Comparing general equilibrium modeling in Julia and GAMS: An example using CSAVE

Y.-H. Henry Chen

Research Scientist, MIT CS3

January 23, 2026

Center for
Sustainability Science
and Strategy

Message from GAMS…

“The reason for GAMS superior performance in [having the shorter model generation time of] this example [IJKLM model] is the use of relational algebra… to process complex queries efficiently.”

— Broihan (2023), GAMS Corporation

Response from Julia…

"… it is not difficult to address it [the bottle neck of Broihan's benchmark Julia model], given that general-purpose languages like Julia and Python have libraries specialized for this task."

— M. Lubin, O. Dowson, J. D. Garcia, J. Huchette, B. Legat (2023), JuMP developers

Questions

- What does a CGE written in Julia look like?
- Which language is faster in running CSAVE?
- How coding strategies may affect Julia’s solving time?

Scope

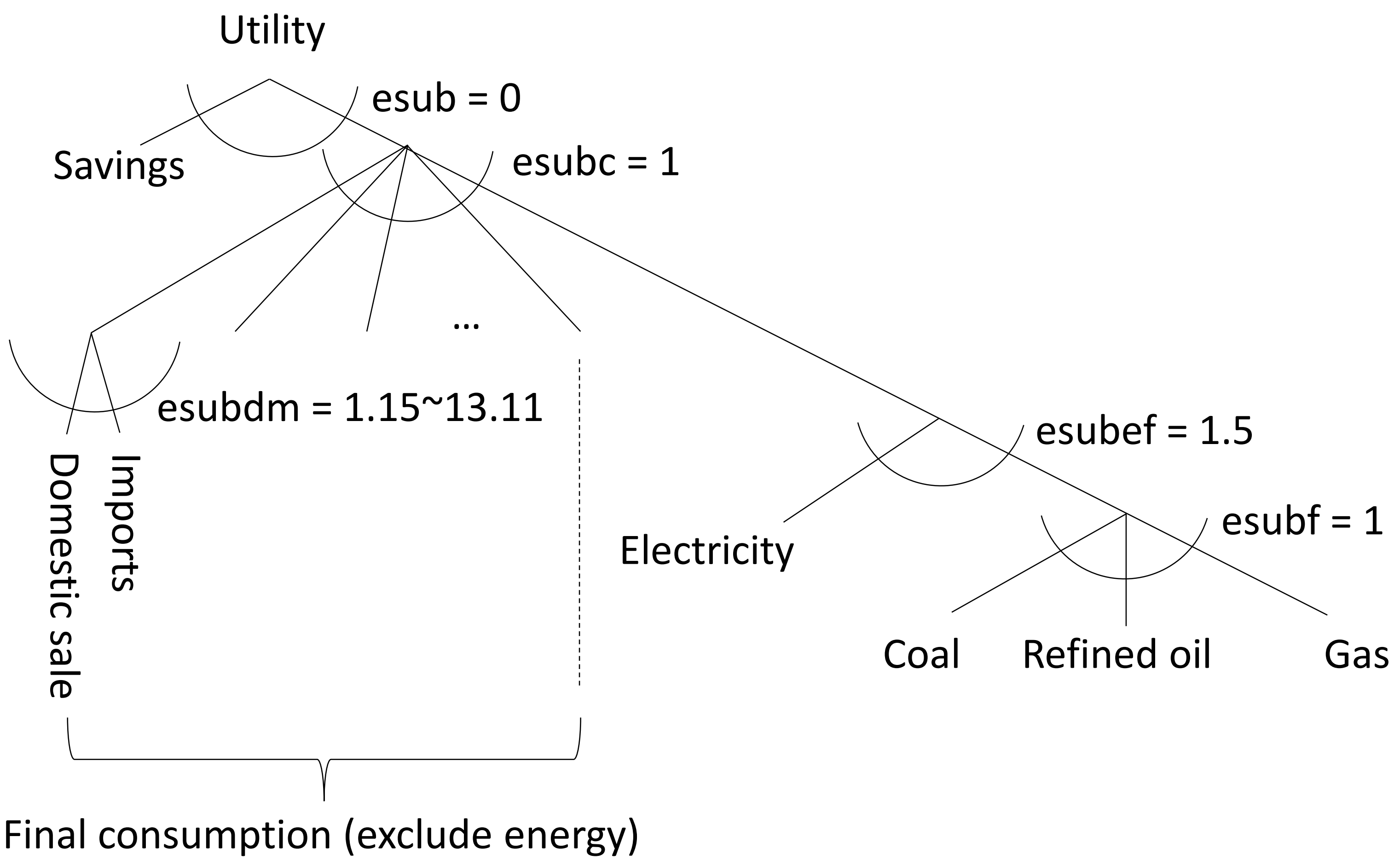
- An exercise based on CSAVE (Chen and Paltsev, 2025): in Julia vs. in GAMS
- Model generation & solving time may vary across models & computers
- Technical details/coding tricks are left for a modeling workshop

Model

- CSAVE: Chen & Paltsev (2025)
  - Motivated by EPPA
  - Derived from GTAPinGAMS
  - In this exercise: multi-sector & multi-region — ↑ resolution for “stress testing”
  - GTAP9 database

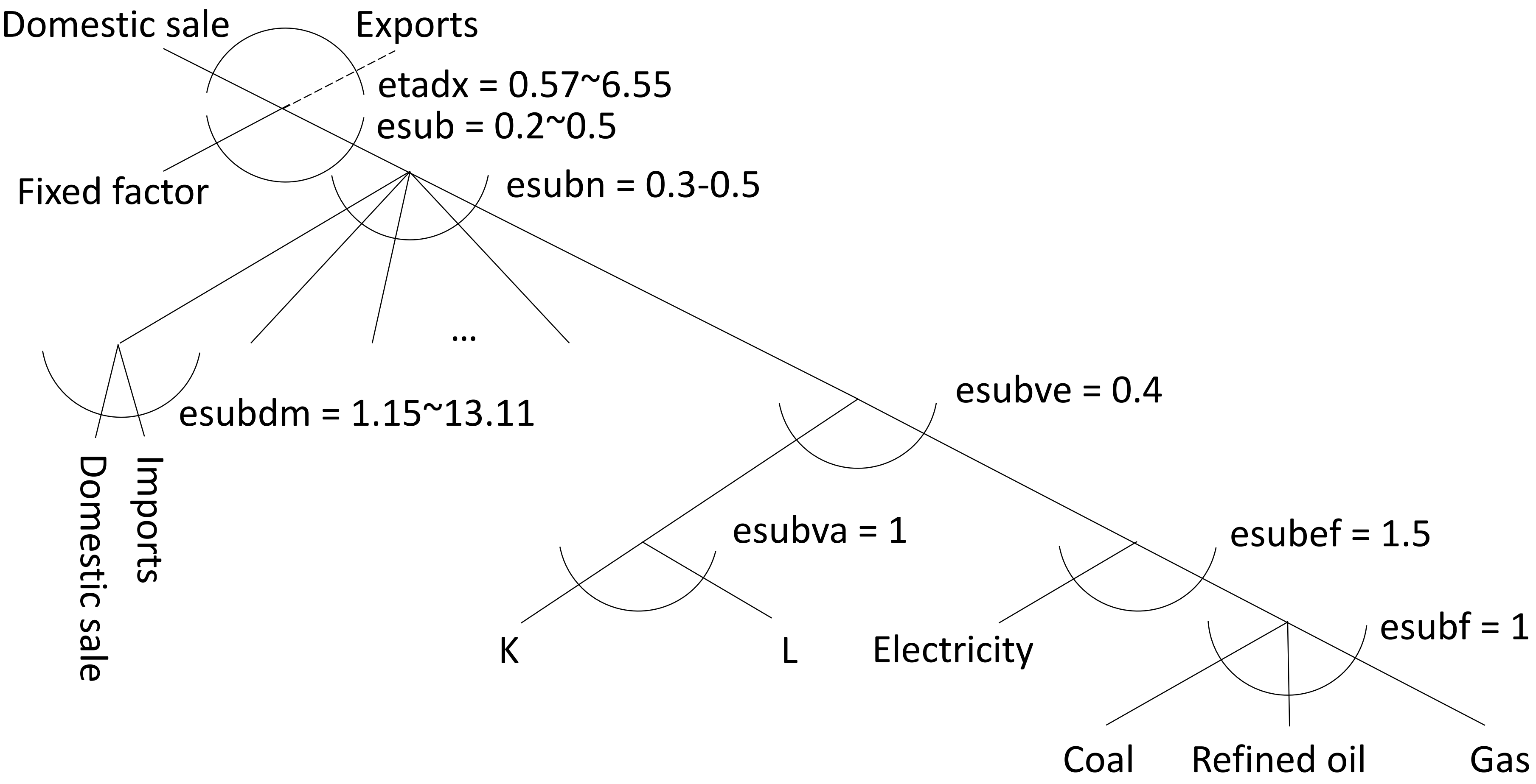
Model

Expenditure function



Model

Cost function for a production sector



Julia vs. GAMS

- Julia
  - Developed by J. Bezanson, S. Karpinski, V. B. Shah, A. Edelman in 2012
  - Open source/free
  - General purpose w/ packages for extension
  - MPSGE.jl is a package of Julia
  - Free PATH license until 12/31/2026 for now
- GAMS
  - Developed by Alex Meeraus at the World Bank in the 1970s
  - Proprietary
  - DSML for optimization problems
  - MPSGE is a sub-system of GAMS
  - Access PATH via GAMS license
- MPSGE.jl
  - D. Anthoff, E. Lazarus, M. Phillipson
- MPSGE
  - Tom Rutherford

Coding in Julia

- Packages used directly include

- I/O & data processing
  - CSV
  - Dataframes
  - JLD2
  - *CSVtoDIC*
  - *GTAP9data*
- Model construction and solving
  - JuMP
  - MPSGE
  - PATHSolver
- Generating figures
  - StatsBase
  - StatsPlots
  - Plots
  - PyPlot
  - Measures
  - Distributions
- Time measurement
  - BenchmarkTools

Coding in Julia

- CSAVEinJulia is packaged and defines its own environment
  - Project.toml — names & identities of the direct dependencies
  - Manifest.toml — the dependency graph, dependency version, where to load

Project.toml

```
[deps]
CSV = "336ed68f-0bac-5ca0-87d4-7b16caf5d00b"
DataFrames = "a93c6f00-e57d-5684-b7b6-d8193f3e46c0"
GTAPdata = "130c15d4-ad42-4bd4-91d3-6787eb393e58"
Ipopt = "b6b21f68-93f8-5de0-b562-5493be1d77c9"
JLD2 = "033835bb-8acc-5ee8-8aae-3f567f8a3819"
JuMP = "4076af6c-e467-56ae-b986-b466b2749572"
MPSGE = "d5dc2f44-7ae2-49e9-bc77-b47b6bca565d"
```

CSAVEinJulia is available on https://github.com/chenyhmitedu/

Manifest.toml

Pin package "GTAPdata" to a specific commit

```
[[deps.GTAPdata]]
deps = ["CSV", "CSVtoDIC", "DataFrames", "JLD2"]
git-tree-sha1 = "d5cb8e98efe4a2d2de8cfabcb153f6ba5ac84558"
repo-rev = "master"
repo-url = "https://github.com/chenyhmitedu/GTAPdata"
uuid = "130c15d4-ad42-4bd4-91d3-6787eb393e58"
version = "0.1.0"
```

Coding in Julia

GAMS' set operation is much simpler and cleaner:

```
vtw(j)       = sum(r, vst(j,r));

vafm(i,g,r) = vdfm(i,g,r)*(1+rtfd(i,g,r))+vifm(i,g,r)*(1+rtfi(i,g,r));
```

Julia uses dictionary (key-value mapping) to accomplish the same task:

```
vtw         = Dict(
                j => sum(vst[(j, r)] for r ∈ set_r)
                for j ∈ set_i
                )

vafm        = Dict(
                (i, g, r) => vdfm[(i, g, r)]*(1+rtfd0[(i, g, r)])+vifm[(i, g, r)]*(1+rtfi0[(i, g, r)])
                for i ∈ set_i, g ∈ set_g, r ∈ set_r
                )
```

Center for
Sustainability Science
and Strategy

Coding in Julia

GAMS/MPSGE

```
$sectors:
y(g,r)$vom(g,r)                       ! Supply
   m(i,r)$vim(i,r)                    ! Imports
   yt(j)$vtw(j)                       ! Transportation services
   E(i,s,r)$vxmd(i,s,r)               ! Exports
   A(i,g,r)$vafm(i,g,r)               ! Armington good i used k


$commodities:
   p(g,r)$vom(g,r)                    ! Domestic output price
   pm(j,r)$vim(j,r)                   ! Import price
   pt(j)$vtw(j)                       ! Transportation services
   pf(f,r)$(evom(f,r)$mf(f))          ! Primary factors rent
   ps(f,g,r)$(sf(f) and vfm(f,g,r))   ! Sector-specific primary
   PX(i,s,r)$vxmd(i,s,r)              ! Price index for exports
   PA(i,g,r)$vafm(i,g,r)              ! Price for Armington goc
   PE(g,r)$vxm(g,r)                   ! Price index for exports

$consumers:
   ra(r)                              ! Representative agent
```

Julia/MPSGE.jl

```
@sectors(MGE, begin
    Y[set_g, set_r],               (description = "Supply")
    M[set_i, set_r],               (description = "Imports")
    YT[set_i],                     (description = "Transportation services")
    E[set_i, set_r, set_r],        (description = "Subsidy and transport service
    A[set_i, set_g, set_r],        (description = "Armington good")
end)

@commodities(MGE, begin
    P[set_g, set_r],               (description = "Domestic output price")
    PM[set_i, set_r],              (description = "Import price")
    PT[set_i],                     (description = "Transportation services")
    PF[set_mf, set_r],             (description = "Non-sector-specific primary f
    PS[set_sf, set_g, set_r],      (description = "Sector-specific primary facto
    PX[set_i, set_r, set_r],       (description = "Price index for exports (incl
    PA[set_i, set_g, set_r],       (description = "Price index for Armington goo
    PE[set_i, set_r],              (description = "Price index for exports (excl
end)

@consumers(MGE, begin
    RA[set_r],                     (description = "Representative agent")
end)
```

Center for Sustainability Science and Strategy

Coding in Julia

GAMS/MPSGE

```
$prod:y(g,r)$vom(g,r)   t:etadx(g)   s:esub(g)   sn(s):esubn(g)   sve(sn):esubve(g)   sva(sve):esubva(g)   sef(sve):esubef(g)   sf(sef):esubf(g)
        o:P(g,r)                q:(vom(g,r)-vxm(g,r))                        a:RA(r)  t:rto(g,r)
        o:PE(g,r)               q:vxm(g,r)                                   a:RA(r)  t:rto(g,r)
        i:PA(i,g,r)$fe(i)       q:vafm(i,g,r)                                                              sf:
        i:PA(i,g,r)$elec(i)     q:vafm(i,g,r)                                                              sef:
        i:PA(i,g,r)$ne(i)       q:vafm(i,g,r)                                                              sn:
        i:ps(sf,g,r)            q:vfm(sf,g,r)   p:(1+rtf0(sf,g,r))           a:ra(r)  t:rtf(sf,g,r)
        i:pf(mf,r)              q:vfm(mf,g,r)   p:(1+rtf0(mf,g,r))           a:ra(r)  t:rtf(mf,g,r)        sva:
```

Julia/MPSGE.jl

```
for g ∈ set_i, r ∈ set_r
    @production(MGE, Y[g, r], [t = etadx[g], s = esub[g], sn => s = esubn[g], sve => sn = esubve[g], sva => sve = esubva[g], sef => sve = esubef[g], sf =
        @output(P[g, r],        vhm[g, r], t, taxes = [Tax(RA[r], rto[g, r])], reference_price = 1-rto0[g, r])
        @output(PE[g, r],       vxm[g, r], t, taxes = [Tax(RA[r], rto[g, r])], reference_price = 1-rto0[g, r])
        [@input(PA[i, g, r],    vafm[i, g, r], sf) for i ∈ set_fe]...
        [@input(PA[i, g, r],    vafm[i, g, r], sef) for i ∈ set_elec]...
        [@input(PA[i, g, r],    vafm[i, g, r], sn) for i ∈ set_ne]...
        [@input(PS[sf, g, r],   vfm[sf, g, r],  s, taxes = [Tax(RA[r], rtf[sf, g, r])],   reference_price = 1 + rtf0[sf, g, r])   for sf ∈ set_sf]...
        [@input(PF[mf, r],      vfm[mf, g, r],  sva, taxes = [Tax(RA[r], rtf[mf, g, r])],   reference_price = 1 + rtf0[mf, g, r])   for mf ∈ set_mf]...
    end)
end
```

Center for Sustainability Science and Strategy

Coding in Julia

GAMS/MPSGE

```
$demand:ra(r)
        d:p("c",r)                q:vom("c",r)
        e:p("c","USA")            q:vb(r)
        e:p("g",r)                q:(-vom("g",r))
        e:p("i",r)                q:(-vom("i",r))
        e:ps(sf,j,r)              q:vfm(sf,j,r)
        e:pf(mf,r)                q:evom(mf,r)
```

Julia/MPSGE.jl

```
for r ∈ set_r
    @demand(MGE, RA[r], begin
        @final_demand(P[:c, r],     vom[:c, r])
        @endowment(P[:c, :USA],     vb[r])
        @endowment(P[:g, r],        -vom[:g, r])
        @endowment(P[:i, r],        -vom[:i, r])
        [@endowment(PF[f, r],       evom[f, r]) for f ∈ set_mf]...
        [@endowment(PS[f, j, r],    vfm[f, j, r]) for f ∈ set_sf, j ∈ set_i]...
    end)
end
```

Center for Sustainability Science and Strategy

Setting

- Data resolution

  - 56x2 — 2-region *USA, ROW*
  - 56x4 — 4-region *USA, EUR, CHN, ROW*
  - 56x8 — 8-region *USA, EUR, ROE, JPN, IND, CHN, ANZ, ROW*
  - 56x16 — 16-region *USA, …, ANZ, TWN, CAN, MEX, BRA, RUS, KOR, IDZ, ASI, ROW*
  - 56x32 — 32-region *USA, [disaggregated EUR: DEU, FRA, GBR, ITA, ESP, …], …, ROW*
  - 56x50 — 50-region *Disaggregate 56x32*
  - 56x61 — 61-region *Disaggregate 56x50*
  - 56x74 — 74-region *Disaggregate 56x61*

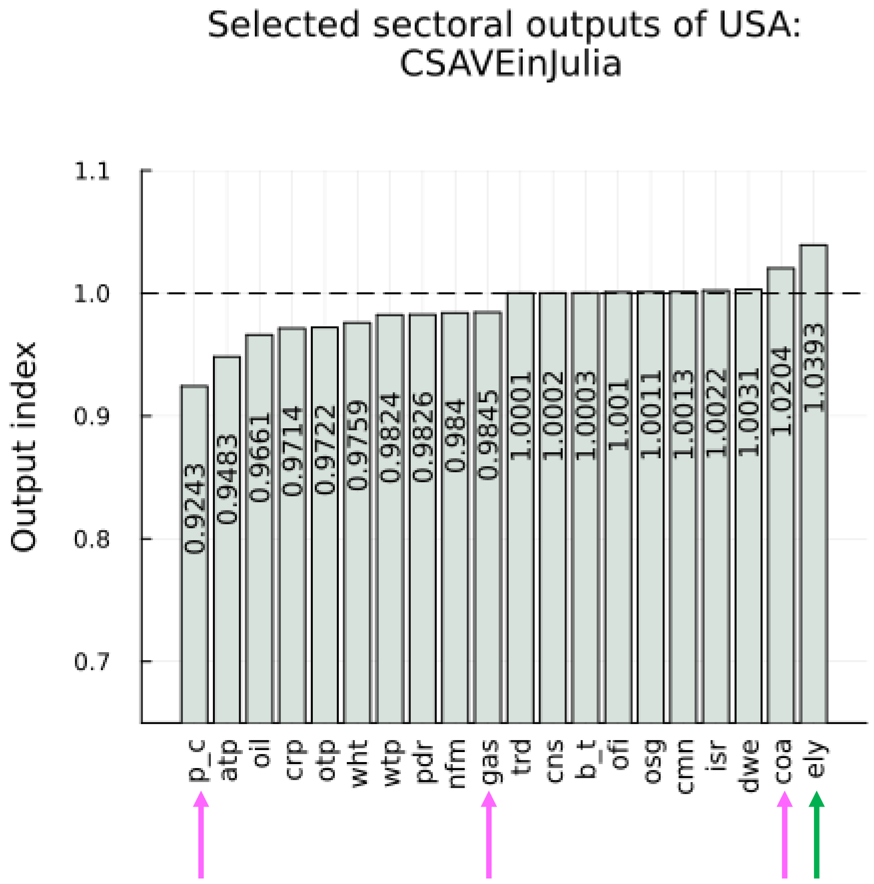
Setting

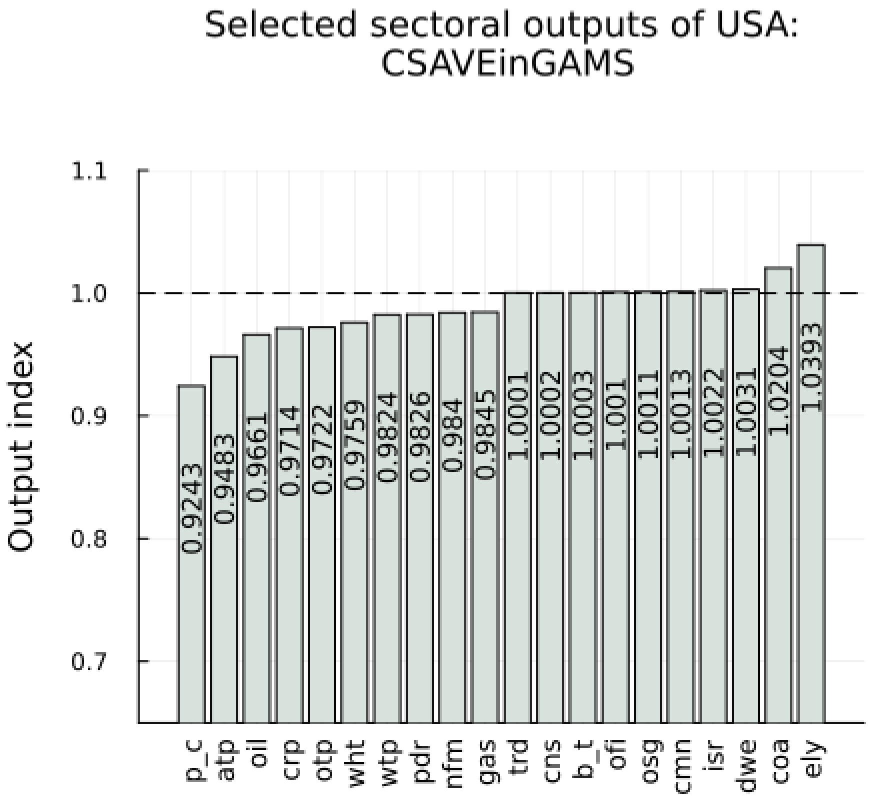
- Scenario
  - US doubles tax rates on fossil fuels use:

    rtfd0[i, g, r] x 2.0 & rtfi0[i, g, r] x 2.0 ; $r \in$ USA, $i \in$ coa, p_c, gas

Simulation

Double fossil fuel tax rates



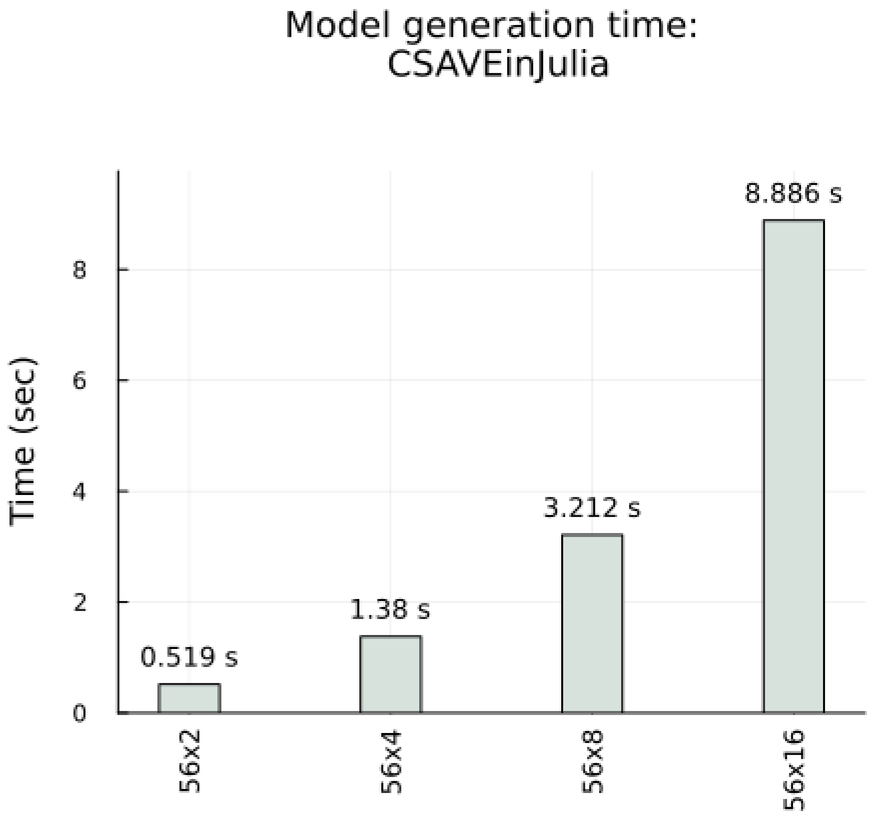


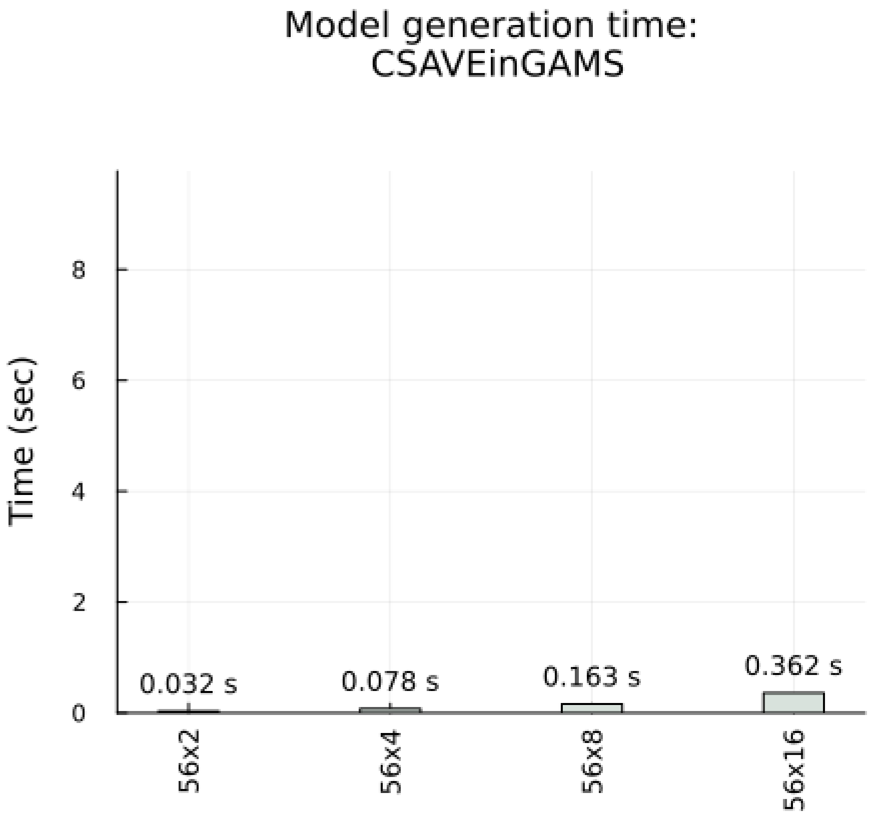
Computer spec

- CPU : AMD Ryzen Threadripper PRO 7955WX 16-Cores 4.5 GHz
- RAM : 128 GB 4800 MT/s

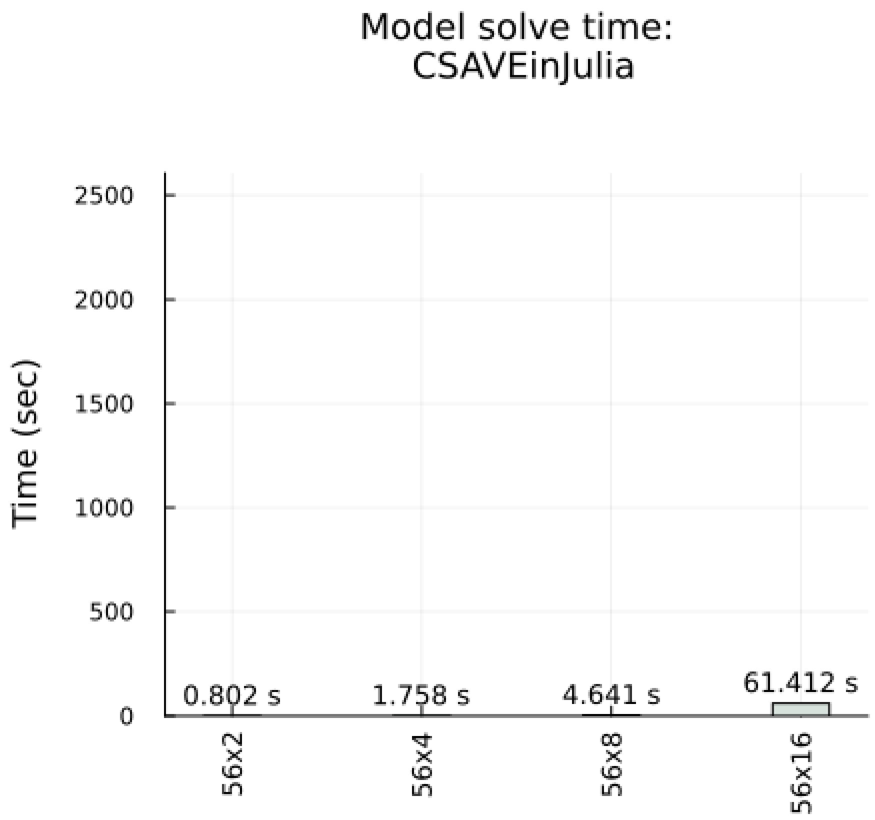
Model generation time

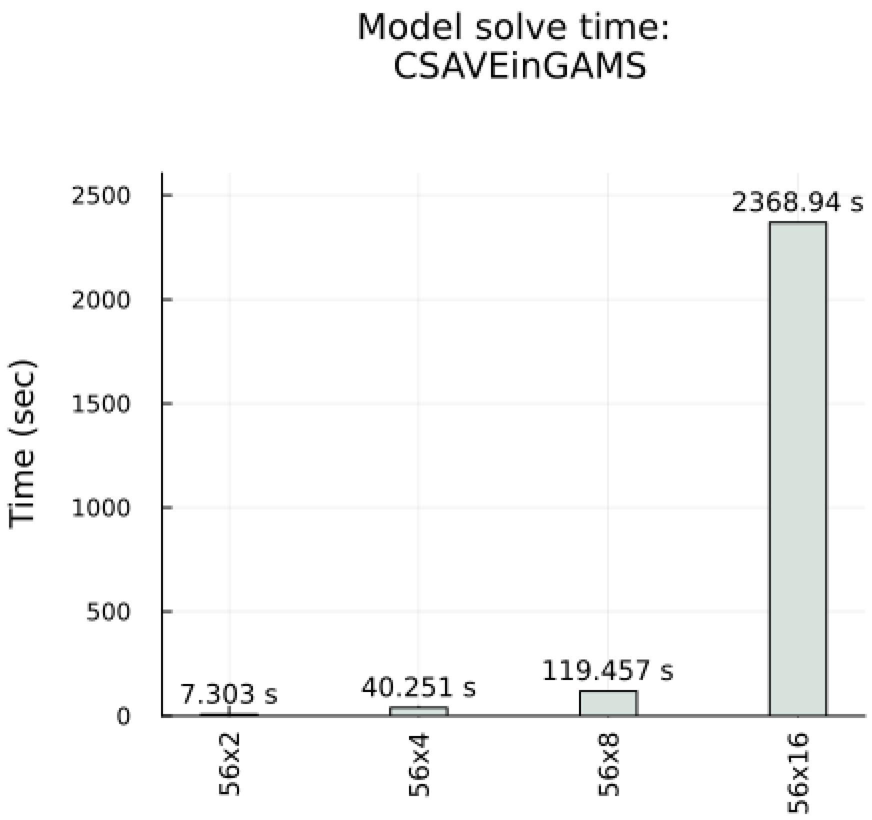
For now, each data resolution setting is run once for demonstration purposes.





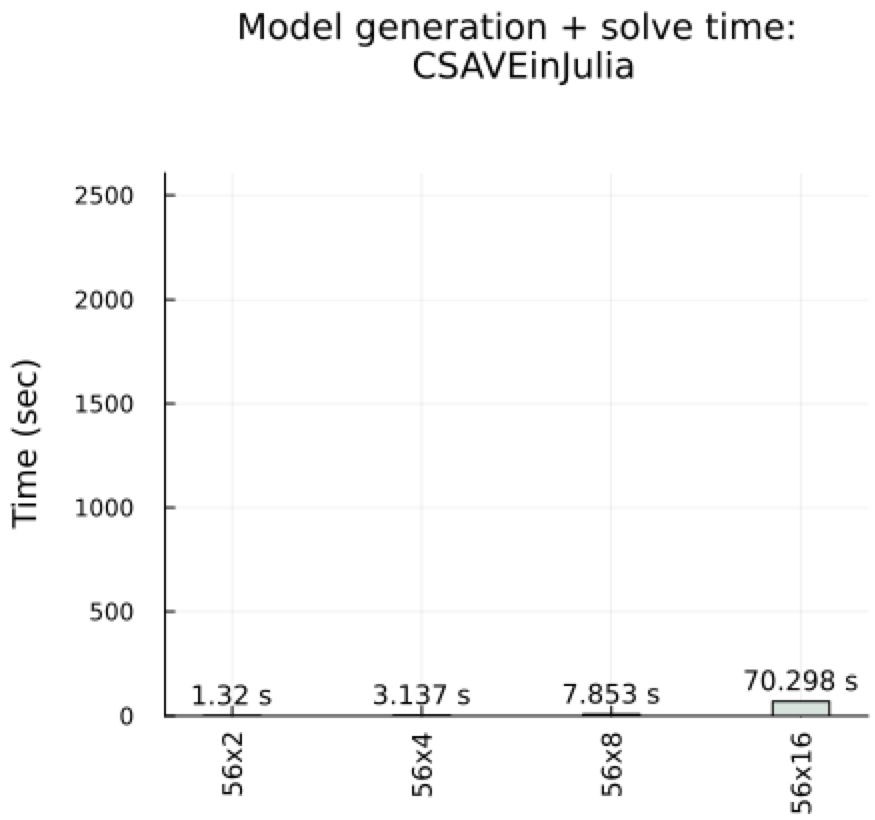
Model solving time

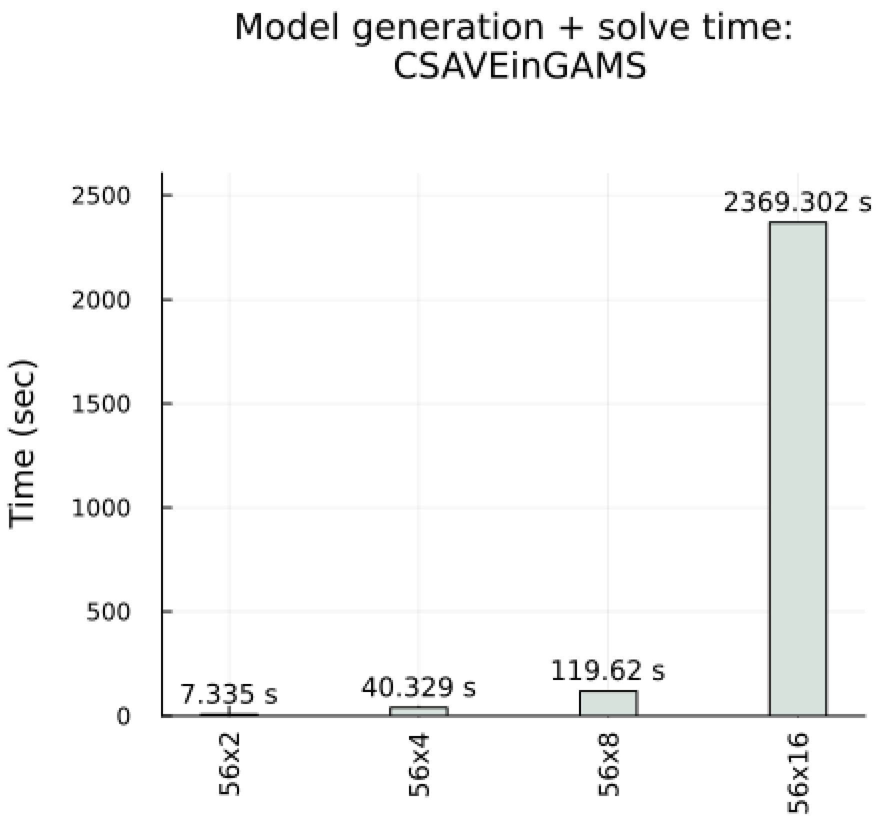




Center for
Sustainability Science
and Strategy

Model generation + solving time





Center for Sustainability Science and Strategy

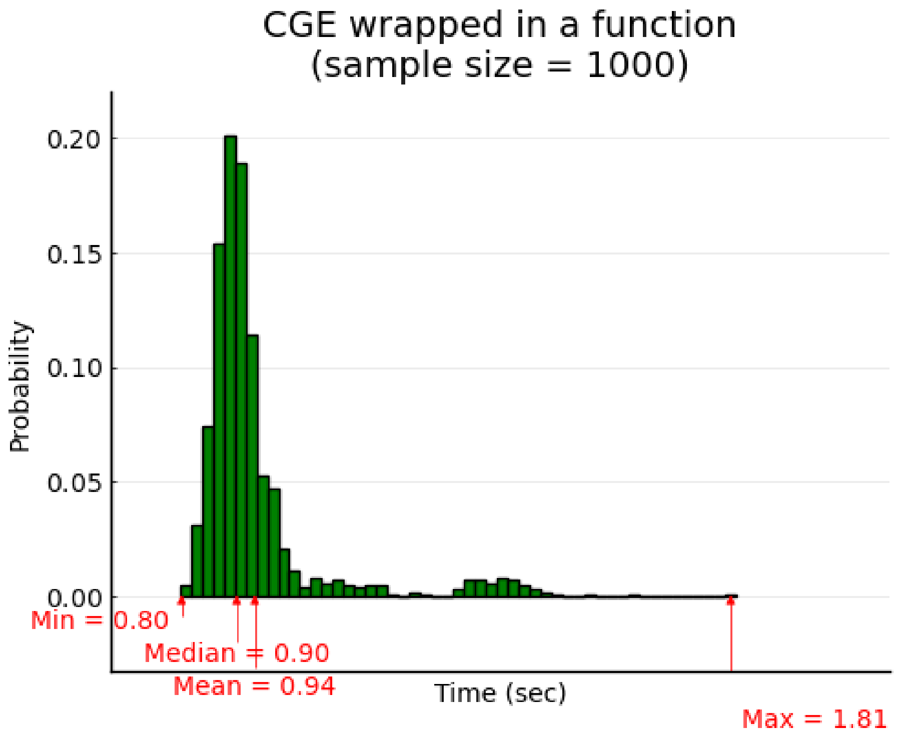
Looking into Julia solving time

"Time distributions are always right-skewed… This phenomena can be justified by considering that the machine noise affecting the benchmarking process is, in some sense, inherently positive…"

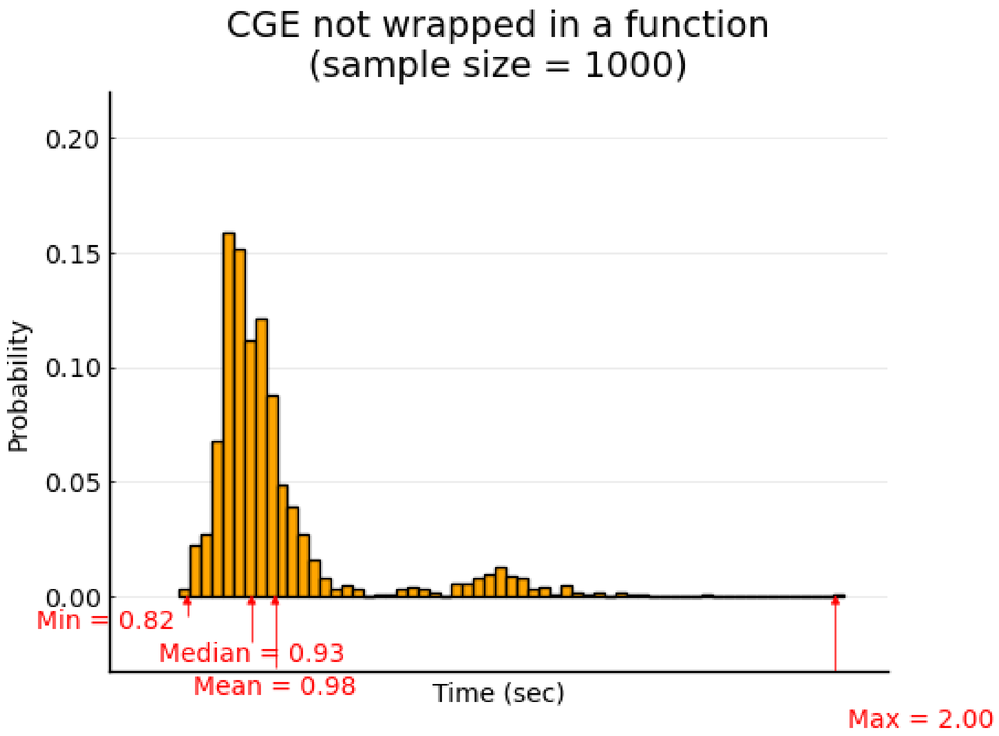
— Chen and Revels (2016)

Looking into Julia solving time

How solving time is affected by coding strategy?



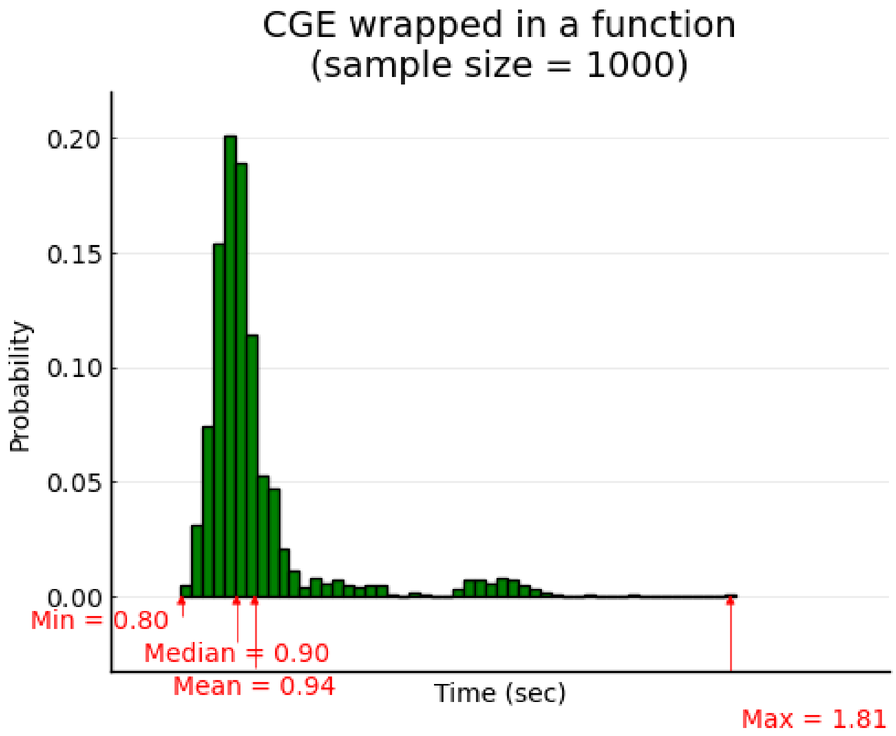
Code lives within the function



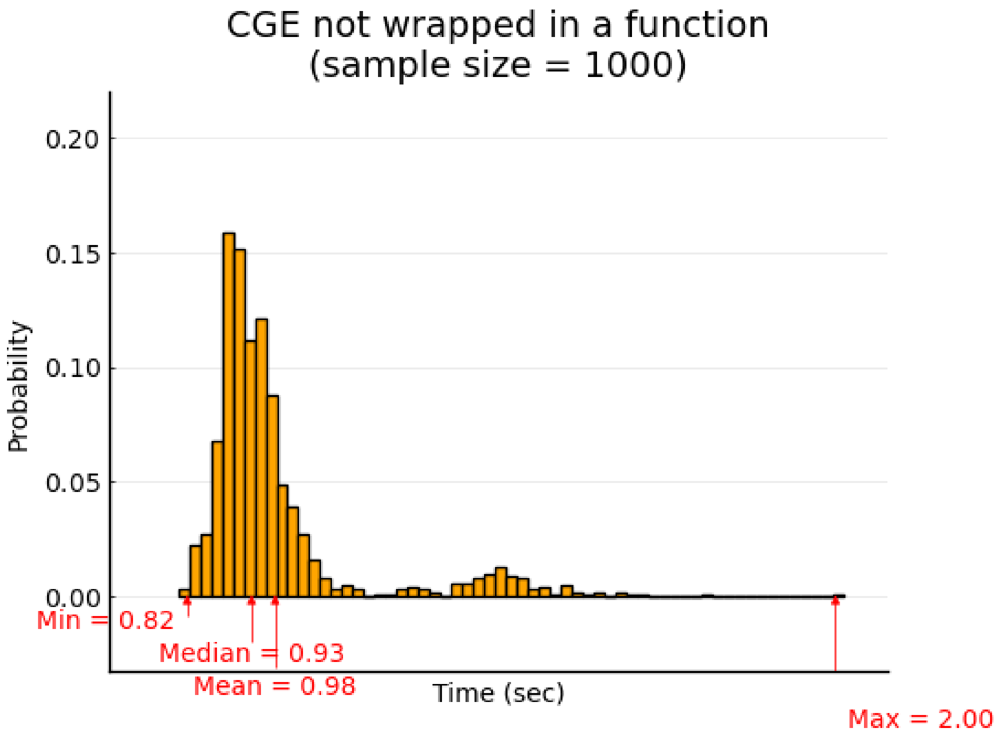
Code lives in global scope

Looking into Julia solve time

Using median to measure central tendency. Question: Is the difference in medians statistically significant?



Code lives within the function



Code lives in global scope

Looking into Julia solve time

Bootstrap resampling with sample size B = 10000 (Efron and Tibshirani, 1994)

```
# 3. Bootstrap Loop
for i in 1:B
    # Resample with replacement
    X1_star = sample(X1, n1, replace=true)
    X2_star = sample(X2, n2, replace=true)

    # Calculate the statistic (difference in medians)
    delta_star = median(X1_star) - median(X2_star)

    # Append "delta_star" to the end of an array ("bootstrap_differences") one element at a time.
    push!(bootstrap_differences, delta_star)
end
```

1000-element Vector{Float64}

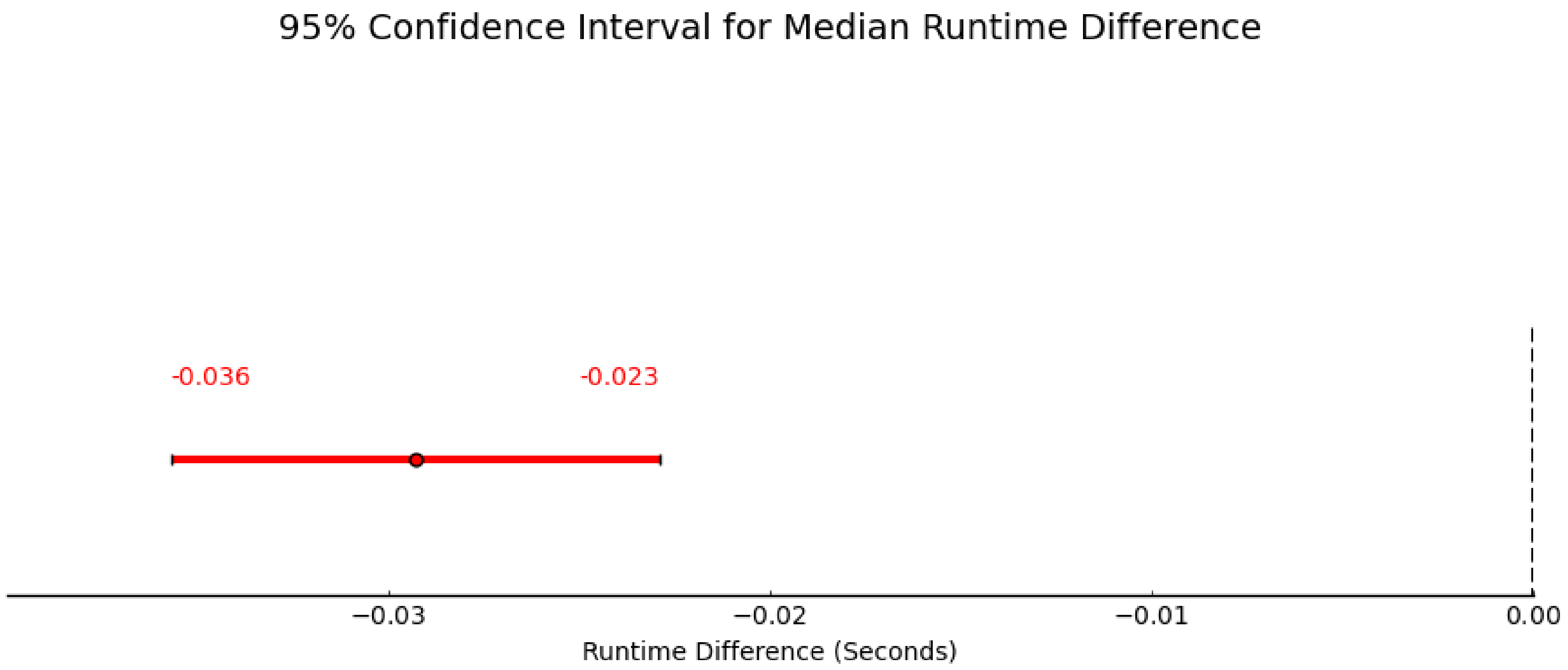
Float64

After the for-loop is done, *bootstrap_differences* is a 10000-element Vector{Float64}

```
# 4. Construct the Confidence Interval (e.g., 95% CI)
CI_lower = quantile(bootstrap_differences, 0.025)
CI_upper = quantile(bootstrap_differences, 0.975)
```

Looking into Julia solve time

Bootstrap resampling with sample size B = 10000 (Efron and Tibshirani, 1994)



Summary

- Julia
    - CSAVEinJulia: slower in model generation
    - CSAVEinJulia: faster in solving
    - More involved in coding

- GAMS
    - CSAVEinGAMS: faster in model generation
    - CSAVEinGAMS: slower in solving
    - Easier to implement

Beyond the conclusion

“A better brush doesn't make a better painter, but a better painter can do more with a better brush.”

— Unknown

Acknowledgement

I am thankful for Mitch Phillipson for sharing his knowledge on CGE modeling in Julia, and for Angelo Gurgel who provides me Julia course materials. All errors are my own.

Thank you!

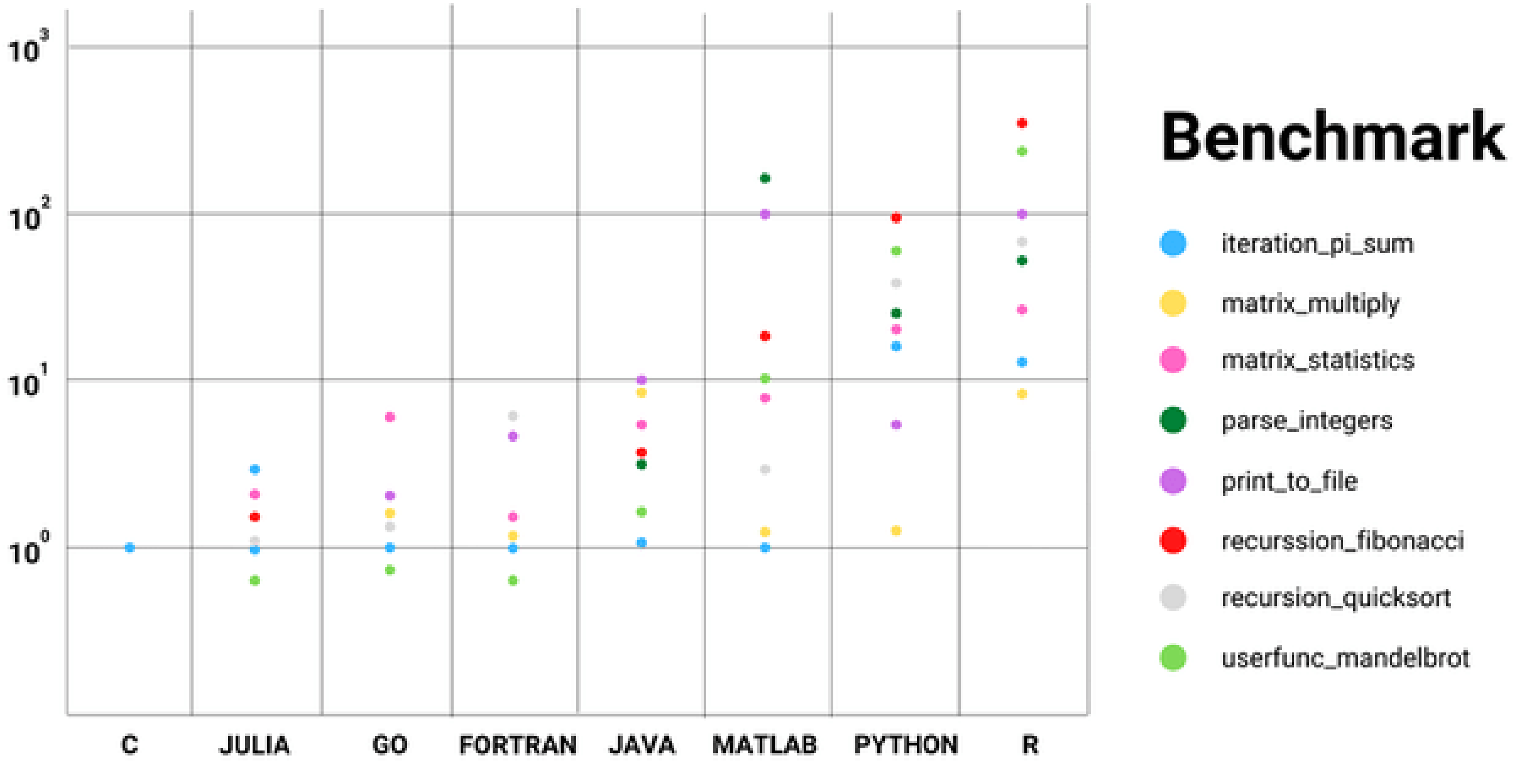
Questions?

Y.-H. Henry Chen

chenyh@mit.edu

Appendix: A comparison

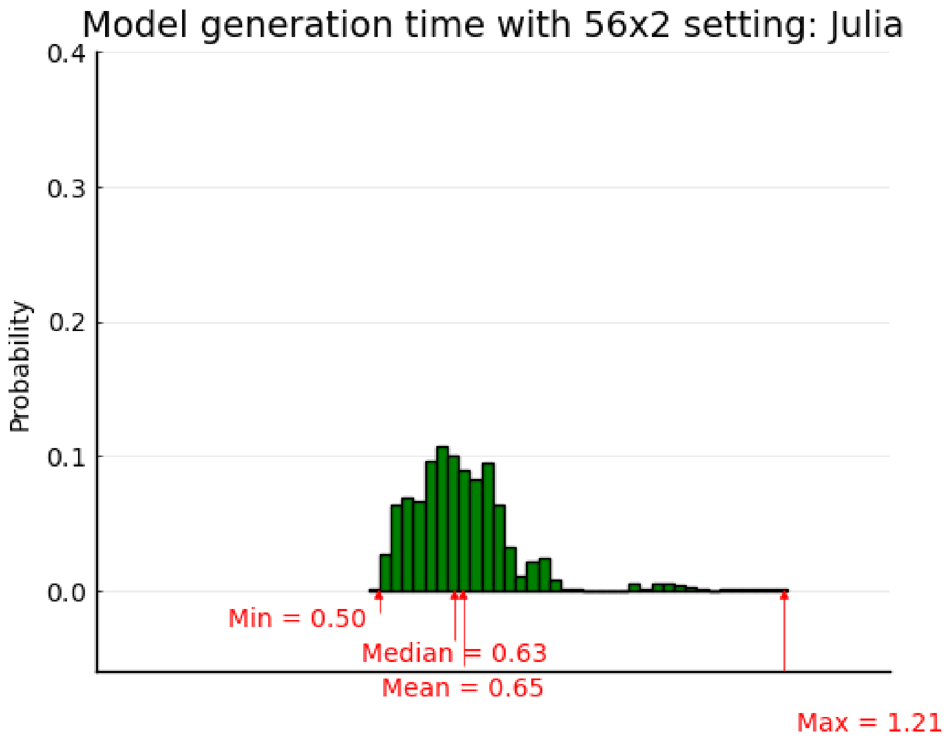
Comparison of Computational Resource for different operations in programming languages

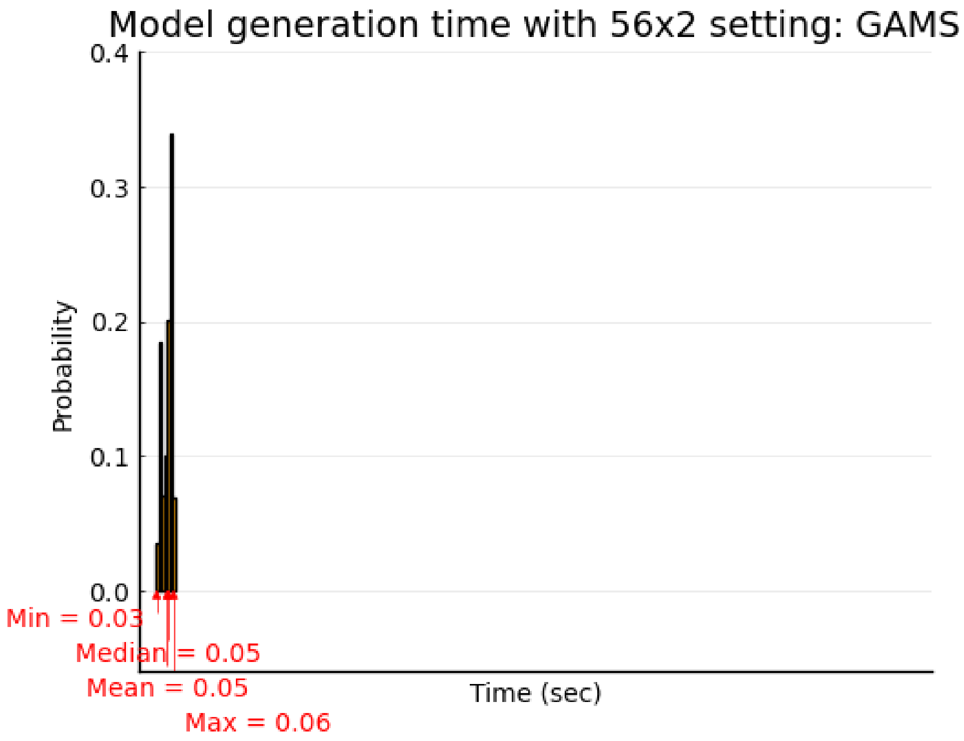


Source: Thakur and Satish (2022)

Appendix: Model generation time difference

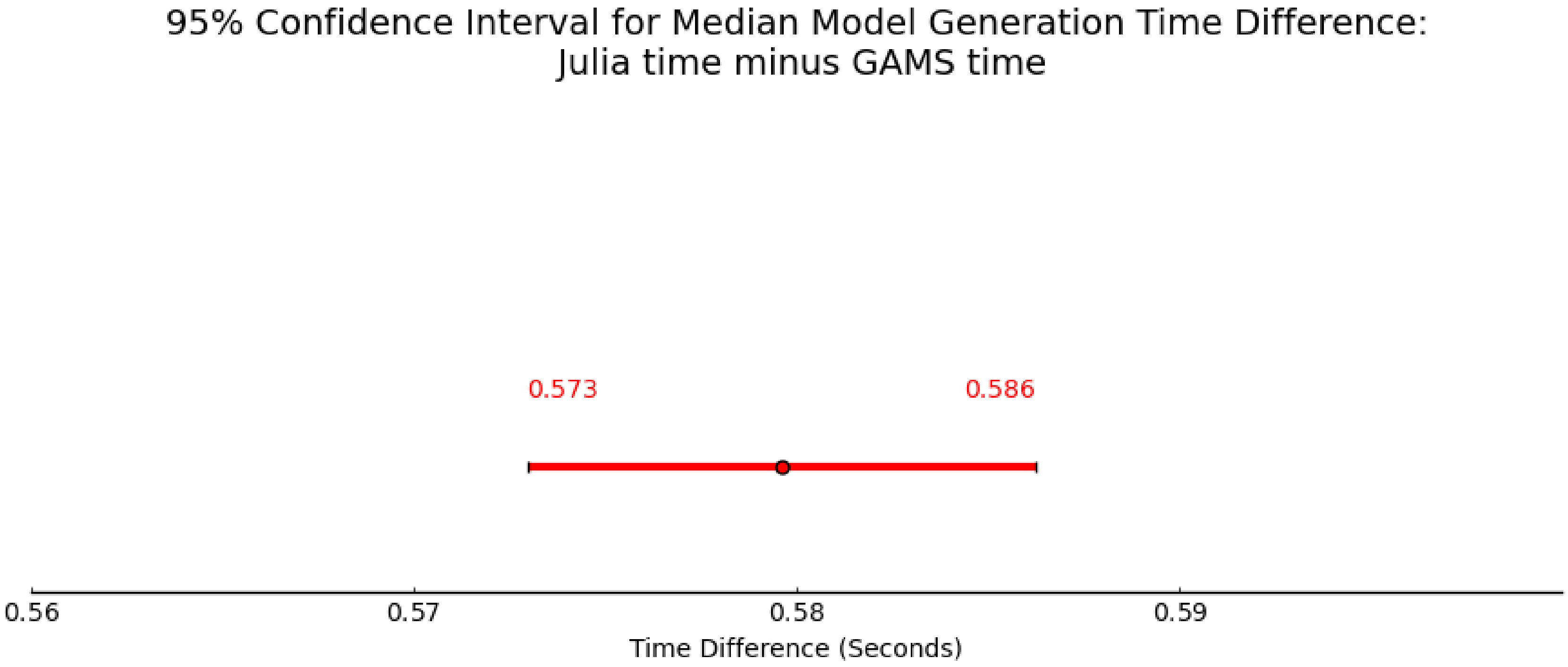
GAMS is faster!





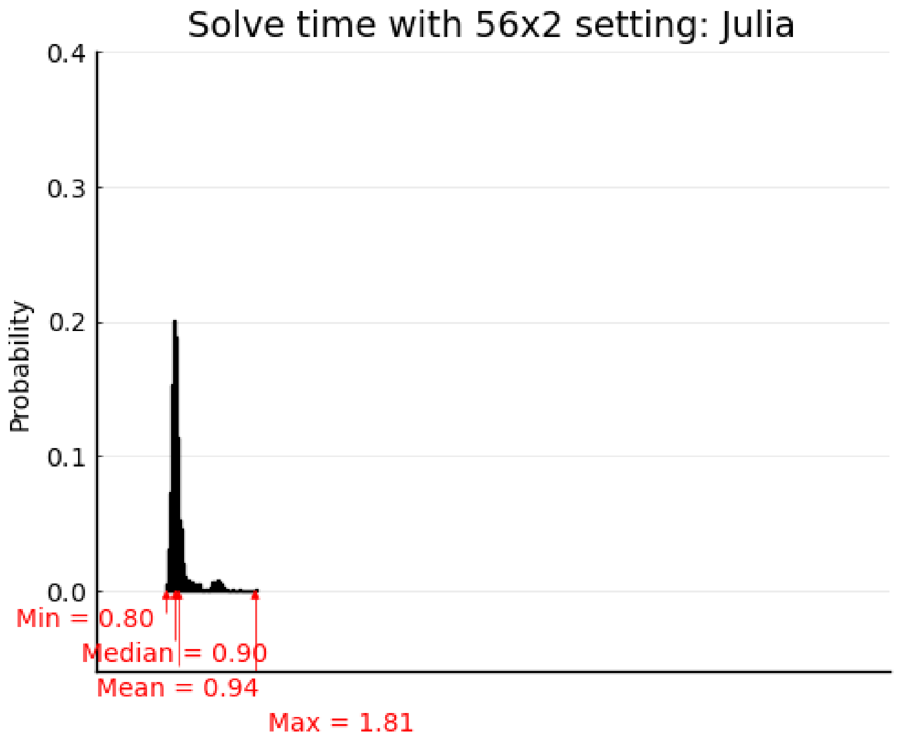
Appendix: Model generation time difference

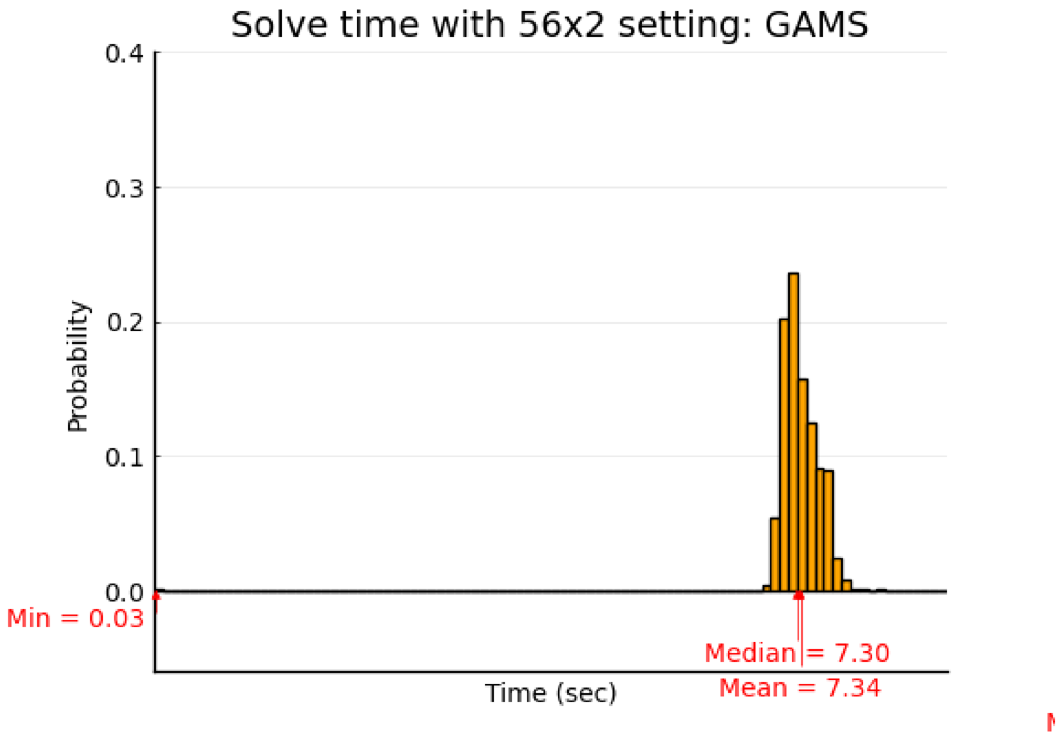
GAMS is faster!



Appendix: Solve time difference

Julia is faster!





Appendix: Solve time difference

Julia is faster!