

Research software development workflows in Julia, applied to JuMP

Frederik Geth, Rahmat Heidarihaei, James Foster

frederik.geth@csiro.au

 frederikgeth 

Solutions Showcases – Friday 23 October

workflow

- set up julia + VSCode

initialize package

develop within
package environment

write unit test

version control and
continuous integration

Why Julia? - Speed!



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

Why Julia?

- free and open source (MIT licensed) designed for scientific / technical computing
- not needed to vectorise code for performance; unvectorised code can also be fast
- designed for parallelism and distributed computation
- efficient support for Unicode, including but not limited to UTF-8
- you can finally use emojis as variables

The screenshot shows a web browser displaying an article from Ars Technica. The title of the article is "The unreasonable effectiveness of the Julia programming language". Below the title, it says "Fortran has ruled scientific computing, but Julia emerged for large-scale numerical work." The author is LEE PHILLIPS, dated 10/9/2020, 10:15 PM. The article features a large image of a virtual conference interface for JuliaCon 2020, which includes a video player for a keynote by Karen Willcox and a sidebar with various sponsors like Moore Foundation, Julia Computing, Zapata, and Vercel. The text below the image reads "Ain't no party like a programming language virtual conference party". The article continues with a paragraph about scientists running into each other online due to canceled in-person events, followed by a section titled "FURTHER READING" with a link to another Ars Technica article about scientific computing's future.

ars TECHNICA

LOOKS LIKE MATH —

The unreasonable effectiveness of the Julia programming language

Fortran has ruled scientific computing, but Julia emerged for large-scale numerical work.

LEE PHILLIPS - 10/9/2020, 10:15 PM

JuliaCon 2020 | Keynote: Scientific Machine Learning | Prof Karen Willcox

Later bekij... Delen

Keynote: Karen Willcox

Karen Willcox

Ain't no party like a programming language virtual conference party

I've been running into a lot of happy and excited scientists lately. "Running into" in the virtual sense, of course, as conferences and other opportunities to collide with scientists in meatspace have been all but eliminated. Most scientists believe in the germ theory of disease.

Anyway, these scientists and mathematicians are excited about a new tool. It's not a [new particle](#) accelerator nor a [supercomputer](#). Instead, this exciting new tool for scientific research is... a computer language.

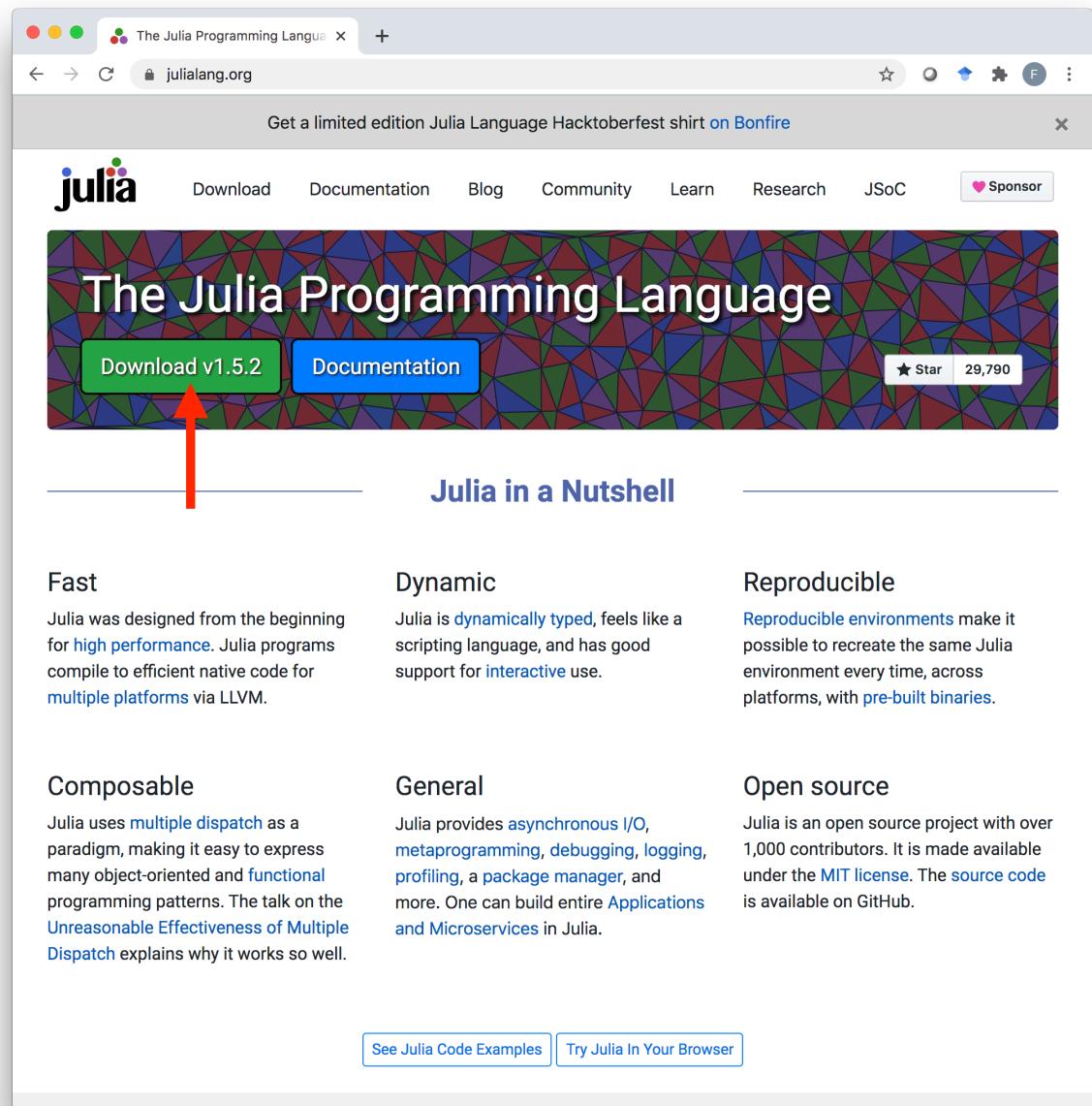
How can a computer language be exciting, you ask? Surely, some are better than others, depending on your purposes and priorities. Some run faster, while others are quicker and easier to develop in. Some have a

FURTHER READING

Scientific computing's future: Can any coding language top a 1950s

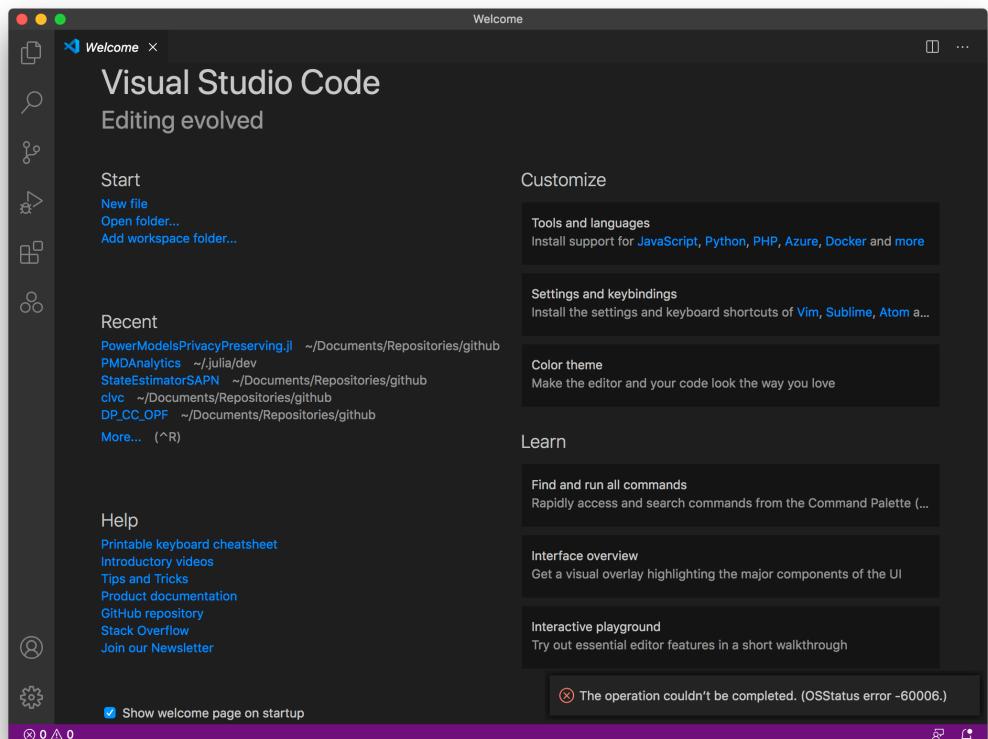
Get Julia

- <https://julialang.org/>
- 1.5.2 (stable)
- mac, windows, linux
- cheatsheets.quantecon.org



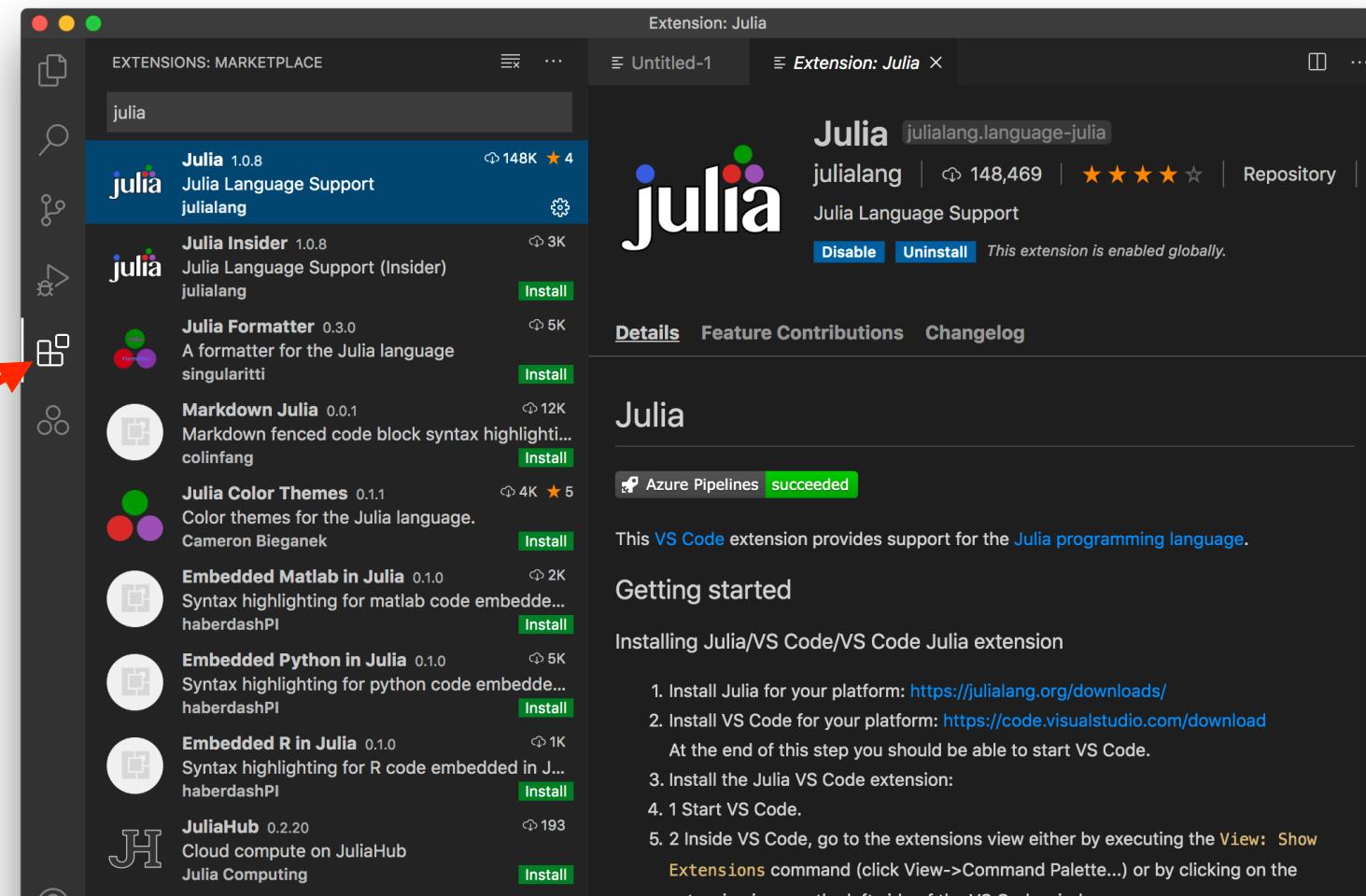
Visual Studio Code

- <https://code.visualstudio.com/>
 - Mac, Windows and Linux



Julia Plugin for VSCode

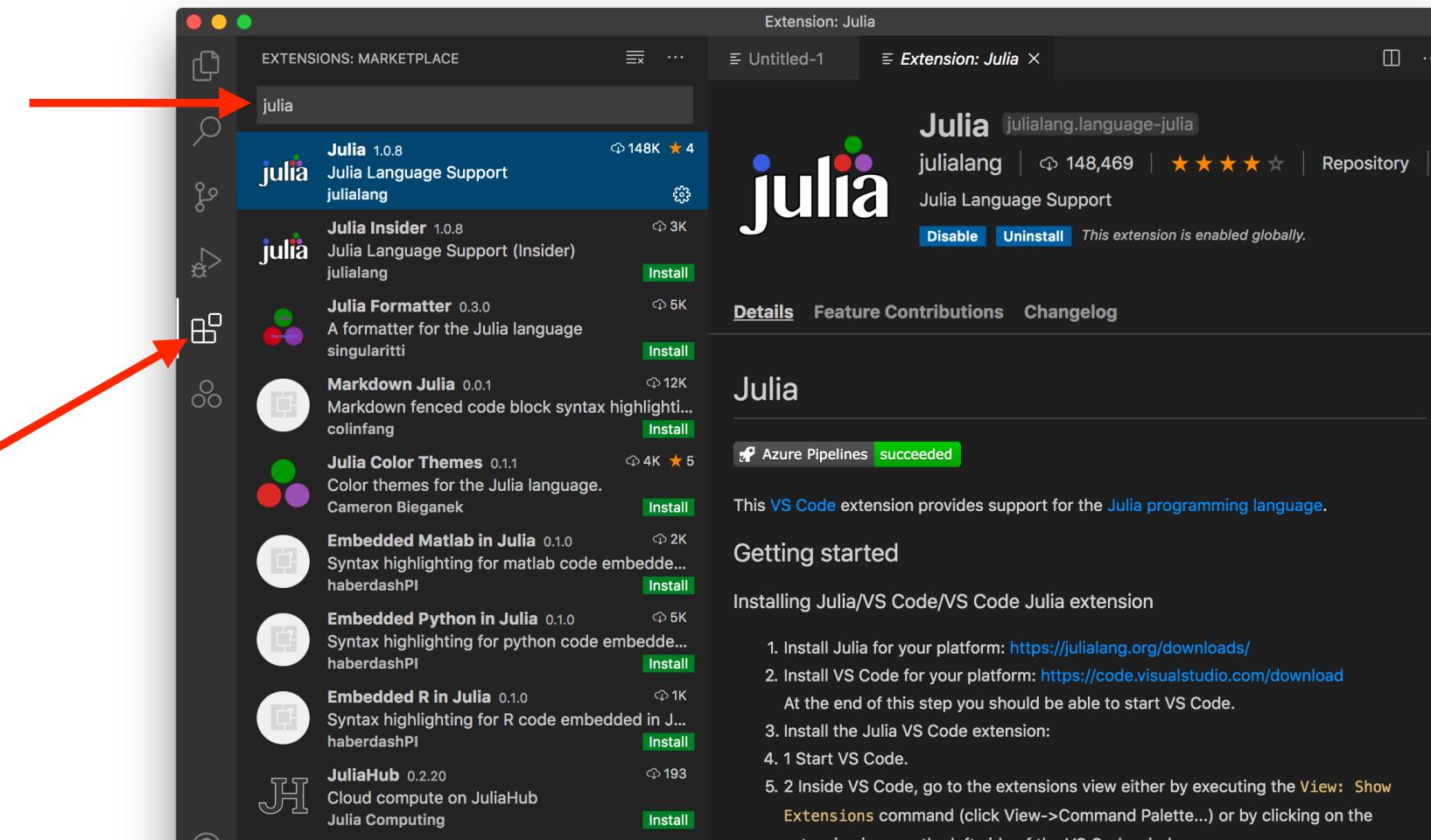
Extensions



Julia Plugin for VSCode

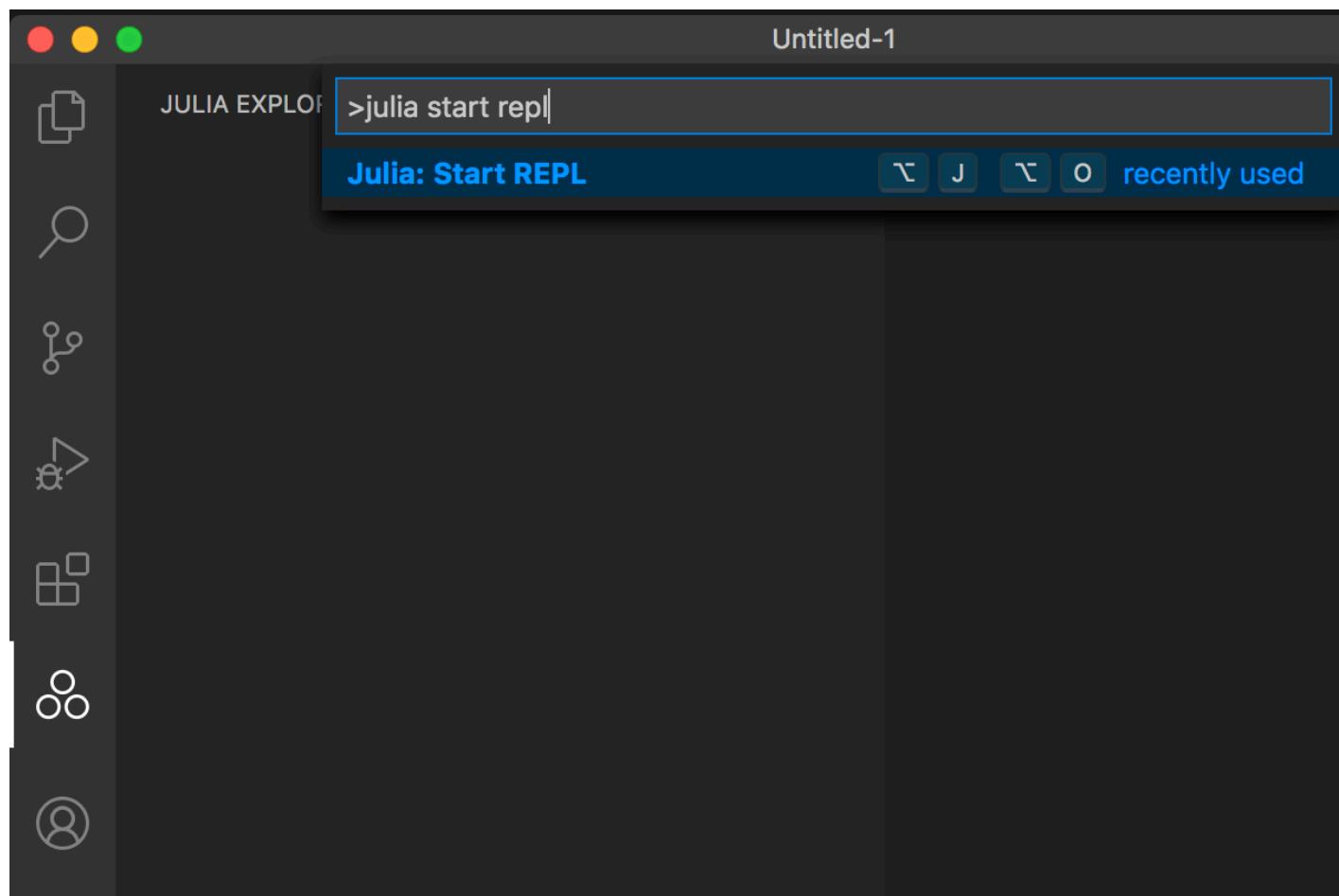
type in ‘Julia’
and install

Extensions



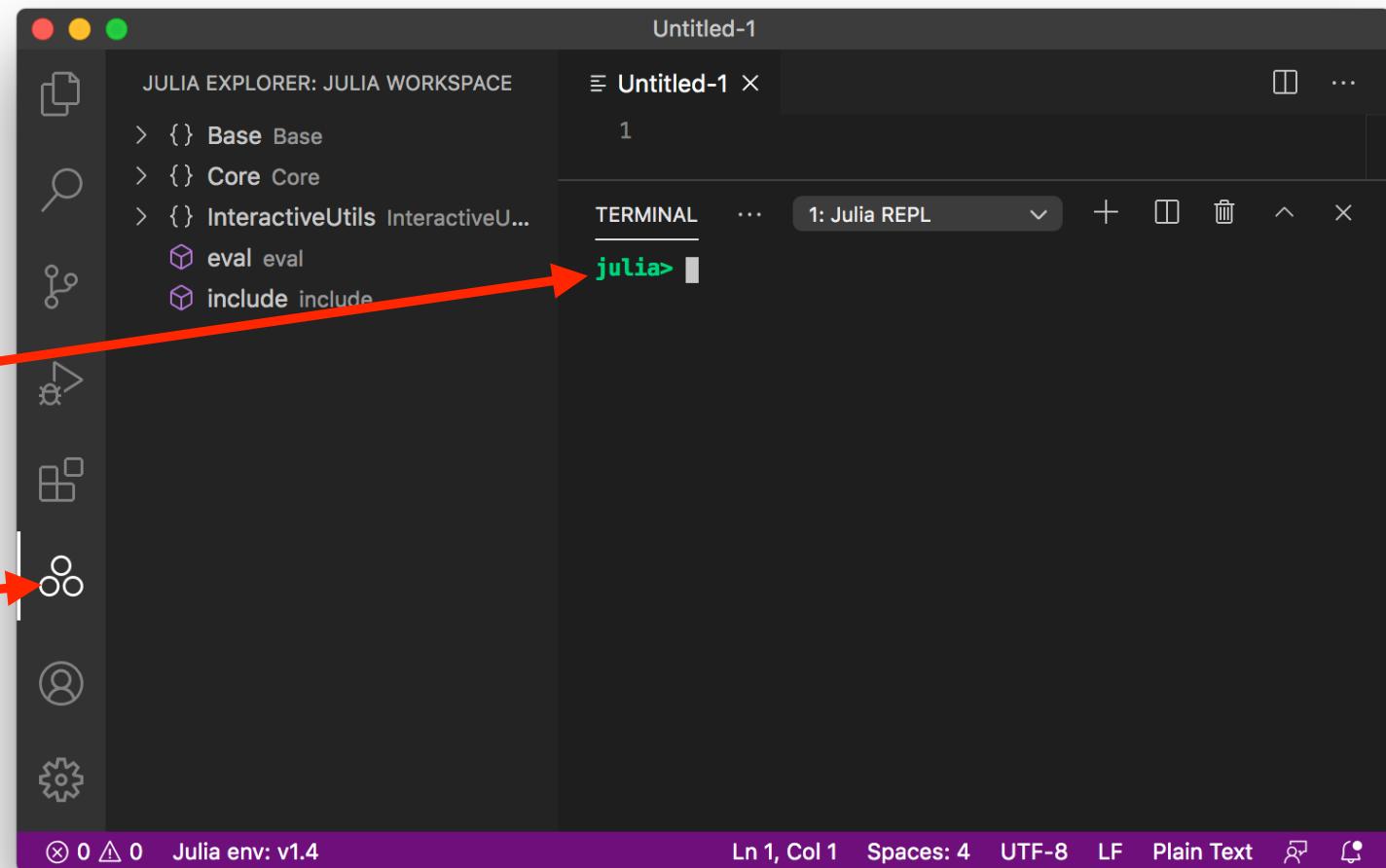
Open REPL in VSCode

- view -> command palette -> type 'julia start REPL' -> enter
 - or alt (option) + j,
alt (option) + o



REPL

- read-eval-print loop
- interactive ‘interpreter’
- with workspace (variable) browser



JULIA EXPLORER: JULIA WORKSPACE

```
> {} Base Base
> {} Core Core
> {} InteractiveUtils InteractiveU...
> [ ] a Array{Int64,1} with 3 eleme...
> [ ] ans Array{Int64,1} with 3 ele...
    eval eval
    include include
> [ ] x Array{Int64,1} with 3 eleme...
```

Untitled-1

TERMINAL ...

1: Julia REPL

```
julia> 2+2
4

julia> pi
π = 3.1415926535897...

julia> a = [1,2,3]
3-element Array{Int64,1}:
 1
 2
 3

julia> x = a
3-element Array{Int64,1}:
 1
 2
 3

julia> a[3]=5
5

julia> x
3-element Array{Int64,1}:
 1
 2
 5

julia> 
```

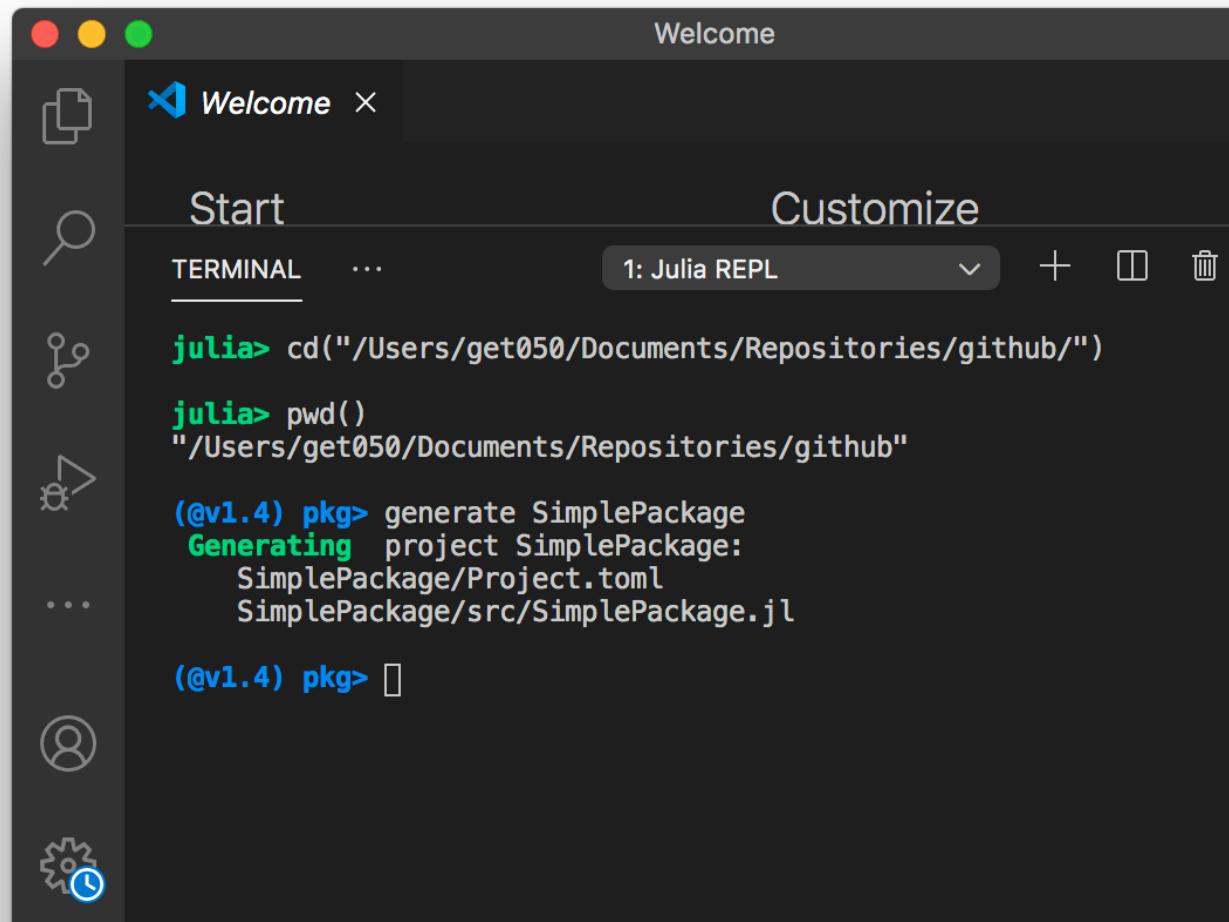
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text ⌂ ⌂

Package setup

- package initialisation workflow
- can be done many different ways, what I present is one of them
- resources:
 - <https://github.com/invenia/PkgTemplates.jl>
 - Developing Julia Packages | Chris Rackauckas: <https://www.youtube.com/watch?v=QVmU29rCjaA>
 - Pkg, Project.toml, Manifest.toml and Environments | Fredrik Ekre: <https://www.youtube.com/watch?v=q-LV4zoxc-E>
 - Using VS Code for Julia development | David Anthoff: <https://www.youtube.com/watch?v=ldhnP00Y1Ks>

generate new package

- navigate to folder where you want to generate a new Package

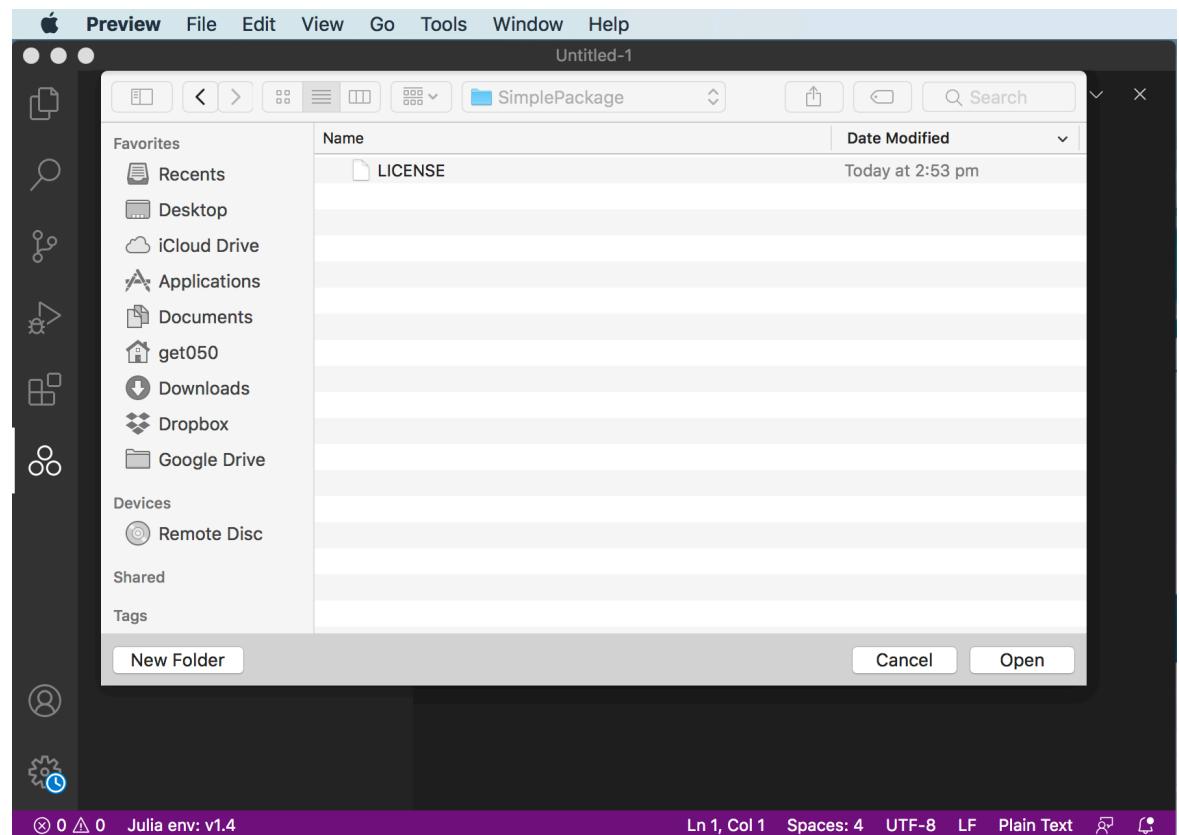


The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with a terminal window open. The terminal title bar says "Welcome" and the main menu bar has "Start" and "Customize". On the left, there's a vertical sidebar with icons for file operations, search, connections, and more. The terminal itself displays the following text:

```
julia> cd("/Users/get050/Documents/Repositories/github/")
julia> pwd()
"/Users/get050/Documents/Repositories/github"
(@v1.4) pkg> generate SimplePackage
Generating project SimplePackage:
SimplePackage/Project.toml
SimplePackage/src/SimplePackage.jl
(@v1.4) pkg> 
```

open folder in VSCode

- file -> open (folder)
- point to ‘SimplePackage’



hello world

- /src contains the package code
- other folders are optional
 - we store a script that uses our module in /script/useSimplePackage.jl

The screenshot shows a Julia development environment with the following structure and code:

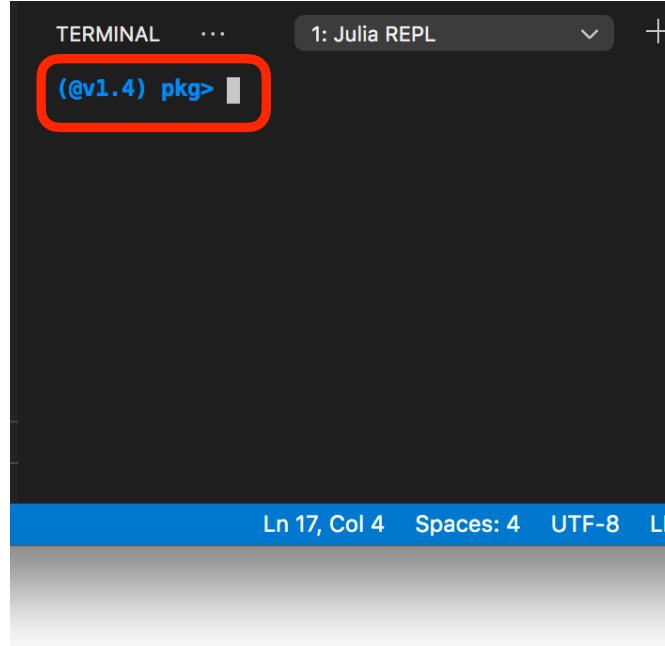
- EXPLORER:** Shows the project tree:
 - SimplePackage
 - script
 - useSimplePackage.jl
 - src
 - SimplePackage.jl
 - Manifest.toml
 - Project.toml
- EDITOR:** The file `useSimplePackage.jl` contains the following code:

```
module SimplePackage
greet() = print("Hello World!")
end # module
```
- TERMINAL:** The terminal shows the command to activate the environment:

```
(@v1.4) pkg> activate .
Activating environment at `~/Documents/Repositories/github/SimplePackage/Project.toml`
```

package manager

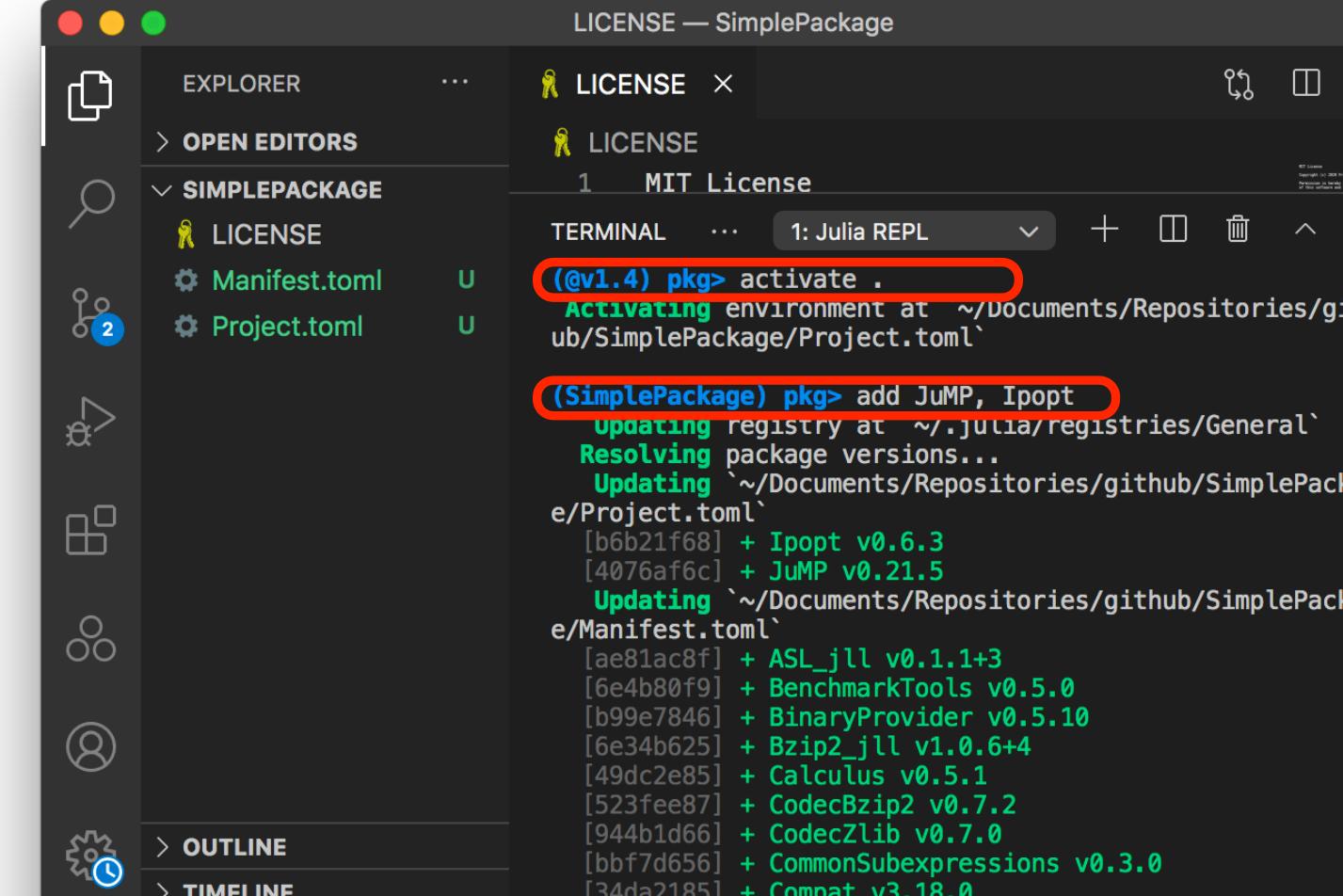
- enter package mode by typing
‘]’ in REPL
 - press backspace to leave
- <https://docs.julialang.org/en/v1/stdlib/Pkg/>



The screenshot shows a terminal window titled "TERMINAL" with a tab labeled "1: Julia REPL". The prompt in the terminal is "@v1.4) pkg>". A red rectangular box highlights the entire prompt area. At the bottom of the terminal window, there is a status bar with the text "Ln 17, Col 4 Spaces: 4 UTF-8 L".

Make environment

- activate environment in current folder
- add ‘JuMP’ and ‘Ipopt’ packages



LICENSE — SimplePackage

TERMINAL 1: Julia REPL

```
(@v1.4) pkg> activate .
Activating environment at ~/Documents/Repositories/github/SimplePackage/Project.toml

(SimplePackage) pkg> add JuMP, Ipopt
Updating registry at ~/.julia/registries/General
Resolving package versions...
Updating `~/Documents/Repositories/github/SimplePackage/Project.toml'
[b6b21f68] + Ipopt v0.6.3
[4076af6c] + JuMP v0.21.5
Updating `~/Documents/Repositories/github/SimplePackage/Manifest.toml'
[ae81ac8f] + ASL_jll v0.1.1+3
[6e4b80f9] + BenchmarkTools v0.5.0
[b99e7846] + BinaryProvider v0.5.10
[6e34b625] + Bzip2_jll v1.0.6+4
[49dc2e85] + Calculus v0.5.1
[523fee87] + CodecBzip2 v0.7.2
[944b1d66] + CodecZlib v0.7.0
[bbf7d656] + CommonSubexpressions v0.3.0
[34da2185] + Compat v3.18.0
```

JuMP example

- example from <https://jump.dev/JuMP.jl/stable/nlp/>
 - Ipopt and JuMP as dependencies

The screenshot shows a Julia development environment with two code editors and a terminal window.

Left Editor: The file `useSimplePackage.jl` contains the following JuMP code:

```
module SimplePackage
    import Ipopt
    import JuMP

    function Rosenbrock()
        model = JuMP.Model(Ipopt.Optimizer)
        JuMP.@variable(model, x, start = 0.0)
        JuMP.@variable(model, y, start = 0.0)

        JuMP.@NLobjective(model, Min, (1 - x)^2 + 100 * (y - x^2)^2)

        JuMP.optimize!(model)
        println("x = ", JuMP.value(x), " y = ", JuMP.value(y))
        return model
    end

end # module
```

Right Editor: The file `useSimplePackage.jl` contains:

```
using SimplePackage
model = SimplePackage.Rosenbrock()
print(model)
```

Terminal: The terminal window shows the output of the script execution:

```
1: Julia REPL
Complementarity.....: 0.00000000000000e+00 0.00000000000000e+00
Overall NLP error....: 2.0183854587685121e-13 2.0183854587685121e-13

Number of objective function evaluations      = 36
Number of objective gradient evaluations     = 15
Number of equality constraint evaluations   = 0
Number of inequality constraint evaluations = 0
Number of equality constraint Jacobian evals = 0
Number of inequality constraint Jacobian evals = 0
Number of Lagrangian Hessian evaluations     = 14
Total CPU secs in IPOPT (w/o function evals) =      2.081
Total CPU secs in NLP function evals        =      1.098

EXIT: Optimal Solution Found.
x = 0.999999999999899 y = 0.999999999999792
Min (1.0 - x) ^ 2.0 + 100.0 * (y - x ^ 2.0) ^ 2.0
Subject to
```

The terminal prompt is `julia>`.

Testing

- add /test/
runtests.jl
- add ‘Test’ as
dependency for
testing
-] test
SimplePackage
 - triggers
runtests.jl

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists files and folders: SimplePackage, runtests.jl (selected), useSimplePackage.jl, Manifest.toml, and Project.toml. The Code Editor shows the contents of runtests.jl and Project.toml. A red box highlights the [extras] and [targets] sections in Project.toml.

```
runtests.jl — SimplePackage
1 using SimplePackage, JuMP, Test
2 model = SimplePackage.Rosenbrock()
3
4 @test isapprox(value(model[:x]), 1.0)
5 @test isapprox(value(model[:y]), 1.0)
6

Project.toml
1 name = "SimplePackage"
2 uuid = "44273272-edb4-4697-817a-9eedfa9aba6d"
3 authors = ["Geth, Frederik (Energy, Newcastle) <"]
4 version = "0.1.0"
5
6 [deps]
7 Ipopt = "b6b21f68-93f8-5de0-b562-5493be1d77c9"
8 JuMP = "4076af6c-e467-56ae-b986-b466b2749572"
9
10 [extras]
11 Test = "8dfed614-e22c-5e08-85e1-65c5234f0b40"
12
13 [targets]
14 test = ["Test"]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Julia REPL + - ×
Constraint violation.....: 0.000000000000000e+00 0.000000000000000e+00
Complementarity.....: 0.000000000000000e+00 0.000000000000000e+00
Overall NLP error.....: 2.0183854587685121e-13 2.0183854587685121e-13

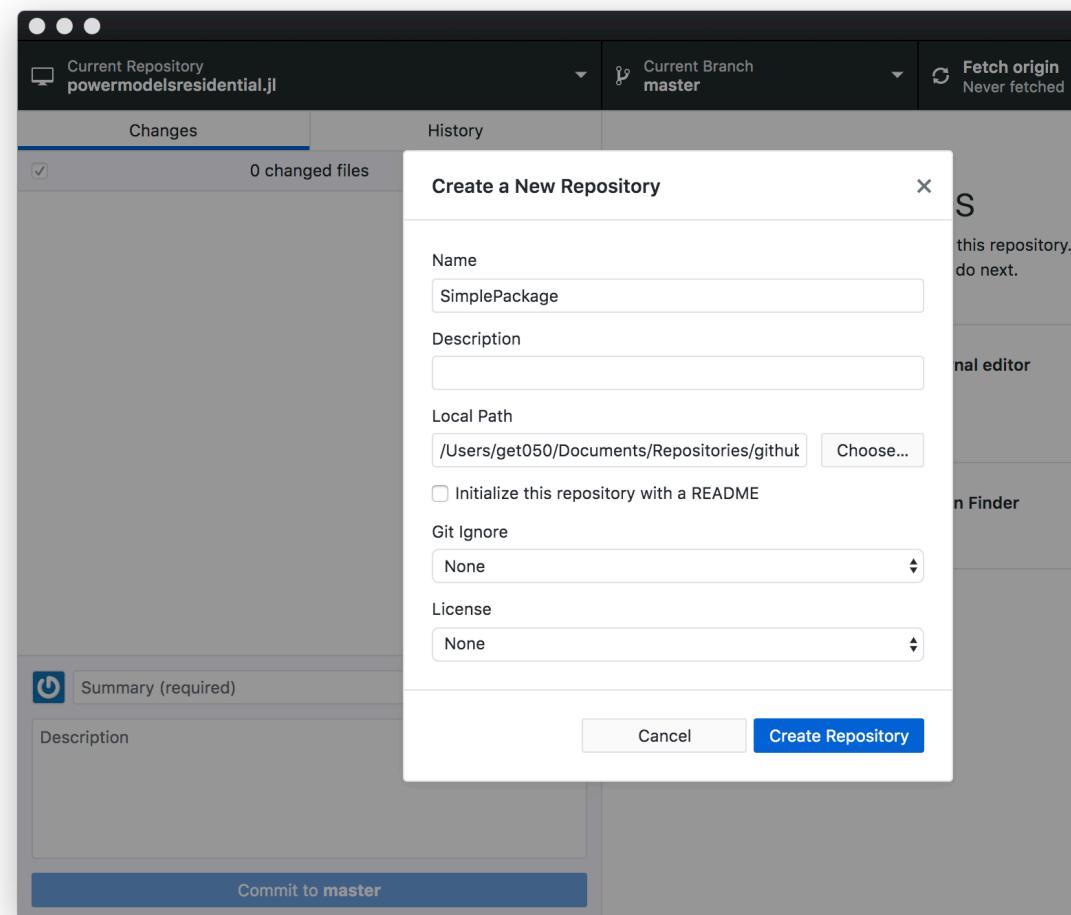
Number of objective function evaluations = 36
Number of objective gradient evaluations = 15
Number of equality constraint evaluations = 0
Number of inequality constraint evaluations = 0
Number of equality constraint Jacobian evaluations = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations = 14
Total CPU secs in IPOPT (w/o function evaluations) = 1.970
Total CPU secs in NLP function evaluations = 1.056

EXIT: Optimal Solution Found.
x = 0.999999999999899 y = 0.999999999999792
Testing SimplePackage tests passed
(SimplePackage) pkg> test SimplePackage
```

Ln 5, Col 37 Spaces: 4 UTF-8 LF Julia Main

Initialize repository

- here Github Desktop
 - File-> New Repository
 - point to ‘SimplePackage’ folder
 - can also use Source Tree, command line



CI

- add config file for Github Actions
 - run tests on the package
 - free if open-source

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure under "SIMPLEPACKAGE". The ".github/workflows" folder contains a file named "main.yml". Both the folder and the file are highlighted with red boxes.
- Editor View:** The file "main.yml" is open. The title bar says "main.yml — SimplePackage". The code content is a GitHub Actions workflow template:

```
.github > workflows > ! main.yml
1 # This is a Julia CI workflow template sourced here:
2 # https://github.com/invenia/PkgTemplates.jl/blob/master/test/fixtures
3 name: CI
4 # Triggers to initiate build process, can modify if we need to
5 on:
6   - push
7   - pull_request
8 jobs:
9   test:
10     name: Julia ${{ matrix.version }} - ${{ matrix.os }} - ${{ matrix.
11       runs-on: ${{ matrix.os }} }
12       strategy:
13         matrix:
14           version:
15             - '1.0'
16             - '1.5'
17             - 'nightly'
18           os:
19             - ubuntu-latest
20             - macOS-latest
21             - windows-latest
22           arch:
23             - x64
24         steps:
25           - uses: actions/checkout@v2
26           - uses: julia-actions/setup-julia@v1
27             with:
28               version: ${{ matrix.version }}
29               arch: ${{ matrix.arch }}
30             - uses: julia-actions/julia-buildpkg@latest
31             - uses: julia-actions/julia-runttest@latest
```

The status bar at the bottom indicates the file is at "Ln 13, Col 14" with "Spaces: 2" and encoding "UTF-8". It also shows "LF" and "YAML" as the current file type.

Actions · frederikgeth/SimplePackage

sch ATTEST | Home CIM100 AeRO frederikgeth (Fred...) OEP

frederikgeth / SimplePackage

Code Issues Pull requests Actions Projects Wiki ...

All workflows New

 Tell us how to make GitHub Actions work better for you with three quick questions.

Give feedback

Filter workflows

1 result

Event Status Branch Actor

✓ **Initial commit**
CI #1: Commit 0cd0486 pushed by frederikgeth
1 hour ago 3m 56s master

Initial commit · frederikgeth/Sir x +

github.com/frederikgeth/SimplePackage/actions/runs/309771831

sch ATTEST | Home CIM100 AeRO frederikgeth (Fred...) OEP model.energy Corporate Gibberi... DS DS

Search or jump to... / Pull requests Issues Marketplace Explore

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights

Initial commit
master Geth, Frederik (Energy, Newcastle) ~ 0cd0486 Re-run jobs

CI on: push

Julia 1.5 - ubuntu-latest - x64

This run Workflow file

1 completed job in 3m 56s Ran 1 hour ago

Artifacts

No artifacts

No artifacts were generated.

Annotations

No annotations

No annotations were created.

Questions?

Frederik Geth, Rahmat Heidarihaei, James Foster

frederik.geth@csiro.au

 frederikgeth 

Solutions Showcases – Friday 23 October