GAMS w/ NEOS and Economic Equilibrium Modeling with Julia/JuMP

Adam Christensen

All the best presentations begin with an outline...

- Our goals
- Building out GAMS/NEOS capabilities
- NEOS Demo
- Extended Economic Modeling
- End-to-End Value Proposition
- Experimental Projects

Our Goals

- *100% open source economic database*
- *Members can execute full build stream*
- *Source of canonical models*
- Knowledge base for extended economic modeling
  - Model library for multiple languages
  - Helper tools/data handling
- End-to-End Value Add
  - Data pre-processing tools
  - Model output reduction
  - Visualization
- Economic Software Incubator

Gettin' the Goods...

- **windc.wisc.edu**
  - Click on "Downloads"
- **WiNDC Flavors**
  - Precompiled GDX
  - JSON (download as a zip archive)
  - Re-build your own GDX from source (windc.zip)
  - All original data source files are available (datasources.zip)

DOWNLOADS

WINDC 1 (RELEASE DATE: DECEMBER 2018)				
PRE-COMPILED CORE WINDC DATABASE	GDX (39.8 MB)		JSON (69.2 MB)	
WINDC BUILD STREAM PACKAGE	windc.zip (84.7 MB)			
FULL DATA SET	datasources.zip (157.9 MB)			
SINGLE DATA SETS	BEA	BEA_2007	CFS	USATradeOnline
	SGF	SEDS	PCE	NASS

Releases going forward...

- Platform independent (Windows, Mac)
- Releases will occur ~1-2 times per year
- Data updates and bug reports will be available
- WiNDC releases will always be tested against different versions of GAMS
- Releases will be numbered

Releases going forward...

- Platform independent (Windows, Mac)
- Releases will occur ~1-2 times per year
- Data updates and bug reports will be available
- WiNDC releases will always be tested against different versions of GAMS
- Releases will be numbered

In a nutshell… we want to make it easy for researchers to reference this database in their own publications

GAMS/NEOS Capabilities

What is NEOS?

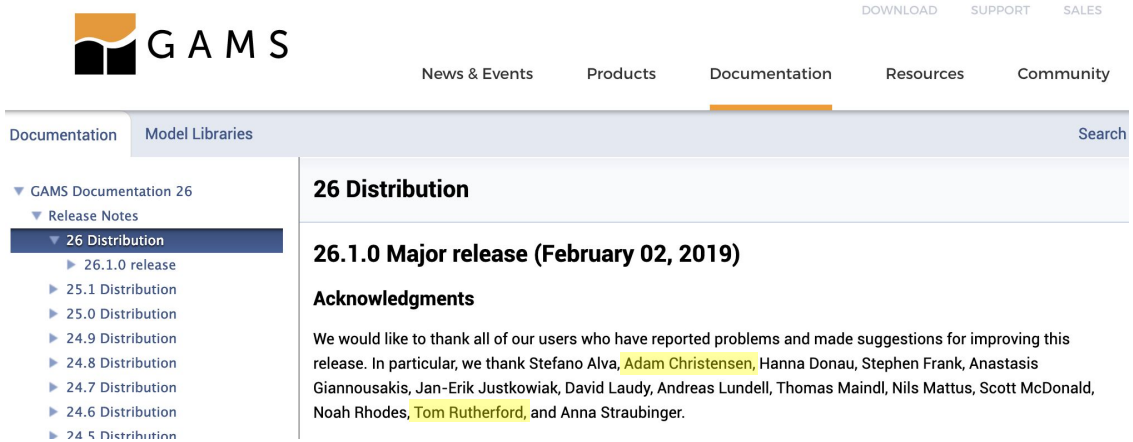
- **N**etwork **E**nabled **O**ptimization **S**ystem
- neos-server.org
- Online access to algorithms that solve many classes of optimization problems
- Jobs can be submitted online through a webform

- **KESTREL** brings your GAMS job to NEOS
  GDX results return as expected

Performance (transit time) penalty -- go get 2 cups of coffee

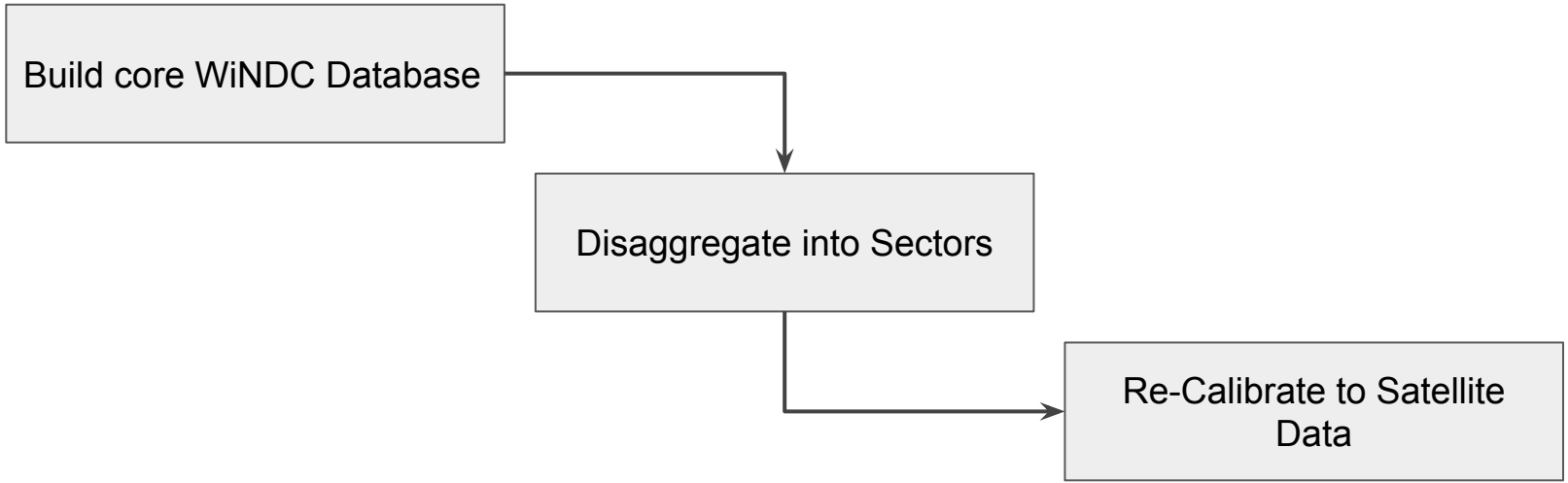
Building out NEOS

- WiNDC members get a base GAMS license to enable NEOS builds
  - Email: adam.christensen@wisc.edu
- WiNDC/NEOS build requires GAMS 26.1.0
  - KESTREL support added for MPSGE models



New Developments

- WiNDC 2.0 database build stream available very soon! (windc.wisc.edu)
  - Updated data (included 2016 data)
  - If building locally -- compatible back to GAMS 24.3.3 (July 2014)
  - More error checking (will stop running if an EXECUTE statement does not finish properly)
  - More modular interface (build, sectoral disaggregation, and re-calibration)



New Developments

- WiNDC 2.0 database build stream available very soon! (windc.wisc.edu)
  - Updated data (included 2016 data)
  - If building locally -- compatible back to GAMS 24.3.3 (July 2014)
  - More error checking (will stop running if an EXECUTE statement does not finish properly)
  - More modular interface (build, sectoral disaggregation, and re-calibration)

To build the core WiNDC database locally:

```
gams run.gms
```

To build with NEOS:

```
gams run.gms --neos=yes
```

Sectoral Disaggregation

- Options are: 405, bluenote, nass, embodiedcarbon, example
  - **405:** full 405 sector disaggregation, not fully functional (small number problems)
  - **bluenote:** electricity power generation, coal mining, petroleum refineries
  - **nass:** farming subsectors (oilseed farming, grain farming, etc.)
  - **embodiedcarbon:** electricity power generation, coal mining, petroleum refineries

To disaggregate the core WiNDC database locally:

```
gams disagg.gms --aggr=bluenote
```

To disaggregate on NEOS:

```
gams disagg.gms --aggr=bluenote --neos=yes
```

Re-Calibration

- Options are: bluenote, nass
  - **bluenote:** calibrate to EIA SEDS data 1997..2014, and 2016
  - **nass:** only 2012 data for now

To re-calibrate the core WiNDC database locally:

```
gams recalibrate.gms --satdata=bluenote --year=XXXX
```

To re-calibrate the core WiNDC database with NEOS:

```
gams recalibrate.gms --satdata=bluenote --year=XXXX --neos=yes
```

Quick NEOS Demo

Open to User Feedback...

- Should be easy to use
- Should be flexible to suit user's needs
- Must maintain platform independence to a high degree

Feedback: adam.christensen@wisc.edu

Extended Economic Modeling

Some things...

- **Julia** is the base language, **JuMP** is the math programming package for Julia

- Codebase on both change frequently -- can be frustrating
  - Julia / JuMP is totally open source

- JuMP offers connections to many solvers
  - Cbc, Clp, CPLEX, CSDP, ECOS, FICO Xpress, GLPK, Gurobi, **Ipopt**, MOSEK, OSQP, SCS, SeDuMi
  - Types: LP, QP, **NLP**, MILP, SOCP, MISOCP, SDP
  - Note: PATHsolver.jl is “available” but not robust

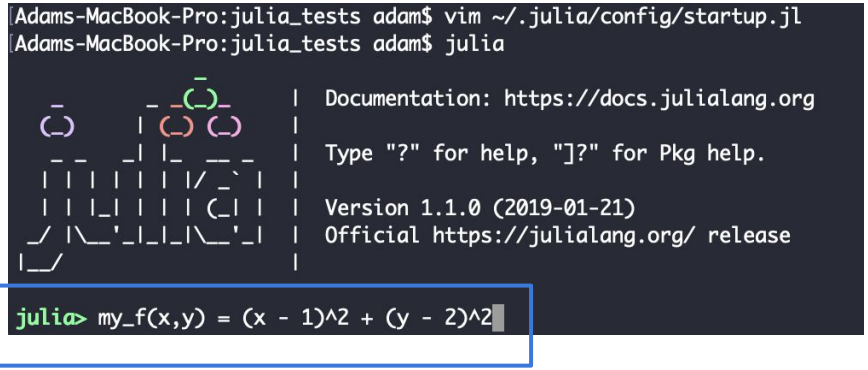
One nice techie thing...

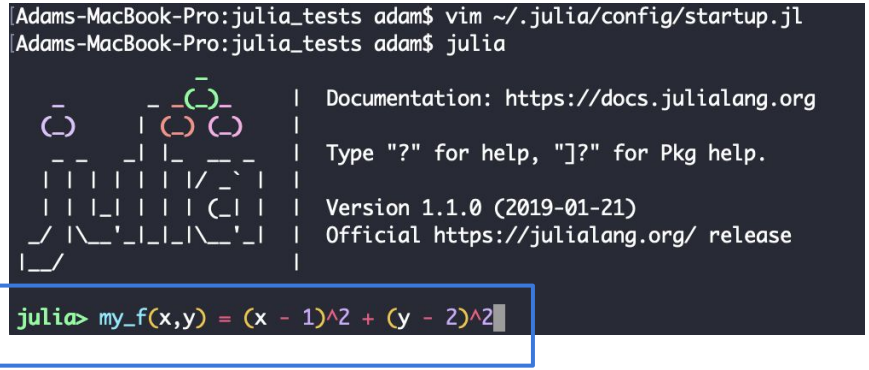
`add OhMyREPL`... then create a `startup.jl` file in...

```
~/.julia/config
```

... that contains

```
using OhMyREPL
```





Getting Data into Julia/JuMP

- Julia/JuMP can approximate “set” notation like GAMS
- Associative arrays (aka “dictionaries”) are the key
  - {key:value} pairs

```
b(j) 'demand at market j in cases'
        / new-york   325
          chicago    300
          topeka     275 /;
```

GAMS

```
julia> b = parse_data("b")
Dict{Any,Any} with 3 entries:
  "new-york" => 325.0
  "chicago"  => 300.0
  "topeka"   => 275.0
```

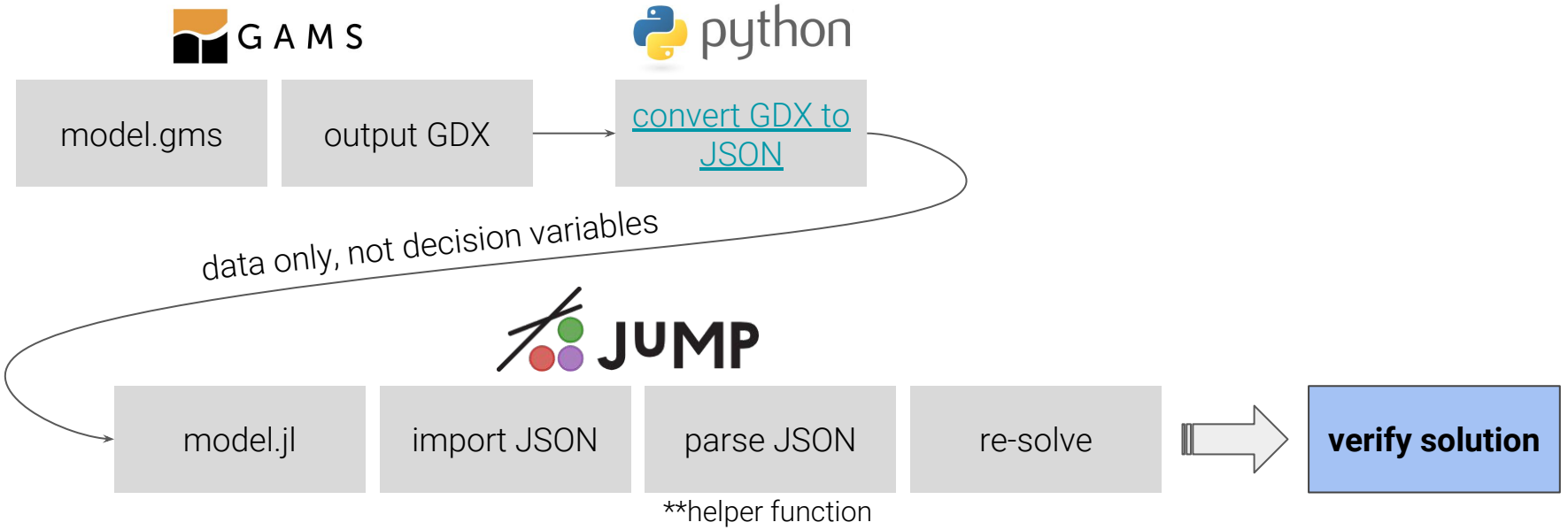
JUMP

JSON files are good for this

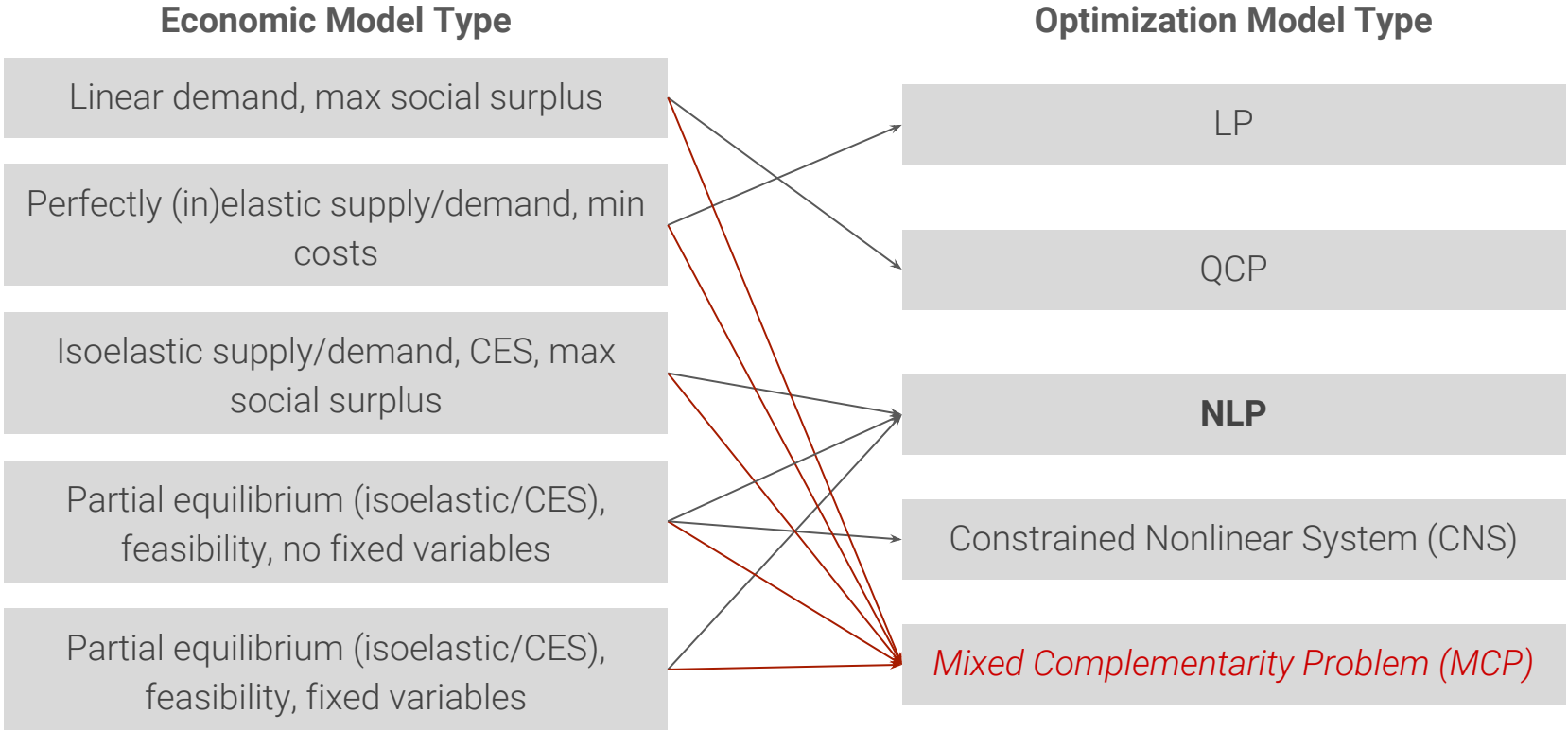
Getting Data into Julia/JuMP


```
{
    "b": {
        "type": "GamsParameter",
        "dimension": 1,
        "domain": [
            "j"
        ],
        "number_records": 3,
        "text": "demand at market j in cases",
        "values": {
            "domain": [
                "new-york",
                "chicago",
                "topeka"
            ],
            "data": [
                325.0,
                300.0,
                275.0
            ]
        }
    }
}
```

Julia/JuMP Development Cycle



Mental Mapping



Example Problem

Partial equilibrium (isoelastic supply, CES)

Regional Trade

Differentiated Goods

No Objective Function (Square System)**

**Fixing variables destroys the square-ness in GAMS thanks to the presolver, thus the need for a zero objective function and the NLP solver (instead of just CNS)

Partial Equilibrium Trade (r→r') Model

- Supply function (calibrated isoelastic)

$$Y_r = \bar{Y}_r \left(\frac{p_r}{\bar{p}_r} \right)^{\eta}$$

- Compensated CES demand function

$$X_{r,r'} = \bar{X}_{r,r'} \left(\frac{C_{r'}}{p_r} \right)^{\sigma_{r'}} C_{r'}^{-\alpha_{r'}}$$

- Cost function (calibrated CES form)

$$C_r = \left[\sum_{r'} \theta_{r',r} \left(\frac{p_r}{\bar{p}_r} \right)^{1-\sigma_r} \right]^{1/(1-\sigma_r)}$$

- Supply & Demand Balancing

$$Y_r = \sum_{r'} X_{r,r'}$$

```
variables P(r)      Equilibrium price,
          Y(r)      Equilibrium supply,
          C(r)      Unit cost,
          X(r,rr)        Demand
          OBJ       Vacuous objective;

equations objdef, output, supply, demand, cost;
output(r)..    Y(r) =e= sum(rr, X(r,rr));
supply(r)..    Y(r) =e= y0(r) * P(r)**eta(r);
demand(r,rr).. X(r,rr) =e= x0(r,rr) * (C(rr)/P(r))**esub(rr) * C(rr)**(-sigma(rr));
cost(r).. C(r) =e= sum(rr, theta(rr,r) * P(r)**(1-esub(r)))**(1/(1-esub(r)));
```

GAMS

```
m = Model(with_optimizer(Ipopt.Optimizer))

@variable(m, P[i in r], start=1)
@variable(m, Y[i in r], start=y0[i])
@variable(m, C[i in r], start=1)
@variable(m, X[i in r, j in r], start=x0[i,j])

@constraint(m, output[i in r], sum(X[i,j] for j in r) == Y[i] )
@NLconstraint(m, supply[i in r], Y[i] == y0[i] * P[i]^eta[i] )
@NLconstraint(m, demand[i in r, j in r], X[i,j] == x0[i,j] * (C[j]/P[i])^esub[j] *
C[j]^(-sigma[j]) )
@NLconstraint(m, cost[j in r], C[j] == sum(theta[i,j] * P[j]^(1-esub[j]) for i in
r)^(1/(1-esub[j])) )
```

JUMP

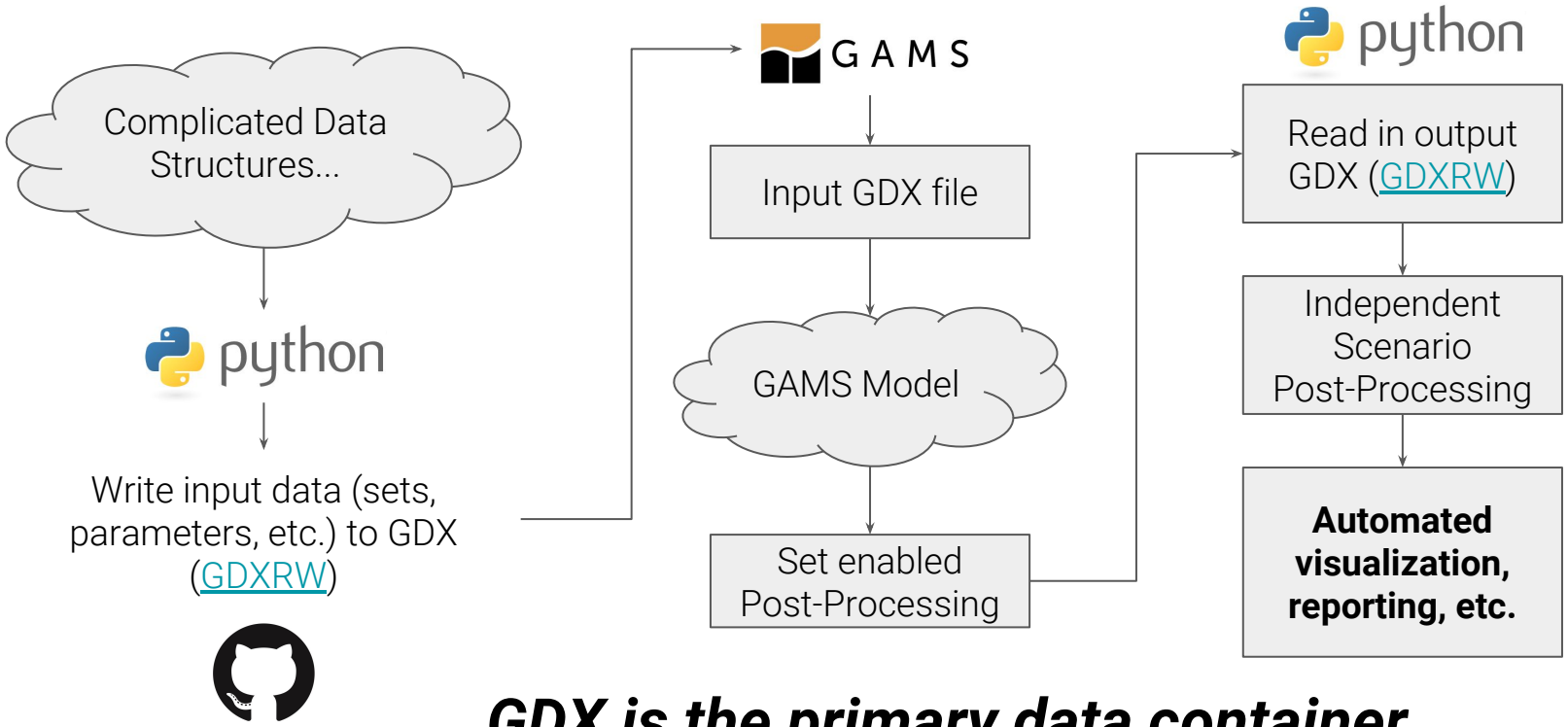
More GAMS → Julia/JuMP Examples Online

https://github.com/uw-windc

- Markusen's M2-3 model (maximize utility, 2 Cobb-Douglas commodities, with rationing)
- Markusen's M2-5 model (maximize utility, 1 good, 1 factor, 1 consumer)
- PIESQCP formulation (William Hogan, 1975) (maximize social welfare)
- (Spatial) Partial Equilibrium (as seen in this presentation)
- More to come...

End-to-End Value

Python/GAMS Workflow...

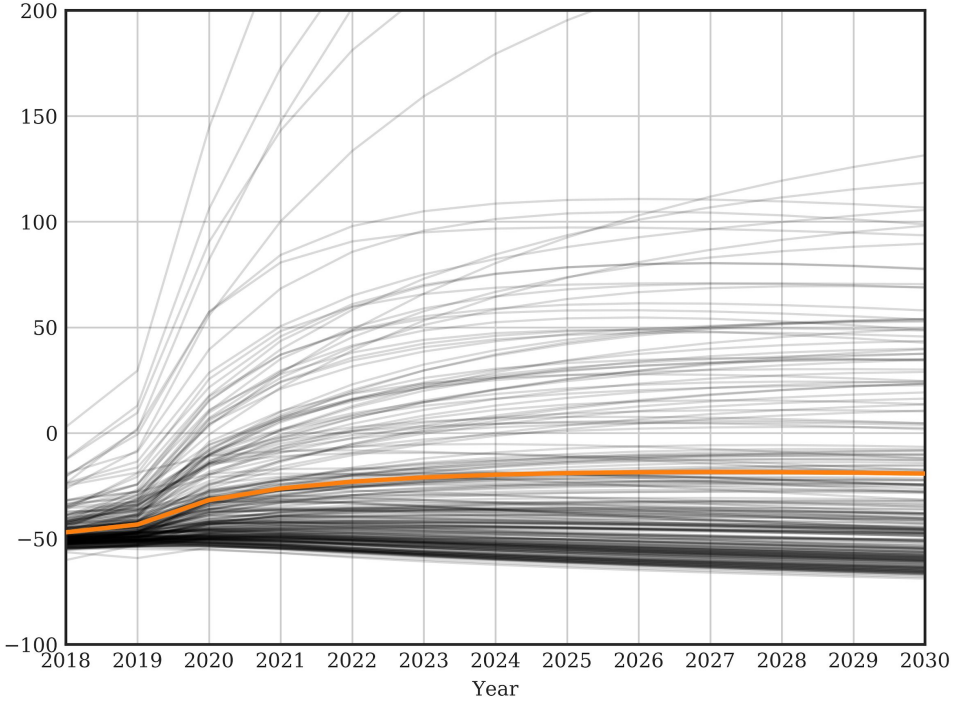


GDX is the primary data container

GDXRW

Example

Automated Plot Creation...



Volume of data can be enormous

All data read directly from the output GDX

Automation reduces chances of silly errors

Advanced Visualization

- Geocode data on the fly, calculate real distances/times on a road network

```
$set key PIzSygmXdMewURGxUD38S2t4VqBQEyVA

execute 'python distance.py --key=%key% --input=query.csv
--output=output.csv';
```

- Connect data directly to maps to debug modelling errors or present results

Mapping Example

Python, Folium, Google API

WiNDC Capabilities

Custom workflows can be designed and implemented

Primary Tools:

- GAMS
- Julia/JuMP
- Python (numpy, scipy, pandas, folium, matlibplot, etc.)
- Expanding our visualization capabilities (D3)

Software Incubator

Pilot-Scale Projects

- Pivot Tables

- Graph-based CES syntax

Pivot Tables

- Incredibly useful to reduce data
- Not easy in GAMS (we want to preserve data as 2D)
- GPivot
  - Can be executed in from a .gms file
  - Reads a GDX for data
  - Can pass a query (SQL-like) in order to create differently scoped pivot tables

Example

Graph-Based CES Functions

- CES functions are an economic powerhouse
- Nested CES functions offer even more flexibility

- Plagued by messy algebra
- MPSGE handles this, but is not available for other platforms (Julia/JuMP)

Investigating ways to rapidly define complicated economic functions

Proof of Concept

- Working in the GAMS framework
- Utilize GAMS EMP (symbolic differentiation & reformulation)

Consumer's Utility Function:

```
EQUATION objConsumer(h);
objConsumer(h)..
consumerUtility(h)
=E=
sum(s, alpha(s,h)**(1/sigmac(h)) *
(x(s,h))**((sigmac(h)-1)/sigmac(h)))**(sigmac(h)/(sigmac(h)-1));
```

Proof of Concept

Define Consumer's Utility Function with Graph Syntax:

```
objConsumer(h)
{'!consumerUtility(h)':['x(s,h)']}
@esub{'consumerUtility':'sigmac(h)'}
@shares{'consumerUtility':'alpha(s,h)'}
```

! ⇒ top node

Proof of Concept

Define Consumer's Utility Function with Graph Syntax:

```
objConsumer(h)
{'!consumerUtility(h)':['x(s,h)']}
@esub{'consumerUtility':'sigmac(h)'}
@shares{'consumerUtility':'alpha(s,h)'}
```

main CES structure
define elasticities for each nest level
define share coefficients

Proof of Concept

Define Consumer's Utility Function with Graph Syntax:

```
objConsumer(h)
{'!consumerUtility(h)':['x(s,h)']}
@esub{'consumerUtility':'sigmac(h)'}
@shares{'consumerUtility':'alpha(s,h)'}
```

main CES structure
define elasticities for each nest level
define share coefficients

Only regex string parsing

```
EQUATION objConsumer(h);
objConsumer(h)..
consumerUtility(h)
=E=
sum(s, alpha(s,h)**(1/sigmac(h)) *
(x(s,h))**((sigmac(h)-1)/sigmac(h)))**(sigmac(h)/(sigmac(h)-1));
```

GAMS/EMP VERSION VERIFIED AGAINST PURE MCP FORMULATION

Thanks.

WiNDC wants to work with you