DDI as a Common Format for Export and Import for Statistical Packages

by Larry Hoyle and Joachim Wackerow[1]

Abstract
One of the roles the DDI standard can perform is to serve as a medium for the transfer of metadata and data across both space and time. A crucial component of this role is the ability to represent the data and metadata contained in common data analysis and management packages. This paper describes an experiment using the program Stat/Transfer to move datasets among five popular packages with DDI Lifecycle as an intermediary format.
We created a dataset in each of the five packages and then exported it to DDI Lifecycle via Stat/Transfer. We also created a DDI Lifecycle instance and an associated delimited dataset, containing as many of the metadata elements found in any of the five packages possible and then exported it to each of the packages. Success or failure to transfer was recorded for a set of generic metadata elements identified in an earlier paper. Using a commercial file transfer program helped identify which metadata elements were transferrable through a generally available machine actionable process.
The experiment revealed some areas for potential improvements to DDI as well as suggestions for data analysis packages and research practices.

Keywords: DDI, data formats, metadata, statistical packages, Stat/Transfer, JMP, R, SAS, SPSS, Stata.

Introduction
Not uncommonly, datasets are initially produced in one software package specific format, whether proprietary, as in an SPSS dataset, or open source, as in an R workspace. As the data are reused, either at a later time or by other researchers in another place they will commonly need to be imported into a different software package. Even within one organization a variety of software tools may be employed. Researchers may have differing needs and personal preferences. Different packages may have unique analysis tools. Software and preferences for software also evolve over time. Data retrieved from an archive after a period of dormancy may very well need to be represented in some new format..

The role of DDI envisioned here is as an intermediate format in the process of moving data and metadata among software packages

An earlier paper (Hoyle et al., 2010) enumerated a list of generic metadata elements that could be represented in at least one of a set of eight common data file formats. No one format was able to represent all of the metadata elements. DDI Lifecycle came closest to being able to represent all of the metadata elements. In what follows, "DDI Lifecycle" refers to DDI Version 3.1, the version used for the experiment described here. Where appropriate, we will describe additional capabilities of DDI 3.2, published in 2014. Some metadata elements were only representable in DDI 3.1 with a workaround (as r:Note elements) making automated discovery challenging. A worthwhile goal for DDI is to be able to contain a machine actionable superset of the metadata elements across the most commonly used array of analysis software. For the "transport" role envisioned here, it is important for DDI to be optimized for clarity and completeness, not necessarily for speed or efficiency..

The Packages

This study used five of the formats which were able to contain the broadest array of metadata. Each of the five packages is able to store metadata some elements not shared by all of the other packages.

- JMP versions 8 and 10 (SAS Institute - JMP)
- R version 2.14 and 3.01 (R Development Core Team, 2009)
- SAS version 9.2 and 9.4 (SAS Institute - SAS)
- SPSS version 19 and 21 (IBM)
- Stata version 11 and 13 (StataCorp)

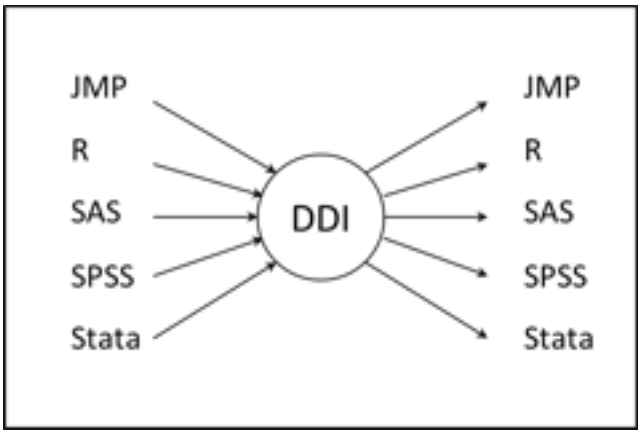


Figure 1 – DDI as an intermediate format

The role of DDI envisioned here is as an intermediate format in the process of moving data and metadata among software packages. Figure 1 shows that role in moving data and metadata among the 5 formats investigated in this project. Given its open nature and to the extent that it comes closest to handling a superset of the metadata managed by the analysis program formats, DDI has an advantage as the intermediate format.

DDI in this role is not necessarily restricted to expression as an XML instance. While the formal specifications of DDI Lifecycle 3.1 and 3.2 are XML schemas, DDI content can be stored physically as an XML file, in an XML database, or even in a relational database. For a discussion of the latter see (Amin et al., 2011). The DDI Alliance is also currently working on two RDF vocabularies (Data Documentation Initiative. 2013a). Future plans for DDI call for a model based specification which can be expressed both in XML and OWL / RDF (Data Documentation Initiative. 2013c). Stat/Transfer versions 11 and 12 were used for the experiment described below.

Transferring data

There are several ways to transfer data and metadata to and from DDI and the five packages. While not practical for datasets of any size, DDI can be hand-edited with an XML editor. There are a number of tools listed on the DDI Tools page (Data Documentation Initiative, 2013b) which can convert metadata to and from DDI and at least one other format. As of version 11, Stat/Transfer can move data and metadata between DDI and 35 other formats. This breadth of coverage was a factor in choosing Stat/Transfer for this experiment. The intent here was neither to endorse nor critique Stat/Transfer, but rather to get a better sense of the current state of the ability to use DDI as a medium for automatic translation of data and metadata across software packages. Since Stat/Transfer (Circle Systems) passes the information through its own internal model, in a sense it is also a sixth format as well as the transfer tool.

The Experiment

Our experiment was designed to address three questions. What metadata are currently possible to transfer with an automated procedure? What metadata elements does DDI support that Stat/Transfer does not? What more could DDI support? Stat/Transfer relies exclusively on the g:ResourcePackage element to contain the metadata in the ddi:DDIInstance it produces. No s:StudyUnit is produced. This usage is consistent with the approach taken by Colectica when importing from SPSS and Stata, the philosophy being that there are no specifically study-level metadata contained in the dataset. This does reveal a shortcoming in the native formats of all of the packages. These files all contain metadata about the structure of the file but almost no metadata

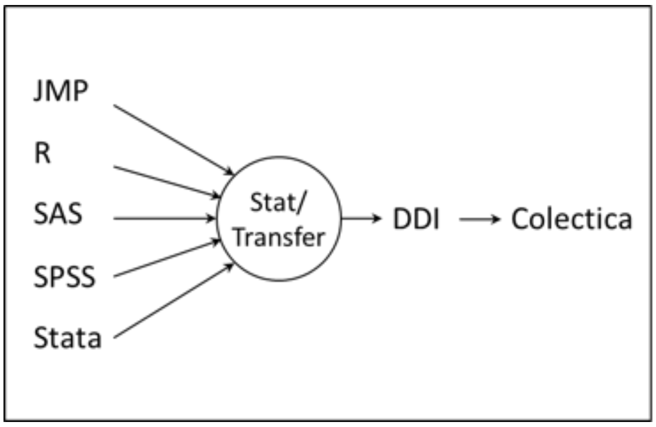


Figure 2 From Packages to DDI Validated with Colectica

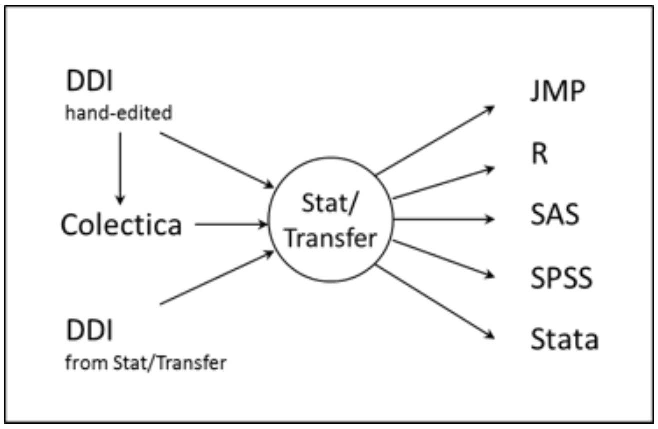


Figure 3 From DDI to Packages

about the actual data or study. Custom attributes on the dataset might offer a mechanism to remedy this shortcoming.

A compatible DDI file used as a source for transfer into the other 5 packages was hand-entered into an XML editor. A separate comma separated variable file contained the associated data. The XML editor validated the metadata against the DDI XML schema. Secondary level validation on the pair of files was performed using Colectica Reader version 3, and Colectica Express version 4. At this time neither Stat/Transfer nor Colectica were capable of using embedded data in the DDI file.

Datasets like the one shown in Figures 4 and 5 were created in R, SPSS, Stata, SAS and JMP. Each of these datasets contained instances of all of the generic metadata elements they could represent. Figure 6, for example, shows the addition of custom attributes named "Concept", "Note", and "Universe" to the sample SPSS dataset, as well as metadata for level of measurement (nominal, ordinal, and scale), and role(input and target).
The hand-edited DDI file was transformed using Stat/Transfer into each of the package's native format. Datasets created in each of the packages were also transformed into DDI. The resulting files were then reviewed for each of the metadata elements considered in the earlier study. Grids like the one shown in Figure 7 were filled in, with a "+" indicating successful metadata transfer, a "-" indicating unsuccessful transfer, and a "~" indicating partial success – such as metadata transferring to an unexpected DDI element. Hatched shaded cells indicate metadata elements not supported by that software package.

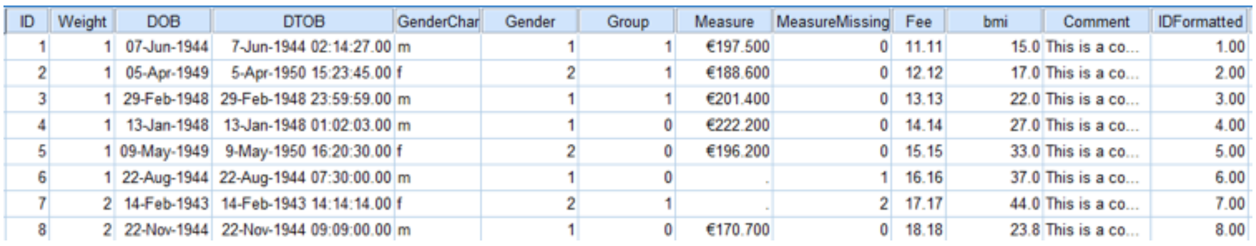
Summary of results

Data, and basic metadata, transferred to and from DDI from all 5 packages. For the most part the elements representable in all of the packages transferred to and from DDI correctly. These elements include:

- dataset name – for all of the packages this came from the name of the file in the host operating system. An R workspace file can contain multiple data frames and other objects.
- variable names
- variable labels – in JMP this becomes a note on the variable
- variable order
- important variable data type (such as date and datetime) – see below for issues related to time zone.
- a missing indicator - see below for issues related to multiple distinct missing values
- labels for numeric values

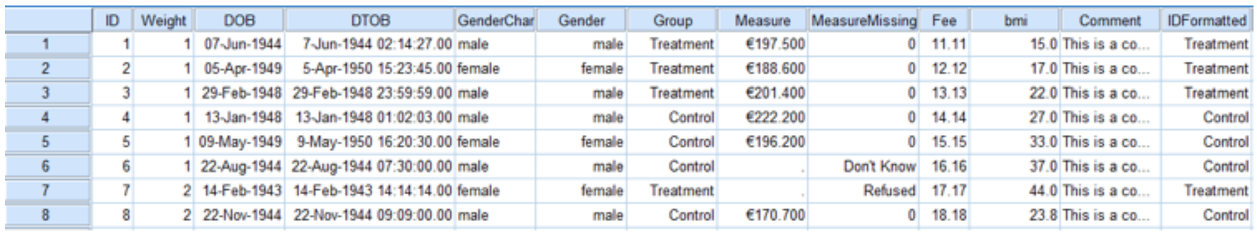
A few metadata elements common to most of the packages never transferred correctly:

- Display formats – This is really no surprise given that there are no standards for display formats across the packages.



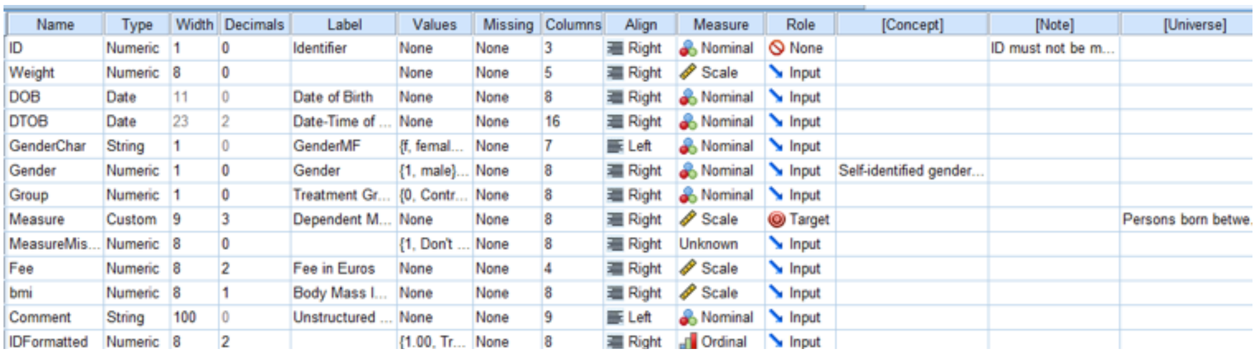
ID	Weight	DOB	DTOB	GenderChar	Gender	Group	Measure	MeasureMissing	Fee	bmi	Comment	IDFormatted
1	1	07-Jun-1944	7-Jun-1944 02:14:27.00	m	1	1	€197.500	0	11.11	15.0	This is a co...	1.00
2	1	05-Apr-1949	5-Apr-1950 15:23:45.00	f	2	1	€188.600	0	12.12	17.0	This is a co...	2.00
3	1	29-Feb-1948	29-Feb-1948 23:59:59.00	m	1	1	€201.400	0	13.13	22.0	This is a co...	3.00
4	1	13-Jan-1948	13-Jan-1948 01:02:03.00	m	1	0	€222.200	0	14.14	27.0	This is a co...	4.00
5	1	09-May-1949	9-May-1950 16:20:30.00	f	2	0	€196.200	0	15.15	33.0	This is a co...	5.00
6	1	22-Aug-1944	22-Aug-1944 07:30:00.00	m	1	0	.	1	16.16	37.0	This is a co...	6.00
7	2	14-Feb-1943	14-Feb-1943 14:14:14.00	f	2	1	.	2	17.17	44.0	This is a co...	7.00
8	2	22-Nov-1944	22-Nov-1944 09:09:00.00	m	1	0	€170.700	0	18.18	23.8	This is a co...	8.00

Figure 4 - The SPSS dataset with value labels hidden



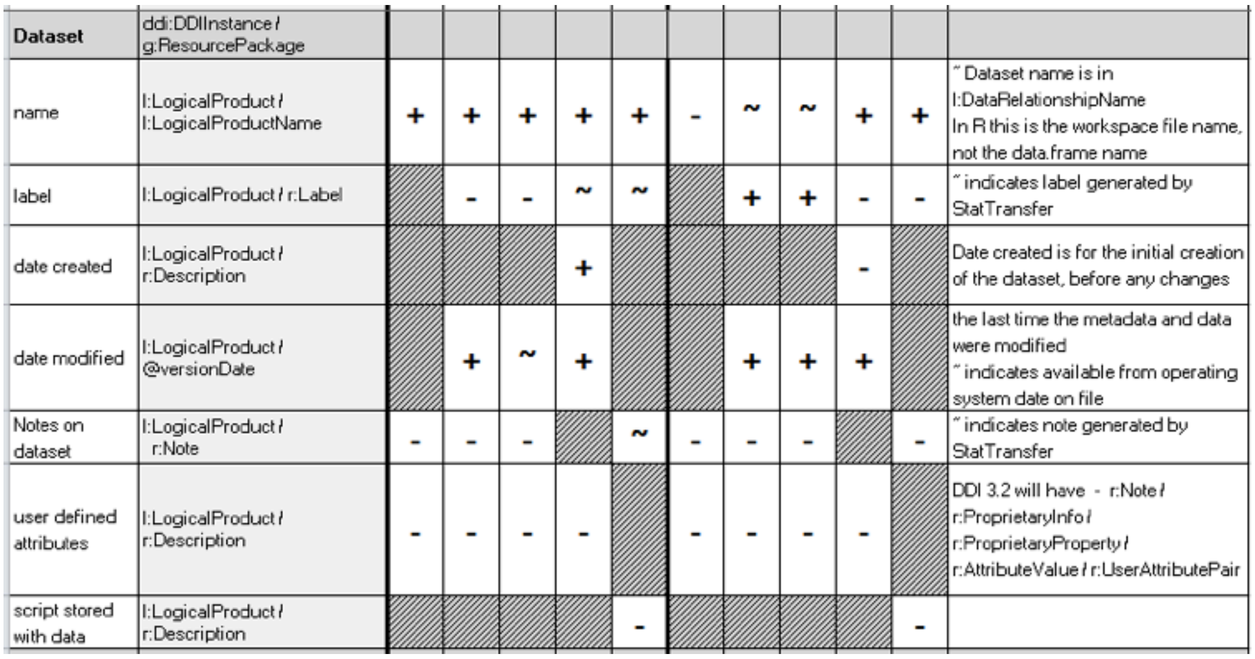
	ID	Weight	DOB	DTOB	GenderChar	Gender	Group	Measure	MeasureMissing	Fee	bmi	Comment	IDFormatted
1	1	1	07-Jun-1944	7-Jun-1944 02:14:27.00	male	male	Treatment	€197.500	0	11.11	15.0	This is a co...	Treatment
2	2	1	05-Apr-1949	5-Apr-1950 15:23:45.00	female	female	Treatment	€188.600	0	12.12	17.0	This is a co...	Treatment
3	3	1	29-Feb-1948	29-Feb-1948 23:59:59.00	male	male	Treatment	€201.400	0	13.13	22.0	This is a co...	Treatment
4	4	1	13-Jan-1948	13-Jan-1948 01:02:03.00	male	male	Control	€222.200	0	14.14	27.0	This is a co...	Control
5	5	1	09-May-1949	9-May-1950 16:20:30.00	female	female	Control	€196.200	0	15.15	33.0	This is a co...	Control
6	6	1	22-Aug-1944	22-Aug-1944 07:30:00.00	male	male	Control	.	Don't Know	16.16	37.0	This is a co...	Control
7	7	2	14-Feb-1943	14-Feb-1943 14:14:14.00	female	female	Treatment	.	Refused	17.17	44.0	This is a co...	Treatment
8	8	2	22-Nov-1944	22-Nov-1944 09:09:00.00	male	male	Control	€170.700	0	18.18	23.8	This is a co...	Control

Figure 5 - The SPSS dataset showing value labels



Name	Type	Width	Decimals	Label	Values	Missing	Columns	Align	Measure	Role	[Concept]	[Note]	[Universe]
ID	Numeric	1	0	Identifier	None	None	3	Right	Nominal	None		ID must not be m...	
Weight	Numeric	8	0		None	None	5	Right	Scale	Input			
DOB	Date	11	0	Date of Birth	None	None	8	Right	Nominal	Input			
DTOB	Date	23	2	Date-Time of ...	None	None	16	Right	Nominal	Input			
GenderChar	String	1	0	GenderMF	{f, femal...	None	7	Left	Nominal	Input			
Gender	Numeric	1	0	Gender	{1, male}...	None	8	Right	Nominal	Input	Self-identified gender...		
Group	Numeric	1	0	Treatment Gr...	{0, Contr...	None	8	Right	Nominal	Input			
Measure	Custom	9	3	Dependent M...	None	None	8	Right	Scale	Target			Persons born betwe.
MeasureMis...	Numeric	8	0		{1, Don't ...	None	8	Right	Unknown	Input			
Fee	Numeric	8	2	Fee in Euros	None	None	4	Right	Scale	Input			
bmi	Numeric	8	1	Body Mass I...	None	None	8	Right	Scale	Input			
Comment	String	100	0	Unstructured ...	None	None	9	Left	Nominal	Input			
IDFormatted	Numeric	8	2		{1.00, Tr...	None	8	Right	Ordinal	Input			

Figure 6 - Variable properties from the SPSS dataset, including user defined properties - Concept, Note, and Universe



Dataset	ddi:DDIInstance / g:ResourcePackage											
name	l:LogicalProduct / l:LogicalProductName	+	+	+	+	+	-	~	~	+	+	~ Dataset name is in l:DataRelationshipName In R this is the workspace file name, not the data.frame name
label	l:LogicalProduct / r:Label	▨	-	-	~	~	▨	+	+	-	-	~ indicates label generated by StatTransfer
date created	l:LogicalProduct / r:Description	▨	▨	▨	+	▨	▨	▨	▨	-	▨	Date created is for the initial creation of the dataset, before any changes
date modified	l:LogicalProduct / @versionDate	▨	+	~	+	▨	▨	+	+	+	▨	the last time the metadata and data were modified ~ indicates available from operating system date on file
Notes on dataset	l:LogicalProduct / r:Note	-	-	-	▨	~	-	-	-	▨	-	~ indicates note generated by StatTransfer
user defined attributes	l:LogicalProduct / r:Description	-	-	-	-	▨	-	-	-	-	▨	DDI 3.2 will have - r:Note / r:ProprietaryInfo / r:ProprietaryProperty / r:AttributeValue / r:UserAttributePair
script stored with data	l:LogicalProduct / r:Description	▨	▨	▨	▨	-	▨	▨	▨	▨	-	

Figure 7 – Example transfer results

- Notes on datasets or variables– Most of the packages have some facility for general notes on a dataset or a variable. These were not successfully transferred.
- User defined attributes on datasets or variables – Several of the packages allow for user defined attributes on the dataset or on individual variables. These did not transfer successfully.
- Measurement level – Several packages allow for the specification of measurement level for variables. The vocabulary for measurement level varies across packages though.
- Weight – SPSS and JMP can store an attribute indicating that a variable functions as a weight. This never transferred.

Results in more detail

Dataset Level

Successful

- Dataset name (except R)
- Date Modified

Issues

In most cases dataset labels, dataset date modified, and value labels for numeric variables transferred. Value labels do not transfer to R, but this is not unexpected, since factors in R are somewhat conceptually different than a labeled variable in the other packages.

For the 5 packages studied, dataset name is typically not included in the dataset file itself, but rather is contained in the name of the file (at the operating system level). If the dataset name is taken from the file name this is a potential problem for R where multiple data frames may be contained in the workspace file. A DDI file should contain that name in l:LogicalProduct/l:LogicalProductName. Additionally, the filename should be captured in pi:PhysicalInstance/pi:DataFileIdentification.
Metadata elements which are not common across the packages did not transfer well, even when possible. These include user defined characteristics of the dataset, notes (which are sometimes a user defined characteristic) and scripts embedded in the dataset (supported only by JMP in this collection). Rule based integrity constraints also did not transfer.

We did not include foreign key constraints or passwords on the sample datasets.

Variable Level

Successful

- Variable name
- Basic data type
- Position
- Variable label
- System missing values
- Value labels – numeric variables
- Value labels – text variables (where possible)

Issues

At the variable level, display formats, such as a leading Euro symbol, did not transfer at all. Given that display formats are not standardized across packages, this is not surprising.
Other elements which did not transfer are: number of decimal positions, scale (not supported in most packages), measurement units, measurement level, variable as a weight, role, user-defined variable attributes, and notes on variables. Some of these, such as weight, and measurement units are really essential for interpreting the data. Others carry meaning beyond cosmetics. Number of decimal places, for example, can connote the level of precision of measurement.

Missing Values

Several packages have the ability to use multiple distinct values to indicate different categories of missing data. Stata and SAS both have a set of "out of band" values to represent distinct missing values. These are displayed as ".A" through ".Z" and "._". SPSS, instead, allows "in band" values to be chosen as missing values, a "9" or a "999", for example. SPSS also allows one range of values to be declared missing, e.g. all values between 90 and 99 inclusive. R has only one missing value "NA", although it also has indicators for infinite numbers "Inf" e.g. the result of 1/0 and "not a number" ("NaN") such as the result of 0/0 . Transferring variables with multiple distinct missing values among packages is not straightforward. DDI3.2 added the ManagedMissingValuesRepresentation element which allows adding a code scheme for missing to a numeric representation. This is consistent with the notion of "sentinel values" in ISO 11404.

One approach to avoid these issues in datasets to be transferred (or archived) is to just use the system missing value in the primary variable and then create a secondary dummy variable with labeled indicators to indicate categories of missing. These dummy variables could also be shared in a resource package.

Multiple Value Labels
Both SAS and Stata keep labels for values separately from variables. This allows multiple variables to share one set of labels. It also allows for alternative labels to be used in different contexts – e.g. longer labels for tabulation rows than tabulation columns, or labels in multiple languages. Each of these packages allows a variable to have an association with one set of labels at a time. Unaffiliated labels can exist in memory during a Stata session but a Stata ".dta" file only stores the sets of labels currently associated with variables. Stata also has a dta XML export format that will export all sets of labels currently defined in a Stata session. SAS can export the sets of labels (called "formats") into a separate dataset, a CNTLIN / CNTLOUT dataset. With both packages the definition of multiple sets of labels can also exist in script files.

An example of multiple formats in SAS follows, with short labels for a variable "gender" in two languages, and a longer set of labels in English. The English value is attached to the variable.

```
proc format cntlout=eddi.sas_Fmts;
  value GENDERen
      1="Male" 2="Female";
  value GENDERde
      1="Männlich" 2="Weiblich";
  value GenderL
      1="Self Identified Male" 2="Self
  Identified Female";
…
  format gender GENDERen.;
```

The Stata example below show the same three sets as value labels.

```
label define GenderE 1 "Male" 2 "Female"
label define GenderG 1 "Männlich" 2 "Weiblich"
label define GenderL 1 "Self Identified Male"
                     2 "Self Identified Female"
label values Gender GenderE
```

In both of the preceding examples the language is represented by a user-defined convention. Labels of a particular language cannot be selected in some general machine actionable way.

DDI is capable of representing these multiple sets of labels in l:CategorySchemes and l:CodeSchemes. DDI links to one l:CodeScheme from a variable, but each l:Category in the l:CategoryScheme linked from that l:CodeScheme may have multiple labels, distinguished by xml:lang and type attributes. There currently appears to be no way though, to indicate which label is the default, or currently associated label.

In the DDI example below the labels for the male code are differentiated by both language, with the "xml:lang" attribute, and type, with the "type" attribute. DDI can associate the whole set with an l:Code, but cannot indicate which label was the currently assigned label.

```
<l:Category id="c1" version="1.0.0"
versionDate="2011-10-26T13:33:00"
  missing="false" >
        <r:Label xml:lang="en-US" type="GENDER"
>male</r:Label>
        <r:Label xml:lang="de" type="GENDER"
>männlich</r:Label>
        <r:Label xml:lang="en-US" type="GENDERL"
>Self Identified Male</r:Label>
```

None of the non-linked sets of labels exported from Stata or SAS to DDI in our experiment.

Role
Several of the packages have a defined variable attribute of "role". This may be used to indicate which variables are eligible to be independent or dependent variables in an analysis, or to indicate more specific functions. Given that the vocabulary for "role" is not standardized across packages, it is not surprising that this metadata element does not transfer. Mapping against a commonly accepted controlled vocabulary would allow machine actionable decisions about comparable or incomparable proprietary terms.

Custom (User Defined) Variable Attributes
With the addition of extended attributes to SAS version 9.4, all of the packages evaluated allowed the definition of custom attributes for variables. None of these were exported to DDI in our tests. DDI 3.2 has a facility for recording these name, value pairs in `r:UserAttributePair elements`. The adoption of a commonly accepted controlled vocabulary by data producers would allow mapping into defined DDI elements.

Labeled Ranges
Both SAS and JMP have the capability to label ranges of values for a variable. In this use SAS formats act as an analog to a normalized structure in a relational database, allowing information to be recorded in only one place (variable). SAS programs can use these formats dynamically to perform analyses on the categorized variables. Since there is currently no good way to represent these in DDI, these did not transfer to DDI. Perhaps it is not best practice to rely on them for archival datasets and instead create additional categorized variables.

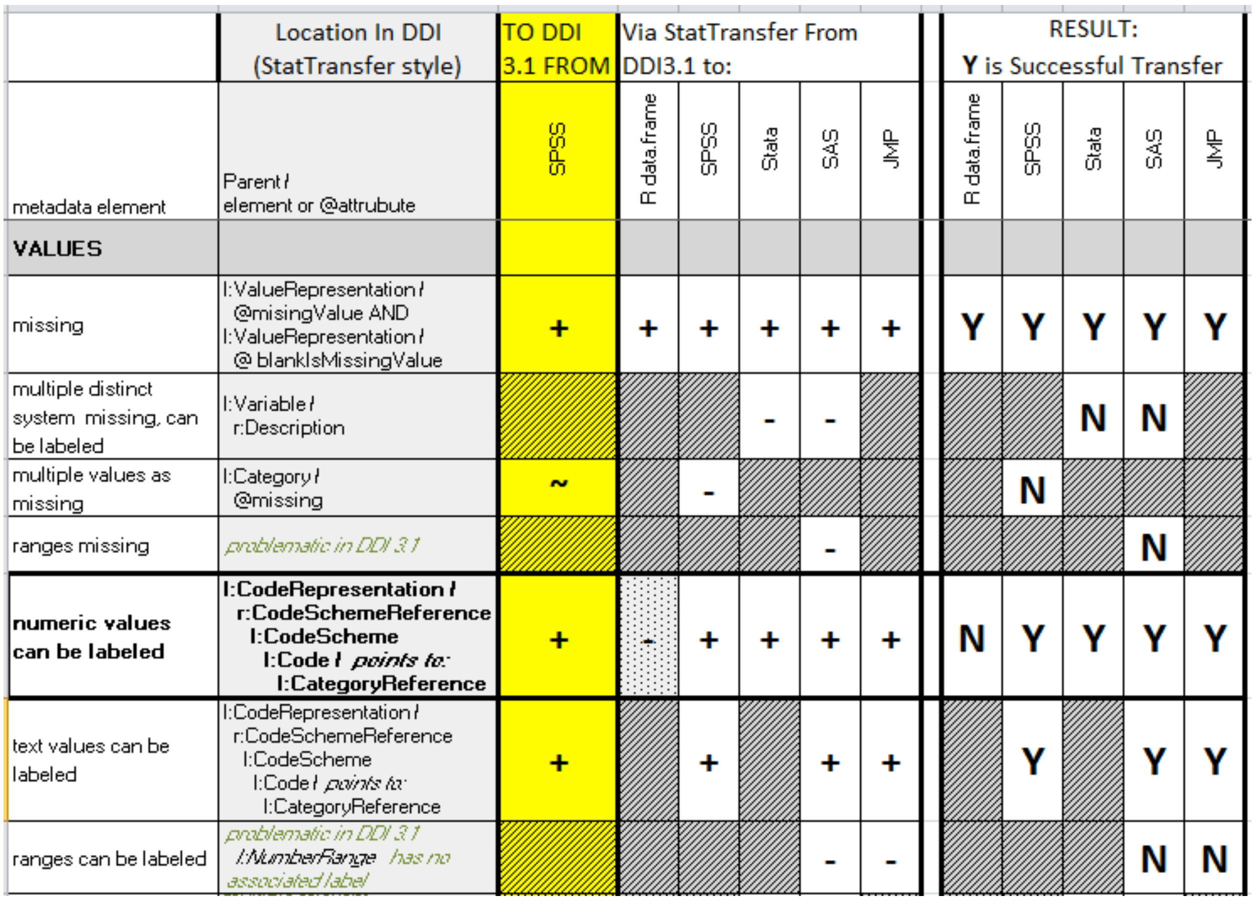
Built-in Display Formats
Some display formats built in to the various packages serve mostly cosmetic functions – left or right alignment, the choice of decimal or thousands separator characters, and so on. Others convey important meaning. Currency symbols, for example, denote units of measurement. Some sort of standardized representation for at least a subset of formats across packages would be very useful.

Date and Time
Date and time conversion was not completely tested. Date and time types are realized in the different packages in various ways. Some (like SAS) have the capability of exporting date and time data in ISO formats. Date and Time formats according to ISO 8601 should be used to assure interoperability of programs (Wikipedia.

ISO 8601). A time value's dependence on a time zone should be carefully checked and documented. Offsets from Coordinated Universal Time (UTC) should be included where meaningful. Stat/Transfer provides a general means to specify the date and time format for export into a CSV file where all values for a variable have the same offset. A format according to ISO can be configured but is not provided by default. A fixed time zone offset can be added to all values for the variable. As an example, if all values for a variable were Central European Daylight Saving Time (CEDT), a time value could be written as: 2009-06-30T18:30:00**+02:00** i.e. 18:30:00 on 30. June 2009 (CEDT). Note that the time zone offset of +2.00 would be a constant across all values for the variable. If observations in the dataset might all have different offsets, the only option would be to add an additional variable containing the offset.

R and DateTime
One issue we encountered with transferring datetime variables in and out of R, was that of non-explicit specification of whether datetime values represented UTC or local values. When the different programs involved made different default assumptions datetime values got shifted by the local offset from UTC. Explicit inclusion of the time zone in datetime values would avoid this problem.

Here to There (and Back Again?)
The grid like shown in Figure 7 can be used to predict what metadata will survive a trip from one package to DDI and then to another package. We created new grids showing that evaluation for each of the packages. In the third to last row of Figure 8, for example, labels for numeric values can be seen to transfer from SPSS through DDI (the yellow column, labeled "To DDI 3.1 FROM") to SPSS, Stata, SAS, and JMP. They would not survive the journey from SPSS to R since R factors do not correspond exactly to labeled numeric variables. The complete set of these grids can be seen in Appendix 3.



	Location In DDI (StatTransfer style)	TO DDI 3.1 FROM	Via StatTransfer From DDI3.1 to:					RESULT: Y is Successful Transfer				
metadata element	Parent / element or @attrubute	SPSS	R data.frame	SPSS	Stata	SAS	JMP	R data.frame	SPSS	Stata	SAS	JMP
VALUES												
missing	l:ValueRepresentation / @misingValue AND l:ValueRepresentation / @ blankIsMissingValue	+	+	+	+	+	+	Y	Y	Y	Y	Y
multiple distinct system missing, can be labeled	l:Variable / r:Description	▨	▨	▨	-	-	▨	▨	▨	N	N	▨
multiple values as missing	l:Category / @missing	~	▨	-	▨	▨	▨	▨	N	▨	▨	▨
ranges missing	*problematic in DDI 3.1*	▨	▨	▨	▨	-	▨	▨	▨	▨	N	▨
numeric values can be labeled	**l:CodeRepresentation / r:CodeSchemeReference l:CodeScheme l:Code / *points to:* l:CategoryReference**	+	-	+	+	+	+	N	Y	Y	Y	Y
text values can be labeled	l:CodeRepresentation / r:CodeSchemeReference l:CodeScheme l:Code / *points to:* l:CategoryReference	+	▨	+	▨	+	+	▨	Y	▨	Y	Y
ranges can be labeled	*problematic in DDI 3.1 l:NumberRange has no associated label*	▨	▨	▨	▨	-	-	▨	▨	▨	N	N

Figure 8 – Example Here to There Grid

Suggestions for DDI
This experiment generated several suggestions for DDI 3.1. These are listed below in rough order from the most specific to the most general.

Multiple labels for a category
A mechanism to indicate whether one of a set of Labels for an l:Category was currently assigned to a variable (or not) would be useful for representing SAS and Stata data. It would also add clarity to the current representation of multiple labels for a Category in DDI. This could perhaps rely on the "type" attribute of the Label element.

The following example shows a set of labels for a gender variable, both in multiple languages and with a "long form" in English.

Which "type" attribute should be selected by default when referencing the Category?

```
<l:Category id="Gm" version="1.0.0" versionDate="2011-
   10-26T13:33:00"
      missing="false">
     <r:Label xml:lang="sv"
   type="GENDERShort">kvinna</r:Label>
     <r:Label xml:lang="de"
   type="GENDERShort">weiblich</r:Label>
     <r:Label xml:lang="en-US"
   type="GENDERShort">female</r:Label>
     <r:Label xml:lang="en-US" type="GENDERLong">Self
   Identified
      Female</r:Label>
   </l:Category>
```

Perhaps this could be specified within the l:CodeRepresentation element. One possibility would be for l:Representation within l:Variable to contain a "PrimaryLabelType" selecting a "type" attribute from the set of labels as in the example below.

```
<l:Representation>
     <l:CodeRepresentation blankIsMissingValue="true"
      classificationLevel="Nominal">
       <r:RecommendedDataType>string</
   r:RecommendedDataType>
       <l: PrimaryLabelType > GENDERshort </l:
   PrimaryLabelType>
       <r:CodeSchemeReference>
         <r:ID>5c706c37-d19b-4b8e-ac6d-
   40094024421f</r:ID>
         <r:IdentifyingAgency>example.org</
   r:IdentifyingAgency>
         <r:Version>1</r:Version>
       </r:CodeSchemeReference>
     </l:CodeRepresentation>
   </l:Representation>
```

Ranges

A facility to assign categories to ranges of coded and numeric variables would allow representation of these features from SAS and JMP datasets. It would also allow a range of values to be labeled as missing. DDI 3.2 allows this for numeric with a ManagedNumericRepresentation. See the Conclusions section for more details.

Precision vs. display information

Numeric output formats can convey an ambiguous amount of information about the precision of measurement of a variable. By convention, a value formatted in "scientific notation" as 1.2345 X10^6 would be considered to have been measured to five digits of precision. The level of precision for same value formatted as a decimal with zero digits to the right of the decimal point (1234500) is not clear. It would be useful for DDI to have an explicit representation of level of precision of measurement.

Integrity Constraints

Several of the packages we evaluated allow for defining some sort of integrity constraint on a variable, either by a list of valid values or by logical expression. An expression like mod(age,1)=0, for example could be used to only accept integer values of age (even though stored as type float). Expressions might also refer to multiple variables, as in an expression limiting years of education to age – 5. SAS also allows foreign key integrity constraints, only allowing entry of values appearing in a column in another table.

DDI 3.1 seems to lack an explicit representation of integrity constraints. DDI 3.2 has a workaround described in the Conclusions section below.

Data Types

The data types defined in the statistical packages generally do not transfer perfectly. Note that data type is distinct from display format. The most commonly used data types in the packages are integer, double precision, Boolean, and character strings. The data types can be captured in DDI in p:PhysicalLocation/p:StorageFormat. The related documentation recommends the use of a controlled vocabulary. One way would be to develop it on the basis of XML Schema data types (W3C 2004) and/or SQL data types (Wikipedia. SQL -- Data Types) which both seem to comprehend the most used forms of data types.

Without a standard for output formats against which to map the various proprietary formats, automated translation from one package to another via DDI is difficult. Documentation for the DDI element r:GenericOutputFormat recommends the use of a controlled vocabulary. Perhaps one could be developed based on Java, C, or Fortran.

Multiple distinct system missing values

DDI could use a more explicit method of representing multiple "out of band" missing values such as used by SAS and Stata, or the Inf, NA or NaN values in R. These values may sometimes also be labeled, necessitating links to l:Categories. DDI 3.2 allows this with the ManagedMissingValuesRepresentation element.

Text descriptions only

Several other generic metadata elements that can be represented in some of the packages can currently only be represented as r:Description or r:Note elements in DDI. More machine actionable representations of these elements would be useful.

Date created -	Some packages (and operating systems) may track not only the date a dataset was last modified, but also its initial creation date. In DDI3.2 this can be recorded in pi:PhysicalInstance/pi:DataFileVersion/@VersionDate or pi:PhysicalInstance/r:Citation/dc:created. Publication date can be recorded separately in pi:PhysicalInstance/r:Citation/r:PublicationDate
Scripts -	Some packages can store scripts in the same structure as the dataset. Currently these can only be represented in DDI as an l:LogicalProduct/r:Description element. A DDI element identifying the script as a script written in a particular language (e.g. R, Jmp Scripting Language) would be useful.
Notes -	Some packages allow a "Note" attribute to be attached to a dataset or variable. While such a note can be preserved in an an l:LogicalProduct/r:Note, it would be more machine actionable to specifically identify the text as having come from a "note" in the original format. Some packages treat a note as just another (name,value) pair. In DDI3.2 this could be recorded with a value

of "note" in the r:AttributeKey element of a r:AttributeKey , r:AttributeValue pair in either an r:ProprietaryProperty or in an r:UserAttributePair.

Value colors -	Some packages allow assignment of colors to particular sets of values or observations. Colors may represent some manually assigned attribute of the data, such as suspected outliers, or, in the case of SPSS, imputed values. While in one sense this could be represented by a code and category scheme, it might be more machine actionable to flag the color values as representing colors in some way.
Filters (Triple-S) -	Triple-S allows a variable to point to another variable as it's "filter". When the filter variable has the value TRUE, the variable pointing to it is available for that case. A workaround for numeric variables is described in the Conclusions section below.

Suggestions for transport programs

Internal model for categories and codes

Some packages, like SPSS or JMP, store value labels as attributes of variables. Others, like Stata and SAS, store sets of labels (formats in SAS) separately and tie them to variables by reference. Since the latter method can represent anything the former does, it should be the basis for the internal model of value labels (categories and codes in DDI).

Detect reuse of sets of value labels (categories and codes)

Once sets of value labels are represented independently from variables and used by reference, they do not need to be defined multiple times. When converting from a representation where a given set of labels might be defined multiple times (e.g. a Likert scale in a survey) to one like DDI where a set can be reused, it is desirable to identify and eliminate the duplication. For one approach to this process see (Wright, 2011).

Multiple sets of categories and codes applicable to a variable

As discussed above, it is possible to have multiple sets of categories and codes applicable to a given variable, for example labels in different languages or of different lengths. With both SAS and Stata only one of the sets can be associated with a given variable. With DDI multiple languages and types can be associated with a variable. An internal model that allows multiple associations and specification of a default or primary set would allow representation of all of the possibilities.

Use the Dataset name when distinct from the file name

In cases like R where the name of the dataset is not necessarily the same as the name of the file containing it, the dataset name should be used.

Generic vs. Proprietary Information

Metadata harvested from a proprietary dataset can be classed into four categories:

1. Generic information that corresponds to a DDI element
2. Generic information for which there is no specific DDI element
3. Proprietary information that corresponds to a DDI element
4. Proprietary information for which there is no DDI element

With DDI 3.1, information of type 2 could only be recorded in an unstructured **r:note**. Information of type 4 could be recorded in a **r:ProprietaryProperty**. With DDI 3.2, information of both types 2 and 4 can be recorded in **r:AttributeKey , r:AttributeValue pairs** – in either an **r:proprietaryProperty** (category 4) or an **r:UserAttributePair** (category2).

Packaging Structure

DDI3.1 allows packaging many elements in either an s:StudyUnit or a g:ResourcePackage. DDI 3.2 adds a third alternative with the ddi:FragmentInstance element. Ideally a transport program should be able to handle any of the three structures. Unfortunately this is not always the case. The DDI community should probably make a recommendation for a preferred structure for transport instances.

Suggestions for statistical and data management software packages

More metadata

All of the packages reviewed here are lacking in the ability to include enough structured metadata in a dataset to actually interpret the data. Many packages allow the attachment of (name, value) pairs of attributes, but without those names and values coming from some sort of structure or controlled vocabulary the metadata have limited interpretability or searchability beyond the data's creators.

Metadata elements such as concept and universe are relevant to a broad array of data. Knowing that a variable was only measured on male children, for example, is important when drawing inferences based on analysis of that variable. For data captured by surveys, at a minimum it is important to know the exact text of the question asked of respondents. Metadata about groups of concepts, questions, and variables are also important. Geographers and others point out that all data are spatial. Metadata about spatial coverage are proving increasingly useful.

Flexible structure

Instances of metadata from many metadata standards are expressible in XML. One possibility would be for statistical and data management packages to add the capability to attach an xml instance from a defined standard to their datasets. For well-known standards like DDI it would also be possible to integrate the use of that XML into their procedures. Survey analysis procedures could incorporate metadata like question text and even question flow. With some software packages it might be possible to include DDI XML in a key value pair.

Suggestions for research practices and preparing archival datasets

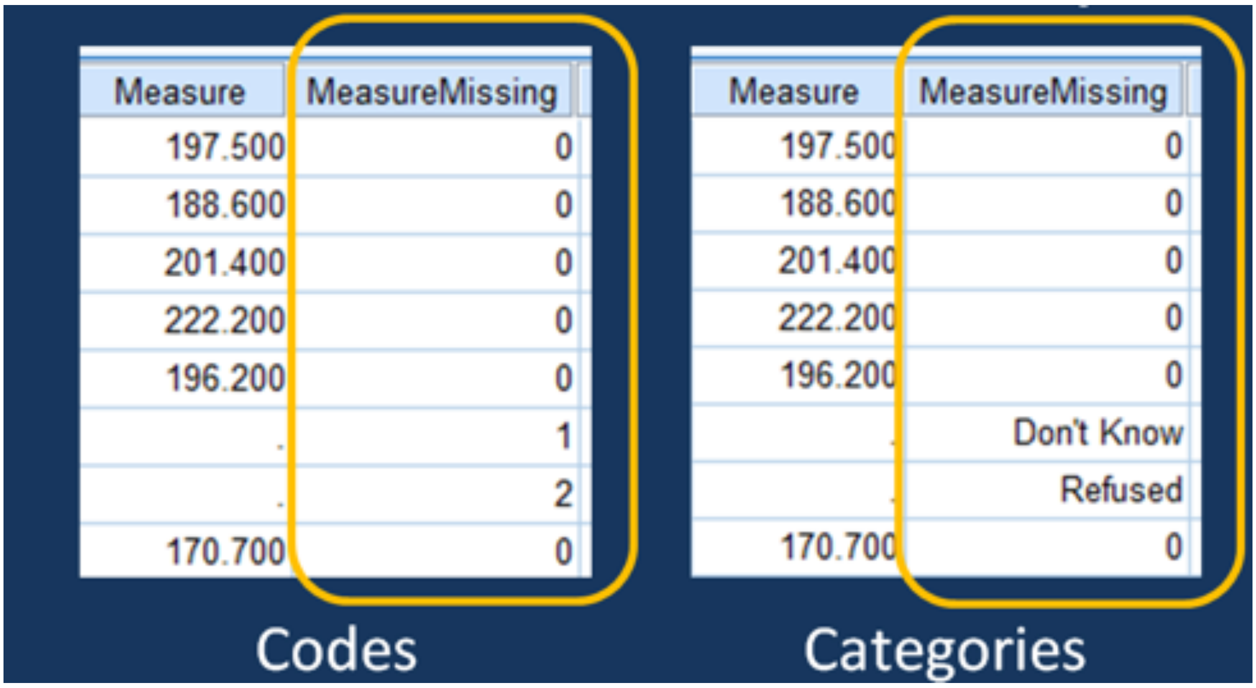
This experiment generated a few suggestions for the practice of preparing archival datasets.

Reason for missing – multiple missing types

Use an auxiliary variable to indicate reason for missing. These could be shared in a g:ResourcePackage. Figure 9 shows a variable "MeasureMissing" which differentiates type of missing for the variable "Measure". The pairing of these two variables could be documented with a l:VariableGroup element.

Alternative formats

Create additional variables for data representable with alternative formats



Measure	MeasureMissing
197.500	0
188.600	0
201.400	0
222.200	0
196.200	0
.	1
.	2
170.700	0

Codes

Measure	MeasureMissing
197.500	0
188.600	0
201.400	0
222.200	0
196.200	0
.	Don't Know
.	Refused
170.700	0

Categories

Figure 9 - An auxiliary variable for "Measure" indicating type of missing

- Long labels
- Languages
- Coded ranges

Doing this for coded ranges requires some additional metadata though. The continuous variable Body Mass Index (BMI), for example, has associated ranges indicating categories such as "underweight", and "obese". These ranges are further broken down in a hierarchy with multiple sub-categories. An additional variable could be recoded from a BMI measurement, but the rules for recoding would also need to be recorded (in a d:GenerationInstruction) in order to ensure that the exact ranges for each category were captured.

User attributes

Use a controlled vocabulary for the names of user attributes (characteristics, properties) where available. If this practice were common, then a much wider range of metadata would be transferrable across packages, without the need for revisions to the programs. One possibility for such a controlled vocabulary might lie in a semantic data form of DDI elements.

Where available, also use a controlled vocabulary for the values of user defined attributes. An example would be to use an attribute named "AnalysisUnit" with values taken from the DDI AnalysisUnit controlled vocabulary. (DDI Controlled Vocabularies Working Group)

Time zones

Where possible, explicitly specify time zone information in datetime values. Datetime values without explicit specification of time zone may be unpredictably interpreted as local times or universal times as they are converted from package to package, leading to changes in the data.

Conclusions

A few general conclusions can be drawn from this experiment:

Adoption of DDI by tools like Stat/Transfer is encouraging. Basic metadata is transferrable among all 5 packages via DDI.

The current state still means that some important metadata that might be contained in proprietary format data files still must be either hand entered into DDI or harvested and entered by user-written code.

No one package has a superset of the other's metadata. Several desirable elements are not universally supported. Some desirable elements like concept and question are not supported by any of the packages, except as user defined (name, value) pairs.
The fact that all of the metadata typically recorded in a proprietary dataset file can be represented in a g:ResourcePackage without an s:StudyUnit reveals a lack of a structured facility for recording information about the origin of that dataset. The development of best practice recommendations for using custom attributes of variables and datasets could be one approach toward remedying this situation.

As mentioned earlier, DDI3.1 allows packaging many elements in either an s:StudyUnit or a g:ResourcePackage and DDI 3.2 adds a third alternative with the ddi:FragmentInstance element. The DDI community should specify one preferred packaging structure to be used as a transport instance.

DDI is almost a superset of the packages considered. Being able to represent a superset of metadata elements across the most commonly used packages is a worthy goal for DDI. Some missing elements like mentioned above should be added for this purpose. Our suggested list of needs follows.

- A facility to define an assigned set of value labels to a variable. This could be done by specifying a preferred type and language for subsetting a category scheme.
- A facility to assign categories to ranges of coded and numeric variables. DDI3.2 now allows a ManagedNumericRepresentation

to have a set of NumberRange elements, each of which can assign a label to the range. ManagedDateTimeRepresentation allows a label to be assigned to a duration. The ManagedTextRepresentation element does not have a corresponding "TextRange" element.

- A method of recording the level of measurement precision to variables, e.g. the number of significant digits. This is distinct from the number of digits to be displayed to the right of the decimal point. As an example the number written as 123000 has 0 digits to the right of the decimal point and carries no information about the precision to which it was measured. If expressed as 1.230E5, though, the convention is that there are four significant digits.
- A facility for recording complex integrity constraints. In DDI3.2 a r:ProcessingInstructionReference can be attached to a l:VariableRepresentation which can then refer to a d:GeneralInstruction. This could be used to describe a constraint as a logical expression that must hold true for the value of the variable. This is a usable workaround, but the semantics are too narrow in that a constraint is a rule for the variable independent of how it is applied. A more general approach in future DDI version could be an Instruction element having a type attribute. The type might take on values of "derivation", or "constraint", or "processing". Note that foreign keys can be described with a RecordRelationship.
- Integrity constraints can also take the form of "Unique" or "Not Null" (the combination being required for a key). These specifications could be added to VariableRepresentation or ValueRepresentation to be inherited by substituted elements.
- Development of controlled vocabularies for data types and output formats
- A more explicit method of representing multiple distinct system missing values. In DDI3.2 this is now possible using a ManagedMissingValuesRepresentation. This is still a significant challenge when moving data from a package supporting multiple missing values to one (e.g. R) which does not.
- A method for explicitly representing scripts stored in a dataset. Currently scripts for specific data transformations can be stored in d:GeneralInstruction. Scripts for other purposes or roles such as analyses or visualization could use some structure. In the future this may become part of a process model in DDI.
- A facility for recording colors attached to either variables or sets of observations, including the color value and an associated concept or category. With DDI3.2 Color assignments for a code could be recorded in a r:UserAttributePair attached to its associated l:Category. For numeric variables a color gradient could be recorded by r:UserAttributePair elements attached to a r:ManagedNumericRepresentation. A work-around for assigning colors to individual values could use a structured r:Label of a r:NumberRange.
- Facilities for a variable to point to a Boolean variable as its filter (missing indicator) and to a categorical variable to indicate different types of missing. For a Boolean filter of a numeric variable, a weight variable is equivalent to a filter indicator, where a weight of 0 indicates missing and a weight of 1 indicates valid. This is not technically correct for a string variable and does not work for the categorical case. What would be desirable is a reference to a variable for which a role could be specified.

Future Work

A longer version of this article is planned for publication in the DDI Alliance Working Paper series. The paper will focus additionally on solutions which are developed in the DDI Moving Forward project on the next generation DDI. The appendix will have an extensive mapping table describing the metadata elements in each package and DDI.

References

ALGENTA TECHNOLOGIES 2012. Colectica Reader - The Free DDI 3 Viewer - DDI Metadata and Survey Design Software.

AMIN, A., BARKOW, I., KRAMER, S., SCHILLER, D. & WILLIAMS, J. 2011. Representing and Utilizing DDI in Relational Databases. DDI Working Paper Series: DDI Alliance. (doi: http://dx.doi.org/10.3886/DDIOtherTopics02).

CIRCLE SYSTEMS Stat/Transfer. (URL: http://www.stattransfer.com/).

DATA DOCUMENTATION INITIATIVE. 2013a. DDI RDF Vocabularies | DDI - Data Documentation Initiative [Online]. Available: http://www.ddialliance.org/Specification/RDF.

DATA DOCUMENTATION INITIATIVE. 2013b. DDI Tools | DDI - Data Documentation Initiative [Online]. Available: http://www.ddialliance.org/resources/tools.

DATA DOCUMENTATION INITIATIVE. 2013c. Future Plans for DDI Development | DDI - Data Documentation Initiative [Online]. Available: http://www.ddialliance.net/ddi-moving-forward-process-summary.

DDI ALLIANCE EXPERT COMMITTEE. 2009. DDI 3.1 XML Schema Documentation (2009-10-18) [Online]. Available: http://www.ddialliance.org/Specification/DDI-Lifecycle/3.1/XMLSchema/FieldLevelDocumentation/.

DDI CONTROLLED VOCABULARIES WORKING GROUP DDI Controlled Vocabularies - Overview Table. DDI Alliance. (URL: http://www.ddialliance.org/Specification/DDI-CV/).

HOYLE, L., WACKEROW, J. & HOPT, O. 2010. DDI 3: Extracting Metadata from the Data Analysis Workflow. In: VARDIGAN, M., EDWARDS, M. & HOYLE, L. (eds.) DDI Working Paper Series -- Use Cases. DDI Alliance, http://www.ddialliance.org/resources/publications/working/usecases: Data Documentation Initiative Alliance. (doi: http://dx.doi.org/10.3886/DDIUseCases04).

IBM IBM SPSS Statistics. (URL: http://www.spss.com).

R DEVELOPMENT CORE TEAM 2009. R: A language and environment for statistical computing. In: COMPUTING, R. F. F. S. (ed.). Vienna, Austria. (URL: http://www.R-project.org).

SAS INSTITUTE - JMP JMP Statistical Discovery Software. SAS Institute. (URL: http://www.jmp.com/).

SAS INSTITUTE - SAS The SAS System. (URL: http://www.sas.com).

STATACORP Stata. (URL: http://www.stata.com/).

W3c. XML Schema Part 2: Datatypes Second Edition W3C Recommendation 28 October 2004 (URL: http://www.w3.org/TR/xmlschema-2/#built-in-primitive-datatypes)

Wikipedia. SQL -- Data Types (URL http://en.wikipedia.org/wiki/SQL#Data_types)

Wikipedia. ISO 8601 (URL:http://en.wikipedia.org/wiki/ISO_8601)

WRIGHT, P. A. 2011. Eliminating Redundant Custom Formats. SAS Global Forum 2011 SAS Institute. (URL: http://support.sas.com/resources/papers/proceedings11/217-2011.pdf).

Appendices

Appendices and possible other related material may be found at http://hdl.handle.net/1808/19900.

A PDF document containing the three appendices may also be accessed directly at https://kuscholarworks.ku.edu/bitstream/handle/1808/19900/DDIasaCommonFormatIassistQAppendices.pdf.

Notes

1 Larry Hoyle is a Senior Scientist at the Institute for Policy & Social Research at the University of Kansas and can be reached by email: LarryHoyle@ku.edu.

Joachim Wackerow is a metadata expert at GESIS - Leibniz Institute for the Social Sciences and can be reached at joachim.wackerow@gesis.org.