Tabular Data File Ingest

Supported File Formats

Tabular Data ingest supports the following file formats: (see the sections below for more information on each of the supported formats)

File format	Versions supported
SPSS (POR and SAV formats)	7 to 22
STATA	4 to 13
R	up to 3
Excel	XLSX only (XLS is NOT supported)
CSV (comma-separated values)	(limited support)

Tabular Data, Representation, Storage and Ingest

This section explains the basics. Of how tabular data is handled in the application. And what happens during the ingest process, as the files uploaded by the user are processed and converted into the archival format in the Dataverse application.

What Happens During this “Ingest”?

The goal of our ingest process is to extract the data content from the user’s files and archive it in an application-neutral, easily-readable format. What does this mean? - Commercial applications such as SPSS and Stata use their own, proprietary formats to encode their files. Some companies publish the specifications of their formats (Thank you Stata - much appreciated!), some don’t (SPSS - yes, we are still frowning at you here at the Dataverse Project). Either way, reading these specially-formatted files requires some extra knowledge or special software. For these reasons they are not considered idea for the purposes of archival preservation. Dataverse stores the raw data content extracted from such files in plain text, TAB-delimited files. The metadata information that describes this content is stored separately,

in a relational database, so that it can be accessed efficiently by the application. For the purposes of archival preservation it can be exported, in plain text XML files, using a standardized, open DDI Codebook format. (more info below)

Tabular Data and Metadata

Data vs. Metadata

A simple example is a numeric data column in a user's Stata file that contains 0s and 1s. These numeric values will be extracted and stored in a TAB-delimited file. By themselves, if you don't know what these values represent, these 1s and 0s are not meaningful data. So the Stata file has some additional information that describes this data vector: it represents the observation values of a *variable* with the *name* "party", with a descriptive *label* "Party Affiliation"; and the 2 numeric values have *categorical labels* of "Democrat" for 0 and "Republican" for 1. This extra information that adds value to the data is *metadata*.

Tabular Metadata in Dataverse

The structure of the metadata defining tabular data variables used in Dataverse was originally based on the DDI Codebook format.

[TODO: a brief explanation of the DataVariable and related objects? A link to a more technical documentation writeup in the developers guide?]

SPSS

SPSS data files (POR and SAV formats).

Supported Versions

Dataverse supports reading of all SPSS versions 7 to 22. But please see the "Limitations" section.

Limitations

SPSS does not openly publish the specifications of their proprietary file formats. Our ability to read and parse their files is based on some documentation online from unofficial sources, and some reverse engineering. Because of that we cannot, unfortunately, guarantee to be able to process *any* SPSS file uploaded.

However, we've been improving this process for a few years by now, and it should be quite robust in the current version of Dataverse. Thus your chances of success - uploading an SPSS files and having it turned into a fully functional tabular data table in the Dataverse - should be reasonably good.

If you are having a problem with a particular SPSS file, please contact our support and we'll see if it's something we could further improve in our SPSS ingest driver.

SPSS Control Cards - not supported

In the past, there have been attempts to support SPSS “control cards”, in addition to the .SAV and .POR files; both in the early VDC software, and in some versions of DVN. A “control card” is a plain text file with a set of SPSS commands that describe the raw data, provided in a separate, fixed-width-columns or comma-separated-values file. For various reasons, it has never been very successful. We weren’t seeing too much demand for this ingest feature, so it was dropped from Dataverse 4.0.

Please contact us if you have any questions and/or strong feelings on this issue, or if you have serious amounts of data exclusively available in this format that you would like to ingest under Dataverse.

Support for Language Encodings in SPSS

Historically, there was no support for specifying a particular language/code page encoding for the data stored in an SPSS file. Meaning, text values in none-ASCII encodings, or non-Latin characters could be entered and stored, but there was no setting to unambiguously specify what language, or what character set it was. By default, Dataverse will try to interpret binary characters as UTF8. If that’s not working - for example, if the descriptive labels and/or categorical values ingest as garbage - and if you know happen to know what encoding was used in the original file, you can now specify it in the Ingest Options. For example, if you know that the text in your SAV file is in Mandarin, and is encoded using the GB2312, specify it as follows:

Upload your file, in the “Edit Files” tab of the Dataset page. Once the file is recognized as SPSS/save, and *before* you click Save, go into the “Advanced Ingest Options”, and select “Simplified Chinese, GB2312” in the nested menu under “Language Encoding” -> “East Asian”.

Stata

Of all the third party statistical software providers, Stata does the best job at documenting the internal format of their files, by far. And at making that documentation freely and easily available to developers (yes, we are looking at you, SPSS). Because of that, Stata is the best supported format for tabular data ingest.

New in Dataverse 4.0: support for Stata v.13 has been added.

R Data Format

Support for R (.RData) files has been introduced in DVN 3.5.

Overview.

R has been increasingly popular in the research/academic community, owing to the fact that it is free and open-source (unlike SPSS and STATA). Consequently, there is an increasing amount of data available exclusively in R format.

Data Formatting Requirements.

The data must be formatted as an R dataframe (data.frame()). If an .RData file contains multiple dataframes, only the 1st one will be ingested (this may change in the future).

Data Types, compared to other supported formats (Stat, SPSS)

Integers, Doubles, Character strings

The handling of these types is intuitive and straightforward. The resulting tab file columns, summary statistics and UNF signatures should be identical to those produced by ingesting the same vectors from SPSS and Stata.

Things that are unique to R:

R explicitly supports Missing Values for all of the types above; Missing Values encoded in R vectors will be recognized and preserved in TAB files, counted in the generated summary statistics and data analysis. Please note however that the Dataverse notation for a missing value, as stored in a TAB file, is an empty string, an not “NA” as in R.

In addition to Missing Values, R recognizes “Not a Value” (NaN) and positive and negative infinity for floating point variables. These are now properly supported by the DVN.

Also note, that unlike Stata, that does recognize “float” and “double” as distinct data types, all floating point values in R are in fact doubles.

R Factors

These are ingested as “Categorical Values” in the DVN.

One thing to keep in mind: in both Stata and SPSS, the actual value of a categorical variable can be both character and numeric. In R, all factor values are strings, even if they are string representations of numbers. So the values of the resulting categoricals in the DVN will always be of string type too.

Another thing to note is that R factors have no builtin support for SPSS or STATA-like descriptive labels. This is in fact potentially confusing, as they also use the word “label”, in R parlance. However, in the context of a factor in R, it still refers to the “payload”, or the data content of its value. For example, if you create a factor with the “labels” of *democrat*, *republican* and *undecided*, these strings become the actual values of the resulting vector. Once ingested in the Dataverse, these values will be stored in the tab-delimited file. The Dataverse DataVariable object representing the vector will be of type “Character” and have 3 VariableCategory objects with the *democrat*, etc. for **both** the CategoryValue and CategoryLabel. (In one of the future releases, we are planning to make it possible for the user to edit the CategoryLabel, using it for its intended purpose - as a descriptive, human-readable text text note).

To properly handle R vectors that are *ordered factors* Dataverse (starting with DVN 3.6) supports the concept of an “Ordered Categorical” - a categorical value where an explicit order is assigned to the list of value labels.

Boolean values

R Boolean (logical) values are supported.

Limitations of R, as compared to SPSS and STATA.

Most noticeably, R lacks a standard mechanism for defining descriptive labels for the data frame variables. In the DVN, similarly to both Stata and SPSS, variables have distinct names and labels; with the latter reserved for longer, descriptive text. With variables ingested from R data frames the variable name will be used for both the "name" and the "label".

Optional R packages exist for providing descriptive variable labels; in one of the future versions support may be added for such a mechanism. It would of course work only for R files that were created with such optional packages.

Similarly, R categorical values (factors) lack descriptive labels too. **Note:** This is potentially confusing, since R factors do actually have "labels". This is a matter of terminology - an R factor's label is in fact the same thing as the "value" of a categorical variable in SPSS or Stata and DVN; it contains the actual meaningful data for the given observation. It is NOT a field reserved for explanatory, human-readable text, such as the case with the SPSS/Stata "label".

Ingesting an R factor with the level labels "MALE" and "FEMALE" will produce a categorical variable with "MALE" and "FEMALE" in the values and labels both.

Time values in R

This warrants a dedicated section of its own, because of some unique ways in which time values are handled in R.

R makes an effort to treat a time value as a real time instance. This is in contrast with either SPSS or Stata, where time value representations such as "Sep-23-2013 14:57:21" are allowed; note that in the absence of an explicitly defined time zone, this value cannot be mapped to an exact point in real time. R handles times in the "Unix-style" way: the value is converted to the "seconds-since-the-Epoch" Greenwich time (GMT or UTC) and the resulting numeric value is stored in the data file; time zone adjustments are made in real time as needed.

Things still get ambiguous and confusing when R **displays** this time value: unless the time zone was explicitly defined, R will adjust the value to the current time zone. The resulting behavior is often counter-intuitive: if you create a time value, for example:

```
timevalue<-as.POSIXct("03/19/2013 12:57:00", format = "%m/%d/%Y %H:%M:%OS");
```

on a computer configured for the San Francisco time zone, the value will be differently displayed on computers in different time zones; for example, as "12:57 PST" while still on the West Coast, but as "15:57 EST" in Boston.

If it is important that the values are always displayed the same way, regardless of the current time zones, it is recommended that the time zone is explicitly defined. For example:

```
attr(timevalue,"tzone")<-"PST"
```

or

```
timevalue<-as.POSIXct("03/19/2013 12:57:00", format = "%m/%d/%Y %H:%M:%OS", tz="PST");
```

Now the value will always be displayed as "15:57 PST", regardless of the time zone that is current for the OS ... **BUT ONLY** if the OS where R is installed actually understands the time zone "PST", which is not by any means guaranteed! Otherwise, it will **quietly adjust** the stored GMT value to **the current time zone**, yet it will still display it with the "PST" tag attached!** One way to rephrase this is that R does a fairly decent job **storing** time values in a non-ambiguous, platform-independent manner - but gives you no guarantee that the values will be displayed in any way that is predictable or intuitive.

In practical terms, it is recommended to use the long/descriptive forms of time zones, as they are more likely to be properly recognized on most computers. For example, “Japan” instead of “JST”. Another possible solution is to explicitly use GMT or UTC (since it is very likely to be properly recognized on any system), or the “UTC+<OFFSET>” notation. Still, none of the above**guarantees** proper, non-ambiguous handling of time values in R data sets. The fact that R **quietly** modifies time values when it doesn’t recognize the supplied timezone attribute, yet still appends it to the **changed** time value does make it quite difficult. (These issues are discussed in depth on R-related forums, and no attempt is made to summarize it all in any depth here; this is just to made you aware of this being a potentially complex issue!)

An important thing to keep in mind, in connection with the DVN ingest of R files, is that it will **reject** an R data file with any time values that have time zones that we can’t recognize. This is done in order to avoid (some) of the potential issues outlined above.

It is also recommended that any vectors containing time values ingested into the DVN are reviewed, and the resulting entries in the TAB files are compared against the original values in the R data frame, to make sure they have been ingested as expected.

Another **potential issue** here is the **UNF**. The way the UNF algorithm works, the same date/time values with and without the timezone (e.g. “12:45” vs. “12:45 EST”) **produce different UNFs**. Considering that time values in Stata/SPSS do not have time zones, but ALL time values in R do (yes, they all do - if the timezone wasn’t defined explicitly, it implicitly becomes a time value in the “UTC” zone!), this means that it is **impossible** to have 2 time value vectors, in Stata/SPSS and R, that produce the same UNF.

A pro tip: if it is important to produce SPSS/Stata and R versions of the same data set that result in the same UNF when ingested, you may define the time variables as **strings** in the R data frame, and use the “YYYY-MM-DD HH:mm:ss” formatting notation. This is the formatting used by the UNF algorithm to normalize time values, so doing the above will result in the same UNF as the vector of the same time values in Stata.

Note: date values (dates only, without time) should be handled the exact same way as those in SPSS and Stata, and should produce the same UNFs.

Excel

Excel files (**New** in Dataverse 4.0!)

Note: only the “new”, XLSX Excel files are supported. We are not planning to add support for the old-style, binary XLS files.

CSV

Ingest of Comma-Separated Values files as tabular data.

Dataverse will make an attempt to turn CSV files uploaded by the user into tabular data.

Main formatting requirements:

The first line must contan a comma-separated list of the variable names;

All the lines that follow must contain the same number of comma-separated values as the first, variable name line.

Limitations:

Except for the variable names supplied in the top line, very little information describing the data can be obtained from a CSV file. We strongly recommend using one of the supported rich files formats (Stata, SPSS and R) to provide more descriptive metadata (informatinve lables, categorical values and labels, and more) that cannot be encoded in a CSV file.

The application will however make an attempt to recognize numeric, string and date/time values in CSV files.

Tab-delimited Data Files:

Tab-delimited files could be ingested by replacing the TABs with commas.