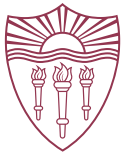


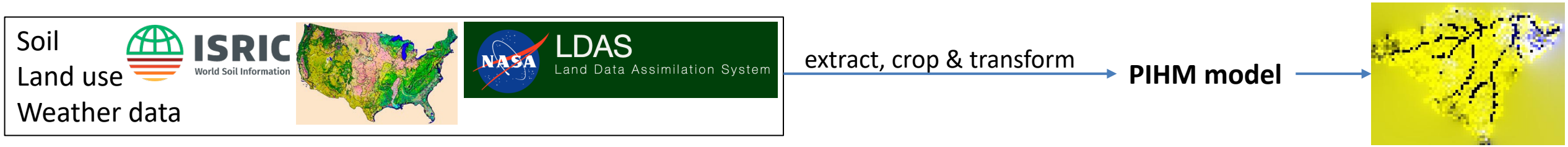
CREATING A **FAIR** DATA CATALOG TO SUPPORT SCIENTIFIC MODELING

Basel Shbita, Binh Vu, Minh Pham, Dan Feldman

Craig Knoblock, Jay Pujara, Yao-Yi Chiang

Motivating Example

- Finding, preparing, and cleaning datasets dominate time devoted to scientific inquiry
  - Example: flooding prediction takes **months** for data prep.



- Need a Data Catalog
  - Retrieve datasets by variables (e.g., SVO) and other metadata (temporal and geospatial coverage)
  - Transform dataset for different use cases with minimal effort
    - Reprojection, cropping, or format conversion
    - Joining multiple datasets
  - Visualize dataset to get insight of the data

Challenges in building the Data Catalog

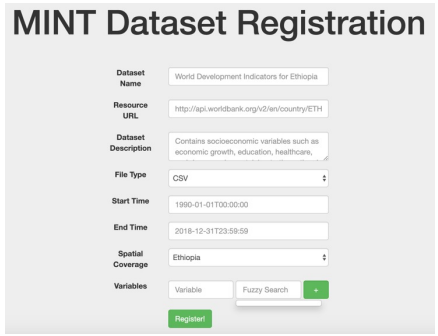
- Registering data in the data catalog
  - Huge number of datasets with **massive amount of data**
  - Lots of **manual effort** for curating the datasets
    - Ambiguous data definition: different communities using **diverse terms** for same phenomena
    - Dataset are stored in **different formats** and layouts: NetCDF, CSV, spreadsheets…
- (Semi)-automatic data transformation
  - Generate & execute a transformation plan based on input and **desired output**
  - Easy to **reuse** existing transformation libraries or add new transformation

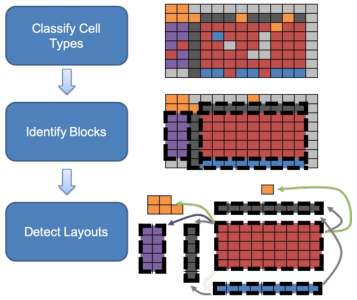
Overall approach

Indicator Name	Units	2005	2006
Cargo Traffic (M.Tor	Million tonne	6915000.00	7100000.00
Cargo Traffic (M.Tor	Million tonne	204929.00	182810.00
Container Traffic (TE	TEUs		
Cargo Traffic (M.Tor	Million tonne	5432353.00	5489586.00
Cargo Traffic, Annua	Volume		
Cargo Traffic, Annua	%		

dataset

Registration





Automated table analysis

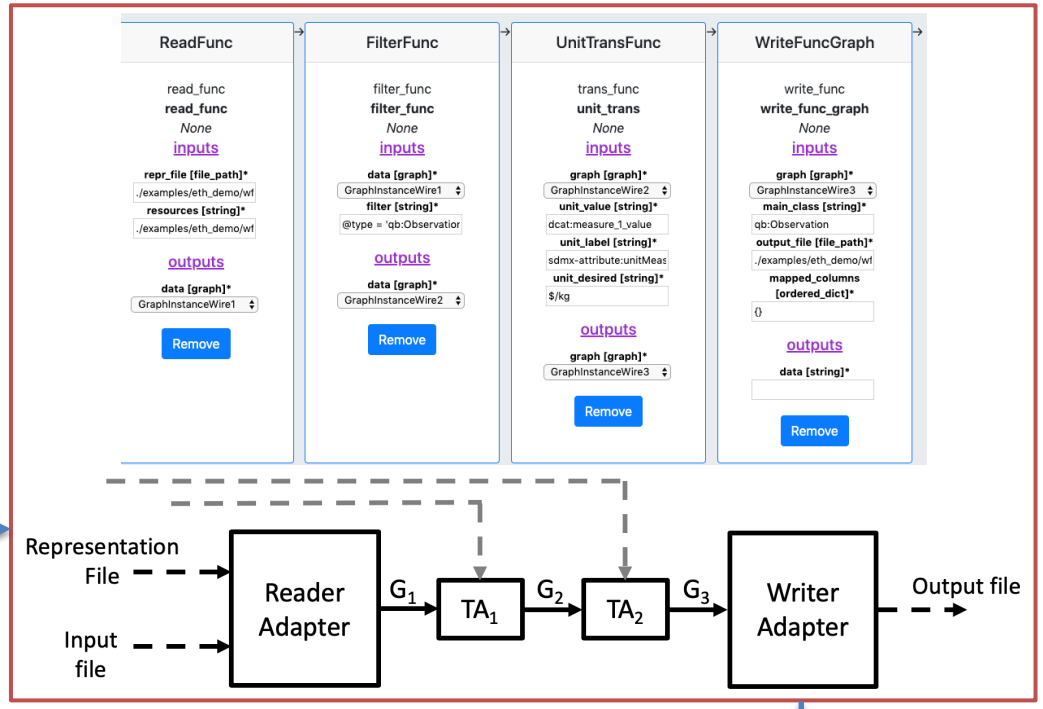
Building dataset representation

Unit detection

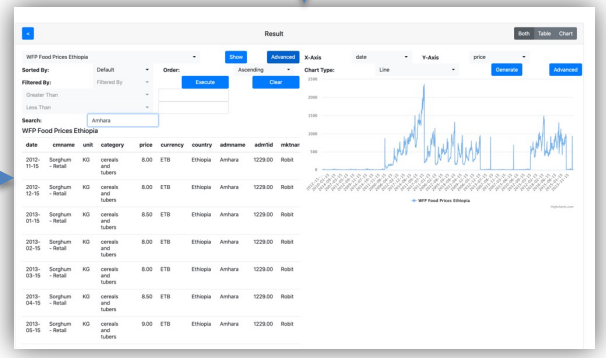
Fuzzy Search

Data Catalog

Data Transformation



Visualization

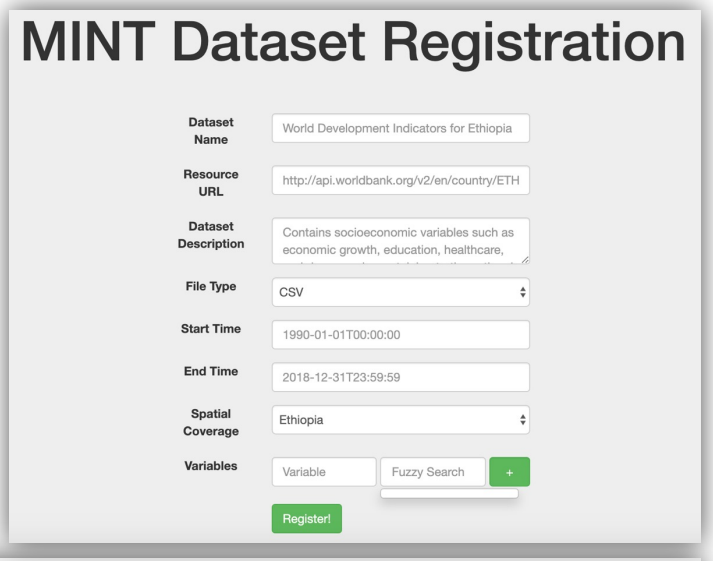


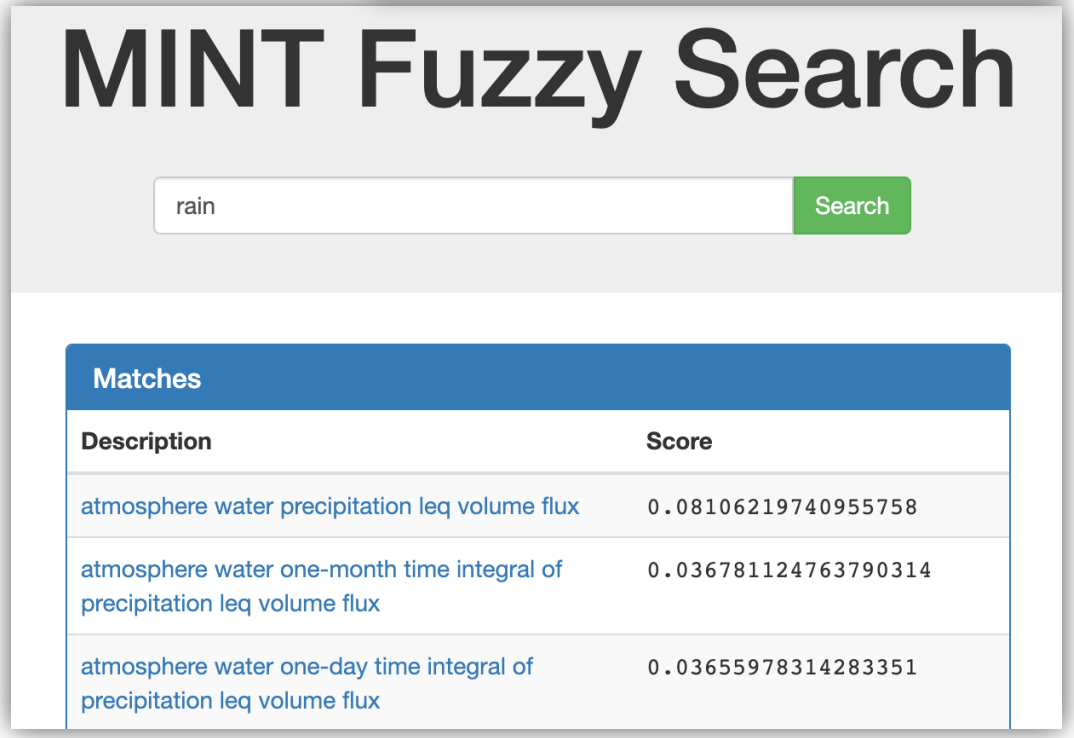
Information Sciences Institute

USC Viterbi School of Engineering

Registering data

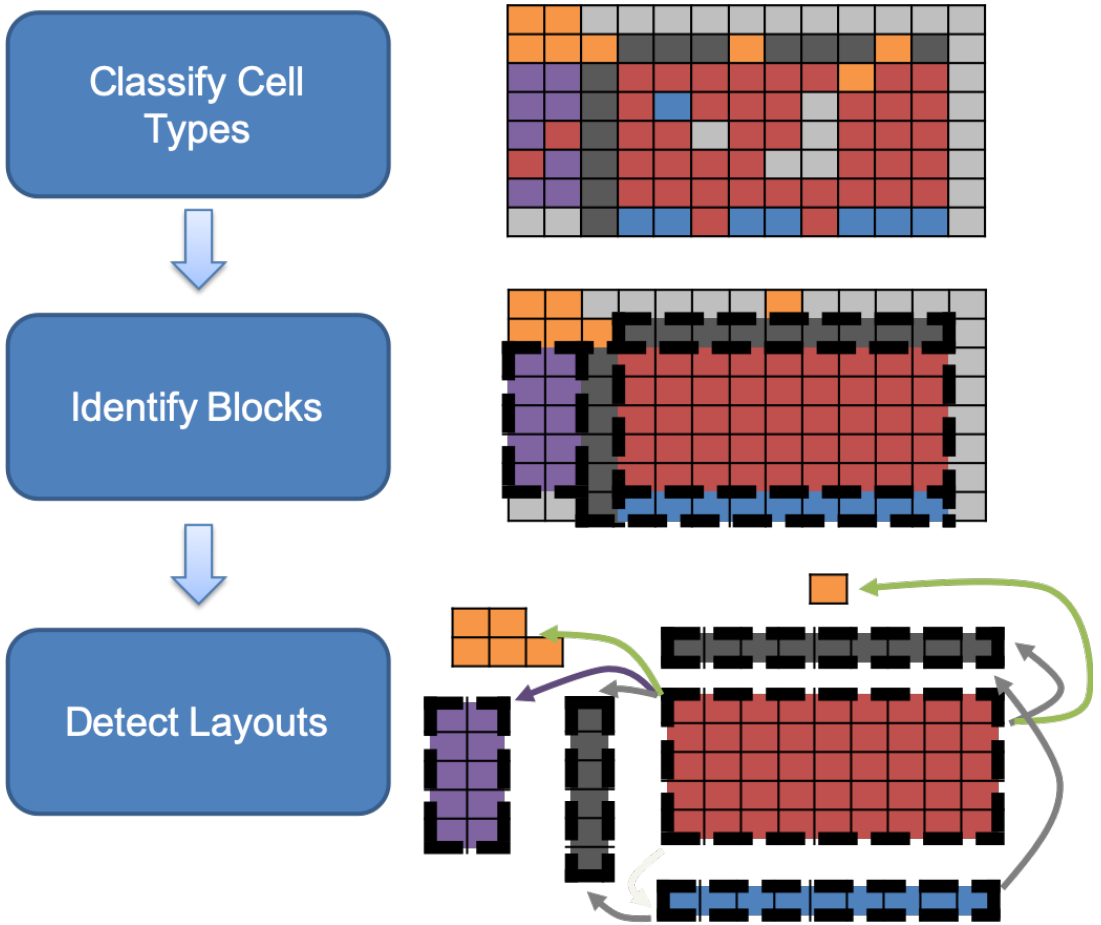
- Dataset registration
  - Can be done via API or UI
  - Provide basic metadata: variable, spatial and temporal coverage
- Finding correct variable (fuzzy search)
  - Semantic knowledge in Wordnet
  - Statistical associations with Word2Vec
  - Topic modeling and other string similarity metrics





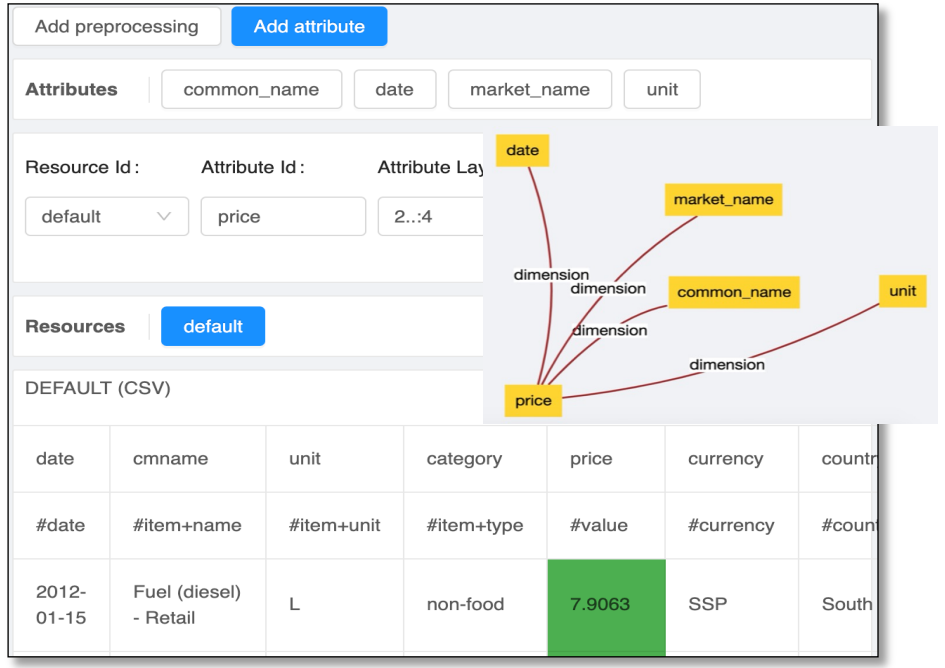
Building rich data understanding

- Automated table analysis [2]
  - Input: tabular datasets
  - Output: dataset layout
    - Header/Attribute/Value blocks
    - Their relationships



Building rich data understanding (cont.)

- Dataset representation (D-REPR [3]). Why?
  - Different **formats** (NetCDF, spreadsheet, CSV, JSON)
  - Different **layouts** (matrix tables, hierarchical tables)
  - **Same interface** to access to many kinds of datasets
- Automatically generate D-REPR file
  - Table understanding
  - D-REPR GUI for curation

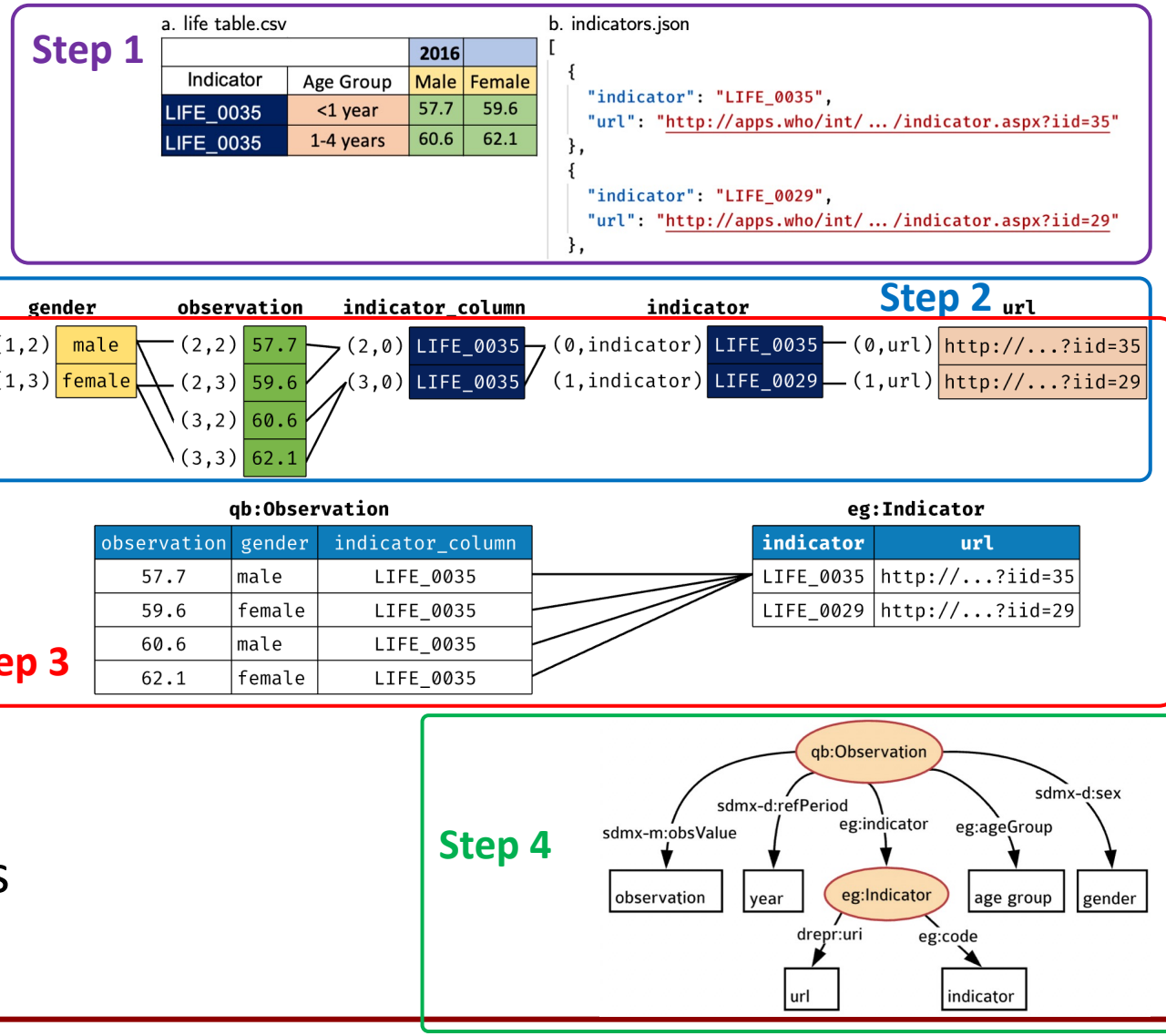


FY 2008	JAN	FEB	MAR
From domestic sugar beets	661,586	485,126	423,775
From imported sugar beets	0	37,160	0
Subtotal	661,586	522,287	423,775
Cane production:			
Florida	321,414	253,438	242,560
...			..
Subtotal	378,919	283,190	289,237
Total	1,040,505	805,476	713,012

Indicator Name	Units	2005	2006
Cargo Traffic (M.Tor	Million tonne	6915000.00	7100000.00
Cargo Traffic (M.Tor	Million tonne	204929.00	182810.00
Container Traffic (TE	TEUs		
Cargo Traffic (M.Tor	Million tonne	5432353.00	5489586.00
Cargo Traffic, Annua	Volume		
Cargo Traffic, Annua	%		

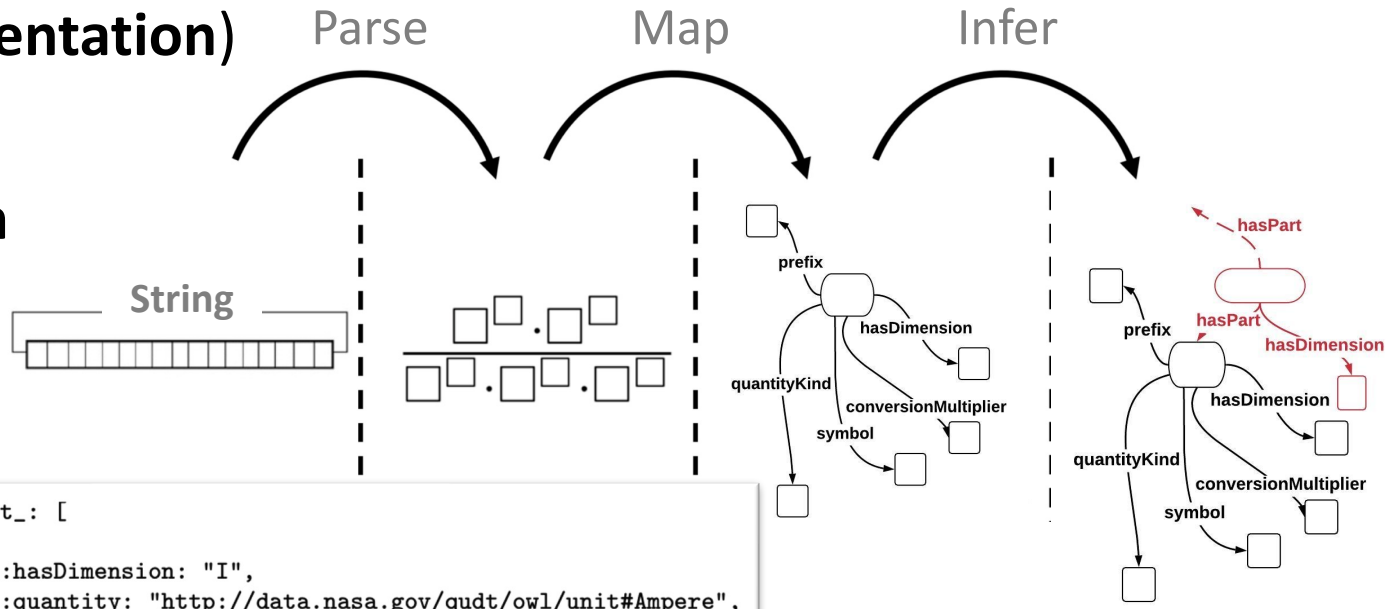
Building rich data understanding (cont.)

- Four steps to create a D-REPR file
  1. Specify resources
  2. List attributes (or variables) in the datasets
  3. Simple rules to join values of the attributes
  4. Map attributes to predicates, classes in domain ontologies



Building rich data understanding (cont.)

- Unit detection (CCUT [1])
  - **Identify**, parse, and map compound units of measurement to QUDT ontology (**semantic representation**)
  - Enable **automatic** unit **conversions**
  - Enhance the dataset **representation**



	A	B	C
2	Year in Production	Units	200
3	Technology Generation		
4	Physical Lgate (Low-Standby-Power)	nm	75
5	EOT (Equivalent Oxide Thickness)	A	22
6	Gate Poly Depletion & Inversion-Layer Thickness	A	8
7	Inversion Gate Dielectric Thickness Value	A	30
8	Maximum Gate Leakage Limit	A/cm^2	4.4E
9	Power Supply Voltage	V	1.2
10	Saturation Threshold Voltage	V	0.5

READ ME | PIDS LSTP Table | Worksheet LSTP | Plots | +

```
qudt:_hasPart_: [
    {
        qudt:hasDimension: "I",
        qudt:quantity: "http://data.nasa.gov/qudt/owl/unit#Ampere",
        qudt:symbol: "A"
    },
    {
        UNK:exponent: "-2",
        UNK:prefix: "http://data.nasa.gov/qudt/owl/unit#Centi",
        UNK:prefix_conversion_multiplier: 0.01,
        qudt:hasDimension: "L",
        qudt:quantity: "http://data.nasa.gov/qudt/owl/unit#Meter",
        qudt:symbol: "cm"
    }
],
qudt:abbreviation: "A cm-2",
qudt:hasDimension: "L-2 I"
```

Arpeggio

Querying data from data catalog

- Search datasets (API-based) by
  - dataset name
  - variables
  - temporal coverage
  - spatial coverage (bounding box)
- Return a subset of data in different format
  - Using data transformation component

Example queries can be found in the Api Demo notebook

Searching for datasets/resources

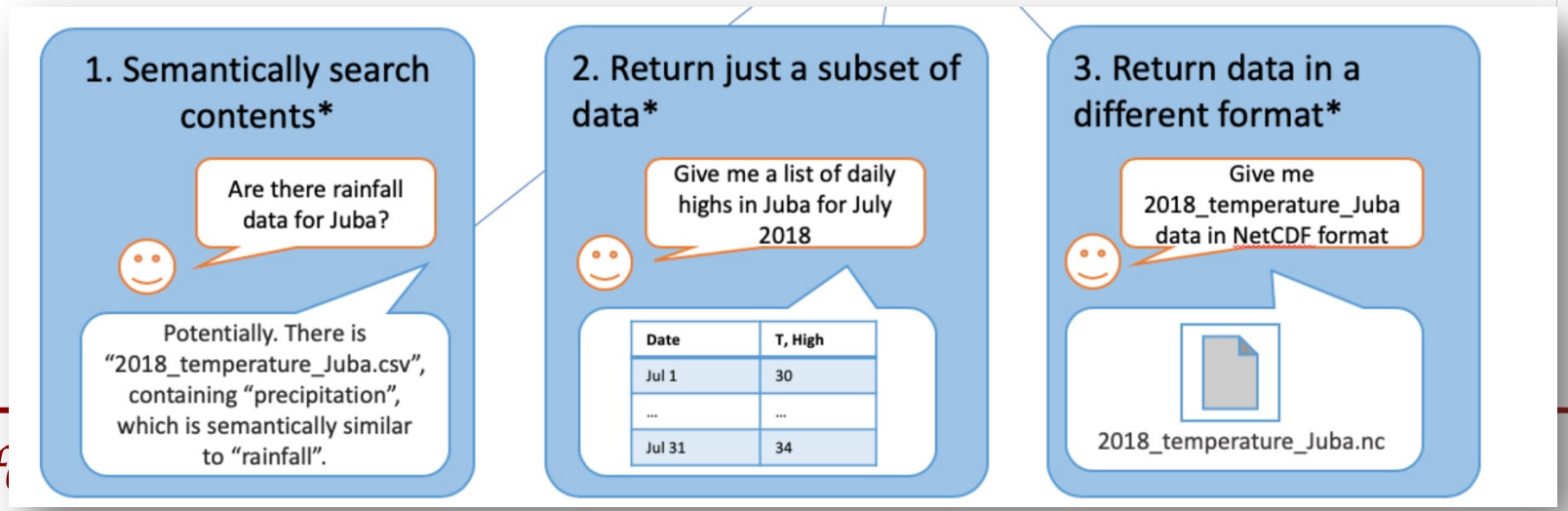
After registering datasets/variables/resources, we can now programmatically search of relevant information. Below, you'll see 3 examples of searching for data using standard variable names, temporal, and spatial coverages. Currently, these are the only search filters we support, but we'll be adding more as we get more feature requests. If you would like to search data catalog by other keywords, please let me know at danf@usc.edu

In [ ]:
```
# 1) Searching by standard_names

search_query_1 = {
    "standard_variable_names__in": [temperature_standard_variable["name"]]
}

resp = requests.post(f"{url}/datasets/find",
                                        headers=request_headers,
                                        json=search_query_1).json()
if resp['result'] == 'success':
    found_resources = resp['resources']
    print(f"Found {len(found_resources)} resources")
    print(found_resources)
```

In [ ]:
```
# 2) Searching by spatial_coverage

# Bounding box search parameter is a 4-element numeric array (in WGS84 coordinate system) [xmin, ymin, xmax, ymax]
# As a reminder, x is longitude, y is latitude
bounding_box = [
    spatial_coverage["value"]["xmin"],
    spatial_coverage["value"]["ymin"],
    spatial_coverage["value"]["xmax"],
    spatial_coverage["value"]["ymax"]
]

search_query_2 = {
    "spatial_coverage__within": bounding_box
}

resp = requests.post(f"{url}/datasets/find",
                                        headers=request_headers,
                                        json=search_query_2).json()
if resp['result'] == 'success':
    found_resources = resp['resources']
    print(f"Found {len(found_resources)} resources")
    print(found_resources)
```

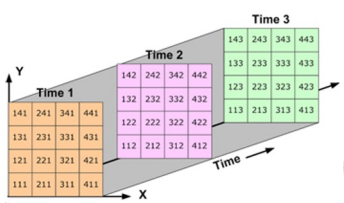
In [ ]:
```
# 3) Searching by temporal_coverage and standard_names

# Bounding box search parameter is a 4-element numeric array (in WGS84 coordinate system) [xmin, ymin, xmax, ymax]
# As a reminder, x is longitude, y is latitude
start_time = "2018-01-01T00:00:00"
end_time = "2018-01-21T23:59:59"

search_query_3 = {
    "standard_variable_names__in": [temperature_standard_variable["name"]],
    "start_time__gte": start_time,
    "end_time__lte": end_time
}

resp = requests.post(f"{url}/datasets/find",
                                        headers=request_headers,
                                        json=search_query_3).json()

if resp['result'] == 'success':
    found_resources = resp['resources']
    print(f"Found {len(found_resources)} resources")
    pp.pprint(found_resources)
```

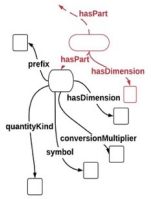
In [ ]:
```
# 4) Searching by dataset_names

search_query_4 = {
    "dataset_names__in": ["Temperature recorded outside my house"]
}

resp = requests.post(f"{url}/datasets/find",
```

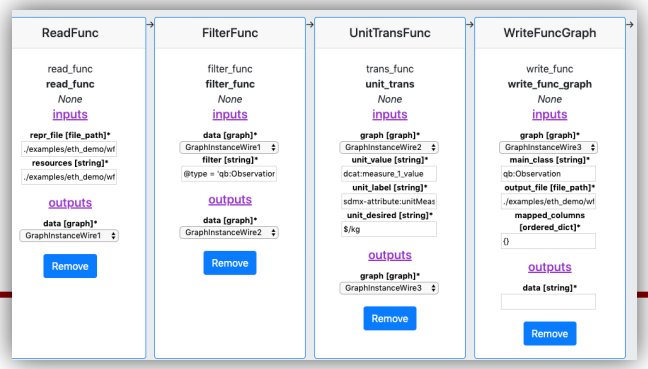


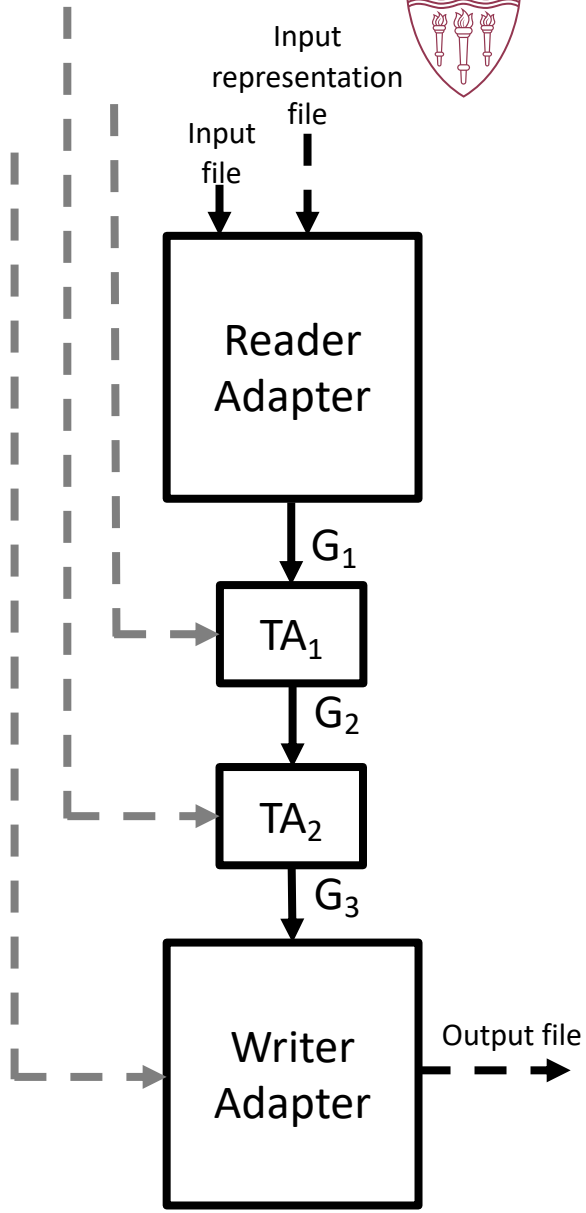
Data Transformation

- Construct **transformation pipeline**
- **Structured Representation**
  - Uses D-REPR representation
    - Supports complex data (N-dim, data across multiple files)
    - Captures and leverages a semantic meaning of the data
    - Can be mapped to any ontology (i.e. SVO, DataCube)
  - Makes the data format-independent
- Building blocks architecture
- No more manual-coding for each transformation
  - Easy to reuse existing building blocks



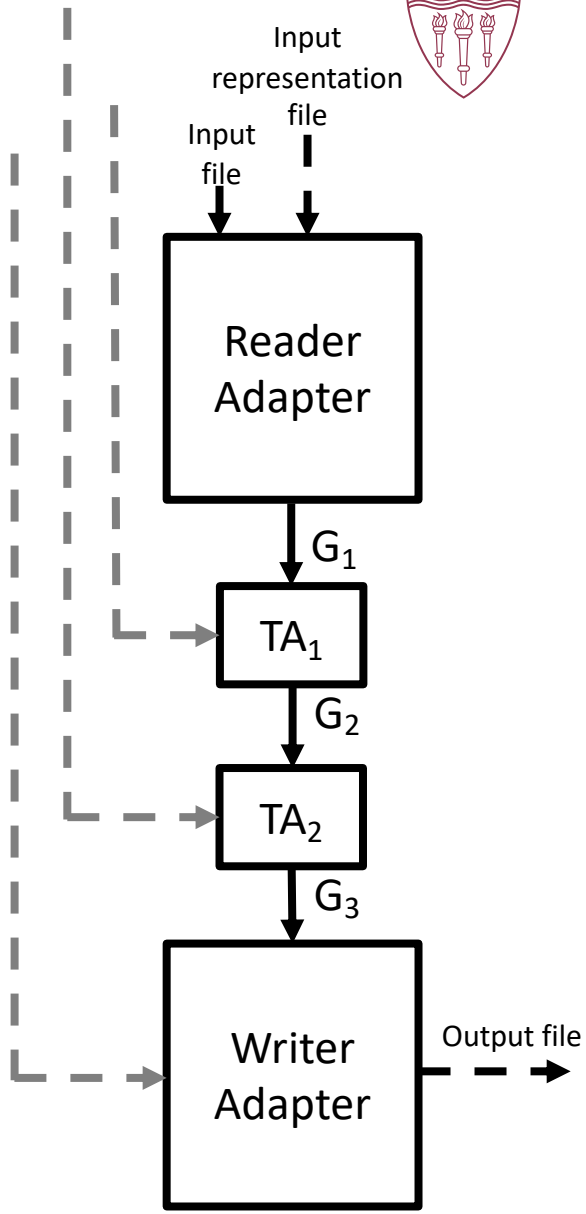






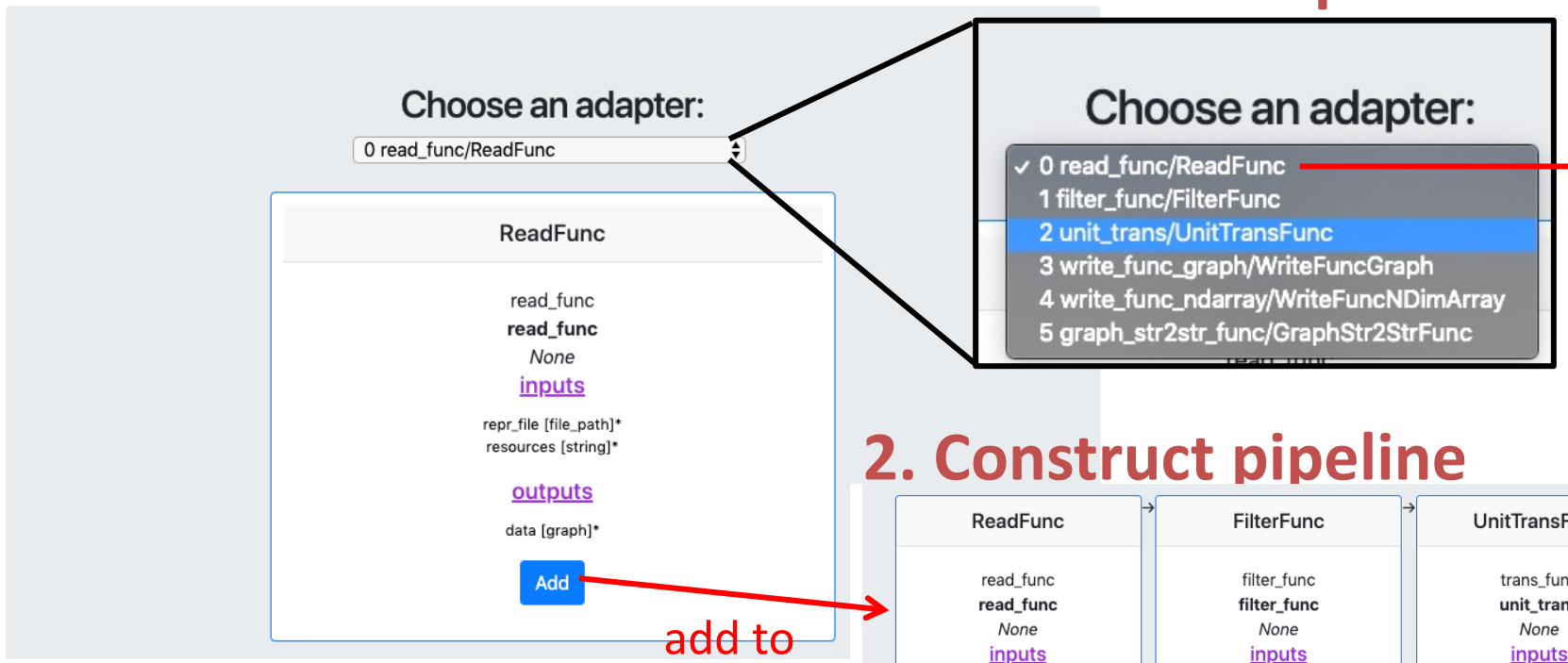
Data Transformation Architecture

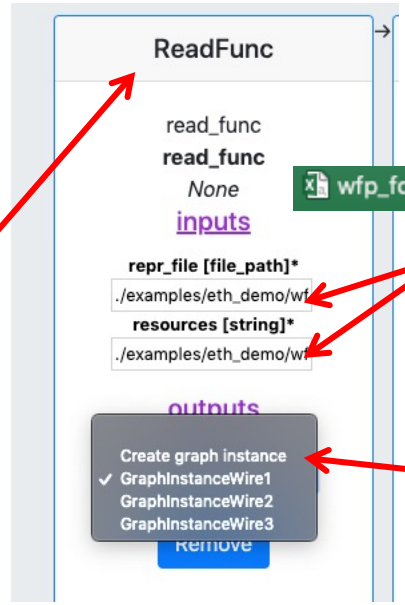
- Adapters (building blocks/'components')
  - Three types:
    - Reader (entry point)
      - Reads input file (data) and description of it (variables, relations, semantics)
    - Transformer
      - Does not materialize the data, just reproduces it
    - Writer (exit point)
      - Writes output file (data) based on a description of it (variables, relations, semantics)
  - Enable input data validation and compatibility checking
- Pipeline
  - Define the required **inputs** for some adapters
  - **Wire** some inputs to outputs ('concatenate' the components)
  - Execute the pipeline!



Data Transformation UI

1. Explore adapters





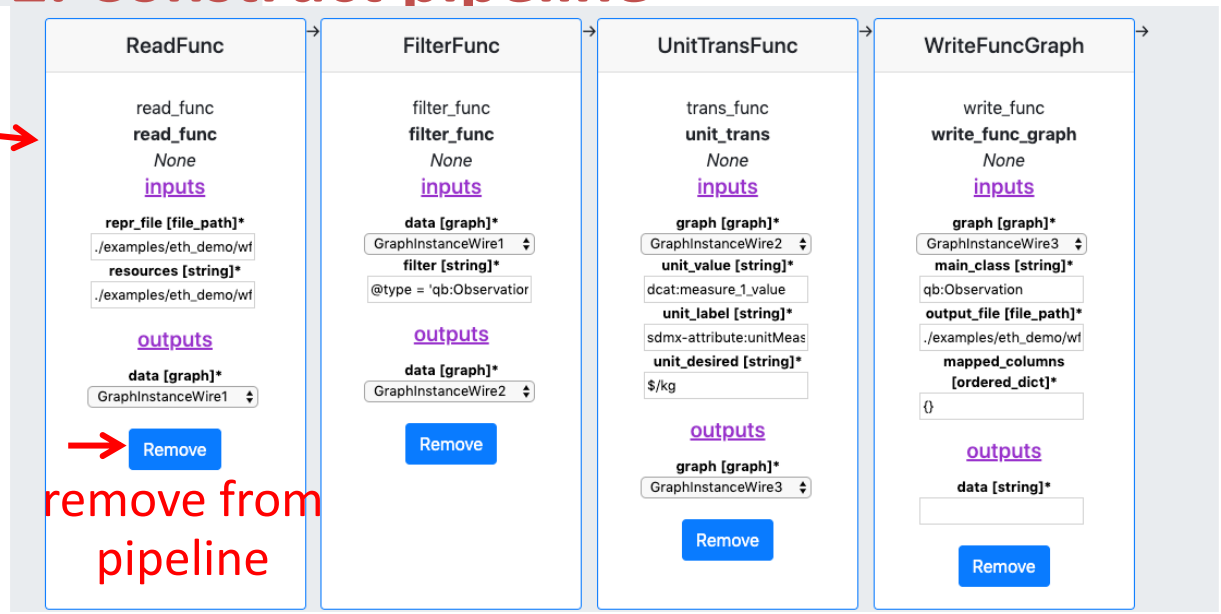
wfp_food_prices_ethiopia

Input files

"wires" = graph instances

2. Construct pipeline

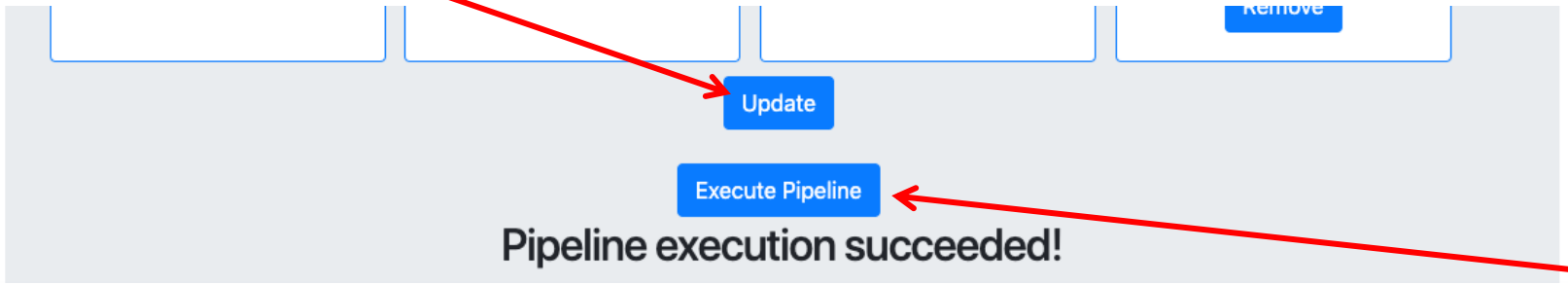
add to pipeline



remove from pipeline

Data Transformation UI (cont'd)

Apply (online) updates and modifications

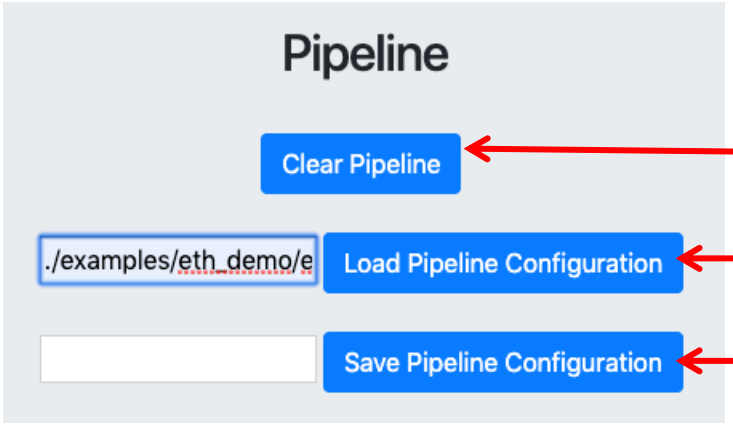


3. Execute pipeline

Generates output file

wfp_food_prices_ethiopia.modified

(optional) Save/Load pipelines



Clear pipeline

Load from a ready config file

Save to config file

Future work

- Dataset Registration
  - **Data discovery** for finding and adding more datasets
  - **Registering high volume** of data sitting behind servers (RESTful API)
  - Improve **automatic data understanding**
    - Table understanding
    - Unit Detection
- Dataset Query
  - GUI for **browsing and visualizing datasets** in the Data Catalog
- Data Transformation
  - Generating transformation plan (semi-)**automatically** (“transformation reasoning”)

Summary

- The Data Catalog allows dataset ***registration, search***, ***transformation*** and ***visualization***
- **Automated** tools to support adding additional **meta-data** to the Catalog
- Transformation framework that allows **constructing**, **executing** and **validating** a **transformation** pipeline
  - Easy to use and easily extended (no need to re-implement a complete flow)
- More information can be found at
  - https://mint-project.org
  - https://github.com/mintproject

References

- [1] B. Shbita, A. Rajendran, J. Pujara, and C. Knoblock, Parsing, Representing and Transforming Units of Measure, in Modeling the World's Systems, 2019
- [2] J. Pujara, A Common Framework for Developing Table Understanding Models, ISWC 2020
- [3] B. Vu, J. Pujara, and C. Knoblock, D-REPR: A Language for Describing and Mapping Diversely-Structured Data Sources to RDF, K-CAP 2019