Capturing Data Provenance from Statistical Software

George Alter
University of Michigan

Jack Gager
Metadata Technology North America Inc.

Pascal Heus
Metadata Technology North America Inc.

Carson Hunter
Metadata Technology North America Inc.

Sanda Ionescu
University of Michigan

Jeremy Iverson
Colectica

H V Jagadish
University of Michigan

Jared Lyle
University of Michigan

Alexander Mueller
University of Michigan

Sigve Nordgaard
Norwegian Centre for Research Data

Ørnulf Risnes
Norwegian Centre for Research Data

Dan Smith
Colectica

Jie Song
University of Michigan

Abstract

We have created tools that automate one of the most burdensome aspects of documenting the provenance of research data: describing data transformations performed by statistical software. Researchers in many fields use statistical software (SPSS, Stata, SAS, R, Python) for data transformation and data management as well as analysis. The C^{2}Metadata ("Continuous Capture of Metadata for Statistical Data") Project creates a metadata workflow paralleling the data management process by deriving provenance information from scripts used to manage and transform data. C^{2}Metadata differs from most previous data provenance initiatives by documenting transformations at the variable level rather than describing a sequence of opaque programs. Command scripts for statistical software are translated into an independent Structured Data Transformation Language (SDTL), which serves as an intermediate language for describing data transformations. SDTL can be used to add variable-level provenance to data catalogues and codebooks and to create "variable lineages" for auditing software operations. Better data documentation makes research more transparent and expands the discovery and re-use of research data.

Submitted 5 April 2021 ~ *Revision received* 21 April 2022 ~ *Accepted* 25 April 2022

Correspondence should be addressed to George Alter, ICPSR, University of Michigan, Institute for Social Research, P.O. Box 1248, Ann Arbor, MI 48106-1248. Email: altergc@umich.edu

An earlier version of this paper was presented at IDCC21, Edinburgh, 19 April 2021

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction

Realizing the promise of research transparency and the FAIR principles (Wilkinson et al., 2016) requires provenance metadata, i.e., documentation of the origins, contents, and meaning of data. Even though most data are "born digital," metadata are usually an afterthought, and the cost of creating detailed metadata is often prohibitive. This paper describes tools that automate the creation of detailed provenance metadata from statistical software, which are widely used for data management and data transformations as well as analysis. Researchers in many fields use statistical software (SPSS, Stata, SAS, R, Python) for data transformation and data management as well as analysis (IBM Corp., 2019; Python Software Foundation, 2019; R Core Team, 2013; SAS Institute, 2015; StataCorp., 2020). Our tools extract variable-level data provenance from scripts used in major statistical software packages and integrate this information into standard metadata formats used by data repositories for data discovery tools, codebooks, question banks, and other services.

Much of the data shared by the social and ecological sciences is maintained in repositories that rely on structured metadata in the Data Documentation Initiative (DDI) (Caporali, Morisset, Legleye, & Richou, 2015; Vardigan, Heus, & Thomas, 2008) and Ecological Metadata Language (EML) (Fegraus, Andelman, Jones, & Schildhauer, 2005) standards. In the social sciences, DDI is used by the Inter-university Consortium for Political and Social Research (ICPSR), the Dataverse Network, the Integrated Public Use Microdata Series (IPUMS), and other U.S. repositories in the Data Preservation Alliance for the Social Sciences (Data-PASS). DDI has also been adopted by the 21 members of the Consortium of European Social Science Data Archives (CESSDA), the Australian Data Archive, and the International Household Survey Network (IHSN), which has conducted thousands of surveys in low and middle-income countries. EML is one of the metadata standards used by the DataONE network of data repositories, the Global Biodiversity Information Facility, the Knowledge Network for Biocomplexity, and the Long-Term Ecological Research Network. Thus, many scientists have used data catalogues and codebooks based on DDI or EML without ever seeing the metadata behind them.

The "Continuous Capture of Metadata for Statistical Data" (C^2Metadata) Project (NSF ACI-1640575) automates the process of describing data transformations. C^2Metadata tools create a workflow for metadata that parallels the workflow that transforms the data. The statistical analysis packages frequently used to manage scientific data have limited metadata capabilities, and they do not support the detailed metadata standards used by data repositories. Consequently, valuable information about the data is lost. We minimize additional work for the data producer by extracting provenance information from the script that transformed the data, and we use that information to update an existing metadata file. Data producers can obtain an updated metadata file and new codebook in minutes by uploading two files to a webpage or running the same tools on their local computer.

C^2Metadata tools translate scripts used by statistical software into an independent Structured Data Transformation Language (SDTL), which serves as an intermediate language for describing data transformations. SDTL describes commands/steps in a program, and it complements metadata standards like DDI and EML that describe the current state of the data. SDTL can be used to:

- Update existing metadata files (e.g., DDI, EML), so that both the original data description and changes to the data are preserved
- Describe variable transformations in natural language for data users who are unfamiliar with the specific software used in variable transformations
- Create "variable lineages" that describe the transformations performed on each variable for use in auditing scripts

- Add variable- and command-level information to PROV-based provenance metadata

The C^{2}Metadata Project has developed an automated workflow that:

1. Extracts data transformation information from scripts for the leading statistical software packages
2. Expresses data transformations in a new Structured Data Transformation Language (SDTL) that is independent of the source languages
3. Incorporates SDTL and human-readable derivatives of SDTL into existing metadata standards (i.e., DDI, EML)
4. Creates an interactive codebook based on the updated metadata

C^{2}Metadata software tools were developed to work with DDI Codebook, which is the light version of DDI used by many data custodians around the globe, such as ICPSR in the United States, CESSDA across Europe, and the IHSN in low- and middle-income countries and international organizations. However, these tools could be adapted to work with DDI Lifecycle, which was developed to record metadata at all stages of data production and dissemination (Vardigan, Granda, & Hoelter, 2016; DDI Alliance, 2020b). SDTL is fully compatible with the way that DDI Lifecycle records variable derivations.

C^{2}Metadata Workflow

An example of an automated metadata workflow based on C^{2}Metadata tools is illustrated in Figure 1. We assume that the user provides two files: a command script in a supported language (SPSS, Stata, SAS, R, Python) and a structured metadata file in a supported metadata standard or format (DDI, EML) describing the data taken as input to the script. The first step is performed by a Parser, which translates the command script into an SDTL script. The SDTL script is sent to an Updater, which also reads the user's metadata file. The Updater communicates with the Pseudocode Translator, an application that creates a natural language version of the SDTL script. The output of the Updater is a revised metadata file that now includes the SDTL and natural language descriptions of all variables modified by the command script. The updated metadata file may be used in a number of different ways. When a data repository receives the updated data file, the updated metadata will be added to its online data catalogue. The data repository may also use a Codebook Formatter to create a static (e.g., pdf) or interactive codebook (e.g., html) for users to download. Each variable in the catalogue or codebook will include a derivation section that describes the origin of the variable and all of the transformations applied to it.

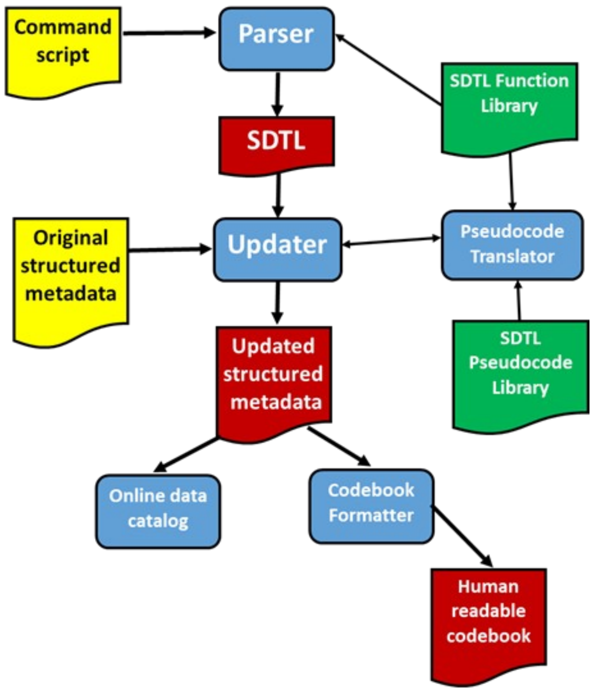


Figure 1 C^{2}Metadata Workflow

Figure 1 includes one Parser and one Updater, but there are actually several versions of each. Each statistical package has its own language that must be parsed and translated into SDTL by a specially designed Parser. Similarly, every metadata standard requires a separate Updater. Since the Parsers and Updaters are separate modules, we can handle any combination of the supported statistical languages and metadata standards.

Figure 1 also includes two files, the Function Library and the Pseudocode Library, which are parts of the SDTL standard. The Function Library is a crosswalk between the syntax for functions (e.g., sine, mean, maximum) in SDTL and in the statistical packages supported by C^{2}Metadata. Although there are thousands of functions, they can all be described by a common template, which simplifies the code in Parsers and Updaters. Similarly, the Pseudocode Library describes how to translate an SDTL command into natural language. The Pseudocode Library provides human-readable text to be inserted before and after variable names, numbers, and other expressions in SDTL commands. The result is a "pseudocode" version of the command that is comprehensible to a person unfamiliar with either the original statistical language or SDTL. Since both the Function Library and the Pseudocode Library are structured data files, they can be modified and expanded as SDTL is updated without changing any application code. The latest versions of the Function Library and Pseudocode Library are in JSON files accessed directly from Gitlab repositories by C^{2}Metadata software modules.

To simplify the C^{2}Metadata workflow for users, we created a Data Transformation Recorder, an online service that orchestrates all of the processes described in Figure 1. The user uploads a command script and one or more XML (DDI or EML) files describing the data before the script was executed. The user must also identify the language of the script and associate the names of data files with the names corresponding to them in the XML file. The Recorder invokes APIs for each of the necessary applications and transmits intermediate results to the next API in the sequence. At the end, the user downloads an SDTL version of the command script, an updated XML file, and an HTML codebook.

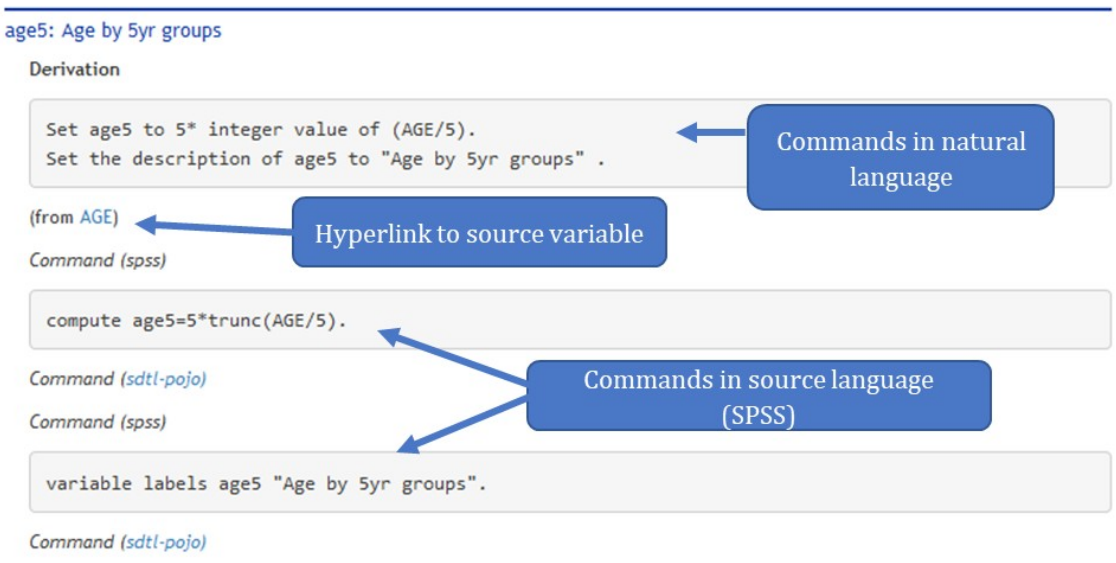
The C^{2}Metadata Project uses elements from DDI Codebook version 2.5 to attach data transformation descriptions to individual variables (DDI Alliance, 2014). A variable can be described by the "derivation" element in the DDI Codebook schema, which has two content elements "drvcmd" (derivation command) and "drvdesc" (derivation description). A derived variable can be described with multiple "drvcmd" elements, which allows us to include both the

SDTL and the source language versions of every command that modified the variable. The "drvdesc" element is used for the natural language (pseudocode) version of each command. Commands that operate on the entire dataset, like AppendDatasets and MergeDatasets, are described with a "fileDerivation", which we expect to be included in the next version of DDI Codebook.

Provenance in Interactive Codebooks

Data are often archived and shared in formats, such as CSV, that must be supplemented by documentation to explain them. In the social sciences, documentation was traditionally in the form of a codebook explaining the origins and meaning of every element in the data (Vardigan & Whiteman, 2007). The advent of structured metadata made it possible to record the same information in a machine-actionable format (XML) that can be used for other purposes, such as populating online catalogues. Data discovery tools, which used to cover only study-level metadata (e.g., title, authors, abstract), can now search variable names, labels, and even values within variables.

We created an interactive version of a codebook to illustrate the new possibilities created by including provenance encoded in SDTL in structured metadata files. Figure 2 is an excerpt describing a derived variable in an interactive codebook. The entry for this variable includes the steps in its creation, which are presented in both a natural language translation of the SDTL and the original source language (SPSS in this example). Pre-existing variables that were used to construct this variable are also presented with hyperlinks pointing to their locations in the codebook. The interactive codebook allows users to choose which information they want to view by opening or closing fields containing natural language, the original source language, and SDTL.


age5: Age by 5yr groups

Derivation

```
Set age5 to 5* integer value of (AGE/5).
Set the description of age5 to "Age by 5yr groups" .
```

(from AGE)

Command (spss)

```
compute age5=5*trunc(AGE/5).
```

Command (sdtl-pojo)

Command (spss)

```
variable labels age5 "Age by 5yr groups".
```

Command (sdtl-pojo)

Commands in natural language
Hyperlink to source variable
Commands in source language (SPSS)

Figure 2. Excerpt from an Interactive Codebook

The interactive codebook allows us to describe variables that passed through intermediate states during the execution of the command script. A data transformation script may modify a variable several times, and some important variables may be dropped before the file is saved. The DDI Updater generates new DDI variable descriptions whenever a variable is transformed in a significant way, and each of these descriptions is assigned an ID that is independent of the variable name. Derived variables can be linked to the relevant states of antecedent variables through variable IDs even if the antecedent variable was changed in later program steps. The DDI standard is also designed to allow a single metadata file to describe multiple data files. The updated XML file includes both the pre- and post-transformation versions of the DDI as well any variable states that were never saved to a file, which the interactive codebook collects into a "Temporary Variables" section. Similar procedures are used in updating EML metadata files.

Since these capabilities are new, the C^{2}Metadata codebook is intended as a prototype to stimulate new ways of using variable-level provenance in data documentation.

Why a Structured Data Transformation Language?

SDTL was created to solve two problems. First, each of the five widely used statistical software packages has its own language, and our planned metadata workflow required a common intermediate language that would work for all of them. In preparation for our NSF proposal, we examined download records at the Inter-university Consortium for Political and Social Research (ICPSR), the largest social science data repository in the U.S. ICPSR offers data for download in the formats of the four statistical packages most common in the social sciences, and researchers are divided among them. SPSS and Stata each account for about 25% of data downloads, and another quarter was divided between SAS and R. Researchers who did not select one of the leading four statistical packages downloaded data in ASCII files and sometimes in Excel. (See Vilhuber (2019) for software used in economics.) Thus, a solution that only worked for one statistical package would reach at most a quarter of the research community. There was a clear need for a common language that could express the commands found in all the statistical packages.

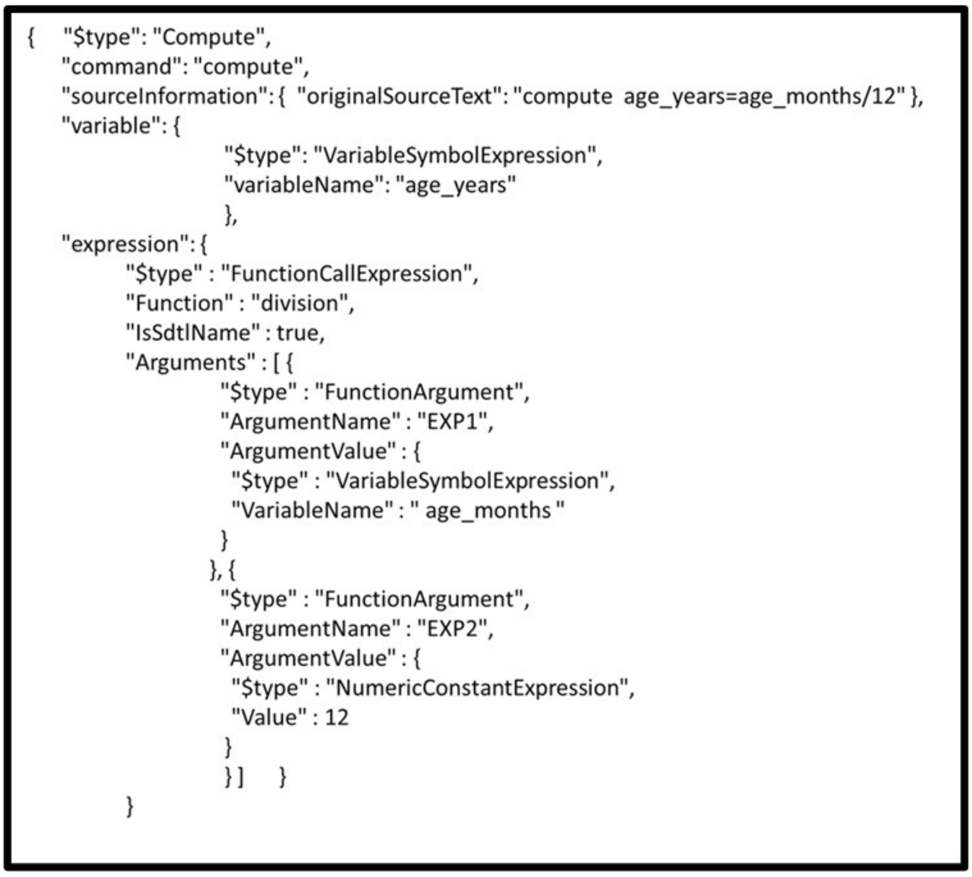
Second, this common language should be in a form that is easy for computers to process. Extracting meaning from a language is a complicated process, and a program customized to each language is required to process scripts into a form that a computer can use. We reduce the costs of sharing and re-using scripts in a common language by making SDTL computer friendly. We developed SDTL in JSON (JavaScript Object Notation), but JSON can be easily translated into other formats used for transmitting complex information among software applications, such as Extensible Markup Language (XML) and Resource Description Framework (RDF).

SDTL is "structured," because it follows a schema with defined tags and delimiters. For example, consider this SPSS command:

```
COMPUTE age_years=age_months/12.
```

This command will create a new variable named "age_years" by dividing the value of variable "age_months" by 12. The SDTL version of this commands is in Figure 3.

SDTL is obviously much more verbose than the SPSS language, but it is also more precise. How do we know that "age_years" and "age_months" refer to variables? Like a spoken language, the SPSS language has syntax rules that allow a person to assign meanings to text like "age_years" based on their order and position in a command. Computers can make these inferences too, but extracting meaning from text is a complicated problem. In SDTL the "$type" tells a computer program that "age_years" and "age_months" are variable names ("VariableSymbolExpression") that refer to columns in the dataset. SDTL relies much more on explicit tagging and less on syntax rules than the languages that it describes.



```
{   "$type": "Compute",
    "command": "compute",
    "sourceInformation": { "originalSourceText": "compute  age_years=age_months/12" },
    "variable": {
                    "$type": "VariableSymbolExpression",
                    "variableName": "age_years"
                    },
    "expression": {
          "$type" : "FunctionCallExpression",
          "Function" : "division",
          "IsSdtlName" : true,
          "Arguments" : [ {
                "$type" : "FunctionArgument",
                "ArgumentName" : "EXP1",
                "ArgumentValue" : {
                 "$type" : "VariableSymbolExpression",
                 "VariableName" : " age_months "
                }
               }, {
                "$type" : "FunctionArgument",
                "ArgumentName" : "EXP2",
                "ArgumentValue" : {
                 "$type" : "NumericConstantExpression",
                 "Value" : 12
                }
                } ]    }
          }
```

Figure 3. Sample SDTL JSON

The SPSS COMPUTE command also uses a number of symbols that play a critical role in the meaning of the command: space, "=", "/", and ".", but these symbols have other meanings in different contexts. In the COMPUTE command "/" means division, but in the following SPSS RECODE command "/" is a separator between two variables that appear in one RECODE command.

```
RECODE age_years (0 THRU 14.999=1) (15 THRU 64.999=2) (65 THRU HI=3)
    / income (0 THRU 19999=1) (20000 THRU 99999=2) (100000 THRU HI=3)
```

The structured nature of SDTL removes ambiguities that would otherwise be resolved by a long list of syntax rules. For example, consider this SPSS command

```
COMPUTE y = 1 + x/5
```

Which operation should be performed first, addition or division? Will the result be [(1 + x)/5] or [1 + (x/5)]? SPSS follows a common convention that division is performed before addition unless a different order of operations is specified by brackets in the formula. In SDTL the order of operations is never ambiguous. As the reader may have noticed in the previous example, arithmetic operations are implemented in SDTL as functions. The expression "x/5" is treated as "division(x, 5)" in SDTL. The basic arithmetic functions in SDTL have two parameters, but each parameter can be a function. This means that SDTL represents "1 + x/5" as "addition(1, division(x,5))". Since the division is nested within the addition, it must be performed first.

SDTL is an international standard maintained by the DDI Alliance (DDI Alliance, 2020c). For a more extended description of SDTL see Alter et al. (2020) and C^{2}Metadata Project (2020).

Translating SDTL into Natural Language

In addition to translating five statistical languages into SDTL, the C^{2}Metadata Project has a simple way of translating SDTL into a more human-friendly form. We have created a set of templates for each SDTL command with text surrounding each of its properties. For example, the template for the SDTL Compute command is

Set {variable} to {expression}.

in which {variable} and {expression} are properties of the command. Each of these properties can be resolved into text, such as a variable name or a number. Using the example in Figure 3, {variable} resolves to "age_years" and {expression} resolves to "age_months/12". The result is

Set age_years to (age_months/12).

Note that {variable} resolves to "age_years" in one step, but the {expression} property is more complicated, as often happens in SDTL. In this case, the expression is a function with two parameters. The Function Library gives this template for division

(EXP1/EXP2),

and we find that EXP1 resolves to a variable named "age_months" and EXP2 resolves to the numeric constant "12". Since SDTL types are often nested several levels deep, resolving SDTL into natural language is a recursive process. The Pseudocode Translator application uses templates like these to convert SDTL into something approximating English.

Templates for SDTL commands are collected in a Pseudocode Library, which is a file in JSON format. (Pseudocode is a term used for the translation of a computer program into language that is easier for humans to decipher.) The Pseudocode Library can be revised and extended without changing any program code in the Pseudocode Translator. Different versions of the Pseudocode Library can be created for other natural languages or special purposes.

Other Uses of SDTL and C^{2}Metadata Tools

SDTL and the PROV Model

Our colleagues in the Whole Tale Project are exploring ways to connect SDTL to the PROV model of provenance. PROV is a family of standards for describing data provenance recommended by the World Wide Web Consortium (Groth & Moreau, 2013). PROV describes the persons and activities that produced and transformed a digital object in a way that can be exchanged and searched on the Web. The original PROV model did not describe variables within datasets or commands within programs, but several extensions of PROV offer more granular approaches to data and data processing, such as ProvONE which was developed by the ecological research community (Cuevas-Vicenttín et al., 2016; see also End to End Provenance Project, 2019; Garijo & Gil, 2013).

SDTL can be serialized into Resource Description Framework (RDF), a format used to share metadata on the semantic web. SDTL RDF can be linked to other provenance models, such as PROV and ProvONE, and analysed by tools like the SPARQL query language. Thomas Thelen and Timothy McPhillips have shown that the following questions can be answered by querying SDTL RDF:

- Which commands affected the values of varX?

- Which variables affected varX?
- Which variables were affected by command Z?
- Which variables were affected by varY?

However, the extensive detail in SDTL RDF makes SPARQL queries long and complex. We are examining ways to map SDTL to ProvONE, which will make queries much simpler.

Translation

SDTL offers a path for translating from one statistical language to another. Organizations often rely on large bodies of code in languages that have become difficult to maintain. Data management scripts in statistical analysis software may use features that are removed in later releases, and younger analysts are often unfamiliar with packages and languages that were prevalent a decade earlier. Under these circumstances, an application that can translate one statistical language into another could be very useful, even if the translation is less than complete. SDTL can be used as an intermediate step in translating between statistical languages. Our project has shown that all five of our target languages can be translated into SDTL, and translating SDTL into these source languages should be a manageable task. If all five languages can be translated to and from SDTL, each language can be translated into the other four.

Reshaping Data Structures

The DDI Alliance is in the process of creating a new standard, DDI Cross Domain Integration (DDI-CDI) (DDI Alliance, 2020a) describing how the same data can be expressed in different formats. Figure 4 shows data in what DDI-CDI calls "Wide" format, in which each row refers to the same unit of observation and each column is a different variable. The same data is shown in Figure 5 in an entity-attribute-value (EAV) format, which is a version of DDI-CDI "Long" format. A row in Long format describes only one attribute of a unit of observation, and the Attribute column identifies the property that is measured by the Value column. DDI-CDI provides standard terms for describing Wide, Long, and two other common data structures and for mapping how values and attribute descriptions are managed in each data structure.

ID	Name	Age	Place of birth	Occupation
9990	Vera	62	Newcastle	Detective
9991	Xavier	49	Madrid	Architect
9992	Yolanda	33	Copenhagen	Baker

Figure 4. Data in Wide Format

ID	Attribute	Value
9990	Name	Vera
9990	Age	62
9990	Place of birth	Newcastle
9990	Occupation	Detective
9991	Name	Xavier
9991	Age	49
9991	Place of birth	Madrid
9991	Occupation	Architect
9992	Name	Yolanda
9992	Age	33
9992	Place of birth	Copenhagen
9992	Occupation	Baker

Figure 5. Data in Long Format

SDTL complements DDI-CDI by providing a way to describe how data are transformed from one data structure to another. The SDTL ReshapeLong command converts data in Wide format (Figure 4) to Long format (Figure 5), and ReshapeWide converts Long format (Figure 5) to Wide format (Figure 4). These commands have different names in various statistical software: reshape (Stata), transpose (SAS), casestovars/varstocases (SPSS), melt (R, Python).

Limitations

The scope of the C^{2}Metadata Project was limited in several ways to keep the project manageable with limited funding and time. We were aware from the start that we could not capture every data transformation feature available in large and complex languages like SAS and R. Our goal has always been to capture 80% to 90% of the commands that researchers use for data management.

A basic limitation of tools developed on the C^{2}Metadata Project is that they operate only on metadata files and do not access any data directly. We rely on a description of the data prior to transformation in a metadata file with a standard format. This decision simplified the creation of Parsers, because they do not need to read and analyze data files, but it did prevent us from implementing some features.

- Metadata files often include descriptive statistics of variables, such as averages and frequency distributions, which are greatly appreciated by researchers. Since the current tools do not access the data, we cannot compute descriptive statistics for variables that have changed.

- Data transformation commands that depend upon the content of the data are not currently implemented. We have specified a ReshapeWide command in SDTL, but it cannot be supported in a metadata-only system. Reshaping data from a "long" to a "wide" format involves changing the unit of observation to a higher level, such as from individuals to households or counties to states. The new data has one row for every case at the group level (household, state) and separate columns for the attributes of every individual within a group. Suppose that data from a census are arranged with one row per person, and we want to reorganize the data to one row per household. The variables for each person in the household will become columns in the new data file, i.e. the age of the first person in the household will be in column Age1, the age of the second person in Age2, and so on. The number of columns for each variable (age, sex, occupation,…) depends upon the number of people in the largest household. Since we cannot know the size of the largest household without accessing the data, reshaping from long to wide is not possible using only metadata.

R and Python are much more open than earlier statistical packages, and the communities supporting each of these languages have contributed thousands of libraries that add new operations and analyses. To limit the scope of our project, we have focused on the "base" and most popular data transformation libraries in each language. The SDTL parser for R is implementing the tidyverse library (Wickham et al., 2019), and the Python parser works with the Pandas library (The pandas development team, 2020).

The limitations described above are due to restrictions of the C^{2}Metadata Project, and they are not due to limitations of SDTL. The main limitation in SDTL is related to data created by analysis commands. For example, regression models typically generate predicted values and residuals, which can be saved as new variables or separate datasets. We expect that support for data created by statistical procedures will be added to SDTL in the future.

Discussion

The C^{2}Metadata Project has demonstrated that it is possible to automate the capture of variable-level provenance metadata. Automation reduces the cost and increases the quality of documentation showing how users of statistical software transformed and manage their data. We have produced a set of applications that convert scripts from five statistical languages into a common intermediate language (SDTL), which is then embedded into two widely used metadata standards. By creating human-readable histories of variables, we provide metadata that is much more detailed and informative than a long list of commands in an unfamiliar language.

We believe that several innovations in our approach are worth noting. First among these is the creation of a Structured Data Transformation Language (SDTL) to serve as a standard way of representing data transformation commands. Since researchers are currently split among at least five statistical software packages, we created a new language that would work with all of them. SDTL is not intended to replace existing statistical languages, rather it is a lingua franca for applications like data catalogues, codebooks, and other data discovery and documentation tools. SDTL is expressed in a structured format (JSON) that is easily read by computer programs, and it is compatible with existing metadata standards.

Second, although SDTL has a small vocabulary, the SDTL Function Library makes it flexible and expandable. Functions are a familiar device in programming languages, and statistical packages rely heavily on functions for many operations, like generating random numbers and computing quantiles of probability distributions. SDTL extends this approach by using functions to describe arithmetic operations, logical conditions, and variables formed by aggregating over rows. The SDTL Function Library maps functions in other languages into their SDTL equivalents. Since all functions follow the same basic syntax, applications that parse other languages can translate functions into SDTL with a minimum of programming code. The Function Library can be expanded without any changes in applications that rely on it.

Third, we have shown that translating SDTL into a human readable form is a simple and extendable process. The Pseudocode Library is set of fill-in-the-blank templates for SDTL commands. Even complicated SDTL commands can be unfolded into properties that consist of pre-defined text, variable names, and numbers.

Finally, additional applications of SDTL are emerging. SDTL can be translated into RDF for use with PROV and other Semantic Web tools. SDTL complements the development of DDI Cross Domain Integration by describing how data can be transformed into different structures and formats. We also see a future for SDTL as an intermediary in translations between statistical languages. Since the source languages have many idiosyncratic features, comprehensive translations are probably not attainable. However, translations covering 80 to 90 percent of a script will be extremely useful for many purposes. For example, many organizations have legacy scripts in statistical languages that their staff no longer understand. Incomplete translations accompanied by human-readable versions of the original scripts can be very helpful in redesigning out of date workflows.

Data availability

No data is associated with this article.

Code availability

C^{2}Metadata applications are available under an open source license from the project Gitlab repository (C^{2}Metadata Project, 2021). Most applications are available as both code (Java, Python, Clojure, C#) and Docker containers.

Acknowledgements

The Continuous Capture of Metadata for Statistical Data Project is funded by National Science Foundation grant ACI-1640575. We are grateful for contributions to this project by Darrell Donakowski, Carl Lagoze, Matthew A. Richardson, Karunakara Seelam, Tom Smith, Yashas Jaydeep Vaidya, and Ole Voldsater. The C^{2}Metadata Project is grateful to the Whole Tale Project (https://wholetale.org/) for their work on integrating SDTL into the ProvONE model, especially Thomas Thelen, Matt Jones, Bertram Ludaescher, Tim McPhillips, and Craig Willis. The Whole Tale is funded by National Science Foundation grant OAC 1541450.

References

Alter, G., Donakowski, D., Gager, J., Heus, P., Hunter, C., Ionescu, S., . . . Voldsater, O. (2020). Provenance metadata for statistical data: An introduction to Structured Data Transformation Language (SDTL). *IASSIST Quarterly, 44*(4). doi:10.29173/iq983

C^{2}Metadata Project. (2020). Structured Data Transformation Language. Retrieved from http://c2metadata.gitlab.io/sdtl-docs/

C^{2}Metadata Project. (2021). Gitlab Repository: c2metadata. Retrieved from https://gitlab.com/c2metadata

Caporali, A., Morisset, A., Legleye, S., & Richou, C. (2015). La mise à disposition des enquêtes quantitatives en sciences sociales : l'exemple de l'Ined. Population (French Edition), 70(3), 567-597. Retrieved from http://www.jstor.org.proxy.lib.umich.edu/stable/24639342

Cuevas-Vicenttín, V., Ludäscher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., & Leinfelder, B. (2016). ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance. Retrieved from http://jenkins-1.dataone.org/jenkins/view/Documentation%20Projects/job/ProvONE-Documentation-trunk/ws/provenance/ProvONE/v1/provone.html

DDI Alliance. (2014). DDI-Codebook 2.5. Retrieved from https://ddialliance.org/Specification/DDI-Codebook/2.5/

DDI Alliance. (2020a, April 4, 2020). DDI Cross Domain Integration (DDI-CDI) Review. Retrieved from https://ddi-alliance.atlassian.net/wiki/spaces/DDI4/pages/860815393/DDI%2BCross%2BDomain%2BIntegration%2BDDI-CDI%2BReview

DDI Alliance. (2020b, April 12, 2022). DDI Lifecycle 3.3. Retrieved from https://ddialliance.org/Specification/DDI-Lifecycle/3.3/

DDI Alliance. (2020c, December 1, 2020). Structured Data Transformation Language. Retrieved from https://ddialliance.org/products/sdtl/1.0

End to End Provenance Project. (2019). Extended Prov JSON. Retrieved from https://github.com/End-to-end-provenance/ExtendedProvJson

Fegraus, E. H., Andelman, S., Jones, M. B., & Schildhauer, M. (2005). Maximizing the value of ecological data with structured metadata: an introduction to ecological metadata language (EML) and principles for metadata creation. Bulletin of the Ecological Society of America, 86, 158–168.

Garijo, D., & Gil, Y. (2013, 17 September 2013). The P-PLAN Ontology. Retrieved from https://www.opmw.org/model/p-plan/

Groth, P., & Moreau, L. (2013). PROV-OVERVIEW: An Overview of the PROV Family of Documents. Retrieved from http://www.w3.org/TR/2013/NOTE-prov-overview-20130430/

IBM Corp. (2019). IBM SPSS Statistics for windows, version 26.0. Armonk, NY: IBM Corp.

Python Software Foundation. (2019). Python Language Reference, version 3.8. Beaverton, OR. Retrieved from https://www.python.org/

R Core Team. (2013). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

SAS Institute. (2015). SAS®9.4 Product Documentation. Cary, NC: SAS Institute Inc. Retrieved from http://support.sas.com/documentation/94/index.html

StataCorp. (2020). Stata Statistical Software: Release 16.1. College Station, TX: StataCorp LP.

The pandas development team. (2020). pandas-dev/pandas: Pandas: Zenodo. Retrieved from https://doi.org/10.5281/zenodo.3509134

Vardigan, M., Granda, P., & Hoelter, L. (2016). Documenting survey data across the life cycle. The SAGE Handbook of Survey Methodology. Los Angeles, CA: SAGE, 443-459.

Vardigan, M., Heus, P., & Thomas, W. (2008). Data documentation initiative: Toward a standard for the social sciences. International Journal of Digital Curation, 3(1).

Vardigan, M., & Whiteman, C. (2007). ICPSR meets OAIS: Applying the OAIS reference model to the social science archive context. Archival Science, 7(1), 73-87. doi:10.1007/s10502-006-9037-z

Vilhuber, L. (2019). Report by the AEA Data Editor. AEA Papers and Proceedings, 109, 718-729. doi:10.1257/pandp.109.718

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D. A., François, R., . . . Hester, J. (2019). Welcome to the Tidyverse. Journal of Open Source Software, 4(43), 1686.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., ... & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, *3*(1), 1-9.