GAMS vs. Python MINIMOD Optimization: A Comparison

Aleksandr Michuda

Technical Summary

- In the summer of 2020, work was undertaken to port the GAMS MINIMOD optimization routine with all associated constraints, to Python
- As of September 2020, all existing functionality has been ported and work has been done to add visualizations to the package
- A comparison of a GAMS robustness script was done as an ultimate test of comparison between GAMS and python
  - Using the nutritional benefits and costs data from Cameroon
- 1000 Monte Carlo simulations were run in both GAMS and Python in order to check for the differences between Python and GAMS
- Results suggest that there are discrepancies, but results are similar.
  - Discrepancies can be explained by differences in the way that GAMS might draw new samples as a result of the simulations

Introduction

The work on Python MINIMOD began in the Summer of 2020 and its intent was to port all existing functionality from the GAMS MINIMOD optimization tool. This included the implementation of a Mixed Integer Programming (MIP) solver for the problem of finding the optimal set of nutritional interventions across time and space.

Setup

The data that was used on Python and GAMS was the same. Some post-processing of the data had to be done in Python in order to mirror the changes done in GAMS, but a check of data differences concludes that the two datasets were the same.

Although Python MINIMOD can create space and time constraints based on string matching in intervention names, the comparison uses the exact interventions that GAMS uses, which are based on manually writing in all interventions. Since the objective in this case is to compare GAMS against Python, not make sure that the constraints were necessarily defined correctly, this was a more prudent thing to do for the comparison.

The GAMS code, in fact was not changed at all. It was only run and the output statistics were saved and will be presented. The only differences between the two settings are as follows:

- Using Python vs. GAMS
  - Even when using the exact same model, using different softwares can lead to differing results. This is mostly going to come from differences in solver settings and the choice of solver, as well as how the inherent probabilistic nature of a solver may lead to different optimal results. The Monte Carlo simulations are meant to protect against those differences.
- Differences in the Models
  - This has been verified to not be an issue, but is here for completeness. It may be that the way that constraints are defined are different between GAMS and Python by the nature of the programming languages being different; extensive testing and iteration with Justin Kagin (the developer of the GAMS code) has shown, however, that the constraints are defined correctly and identically across Python and GAMS.

The constraints that were used across each model are as follows:

- Constraining all fortified oil, fortified cube, unenhanced oil and fortified maize interventions to be national
- Allowing three year startup costs for fortified cube and maize.
  - These choices were made directly as a result of the GAMS code.
- Using a national intervention in all 10 years of capsules (vitamin A?) and unenhanced oil as the Business as Usual (BAU*).

Since the effective coverage data has standard errors for each interventions, a normal distribution was constructed around

Python vs GAMS

After running 1,000 simulations, we found that all simulations converged and found optimal solutions. Table 1 shows the percentage of each intervention across the two platforms. It seems that the percentage of Cube is very similar, but there is discrepancy for unenhanced oil as well as fortified maize.

%	GAMS	Python
VA Oil	100	77.5
VA Cube	88.1	90.1
Maize	3.6	22.5
DW (Unenhanced) Oil	76.8	17.5

Apart from that, we can also give a point estimate of costs and benefits with a confidence interval. fig. 1 illustrates that the differences in this case are very similar.

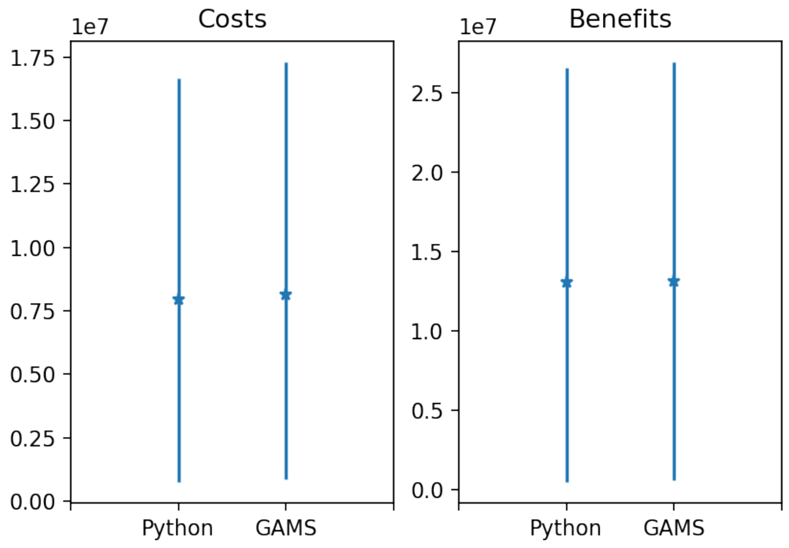

Figure 1: Total Costs and Benefits with Confidence Intervals, Python vs. GAMS

We can also plot the change over time of the first draw of the GAMS simulations with all the draws of the Python simulations. *2 shows benefits across time and* @fig:traj__cost shows the same for costs.

Appendix

Below we present some visualizations from the Python Simulations.

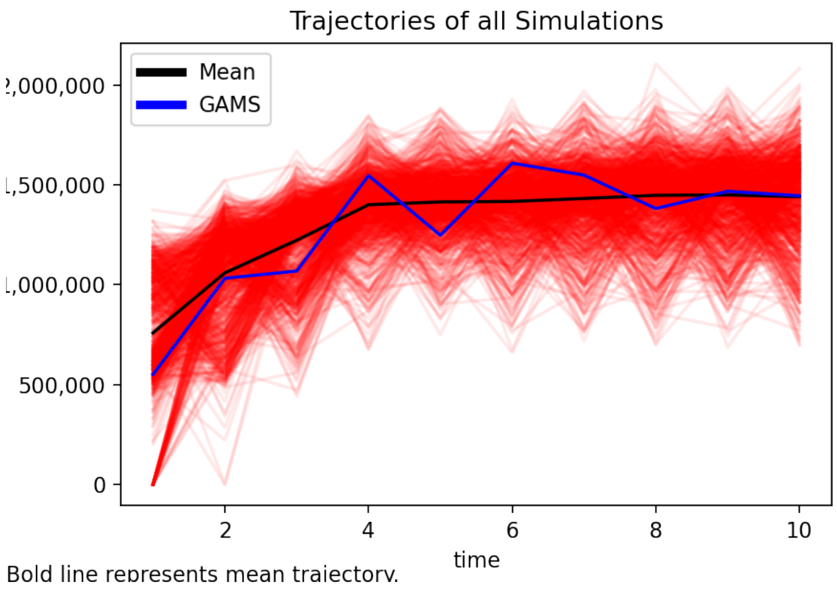

Figure 2: Benefits across all Simulations, Python vs. GAMS

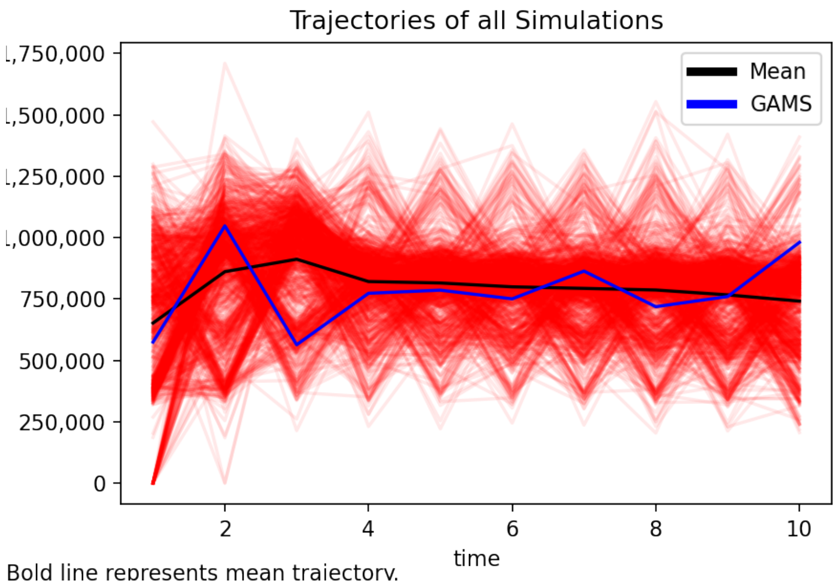

Figure 3: Costs across all Simulations, Python vs. GAMS

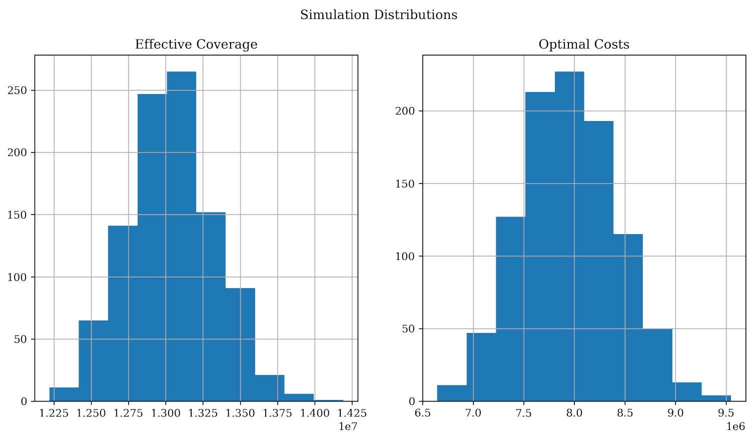

Figure 4: Distribution of Optimal Effective Coverage and Optimal Costs for Simulations