

ÖGOR Summer-Workshop for PhD-candidates and Post-Docs

An introduction to Julia and JuMP for Operations Research

Prof. Dr. Xavier Gandibleux

Nantes Université – France
Département Informatique – Faculté des Sciences et Technique

Topic 1

Getting started Software environment



Software environment: check list

Base:

- ▶ Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- ▶ Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- ▶ Language: **Julia 1.9.1**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, HiGHS, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- ▶ Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- ▶ Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- ▶ Language: **Julia 1.9.1**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, HiGHS, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- ▶ Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- ▶ Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- ▶ Language: **Julia 1.9.1**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, HiGHS, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ interacting with repository: **GitHub Desktop**

Software environment: check list

Base:

- ▶ Operating system: **MacOS**, Windows, Linux (ubuntu), etc.
- ▶ Browser: **Firefox**, Safari, Chrome, etc.

Apps to install on your computer:

- ▶ Language: **Julia 1.9.1**
- ▶ Editor: **Visual Studio Code (VSCode)**, Emacs, etc.

Apps to install for optimisation needs:

- ▶ Open-source MILP solver: **GLPK**, HiGHS, etc.
- ▶ Commercial MILP solver: **Gurobi**, Cplex, etc.

Apps used (installation not mandatory but recommended):

- ▶ interacting with repository: **GitHub Desktop**

Language: Julia 1.9.1

<https://julialang.org/>

The screenshot shows the official website for The Julia Programming Language. The header features the Julia logo and navigation links for Download, Documentation, Learn, Blog, Community, Contribute, and JSOC. A "Sponsor" button is also present. The main banner has a geometric pattern and displays the text "The Julia Programming Language" along with "Download" and "Documentation" buttons. A GitHub star count of 42,690 is shown. Below the banner, the page is titled "Julia in a Nutshell" and lists several key features: Fast, Dynamic, Reproducible, Composable, General, and Open source, each with a brief description.

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](#).

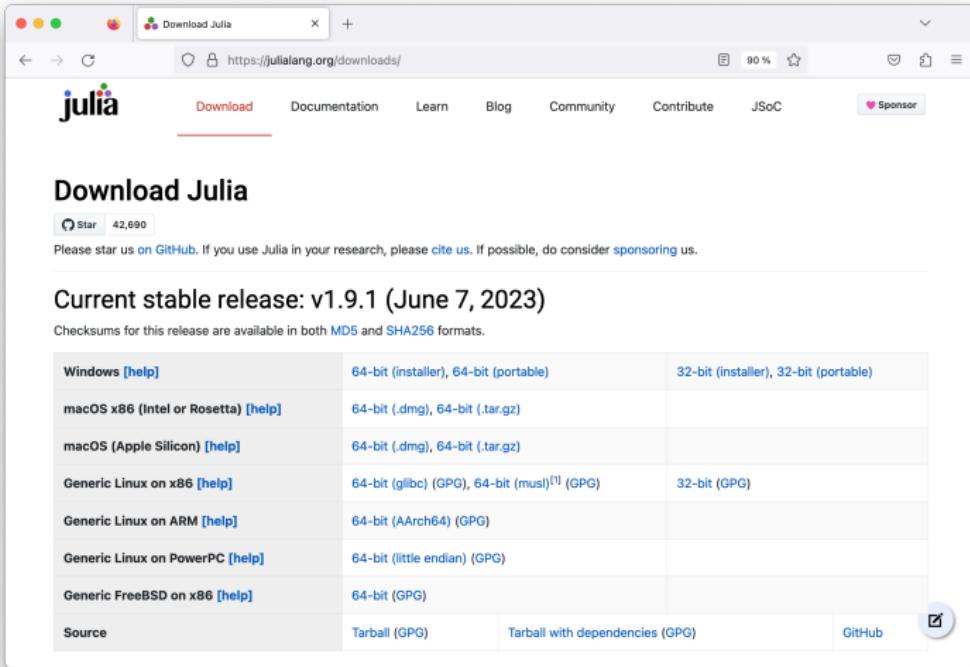
Composable
Julia uses [multiple dispatch](#) as a paradigm, making it easy to express many object-oriented and functional programming patterns. The talk on the [Unreasonable Effectiveness of Multiple Dispatch](#) explains why it works so well.

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.

Language: Julia 1.9.1

<https://julialang.org/downloads/>



The screenshot shows a web browser window displaying the Julia download page at <https://julialang.org/downloads/>. The page features a navigation bar with links for Download, Documentation, Learn, Blog, Community, Contribute, and JSOC. A "Sponsor" button is also present. The main content area is titled "Download Julia" and includes a GitHub star count of 42,690. A message encourages users to star the project on GitHub, cite it if used in research, and consider sponsoring. The "Current stable release: v1.9.1 (June 7, 2023)" is highlighted. Checksums for this release are available in MD5 and SHA256 formats. Below this, a table lists download links for various platforms:

Platform	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)
Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)	
macOS (Apple Silicon) [help]	64-bit (.dmg), 64-bit (.tar.gz)	
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl) ^[1] (GPG)	32-bit (GPG)
Generic Linux on ARM [help]	64-bit (AArch64) (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)

A GitHub icon is located in the bottom right corner of the table.

Language: Julia 1.9.1

<https://julialang.org/downloads/>

The screenshot shows a web browser window with the URL <https://julialang.org/downloads/platform/#macos>. The page content is as follows:

macOS

On macOS, a `julia-1.9.1-mac64.dmg` file is provided, which contains `Julia-1.9.app`. Installation is the same as any other Mac software: drag the `Julia-1.9.app` to Applications Folder's Shortcut. The Julia download runs on macOS 10.9 Mavericks and later releases. You can build from source for macOS 10.6 Snow Leopard (possibly earlier versions as well) and 32-bit but neither are fully supported.

You can launch Julia by opening the Julia app like any other application.

Optional: Add Julia to PATH

If you want to launch Julia from the command line, first open a new terminal window, then run the following snippet from your shell (e.g., using the Terminal app, not inside the Julia prompt).

```
sudo mkdir -p /usr/local/bin  
sudo rm -f /usr/local/bin/julia  
sudo ln -s /Applications/Julia-1.9.app/Contents/Resources/julia/bin/julia /usr/local/bin/julia
```

This code creates a symlink to a Julia version (here 1.9) of your choosing. To launch Julia, simply type `julia` inside your shell and press return.

Uninstallation

You can uninstall Julia by deleting `Julia.app` and the packages directory in `~/.julia`. Multiple Julia.app binaries can co-exist without interfering with each other. If you would also like to remove your preferences files, remove `~/.julia/config/startup.jl` and `~/.julia/logs/repl_history.jl`.

Language: Julia 1.9.1

<https://julialang.org/downloads/>

Platform Specific Instructions X

https://julialang.org/downloads/platform/#windows

Windows

Julia is available for Windows 7 and later for both 32 bit and 64 bit versions.

We highly recommend running Julia using a modern terminal, such as installing the [Windows Terminal from the Microsoft Store](#).

Install Julia in 4 simple steps!

1. Download the Windows Julia installer from <https://julialang.org/downloads/>. Note, the 32-bit Julia binaries work on both 32-bit and 64-bit Windows (x86 and x86_64), but the 64-bit Julia binaries only run on 64-bit Windows (x86_64).
2. Run the installer
3. Check the [Add Julia To PATH](#) to automatically add Julia to Environment Variables

Installer - Julia 1.7.0

Select Additional Tasks

Select additional tasks to perform:

Icons:

Create a Desktop shortcut
 Create a Start Menu entry

Other:

Add Julia to PATH

Language: Julia 1.9.1

<https://julialang.org/downloads/>

The screenshot shows a web browser window with the URL https://julialang.org/downloads/platform/#linux_and_freebsd. The page title is "Linux and FreeBSD". A note at the top states: "It is strongly recommended that the official generic binaries from the downloads page be used to install Julia on Linux and FreeBSD. The following set of commands downloads the latest version of Julia into a directory named `julia-1.9.1`". Below this is a code block:

```
 wget https://julialang-s3.julialang.org/bin/linux/x64/1.9/julia-1.9.1-linux-x86_64.tar.gz  
 tar zxfv julia-1.9.1-linux-x86_64.tar.gz
```

Running Julia

The generic Linux and FreeBSD binaries do not require any special installation steps, but you will need to ensure that your system can find the `julia` executable. The directory where Julia is installed is referred to as `<Julia directory>`.

To run Julia, you can do any of the following:

- Invoke the `julia` executable by using its full path: `<Julia directory>/bin/julia`
- Create a symbolic link to `julia` inside a folder which is on your system `PATH`
- Add Julia's `bin` folder (with full path) to your system `PATH` environment variable

To add Julia's `bin` folder (with full path) to `PATH` environment variable, you can edit the `~/.bashrc` (or `~/.bash_profile`) file. Open the file in your favourite editor and add a new line as follows:

```
export PATH="$PATH:/path/to/<Julia directory>/bin"
```

Apart from this, there are several ways through which you can change environment variable. You can follow [this guide](#) to find out a way convenient for you.

Uninstallation

Julia installs all its files in a single directory. Deleting the directory where Julia was installed is sufficient. If you would also like to remove your packages, remove `~/.julia`. The startup file is at `~/.julia/config/startup.jl` and the history at `~/.julia/logs/repl_history.jl`.



Language: Julia 1.9.1

Just click on the app icon

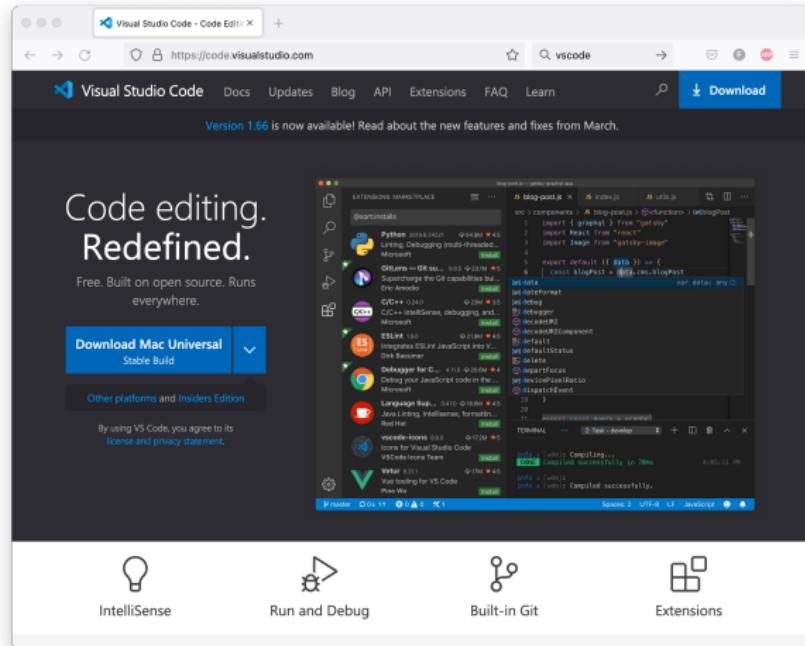


A screenshot of a terminal window titled "xavierg — julia — 80x24". The window shows the following text:

```
[MacBook-de-xavierg-2:~ xavierg$ julia
   Documentation: https://docs.julialang.org
   Type "?" for help, "]?" for Pkg help.
   Version 1.9.1 (2023-06-07)
   Official https://julialang.org/ release
julia> ]
```

Editor: Visual Studio Code

<https://code.visualstudio.com/>

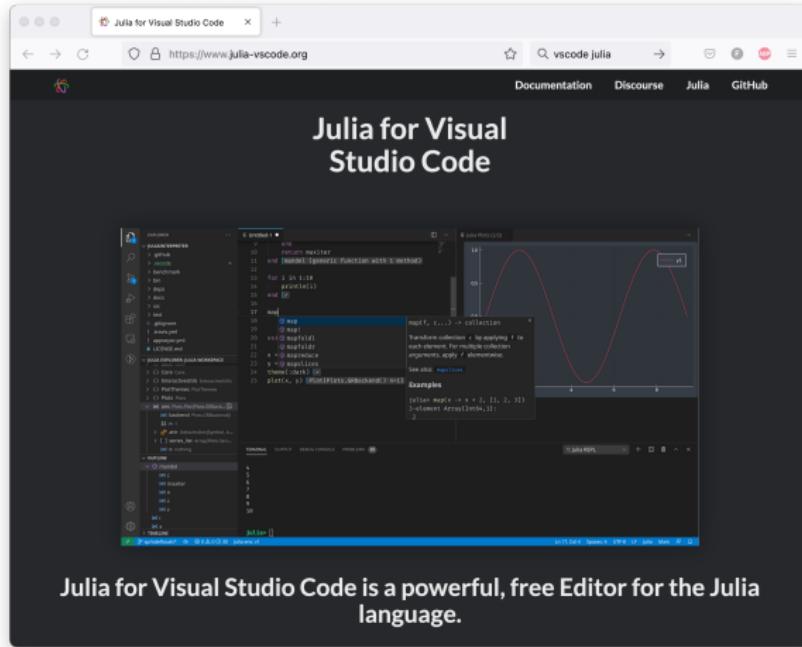


extended for Julia

<https://www.julia-vscode.org/>

Editor: Visual Studio Code

<https://code.visualstudio.com/>



Julia for Visual Studio Code is a powerful, free Editor for the Julia language.

extended for Julia

<https://www.julia-vscode.org/>



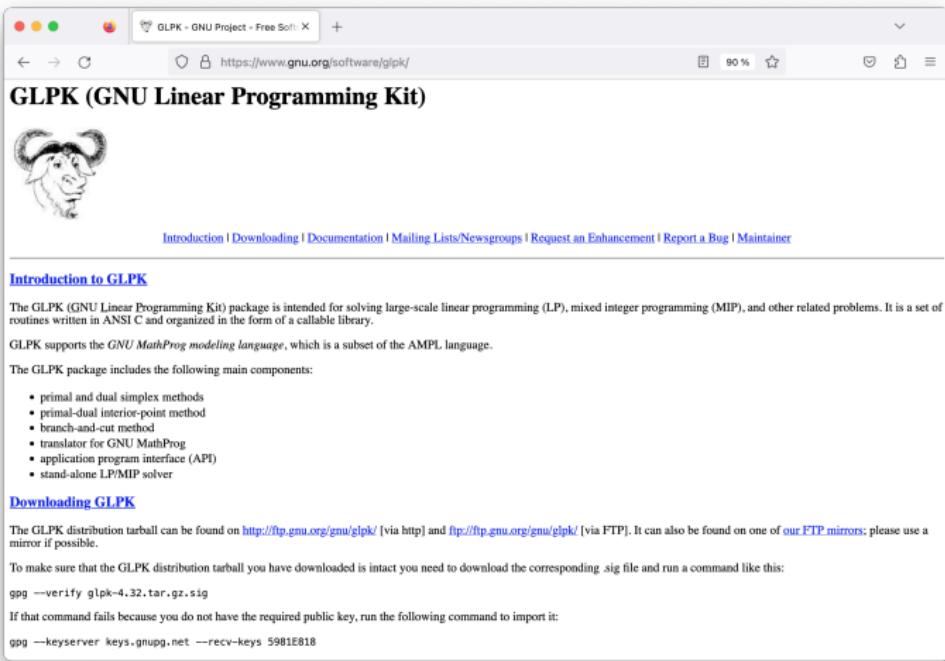
Editor: Visual Studio Code

<https://code.visualstudio.com/>

The screenshot shows the Visual Studio Code Extension Marketplace. In the search bar at the top, the word "Julia" is typed. On the left sidebar, there is a list of extensions related to Julia, including "Julia Language Support" by julialang, "Markdown Julia" by colinfang, "Julia Insider" by julialang, "Julia Color Theme" by CameronBiegeman, "julia light" by haberdashPI, "Embedded Python..." by haberdashPI, "Embedded Matlab I..." by haberdashPI, "Embedded R in Julia" by haberdashPI, "Jupyter" by Microsoft, "Tabnine AI A..." by TabNine, "JuliaHub" by Julia Computing, and "juliaTextMate" by canelhasmateus. The main content area displays the "Julia v1.6.17" extension by julialang, which has 336,750 installs and a 5-star rating from 70 reviews. It is described as providing support for the Julia programming language. Below the extension details, there are tabs for "Détails", "Contributions", "Journal des modifications", and "État du runtime". To the right, there are sections for "Catégories" (Programming Languages, Snippets, Linters, Debuggers, Data Science, Notebooks, Machine Learning) and "Ressources" (Place de marché, Dépot, Licence, julia-vscode.org). At the bottom, there is a section titled "Informations sur la Place de marché" with details about the extension's publication date (21/06/2016), last update (07/04/2022), and last check (19:16:44). A note at the bottom states: "If you have installed Julia into a standard location on Mac or Windows, or if the Julia binary is on your PATH, the Julia VS Code extension should automatically find your Julia installation."

Open Source MILP solver: GLPK

<https://www.gnu.org/software/glpk/>



The screenshot shows a web browser displaying the GLPK (GNU Linear Programming Kit) homepage. The URL in the address bar is <https://www.gnu.org/software/glpk/>. The page title is "GLPK (GNU Linear Programming Kit)". A cartoon character with a horned head and a beard is on the left. Below the title, there is a horizontal menu with links: Introduction | Downloading | Documentation | Mailing Lists/Newsgroups | Request an Enhancement | Report a Bug | Maintainer. The main content area starts with a section titled "Introduction to GLPK". It states that the GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library. It supports the *GNU MathProg modeling language*, which is a subset of the AMPL language. The GLPK package includes the following main components:

- primal and dual simplex methods
- primal-dual interior-point method
- branch-and-cut method
- translator for GNU MathProg
- application program interface (API)
- stand-alone LP/MIP solver

Downloading GLPK

The GLPK distribution tarball can be found on <http://ftp.gnu.org/gnu/glpk/> [via http] and <ftp://ftp.gnu.org/gnu/glpk/> [via FTP]. It can also be found on one of [our FTP mirrors](#); please use a mirror if possible.

To make sure that the GLPK distribution tarball you have downloaded is intact you need to download the corresponding .sig file and run a command like this:

```
gpg --verify glpk-4.32.tar.gz.sig
```

If that command fails because you do not have the required public key, run the following command to import it:

```
gpg --keyserver keys.gnupg.net --recv-keys 5981E818
```



Installing GLPK

On macOS

The easy way to install GLPK with one of these two package manager:

- ▶ homebrew
 - ▶ install homebrew: <https://brew.sh/>
 - ▶ install GLPK: <https://formulae.brew.sh/formula/glpk>
- ▶ macport
 - ▶ install macport: <https://www.macports.org/>
 - ▶ install GLPK: <https://ports.macports.org/port/glpk/>

On windows

Follow the instructions given here:

- ▶ <http://winglpk.sourceforge.net/>

On linux (ubuntu)

Follow the instructions given here:

- ▶ <https://howtoinstall.co/en/glpk>



Commercial MILP solver: Gurobi

<https://www.gurobi.com/>

The screenshot shows the Gurobi Optimization website. At the top, there's a navigation bar with links for Documentation, Downloads & Licenses, Support, Register, Login, and a Free Trial button. Below the navigation is a main heading "Optimization for the Entire Business". Underneath it, a sub-section for "Data Scientist" is highlighted. To the left of the main content area is a large portrait of a woman with blonde hair, wearing a white turtleneck sweater. The main content area is divided into five horizontal sections corresponding to different professional roles: Data Scientist, Operations Researcher, Academic, Business Professional, and Computer Scientist. The "Data Scientist" section is currently active. It contains a brief description of how optimization is becoming essential for data scientists, followed by four levels of resources: Level 1 – Introduction for Data Scientists, Level 2 – Resources for Beginners, Level 3 – Resources for Intermediate Users, and Level 4 – Resources for Advanced Users.

Installing Gurobi (academic license)

Gurobi software downloads and license center:

<https://www.gurobi.com/downloads/>

Video for

- ▶ macOS:

<https://www.youtube.com/watch?v=ZcL-NmckTxQ>

- ▶ windows:

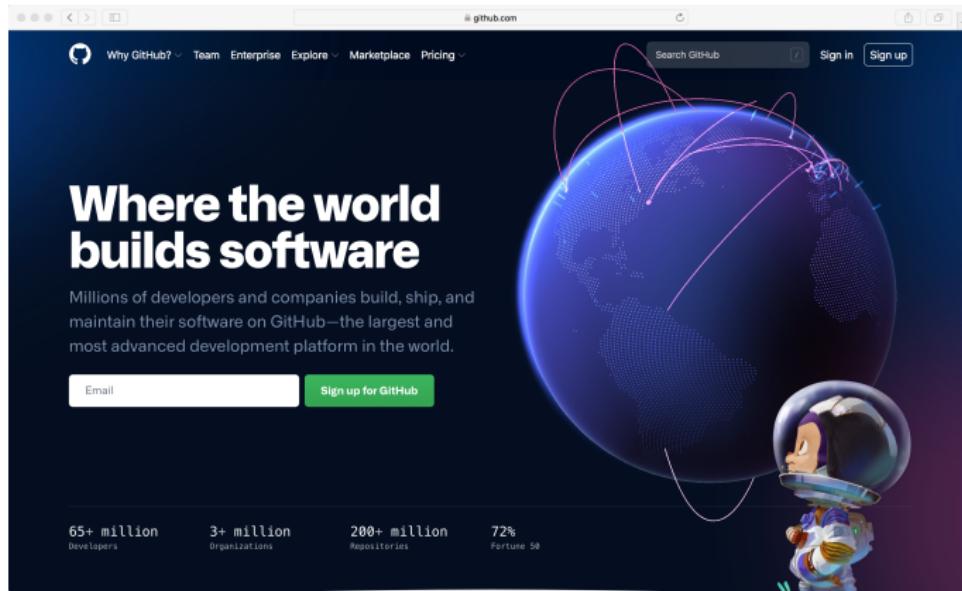
<https://www.youtube.com/watch?v=fQVxuW0iPpI&t=0s>

- ▶ linux (ubuntu):

<https://www.youtube.com/watch?v=yNmeG6Wom1o&t=0s>

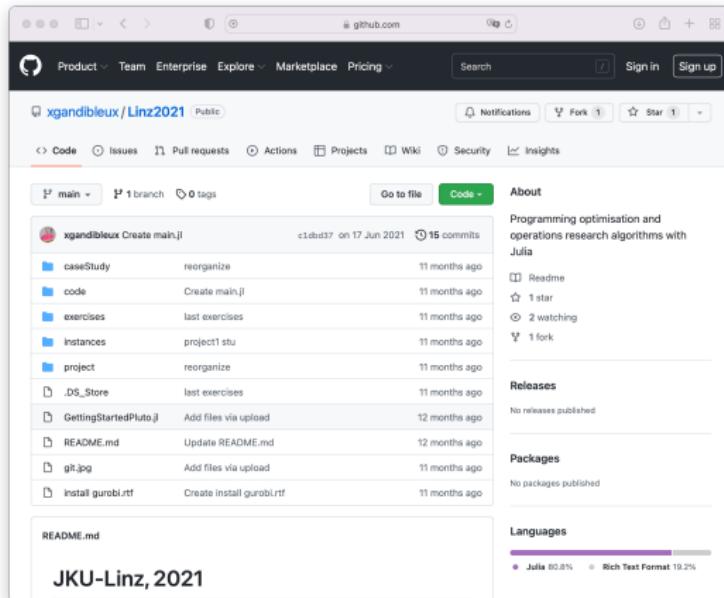
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



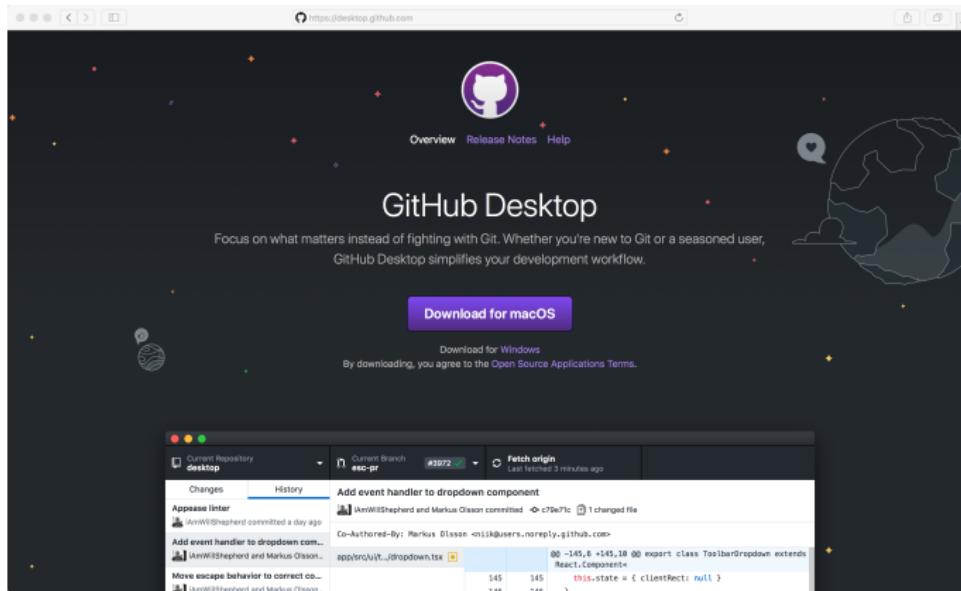
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



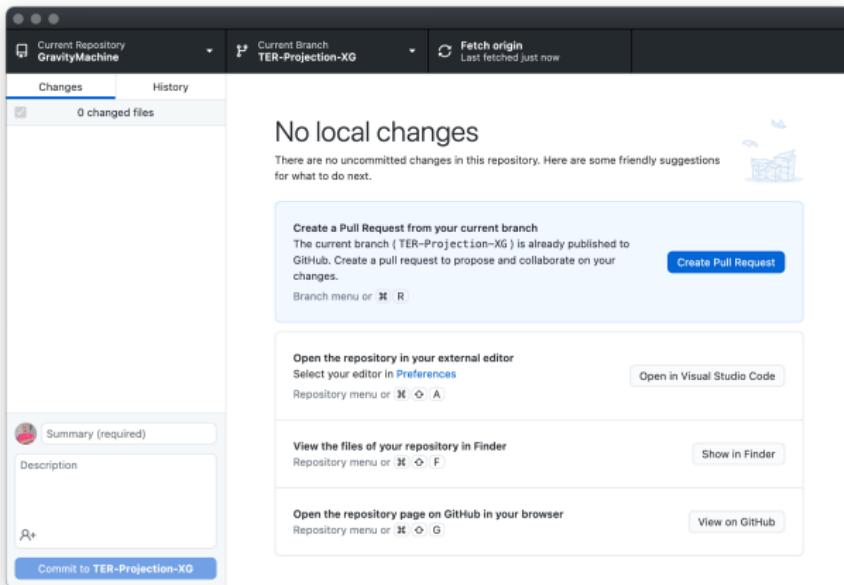
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>



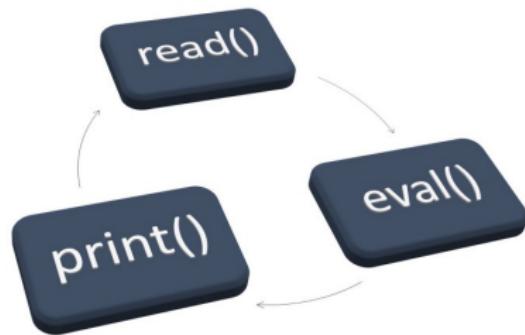
Repository: GitHub and the app GitHub Desktop

<https://github.com> and <https://desktop.github.com>





Getting started
R ead E valuate P rint L oop



REPL: normal mode

Just click on the app icon



A screenshot of a macOS application window titled "xavierg — julia — 80x24". The window contains the following text:

```
MacBook-de-xavierg-2:~ xavierg$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]?" for Pkg help.
Version 1.9.1 (2023-06-07)
Official https://julialang.org/ release
julia>
```

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program

`include("filename.jl")`

Interrupt execution

`[Ctrl] + [C]`

Recall last result

`ans`

Previous-run command

Up key

Search backwards into history

`[Ctrl] + [r]`

Tab-completion

characters + [Tab]

Clear screen

`[Ctrl] + [L]`

Exit REPL

`exit()` or `[Ctrl] + [D]`

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program

`include("filename.jl")`

Interrupt execution

`[Ctrl] + [C]`

Recall last result

`ans`

Previous-run command

Up key

Search backwards into history

`[Ctrl] + [r]`

Tab-completion

characters + [Tab]

Clear screen

`[Ctrl] + [L]`

Exit REPL

`exit()` or `[Ctrl] + [D]`

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program

`include("filename.jl")`

Interrupt execution

`[Ctrl] + [C]`

Recall last result

`ans`

Previous-run command

Up key

Search backwards into history

`[Ctrl] + [r]`

Tab-completion

characters + [Tab]

Clear screen

`[Ctrl] + [L]`

Exit REPL

`exit()` or `[Ctrl] + [D]`

REPL: normal mode (commands)

Some commands to know:

Load (and run) a program	<code>include("filename.jl")</code>
Interrupt execution	<code>[Ctrl] + [C]</code>
Recall last result	<code>ans</code>
Previous-run command	<code>Up key</code>
Search backwards into history	<code>[Ctrl] + [r]</code>
Tab-completion	<code>characters + [Tab]</code>
Clear screen	<code>[Ctrl] + [L]</code>
Exit REPL	<code>exit() or [Ctrl] + [D]</code>

REPL: help mode

```
julia> ?
```



```
xavierg — julia — 80x24
[MacBook-de-xavierg-2:~ xavierg$ julia
   Documentation: https://docs.julialang.org
   Type "?" for help, "]??" for Pkg help.
   Version 1.9.1 (2023-06-07)
   Official https://julialang.org/ release

[help?> VERSION
search: VERSION VersionNumber versioninfo pkgversion reverseind

VERSION

A VersionNumber object describing which version of Julia is in use. See also
Version Number Literals.

julia> ]
```

REPL: shell mode

julia> ;



The screenshot shows a terminal window titled "xavierg — julia — 80x24". The window contains the following text:

```
[MacBook-de-xavierg-2:~ xavierg$ julia
      Documentation: https://docs.julialang.org
      Type "?" for help, "]?" for Pkg help.
      Version 1.9.1 (2023-06-07)
      Official https://julialang.org/ release

[shell> pwd
/Users/xavierg

shell> ]
```

Return to normal mode: [Backspace] on empty line

REPL: package manager mode

julia>]



The screenshot shows a terminal window titled "xavierg — julia — 80x24". The window contains the following text:

```
[MacBook-de-xavierg-2:~ xavierg$ julia
  Documentation: https://docs.julialang.org
  Type "?" for help, "]?" for Pkg help.
  Version 1.9.1 (2023-06-07)
  Official https://julialang.org/ release
  [__]

[@v1.9] pkg> status
Status `~/julia/environments/v1.9/Project.toml'
[7073ff75] IJulia v1.24.2

[@v1.9] pkg>
```

Return to normal mode: [Backspace] on empty line

REPL: package manager mode (commands)

Some commands to know:

add package(s) to project	add
remove package(s) from project	remove or rm
summarize contents	status or st
update packages	update or up
help online	help



Getting started

Packages

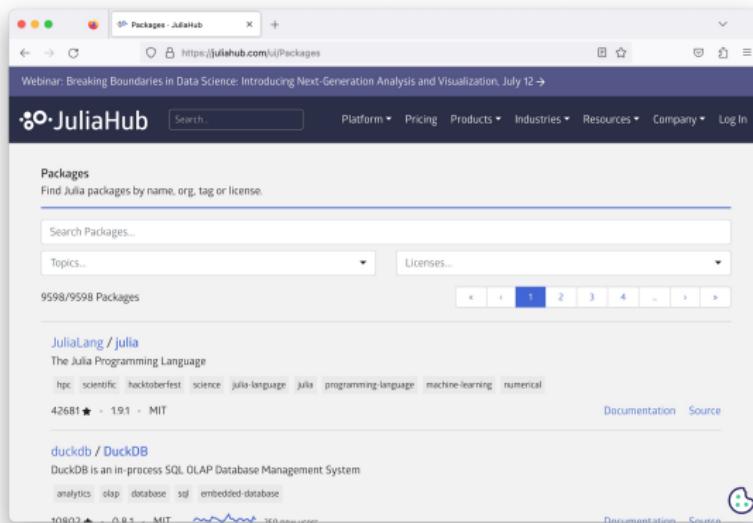


Packages

Visual interface for exploring the packages:

Julia hub

<https://juliahub.com/ui/Packages>



Alternative:

Julia Packages

<https://juliapackages.com/>



Packages

A selection of useful packages:

Base library:

Pkg	primitives for managing packages
Random	primitives for random number
LinearAlgebra	primitives for linear algebra
Printf	primitives for C-like format to display on screen

External library:

PyPlot	Plotting for Julia based on matplotlib
Plots	a high-level plotting package
JuMP	Modeling language for Mathematical Optimization
GLPK	interface for GLPK solver
Gurobi	interface for Gurobi solver
CPLEX	interface for the CPLEX solver
vOptGeneric	Multiobjective linear optimization solver; generic problems
vOptSpecific	Multiobjective linear optimization solver; specific problems

Packages

A selection of useful packages:

Base library:

Pkg	primitives for managing packages
Random	primitives for random number
LinearAlgebra	primitives for linear algebra
Printf	primitives for C-like format to display on screen

External library:

PyPlot	Plotting for Julia based on matplotlib
Plots	a high-level plotting package
JuMP	Modeling language for Mathematical Optimization
GLPK	interface for GLPK solver
Gurobi	interface for Gurobi solver
CPLEX	interface for the CPLEX solver
vOptGeneric	Multiobjective linear optimization solver; generic problems
vOptSpecific	Multiobjective linear optimization solver; specific problems

Packages

A selection of useful packages:

Base library:

Pkg	primitives for managing packages
Random	primitives for random number
LinearAlgebra	primitives for linear algebra
Printf	primitives for C-like format to display on screen

External library:

PyPlot	Plotting for Julia based on matplotlib
Plots	a high-level plotting package
JuMP	Modeling language for Mathematical Optimization
GLPK	interface for GLPK solver
Gurobi	interface for Gurobi solver
CPLEX	interface for the CPLEX solver
vOptGeneric	Multiobjective linear optimization solver; generic problems
vOptSpecific	Multiobjective linear optimization solver; specific problems

Adding a package (from the external library)

Commands to invoke **once** before the first use of the given package:

Method 1:

```
(@v1.9) pkg> add PyPlot  
(@v1.9) pkg> add JuMP,GLPK
```

Method 2:

```
julia> using Pkg  
julia> Pkg.add("PyPlot")  
julia> Pkg.add("JuMP")  
julia> Pkg.add("GLPK")
```

Adding a package (from the external library)

Commands to invoke **once** before the first use of the given package:

Method 1:

```
(@v1.9) pkg> add PyPlot  
(@v1.9) pkg> add JuMP,GLPK
```

Method 2:

```
julia> using Pkg  
julia> Pkg.add("PyPlot")  
julia> Pkg.add("JuMP")  
julia> Pkg.add("GLPK")
```

Using a package

Commands to invoke **each time** before the first use of the given package:

```
julia> using Pkg  
julia> using JuMP,GLPK  
julia> using Printf  
julia> using LinearAlgebra
```



Getting started

Let's try



Running instructions into the REPL

1) Click on the app icon...



... the REPL is ready!

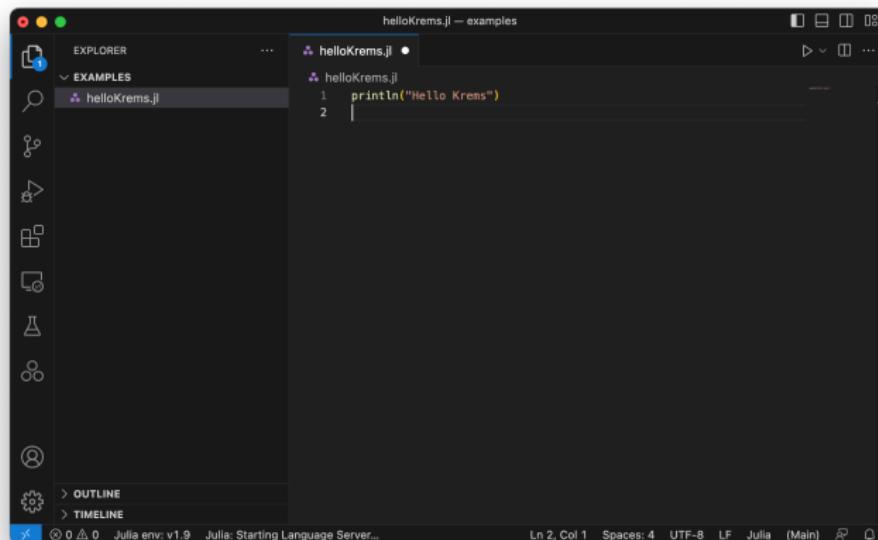
2) Type your instructions...

```
julia> println("Hello Krems")
```

... done!

Running a program into the REPL

Program prepared with a text editor (here VScode)...



...to load and execute into the REPL :

```
julia> include("hello.jl")
```

Running a program into an IDE (VScode)

Program prepared with a text editor (here VScode)...

...to load and execute into the VScode terminal:

The screenshot shows the Visual Studio Code (VScode) interface with a dark theme. On the left is the Explorer sidebar, which has a 'EXAMPLES' folder open containing a file named 'helloKrems.jl'. The main central area displays the contents of this file:

```
helloKrems.jl -- examples
helloKrems.jl •
helloKrems.jl
1 println("Hello Krems")
2
```

Below the code editor, the 'TERMINAL' tab is selected, showing the output of the program's execution:

```
Hello Krems
julia>
```

The status bar at the bottom indicates the current environment and terminal settings:

Ln 2, Col 1 Spaces: 4 UTF-8 LF Julia Main

Running Julia code into a notebook (jupyter)

Jupyter <https://jupyter.org/>

IJulia has to be installed
(<https://github.com/JuliaLang/IJulia.jl>)

Steps to follow:

```
(@v1.9) pkg> add IJulia
```

```
julia> using IJulia
julia> notebook()
```

answer yes to the questions

Done!

Running Julia code into a notebook (jupyter)

Jupyter <https://jupyter.org/>

IJulia has to be installed

(<https://github.com/JuliaLang/IJulia.jl>)

Steps to follow:

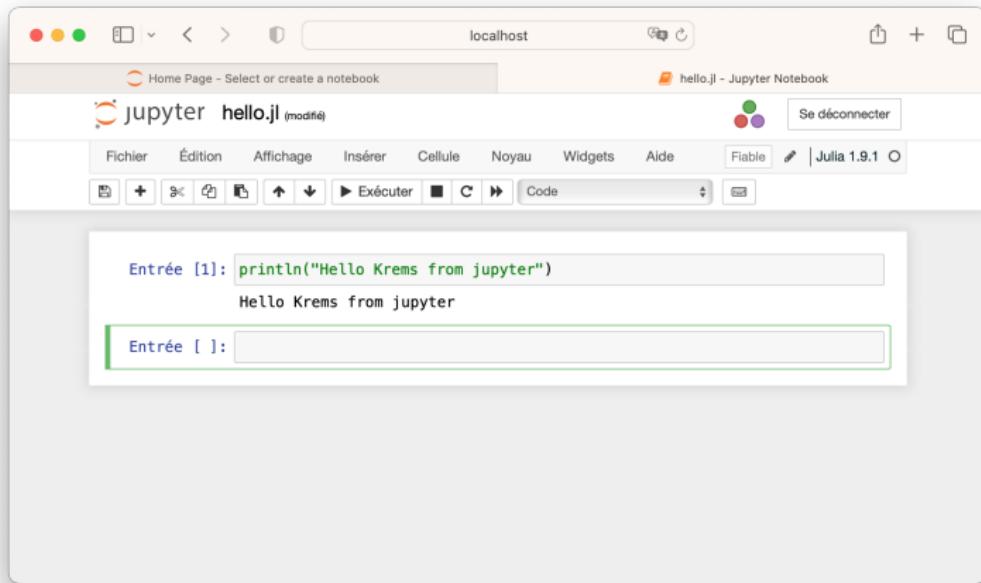
```
(@v1.9) pkg> add IJulia
```

```
julia> using IJulia  
julia> notebook()
```

answer yes to the questions

Done!

Running Julia code into a notebook (jupyter)



Running Julia code into a notebook (Pluto)

Pluto <https://github.com/fonsp/Pluto.jl>

Install: <https://www.youtube.com/watch?v=C4QhZcX34mI>

Steps to follow:

```
(@v1.9) pkg> add Pluto
```

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

Done!

Running Julia code into a notebook (Pluto)

Pluto <https://github.com/fonsp/Pluto.jl>

Install: <https://www.youtube.com/watch?v=C4QhZcX34mI>

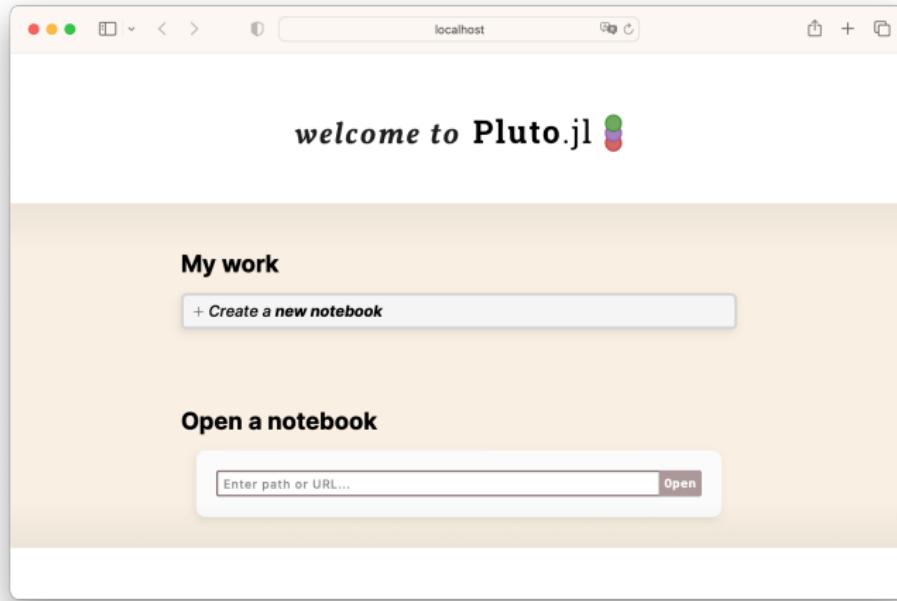
Steps to follow:

```
(@v1.9) pkg> add Pluto
```

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

Done!

Running Julia code into a notebook (Pluto)



Running Julia code into a notebook (2)

The screenshot shows a web-based Julia notebook interface. At the top, there's a toolbar with standard window controls (red, yellow, green buttons), a dropdown menu, and a search bar labeled "localhost". Below the toolbar, the title "Pluto.jl" is displayed next to a logo, with a "Save notebook..." button to its right. A "File" menu icon is also present. The main area contains a code cell with the following content:

```
+ println("Hello Krems from Pluto") |
```

Below the code cell, the output is shown in a dark box:

```
Hello Krems from Pluto
```

Next to the output box is a small circular icon with a "P" and a timer icon indicating the execution time: "19.8 ms". At the bottom of the interface, there are navigation buttons for "FAQ", "Live Docs", and "Status". On the far left, there's a "Send" button. In the bottom right corner, there's a feedback section with a "How can we make Pluto.jl better?" input field, an "Instant feedback..." button, and a "Send" button.



Let's go for installing the apps and do your first runs

