The state of JuMP

Miles Lubin (Hudson River Trading*)

JuMP-dev 2024

*Speaking in a personal capacity. Views and opinions expressed do not necessarily reflect those of HRT.

What is JuMP?

Part of the zoo of algebraic modeling languages

AIMMS

AMPL

CVX RESEARCH

CVXPY

GAMS

LINGO
the modeling language and optimizer

MAXIMAL
SOFTWARE

Mini
Zn

PYOMO

CMPL, CPLEX Concert, GNU MathProg, Gurobi C++/Python API, linopy, MATLAB “problem-based optimization workflow”, Mosek Fusion, MOSEL, ompr, OPTMODEL, PuLP, PyOptInterface, Python-MIP, YALMIP, ZIMPL, …

What is JuMP?

An algebraic modeling language in Julia

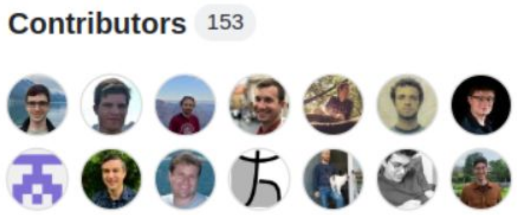
```
using JuMP, Ipopt
function constrained_linear_regression(A::Matrix, b::Vector)
    model = Model(Ipopt.Optimizer)
    @variable(model, 0 <= x[1:size(A, 2)] <= 1)
    @variable(model, residuals[1:size(A, 1)])
    @constraint(model, residuals == A * x - b)
    @constraint(model, sum(x) <= 1)
    @objective(model, Min, sum(r^2 for r in residuals))
    optimize!(model)
    return value.(x)
end
A, y = rand(30, 20), rand(30)
x = constrained_linear_regression(A, b)
```

Why is JuMP interesting?

- Nice syntax
- Comprehensive documentation
- Vibrant community
- Open source
- Solver independent (50+ supported solvers)
- Embedded in Julia
- Supports interacting with solvers while they’re running
- Low overhead for model generation
- Extensible to new solvers
- Extensible to new problem classes

Who is (some of) JuMP?

https://github.com/jump-dev/JuMP.jl/graphs/contributors


Contributors 153

In the last 12 months of github.com/jump-dev…

>695,000
downloads of jump-dev packages

1,174
pull requests opened

52
unique contributors

373
issues opened

Milestones from the past year

- Between July 2023 and July 2024: **1.13 ➔ 1.22**
- Three roadmap items completed
- Supported solvers increased from 43 to 54
- Convex.jl backend rewrite
- Julia community prize
- Transitioned from NSF funding (new: LANL, PSR)
- Second JuMP baby

Major technical improvements

- New Nonlinear API
- Macro refactoring

Improving nonlinear programming support in JuMP

https://jump.dev/JuMP.jl/stable/manual/nonlinear/

```
using JuMP
model = Model()
@variable(model, x[1:2])
@objective(model, Min, x[2]^3 * sin(x[1])^x[2])
my_func(y) = sum(2^y[1] .+ exp.(y))
@expression(model, expr, 2 * my_func(x))
@constraint(model, expr <= 100)
@constraint(model, sqrt(x' * x) <= 1)
```

Nonlinear complementarity

https://jump.dev/JuMP.jl/stable/tutorials/nonlinear/complementarity/

```julia
using JuMP, PATHSolver
model = Model(PATHSolver.Optimizer)
@variable(model, 0 <= x[1:2] <= 1, start = 0.5)
# To add a constraint of the form `F(x) ⊥ x` do
@constraint(model, x[2]^3 - x[1] ⊥ x[1])
@constraint(model, 1 - exp(x[1]) ⊥ x[2])
optimize!(model)
value.(x)
```

Smaller items

Parameters

```julia
using JuMP
model = Model()
@variable(model, x in Parameter(2))
@variable(model, y[1:2])
@constraint(model, sum(y) >= x)
set_parameter_value(x, 3)
```

Use with extensions like ParametricOptInterface.jl

DimensionalData

https://jump.dev/JuMP.jl/stable/extensions/DimensionalData/

```
using JuMP, DimensionalData
model = Model()
@variable(
    model,
    x[i = 2:4, j = ["a", "b"]] >= i,
    container = DimensionalData.DimArray,
)
x[At(2), At("a")]
x[1, 1]
```

Boolean SAT

https://jump.dev/JuMP.jl/stable/manual/constraints/#Boolean-constraints

```
using JuMP, MiniZinc
model = GenericModel{Bool}(() ->
    MiniZinc.Optimizer{Bool}("chuffed"))
@variable(model, x[1:2])
@constraint(model, x[1] || x[2] := true)
@constraint(model, x[1] && x[2] := false)
optimize!(model)
value.(x)
```

lp_matrix_data

```julia
using JuMP
model = Model()
@variable(model, x[1:2])
@constraint(model, x[1] + 2 * x[2] <= 1)
@objective(model, Max, x[2])
data = lp_matrix_data(model)
```

$$c^\top x + c_0$$
$$b_l \leq Ax \leq b_u$$
$$x_l \leq x \leq x_u$$

`is_solved_and_feasible`

- Does what its name says
- Shortcut for checking MOI termination and solution statuses

Also

- Convex.jl backend rewrite
- Gurobi automated install
- New solvers/interfaces:
  - MINOTAUR
  - PolyJuMP.QCQP
  - Octeract
  - MiniZinc
  - Percival
  - Manopt
  - SDPLR
  - Optim
  - MAiNGO
  - DSDP

New roadmap items

- Nonlinear expressions with vector inputs and outputs
- First-class support for units
  - `@variable(model, x, u"m/s")`
- (Maybe more after this workshop!)

2.0?

- Haven’t seen a good reason to make breaking changes so far
- Compare with numpy (16 years from 1.0 to 2.0!)

JuMP-inspired packages

I’d love to use JuMP but I’m coding in ___

- C++/Python: MathOpt (Google OR-Tools)
- Python: PyOptInterface
- R: ompr

Great!

Thank you!

Go to jump.dev for more information

NUMFOCUS
OPEN CODE = BETTER SCIENCE

PSR

GERAD

Google Open Source

Los Alamos
NATIONAL LABORATORY

JuliaHub

RelationalAI

And the whole JuMP community!