Getting the Best of Both Worlds

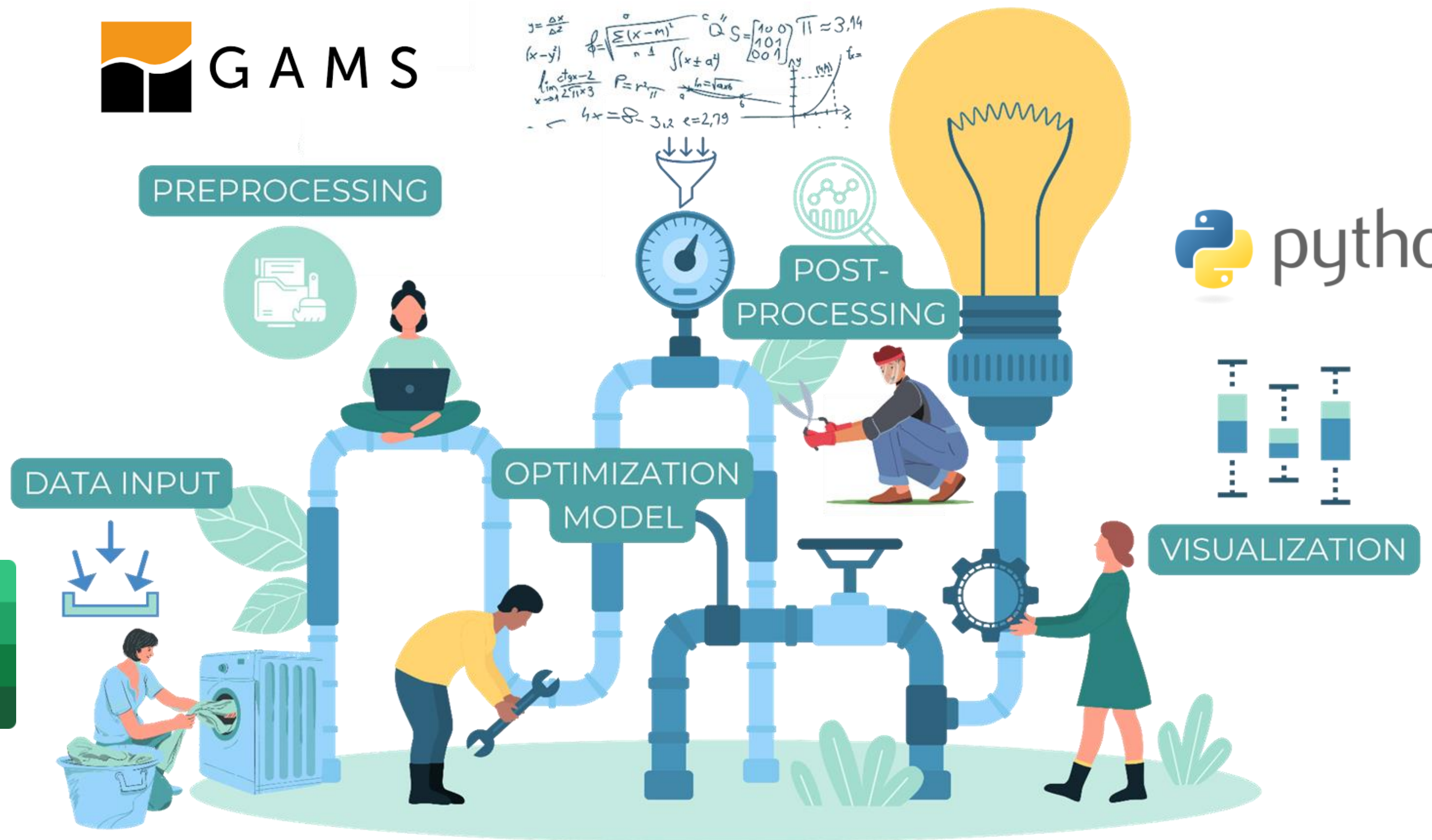
Ways to Combine Python's Flexibility with a Domain Specific Modeling Language in Applied Operations Research

Justine Broihan GAMS Development Corp. GAMS Software GmbH

Solving Complex Real-World Problems

- Is not all about the model itself
- Data cleaning
- Data manipulation
- Data visualization

Optimization Pipeline



Let's be honest

- Data manipulation, pre- and postprocessing can be time-consuming
- Different tools/languages have different strengths/weaknesses
- Choosing the best tools for your pipeline while keeping the interaction between them convenient can get difficult

Streamlining the Optimization Pipeline

- TRANSFER
- CONTROL
- EMBEDDED CODE </>
- GAMSPY

&

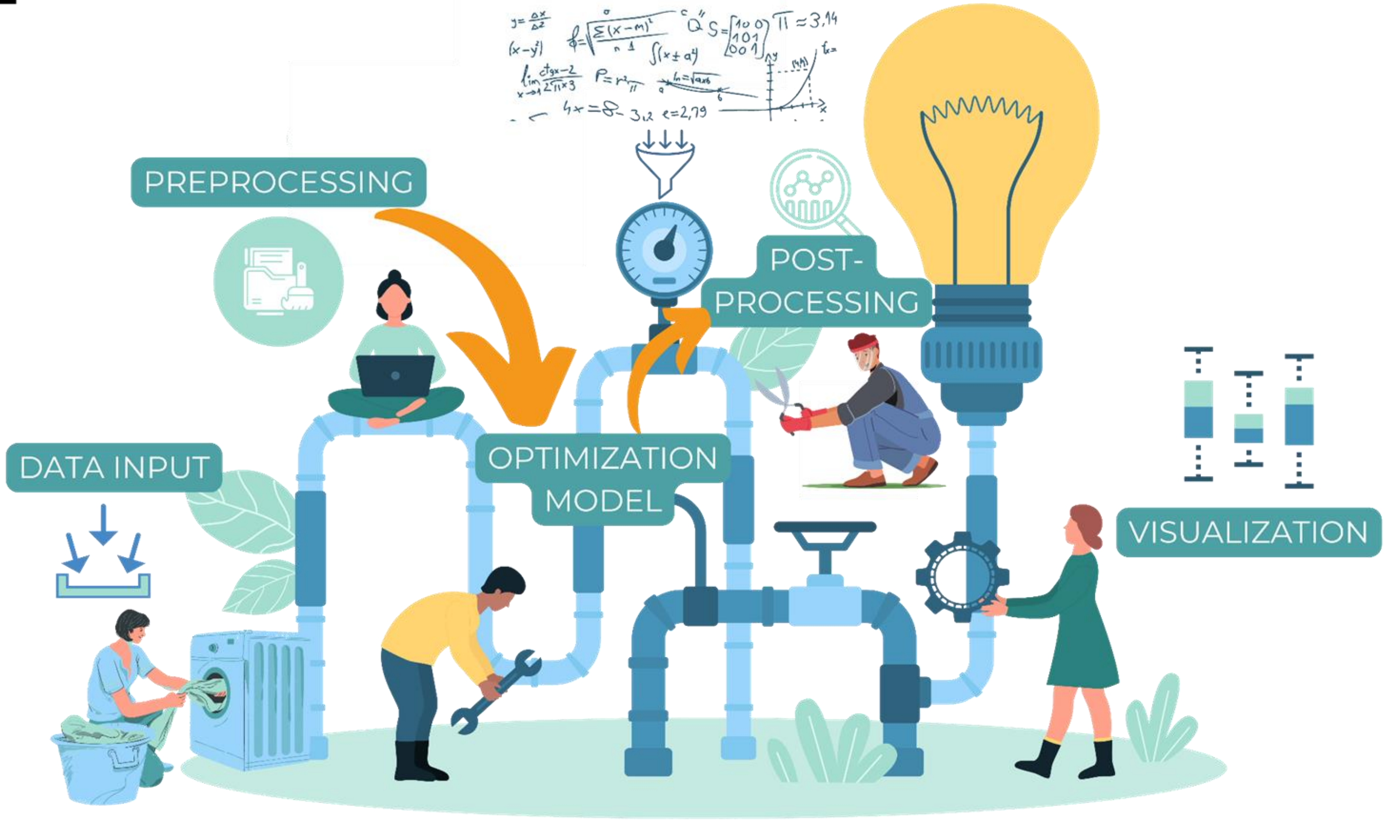
- API focusing on moving data from and to GAMS
- Connects to GDX (file based) and GMD (in memory)

- Available for

TRANSFER

GAMS

MODEL – SOLVE – DEPLOY



TRANSFER

GAMS
MODEL – SOLVE – DEPLOY

- Easy to install

```
pip install gams[transfer] --find-links [PATH TO GAMS]\api\python\bdist
```

- Works seamlessly with pandas NumPy
- Allows reading and writing GDX files

TRANSFER

GAMS

MODEL – SOLVE – DEPLOY

preprocessing.py

```python
from gams import transfer as gt
import pandas as pd

m = gt.Container()

# create the sets i, j
i = gt.Set(m, "i", records=["seattle", "san-diego"], description="supply")
j = gt.Set(m, "j", records=["new-york", "topeka"], description="markets")
```

TRANSFER

GAMS

MODEL – SOLVE – DEPLOY

preprocessing.py

```python
# add "d" parameter -- domain linked to set objects i and j
d = gt.Parameter(m, "d", [i, j], description="distance in thousands of miles")

# create some data as a generic DataFrame
dist = pd.DataFrame(
    [
        ("seattle", "new-york", 2.5),
        ("seattle", "topeka", 1.8),
        ("san-diego", "chicago", 1.8),
        ("san-diego", "topeka", 1.4),
    ],
    columns=["from", "to", "thousand_miles"],
)

# setRecords will automatically convert the dist DataFrame into a standard DataFrame format
d.setRecords(dist)
```

TRANSFER

GAMS

MODEL – SOLVE – DEPLOY

preprocessing.py

```python
# write the GDX
m.write("out.gdx")
```

Streamlining the Optimization Pipeline

GAMS
MODEL – SOLVE – DEPLOY

TRANSFER

CONTROL

EMBEDDED CODE </>

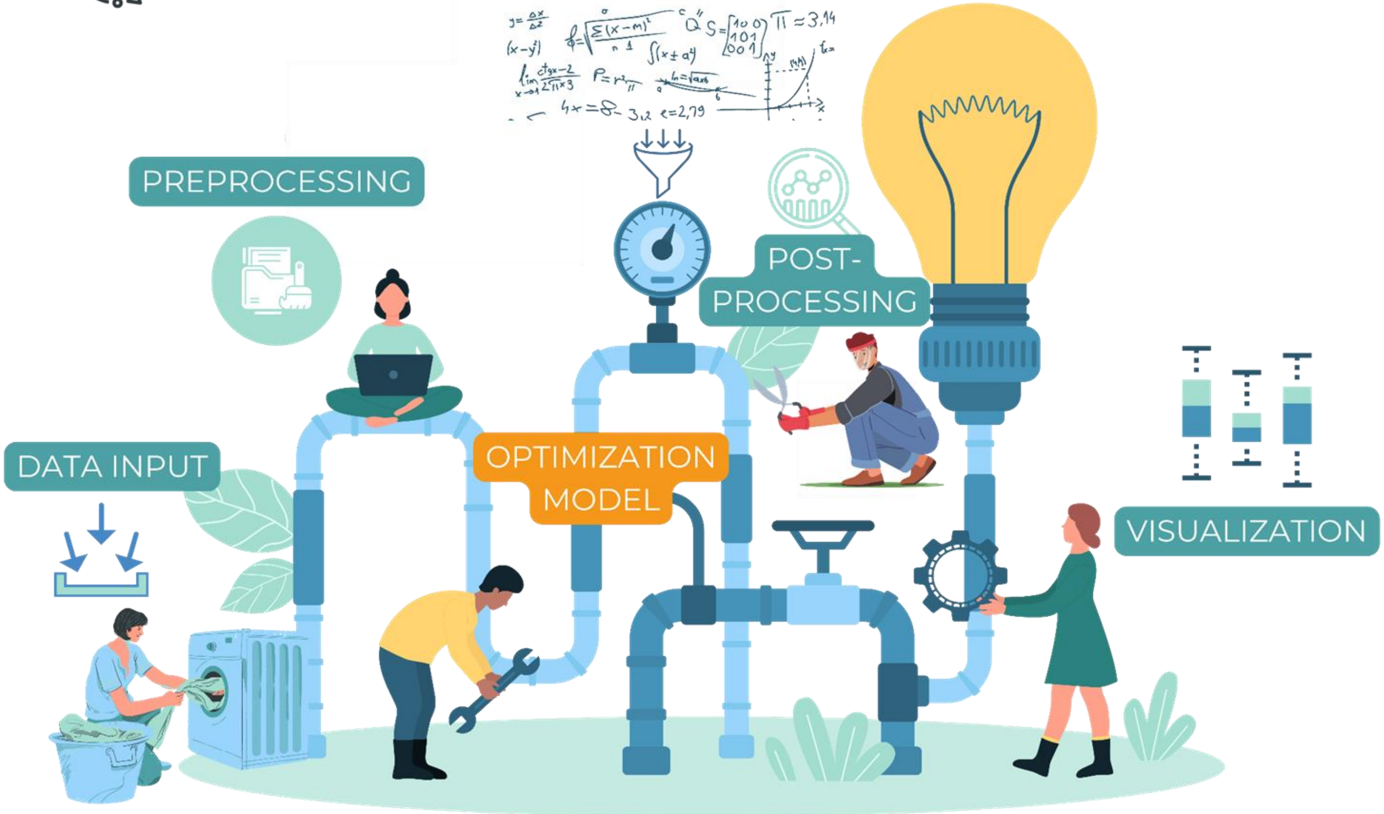
GAMSPY

&

CONTROL

- API focusing on controlling the GAMS system
  - Create and run GAMS models (`GAMS Jobs`)
  - Solve a sequence of closely related model instances (`GamsModelInstance`)

CONTROL


PREPROCESSING
POST-
PROCESSING
DATA INPUT
OPTIMIZATION
MODEL
VISUALIZATION

CONTROL

GAMS
MODEL – SOLVE – DEPLOY

solve.py

```python
import gams

# create a workspace
ws = gams.GamsWorkspace()

# point to .gms file
job = ws.add_job_from_file("trnsport.gms")

# define options
opt = ws.add_options()
opt.all_model_types = "xpress"

# solve
job.run(opt)
```

Streamlining the Optimization Pipeline

GAMS
MODEL – SOLVE – DEPLOY

TRANSFER

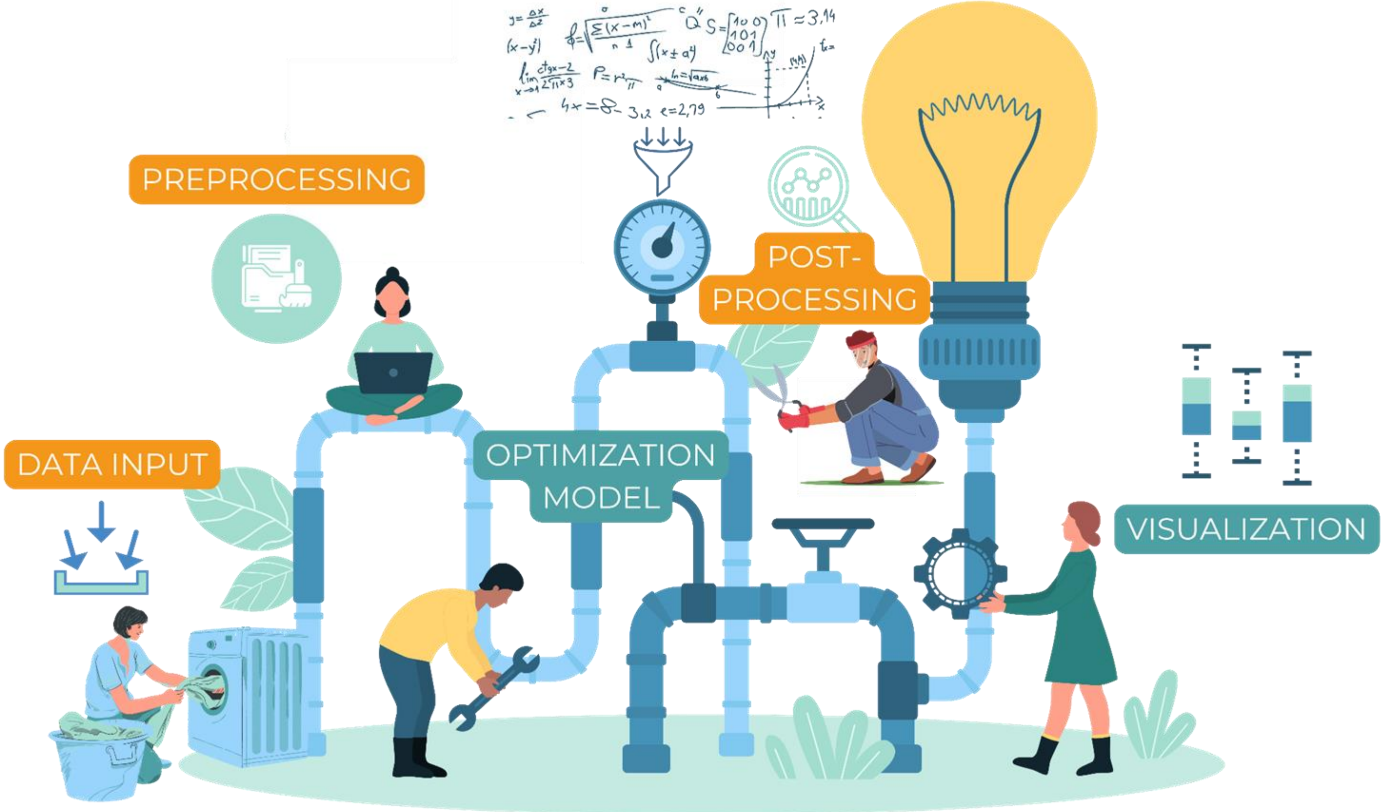
CONTROL

EMBEDDED CODE </>

GAMSPY

EMBEDDED CODE </>

GAMS
MODEL – SOLVE – DEPLOY



EMBEDDED CODE </>

GAMS
MODEL – SOLVE – DEPLOY

- Leverages Pythons strength inside a `.gms` file
  - Allows the use of external code (e.g. Python) during compile and execution time
  - GAMS symbols (sets, parameters, etc.) are shared with the external code

EMBEDDED CODE </>

GAMS MODEL – SOLVE – DEPLOY

Assign values to sets and parameters

Some sort of shortest path algorithm

Rest of the .gms file

```
Set i 'vertices';
Alias (i,j);

Parameters
G(i,j)                      'Length of an edge'
shortest_distance(i,j)      'Shortest distance from i to j'
;

...

EmbeddedCode Python:
def dijkstra(G, starting_node, end_node):
    ...

vertices = list(gams.get('i'))
graph = list(gams.get('G'))
shortest_distance = list()

for source in vertices:
    for sink in vertices:
        dist = dijkstra(graph, source, sink)
        shortest_distance.append((source, sink, dist))

gams.set("shortest_distance", shortest_distance)
endEmbeddedCode shortest_distance

display shortest_distance;

...
```

Streamlining the Optimization Pipeline

GAMS
MODEL – SOLVE – DEPLOY

TRANSFER

CONTROL

&

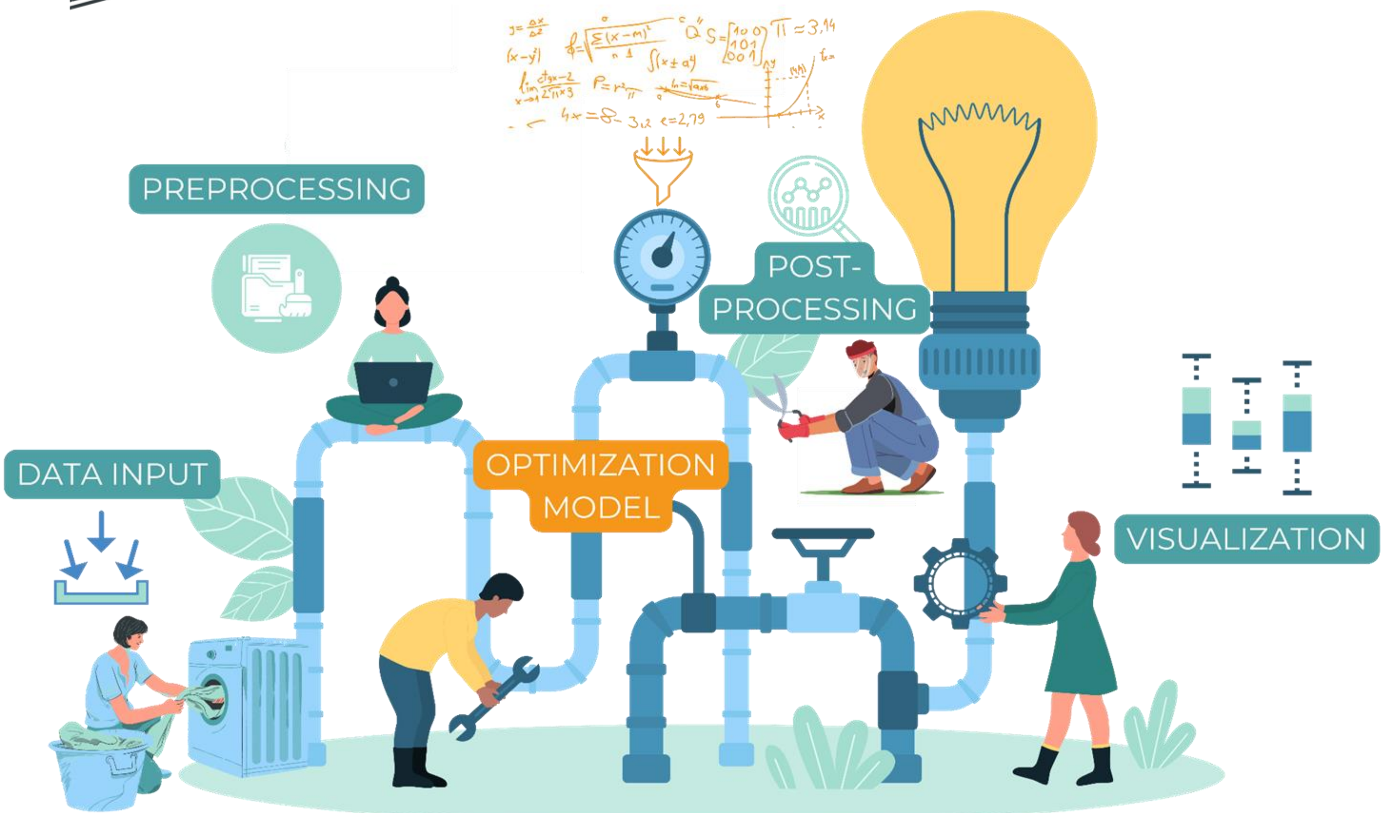
EMBEDDED CODE </>

GAMSPY

GAMSPY

GAMS

MODEL – SOLVE – DEPLOY



GAMSPY

GAMS
MODEL – SOLVE – DEPLOY

Declaration of sets, parameters, and variables as with gams.transfer

transport.py

```python
from gamspy import Set, Parameter, Variable, Equation, Model, Container, Sum

container = Container()

...

supply = Equation(container, name="supply", domain=[i])
supply[i] = Sum(j, x[i,j]) <= a[i]

demand = Equation(container, name="demand", domain=[j])
demand[j] = Sum(i, x[i,j]) >= b[j]

obj = Sum((i,j), c[i,j] * x[i,j])

transport = Model(
                container,
                name="transport",
                equations=['supply','demand'],
                objective=obj,
                sense="min",
                problem="LP")
transport.solve()
```

Contact Us

jbroihan@gams.com
www.gams.com

@GamsSoftware

https://www.linkedin.com/company/gams-development

More GAMS Talks

- GAMS Engine SaaS:
  TE-16, Thursday, 15:50-17:20
- GAMS MIRO:
  FA-16, Friday, 8:30-10:00

Visit us at our booth!

GAMS

MODEL – SOLVE – DEPLOY