GAMS Transfer

A Unified System to Move Data

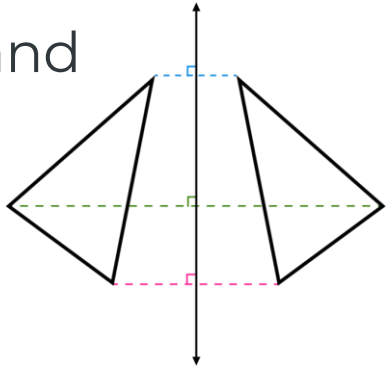
Adam Christensen

Atharv Bhosekar

Renke Kuhlmann

Self-Reflection

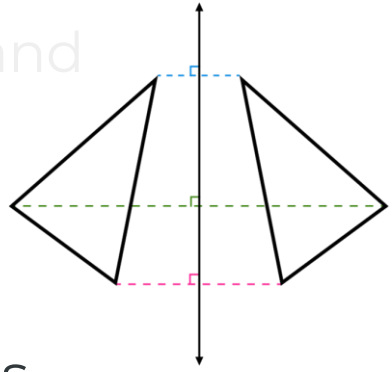
- It should be easier to move data to/from GAMS
- Open source data science tools are powerful and popular



- Should harmonize syntax across platforms
- Should have convenience functions to make environment-specific data conversions easy
- Must perform, but perhaps not at all costs

Self-Reflection

- It should be easier to move data to/from GAMS
- Open source data science tools are powerful and popular



- Should harmonize syntax across environments
- Should have convenient functions to make environment-specific data conversions easy
- Must perform, but perhaps not at all costs

What is it?

- API
- Connects to GDX (file based) and GMD (in memory)
- `gamstransfer` will exist for several languages
  - Python (to be released with GAMS 37)
  - Matlab/Octave (to be released with GAMS 37)
  - R (release date TBD)
  - perhaps others

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

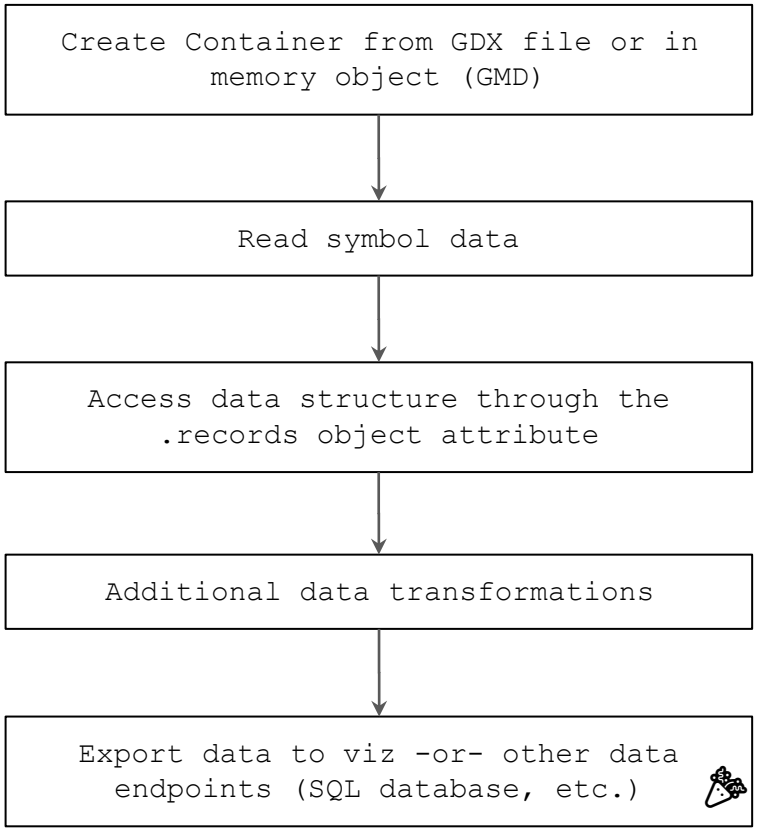
Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Design Philosophy

- Centered around a data `Container`
- Symbol objects are created & added to the `Container`
  - Meta data (i.e., descriptions, etc.)
  - Record data (i.e., the actual data)
- Symbol objects in the `Container` can be linked together
  - Enables domain violation detection, implicit set growth, etc.
- Symbol records have a standardized format
  - Python: Pandas DataFrame, numpy arrays
  - Matlab: struct, table, dense matrix, sparse matrix
- Record format leverages a large collection of I/O tools
- C/C++ read/write power calls make this fast
- `Container` object owns the bulk read/write methods

Read: example workflow



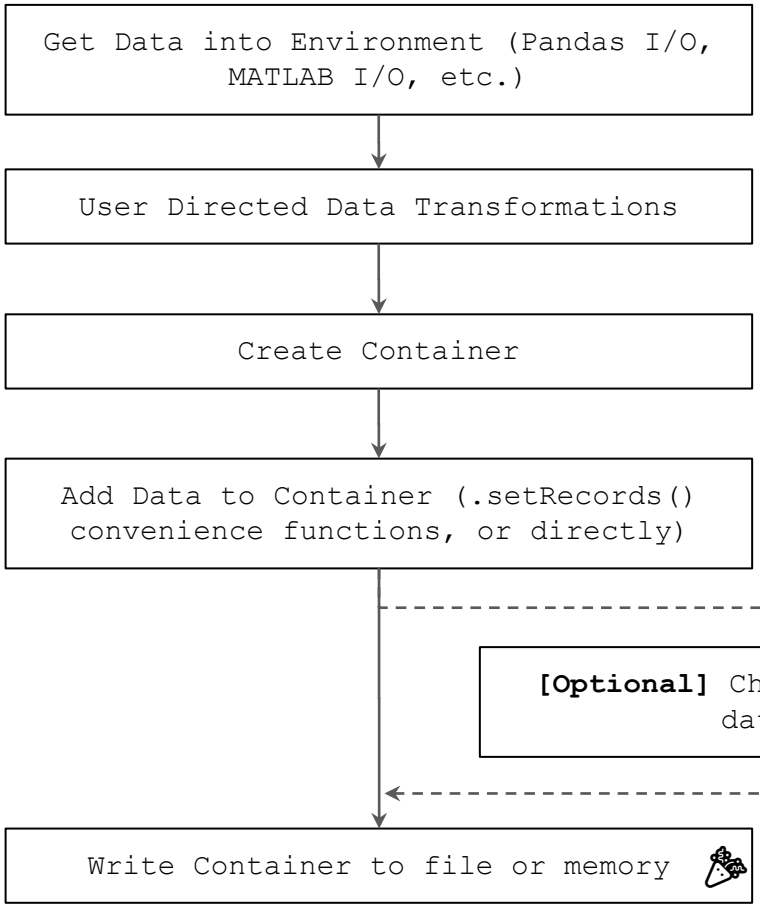
As a convenience a user could pass a GDX file string to the Container constructor and read all the data

Read all symbols easily with one bulk call, or specify a subset of symbols. The read function accepts a data startpoint -- enables adding symbols from multiple sources

groupby, pivot_table, melt, stack, unstack, *plus* others!

plotly/Dash, matplotlib, bokeh, Geoplotlib, etc.

Write: example workflow



Excel, csv, SQL, html, txt, json, feather, *plus* others!

groupby, pivot_table, melt, stack, unstack, *plus* others!

Can be created from an existing object or create an empty Container

Data can be added to a container to perform a symbol merge operation

`*gamstransfer` includes some data debug features

Why use `gamstransfer`?

- Data-only API
  - not a “model” API
  - `gamstransfer` is not a replacement for other APIs we offer
- Emphasis on user convenience
  - Feedback/experience says that users “just want the data!” 
  - Bulk read and write calls
  - Better structural integration to different environments
- `gamstransfer Containers` are state aware
  - Enables new functionality
  - Do GAMS-like things, but in a convenient data environment

Users that want to move data between environments will be interested in `gamstransfer`

Demo