsby\Assist.txt - Notepad
File Edit Format View Help
#Date:2020-06-14
#20:08:29
# multiple linear regression analysis
C:\Users\HInoue\AppData\Local\Programs\Julia\Julia-1.4.0\bin\julia.exe

using DataFrames
using GLM
using Distributions
using Missings
data=DataFrame(
John=[1, 3, 4, 6, 7, 6, 3, 4, 2],
Jane=[14, 3, 23, 34, 21, 5, 16, 2, 7],
Doe=[5, 2, 3, 23, 8, 3, 1, 12, 5],
Smith=[12, 19, 15, 13, 22, 34, 20, 10, 36])
data=dropmissing(data)
lm(@formula(John~Jane+Doe+Smith),data)

```

Figure 11. Generated script can be saved to the PC.

The script generated for Multiple Linear Regression in Julia is saved. The file is saved to C:\Users\Public.

especially the functions incorporated into it, will likely be influenced by the applications for which Julia is used in the future.

2. Comparison with other software that handles the Julia language itself

At present, Jupyter Notebook is the typical development environment preferred for developing the Julia source code [14]. In particular, Jupyter Notebook can be used to develop, edit, and reuse Julia source code; in addition, it can also be used to re-execute saved source codes.

Furthermore, it can also be used for programming with Julia aside from just using its statistical analysis packages. However, configuring the environment in Jupyter Notebook and performing the supplemental installation of Julia for Jupyter Notebook can be complicated. In contrast, the tool developed in this study, J-Assist, can execute selected statistical analysis functions in Julia simply from a preset menu. Moreover, to use J-Assist, Microsoft Excel is required.

However, the statistical analysis techniques available in the J-Assist menu can be performed

Table 1. Types of statistical analysis functions available in J-Assist.

One sample t-test
Two sample t-test
Assuming equal variances
Not assuming equal variances
Paired t-test
Correlation coefficient
Pearson chi-squared test
Z-test
Assuming equal variances
Not assuming equal variances
F-test
Fisher exact test
Binomial test
Wilcoxon signed-ranks test
normal approximation (-)
normal approximation (+)
Mann-Whitney test
normal approximation (-)
normal approximation (+)
Anderson-Darling test
Kolmogorov-Smirnov test
Multiple linear regression
Multiple logistic regression
Principal component analysis
Factor analysis

using Julia without any knowledge of the Julia programming language or background in programming. Installation and environment configuration for J-Assist also do not require complex operations.

Thus, either Jupyter Notebook or J-Assist can be used for analysis depending on the background of the user and considering their analysis objectives; in particular, J-Assist can be used by users who are unfamiliar with Julia or programming and want to quickly perform statistical analyses using Julia, while Jupyter Notebook can be used by those users who want to develop complex programs or conduct other complicated analyses in Julia.

3. Research limitations

Our study has the following two limitations. The first limitation is related to J-Assist alone. First, in J-Assist, users have to select the data

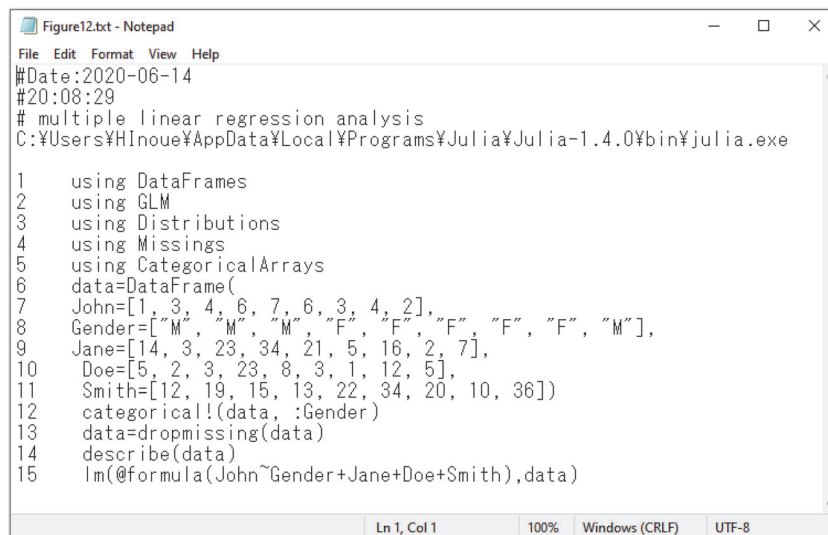
range for analysis while simultaneously viewing the corresponding Microsoft Excel worksheet; currently, the software is not equipped to display or allow editing of data directly from its interface. It is noteworthy that incorporating these display and editing functions would likely require a substantial expansion of the program's features.

As an alternative to the editing function, the number of options supported by the analysis desired by the user could be increased. For example, if a user wants to use a dataset summary and categorical variables for multiple regression analysis, the code written by the user would look like Figure 12. Figure 12 is similar to Figure 11 with the addition of lines 5 and 12. These additional descriptions are code from other Julia regression analysis packages. This Julia package was not included in GLM, Julia's regression analysis package, and therefore, it was not supported by J-Assist. More options should be available to the user to make it easier to accommodate analysis variations.

The second limitation is related to Julia itself. The analysis menu was created by carefully selecting functions with stable performance releases.

The function described in Julia's documentation for that package was considered stable. Among them, the author was able to actually execute one with Julia's REPL and confirm that the operation was selected [10][11][12]; thus, the number of available analysis functions is limited. For example, the current version of J-Assist does not provide options for some frequently used analysis techniques such as ANOVA or Survival Analysis because the necessary packages are not included in Julia 1.4 with their functions either currently in development or not showing stable performance.

In addition, Julia updates may increase the editable settings for analysis. As indicated by the Multiple Linear Regression analysis results shown in Figure 10, the current parameter and



```

File Edit Format View Help
#Date:2020-06-14
#20:08:29
# multiple linear regression analysis
C:\Users\HInoue\AppData\Local\Programs\Julia\Julia-1.4.0\bin\julia.exe

1 using DataFrames
2 using GLM
3 using Distributions
4 using Missings
5 using CategoricalArrays
6 data=DataFrame(
7   John=[1, 3, 4, 6, 7, 6, 3, 4, 2],
8   Gender=["M", "M", "M", "F", "F", "F", "F", "F", "M"],
9   Jane=[14, 3, 23, 34, 21, 5, 16, 2, 7],
10  Doe=[5, 2, 3, 23, 8, 3, 1, 12, 5],
11  Smith=[12, 19, 15, 13, 22, 34, 20, 10, 36])
12 categorical!(data, :Gender)
13 data=dropmissing(data)
14 describe(data)
15 lm(@formula(John~Gender+Jane+Doe+Smith),data)

Ln 1, Col 1    100% Windows (CRLF) UTF-8

```

Figure 12. An example of code written by a user to use a dataset summary and categorical variables for multiple regression analysis.

Line numbers have been added for convenience.

output settings for analyses using Julia may be restricted based on the type of analysis. This is because scripts are generated based on the Julia function documentation thereby limiting the scope of the analyses [11].

Consequently, it will be necessary to add to and update the code for editable parameter and output settings in J-Assist independently if additional packages are implemented or existing packages are updated in Julia in the future.

4. Directions for future developments

In the future, J-Assist can be further developed through three potential steps; these steps consider only the development of the J-Assist program itself and do not depend on future developments in Julia.

First, J-Assist can be extended to non-Windows PCs by enabling the use of data stored in file formats other than the Microsoft Excel format. Second, functionality to open and execute Julia scripts previously written and saved in J-Assist could be added. This would likely become technically feasible through the realization of the first

step. The third step is to enable copying and pasting of the analysis results, including figures and tables, from REPL.

These steps will likely be incorporated based on the future developments in Julia while differentiating J-Assist from Jupyter Notebook.

Acknowledgments

The authors would like to thank Editage for their support and assistance in translating and revising the English version of this manuscript. The authors are grateful to Masanori Kondo, Assistant Professor, Faculty of Healthcare Management, Niigata University of Health and Welfare for the suggestions regarding the format of the flowchart in this manuscript.

Conflicts of interest

There are no conflicts of interest to disclose.

References

1. JuliaLang.org contributors; The Julia Language. Available from: <https://julialang.org/> (accessed April 24, 2020)

2. Sato J, Washizawa Y. An Introduction of Julia language. The Journal of The Institute of Image Information and Television Engineers. 2017; 71: 74-77. (in Japanese)
3. Shindo H, Sato K. Learn Julia Programming from Scratch. Tokyo: CORONA PUBLISHING CO., LTD.; 2020; 1-6. (in Japanese)
4. Opensource.org; The MIT License. Available from: <https://opensource.org/licenses/MIT> (accessed April 24, 2020)
5. GitHub, Inc.; The Julia Language. Available from: <https://github.com/JuliaLang> (accessed April 24, 2020)
6. Kanda Y. Investigation of the Freely Available Easy-To-Use Software ‘EZR’ for Medical Statistics. Bone Marrow Transplant. 2013; 48: 452-458.
7. Kanda Y. Statistical analysis using freely-available “EZR (Easy R)” software. Jpn J Clin Hematol. 2015; 56: 2258-2266. (in Japanese)
8. Division of Hematology, Saitama Medical Center, Jichi Medical University. Free statistical software: EZR on R commander. Available from: <http://www.jichi.ac.jp/saitama-sct/SaitamaHP.files/statmedEN.html> (accessed June 5, 2020)
9. GitHub, Inc.; Missing Data. Available from: <https://juliadata.github.io/DataFrames.jl/stable/man/missing/> (accessed April 24, 2020)
10. JuliaStats contributors.; HypothesisTests.jl. Available from: <https://juliastats.org/HypothesisTests.jl/stable/> (accessed April 24, 2020)
11. JuliaStats contributors.; GLM. Available from: <https://juliastats.org/GLM.jl/latest/manual/> (accessed April 24, 2020)
12. Dahua Lin.; Welcome to MultivariateStats’s documentation. Available from: <https://multivariatestatsjl.readthedocs.io/en/stable/index.html> (accessed April 24, 2020)
13. Anshul Joshi. Julia for Data Science. Birmingham: Packt Publishing Ltd.; 2016; 165-214.
14. Kaminski B, Szufel P. Julia 1.0 Programming Cookbook: Over 100 numerical and distributed computing recipes for your daily data science workflow. Birmingham: Packt Publishing Ltd.; 2018; 52-63.