C²METADATA

Continuous Capture of Metadata

Supported by the Data Infrastructure Building Blocks (DIBBS) program of the National Science Foundation through grant NSF ACI-1640575

Extending the PROV Model to Data Transformation Scripts

George Alter

SDTH: Structured Data Transformation History

- SDTH answers basic questions about a data transformation script:
  - **What dataframes/variables affected the values of variable X or dataframe Y?**
  - **What dataframes/variables were affected by variable X or dataframe Y?**
  - **What commands affected the values of variable X or dataframe Y?**
  - **What commands were affected by variable X or dataframe Y?**

- SDTH is an extension of the W3C PROV standard for provenance
  - Expressed as RDF triples (subject, predicate, object)
  - Can be queried with SPARQL
  - Can be combined with other PROV statements

W3C PROV

- Key concepts in PROV are very abstract
  - Activity
  - Entity
  - Activities can use an Entity and can generate a new Entity

- Focuses on origin, ownership, and <u>process</u>
  - PROV is a history of data flowing through processes
  - A process has inputs and outputs
  - Operations within a process are not described

W3C PROV: Limitations

- A PROV process is a black box
  - PROV activities are only described by inputs and outputs
  - **PROV entities do not change.**
  - **Activities create new entities**

- PROV does not meaningfully describe data sets
  - No concept corresponding to a data matrix (e.g., dataframe)
  - No way to represent a variable or record within a data structure

- **SDTH extends PROV to data sets and data transformations**

Variable names vs. VariableInstances

- Variables are not PROV entities
  - PROV entities are stable and immutable
  - Variables in procedural languages (SPSS, R, etc.) change all the time
    - **A variable is a container with a variable name**
    - **Contents of a variable may change, but the variable name remains the same**
- SDTH introduces **instances**
  - An instance is a specific state of a data structure (variable, dataframe, file)
  - **A VariableInstance is a specific set of values**
  - If a value in a variable changes, a new VariableInstance is created
  - A variable name may be associated with many VariableInstances

Names and Instances: A Simple Example

- A simple data transformation script:

```
Compute varX = 10
Compute varX = 2*varX
```

- One variable **name**: “varX”
- Two variable **instances**: 10, 20

Entities and Predicates in SDTH

SDTH Entities
SDTH Predicates

- RDF consists of triples: (Subject, Predicate, Object)
  - Subject and Object must be PROV entities
  - Predicates are PROV activities or attributes of entities
- An SDTH **Program hasProgramSteps**
- An SDTH **ProgramStep**
  - Acts upon data instances
    - **FileInstance**
    - **DataframeInstance**
    - **VariableInstance**
  - **Has source code**
- A data instance
  - **Has** a **name** in the source code
  - May be **derived from** a previous data instance
    - derivedFrom is used when values change
  - May be **elaborated from** a previous data instance
    - elaboratedFrom is used when attributes of variables change
    - E.g., a numeric variable is converted to string

Example: Two Python Commands

```
PersonalData = PersonalData.assign(HHsize=PersonalData['PPHHSIZE'] )
```

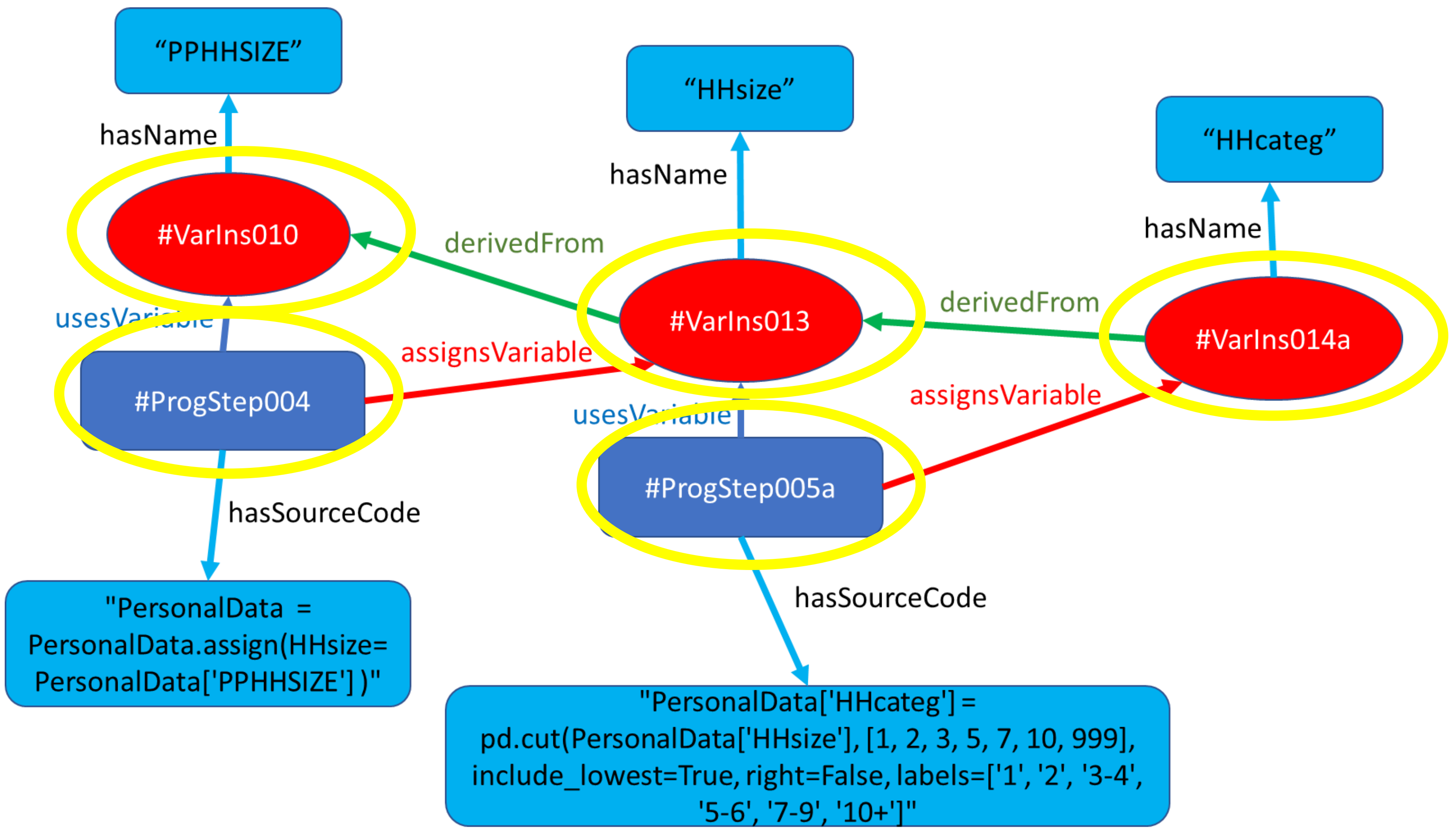
- Adds a new variable named ‘HHsize’ to dataframe ‘PersonalData’
- Assigns values from ‘PPHHSIZE’ to ‘HHsize’

```
PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize’],
            [1, 2, 3, 5, 7, 10, 999],
            include_lowest=True, right=False,
            labels=['1', '2', '3-4', '5-6', '7-9', '10+'] )
```

- Adds a new variable named ‘HHcateg’ to dataframe ‘PersonalData’
- Assigns values from ‘HHsize’ to ‘HHcateg’
- Recodes ‘HHcateg’ into six categories

Graph of SDTH for Example

Entities are assigned identifiers

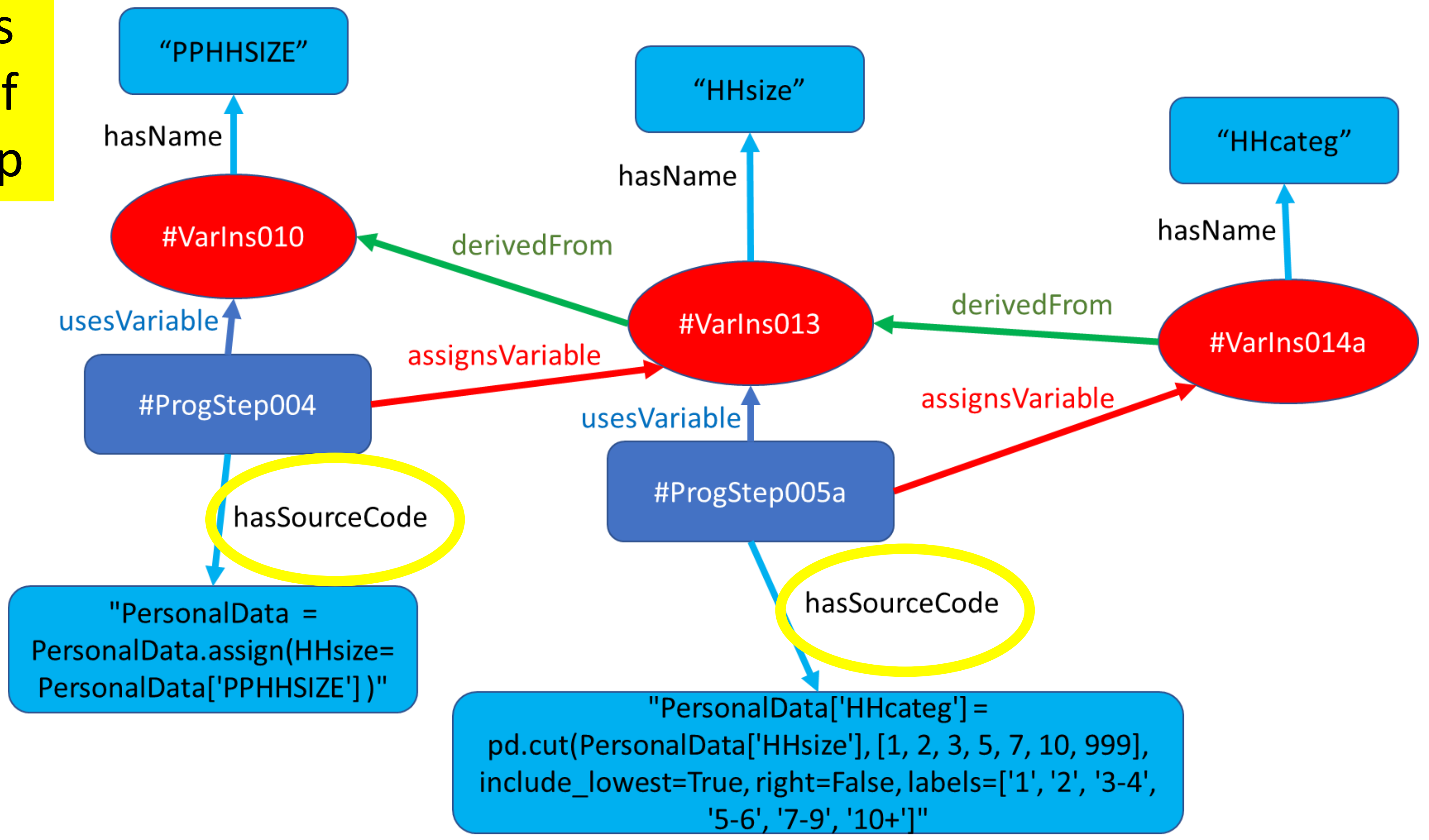


"PersonalData = PersonalData.assign(HHsize= PersonalData['PPHHSIZE'])"

"PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999], include_lowest=True, right=False, labels=['1', '2', '3-4', '5-6', '7-9', '10+']"

Graph of SDTH for Example

Source code is an attribute of a ProgramStep

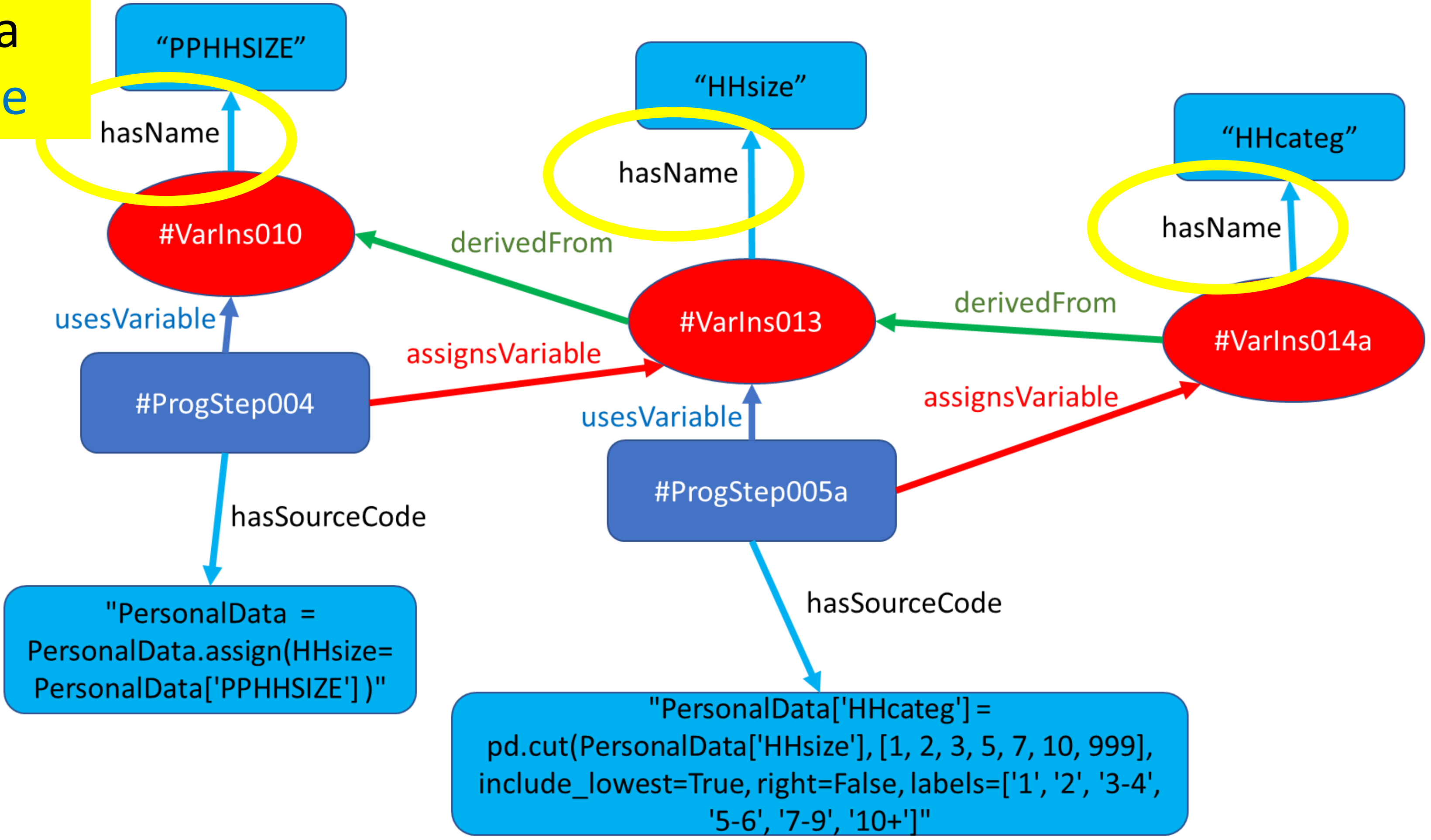


"PersonalData = PersonalData.assign(HHsize= PersonalData['PPHHSIZE'])"

"PersonalData['HHcateg'] = pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999], include_lowest=True, right=False, labels=['1', '2', '3-4', '5-6', '7-9', '10+']"

Graph of SDTH for Example

A variable name is an attribute of a VariableInstance

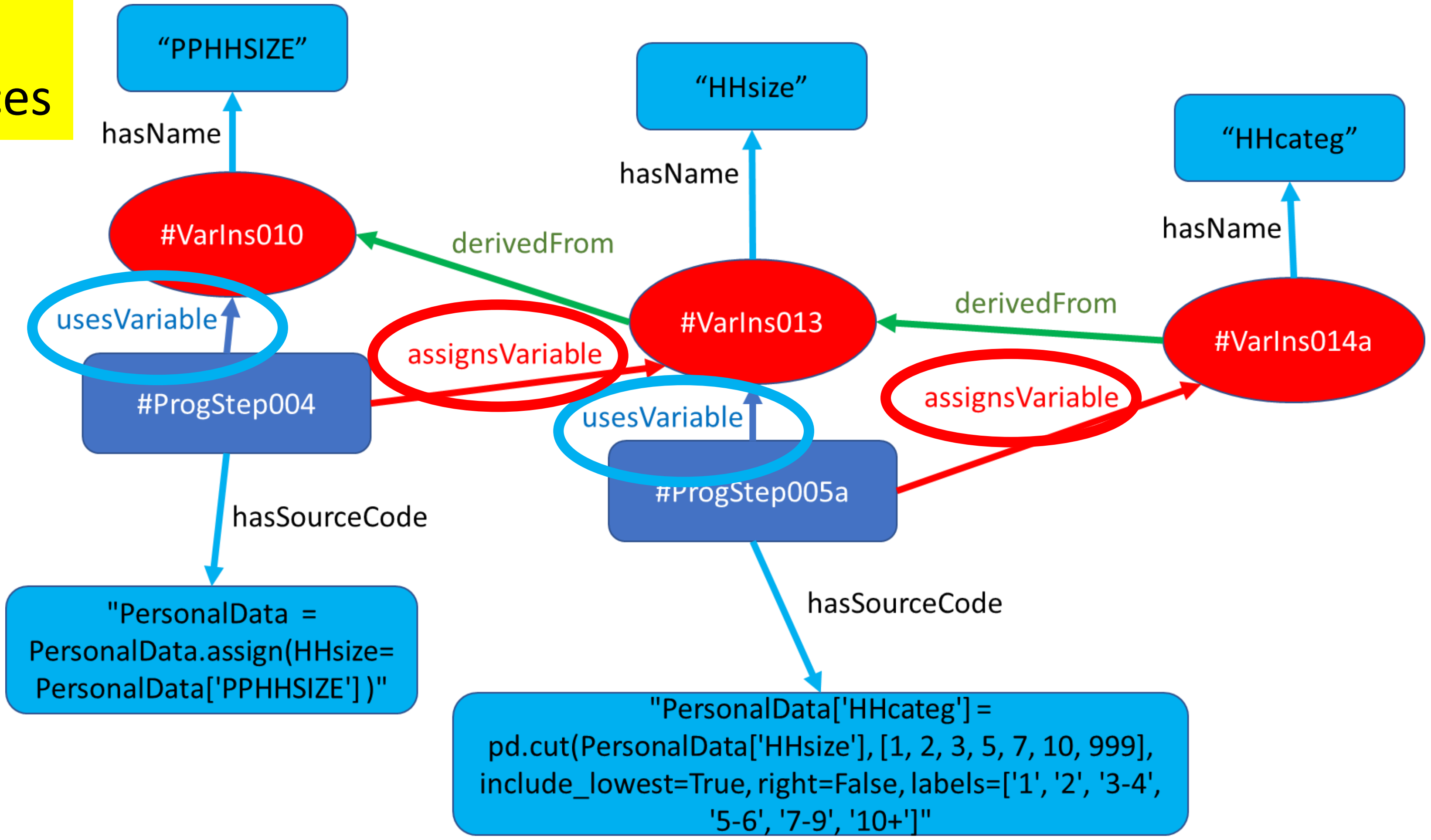


```
"PersonalData  =
PersonalData.assign(HHsize=
PersonalData['PPHHSIZE'] )"
```

```
"PersonalData['HHcateg'] =
pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999],
include_lowest=True, right=False, labels=['1', '2', '3-4',
'5-6', '7-9', '10+']"
```

Graph of SDTH for Example

ProgramSteps
use and **assign**
VariableInstances

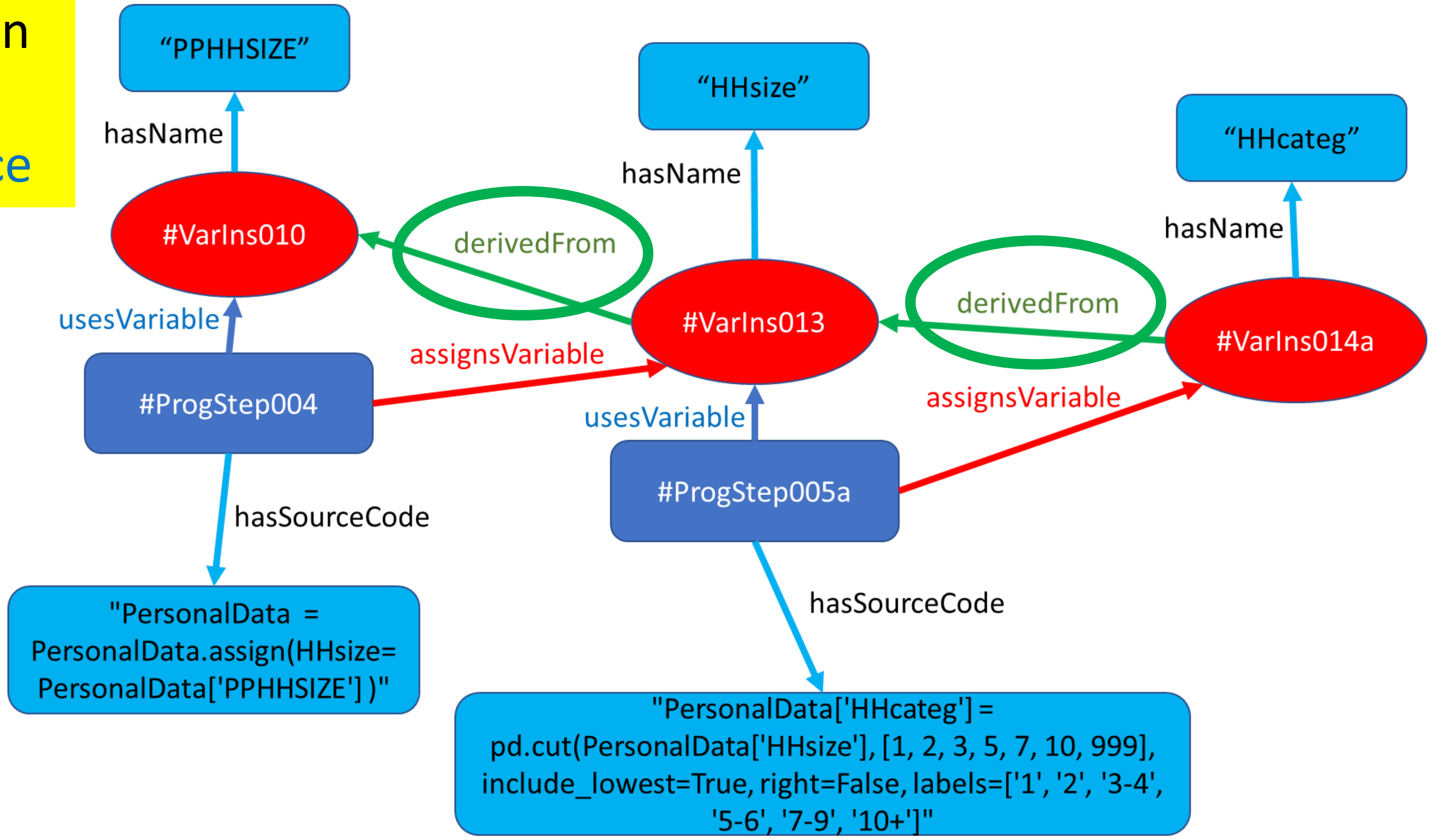


```
"PersonalData =
PersonalData.assign(HHsize=
PersonalData['PPHHSIZE'] )"
```

```
"PersonalData['HHcateg'] =
pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999],
include_lowest=True, right=False, labels=['1', '2', '3-4',
'5-6', '7-9', '10+']"
```

Graph of SDTH for Example

derivedFrom shows the origin of a VariableInstance



```
"PersonalData =
PersonalData.assign(HHsize=
PersonalData['PPHHSIZE'] )"
```

```
"PersonalData['HHcateg'] =
pd.cut(PersonalData['HHsize'], [1, 2, 3, 5, 7, 10, 999],
include_lowest=True, right=False, labels=['1', '2', '3-4',
'5-6', '7-9', '10+']"
```

SPARQL: What variables affected HHcateg?

A simple SPARQL query gives us the names of all variables that affected 'HHcateg', directly or indirectly

```
SELECT distinct  ?sname ?oname
   WHERE {
         ?s  sdth:wasDerivedFrom+ ?o .
                        ?s  sdth:hasName ?sname .
                        ?o  sdth:hasName ?oname .
      FILTER (?sname = "HHcateg")
                        }
```

SPARQL queries operate on VariableInstances, but they report variable names

Output of a SPARQL Query

Subject variable name (?sname)	Object variable name (?oname)
HHcateg	HHsize
HHcateg	PPHHSIZE

SDTH is a complement of SDTL

- **Our research group also developed SDTL**
- **SDTL and SDTH solve different aspects of a common problem**
  - **How do we describe the ways that data have been transformed and manipulated?**
- **SDTL: Structured Data Transformation Language**
  - SDTL is a language for describing data transformations by statistical analysis software
  - SDTL was created to add variable-level provenance to metadata
  - Many different languages can be translated into SDTL
  - SDTL is machine-actionable (e.g., JSON), which simplifies writing software tools
- **SDTH: Structured Data Transformation History**
  - SDTH is a simple way to query scripts
  - SDT**H** is much less detailed than SDT**L**
  - SDTH extends the W3C PROV standard to describe data
    - SDTH can be combined with other PROV compatible provenance metadata

SDTH: Conclusions

- SDTH answers four basic questions about data transformation scripts
- By moving to the PROV model, SDTH makes querying much easier
- SDTH extends the PROV model with data objects
- PROV and SDTH require a change in perspective
  - From variable names, which point to containers for values
  - To variable instances, which are specific sets of values

How will SDTL and SDTH be maintained?

- DDI Alliance is a standards organization that maintains DDI and related standards
- SDTL has been adopted by the DDI Alliance
- SDTH is being documented for review by the DDI Alliance
- All software developed by the C^2Metadata Project is open source on Gitlab

Reference links

- Publications:
  - Alter, G. C., Donakowski, D., Gager, J., Heus, P., Hunter, C., Ionescu, S., Iverson, J., Jagadish, H. V., Lagoze, C., Lyle, J., Mueller, A., Revheim, S., Richardson, M. A., Ørnulf, R., Seelam, K., Smith, D., Smith, T., Song, J., Vaidya, Y. J., & Voldsater, O. (2020). Provenance metadata for statistical data: An introduction to Structured Data Transformation Language (SDTL). *IASSIST Quarterly*, *44*(4). https://doi.org/10.29173/iq983
  - Alter, G. C., Gager, J., Heus, P., Hunter, C., Ionescu, S., Iverson, J., Jagadish, H. V., Lyle, J., Mueller, A., Nordgaard, S., Risnes, O., Smith, D., & Song, J. (2021). Capturing Data Provenance from Statistical Software. *International Journal of Digital Curation*, *16*(1), Article 1. https://doi.org/10.2218/ijdc.v16i1.763
  - Song, J., Alter, G., & Jagadish, H. V. (2019). *C 2 Metadata: Automating the Capture of Data Transformations from Statistical Scripts in Data Documentation*. 2005–2008. https://doi.org/10.1145/3299869.3320241
- SDTL product page: https://ddialliance.org/products/sdtl
- SDTL User Guide: http://c2metadata.gitlab.io/sdtl-docs/master/
- SDTL Working Group: https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/899547182/SDTL+-+Structured+Data+Transformation+Language

Thank You!

Supported by National Science Foundation grant NSF ACI-1640575

WHOLE Tale

ICPSR

NORC 75 at the UNIVERSITY of CHICAGO

colectica

mtna

NSD NORWEGIAN CENTRE FOR RESEARCH DATA

ANES American National Election Studies