# Feedback from a Julia user

Research and teaching activities in operational research

# What have I been using Julia for?

- Several research projects:
  - Implementing stochastic/robust optimisation models
  - Solving water-resource problems — ReservoirManagement.jl
  - Optimising production and workers' shifts — IndustrialProcessFlexibilisation.jl
  - Improving combinatorial bandits — CombinatorialBandits.jl
- Teaching:
  - Discrete optimisation (twice) — OptimisationTeachingKit

# Research with Julia and JuMP

# Discovering Julia and JuMP

- I started with Julia 0.3, in September 2014
- Several implementations of stochastic and robust optimisation models
  - Including Benders' decomposition
- Rapid development
  - Albeit low experience on my side

# ReservoirManagement.jl

- Goal of the project: determine how to manage water reservoirs (dams)
- Very similar constraints between two dams
  - Hence very similar optimisation models, use just a few parameters
  - Allow the user to *declare* the parameters

# ReservoirManagement.jl

- Example usage:

```
Vesdre = NaturalRiver(name="Vesdre", scenarios=…,
environmental_flow=0.5m^3/s…)

purpose = DeterministicPurposes(drinkingWater=0.5m^3/s)

out = ConstantDamOutputs(bottomOutlets=100m^3/s)

VesdreReservoir = Reservoir(name="Vesdre",
    capacity=(2_500_000.m^3, 25_000_000.m^3),
    purposes=purpose, outputs=out, rivers_in=[Vesdre])
```

# ReservoirManagement.jl

- Long term maintenance?
  - Development started in Julia 0.3
    - A prototype that evolved into a library
  - Migration to 0.4 done in a few days
  - Migration to 0.5 cancelled for multiple reasons
    - Not sufficient test coverage
    - Unexplained errors in the existing code
    - #18725 broke my usual workflow

# IndustrialProcessFlexibilisation.jl

- Goal of the project: exploit electricity flexibility in industrial sites
  - Consume electricity when it is really cheap
  - Have workers on site when they are required
- Plants can be very different one from the other
  - Different machines, routes within the plant, etc.
  - More complex to represent in data structures

# IndustrialProcessFlexibilisation.jl

- How to build a model based on such a representation?
  - Usually: objects with methods
  - Not possible in Julia: no “object model”
- Rather: multiple dispatch
  - Constructor: builds the required optimisation variables
  - postConstraints: adds constraints for each object
    - Exploits multiple dispatch to adapt to object type
  - ➢ Users can provide their own objects!

# Teaching with Julia and JuMP

# Why use Julia and JuMP?

- Context: a discrete optimisation course
  - Mostly: work with MIP models, a bit heuristics

- Previous iterations of the course:
  - AMPL for the exercise sessions
    - Nice syntax for modelling, hard to do anything else
  - Java with the CPLEX API for the project
    - Nonintuitive API

# Why use Julia and JuMP?

Since 2015: Julia and JuMP everywhere

- More consistent: one language to rule them all
- Easier installation
  - GMPL implemented enough of AMPL
    No official binary for Windows (WinGLPK)
  - How to get a CPLEX license for students?
    (Now, much simplified with OnTheHub)

# Why use Julia and JuMP?

- With JuMP:
  - Completely free (and open source)
  - Cbc is always easy to install, on all platforms
- What about the competition?
  - CVX, YALMIP: based on MATLAB, not free
  - Pyomo: installing any solver is much harder

# How did students react?

First year: Julia 0.3 (September 2015)

- Many installation problems (half of the students)
- Not so helpful error messages
- 11 groups out of 15 used Julia for the project
  - 3 groups: exclusively Julia
  - 2 groups of non-computer science/engineer students

# How did students react?

Second year: Julia 0.4 (September 2016)

- Very few installation problems
- Overall more complex projects
  - Several groups used callbacks, none the previous year
- 20 groups out of 20 used Julia for the project
  - Up from 75% the previous year
  - 16 exclusively Julia (80%)
    - Up from 20% the previous year
  - One third of non-computer science/engineer students

# Conclusion

# Conclusion

Julia has evolved quite a bit since 2014

- More mature:
  - Better error messages
  - More polished environment (including IDEs)
- Still not 100% ready for prime time:
  - Where is the debugger?
  - But already useable for a wide audience

# Questions?
# Remarks?