CONSIRT

Cross-National Studies:
Interdisciplinary Research and Training Program

Working Papers Series

CONSIRT Labs: Methodology of Survey Data Harmonization

Working with Big Data: Experiences with the Cross-national Survey Data Harmonization Project

Przemek Powałko, Polish Academy of Sciences

Original: October 2014
Updated: January 2015

Working with Big Data: Experiences with the Cross-national Survey Data Harmonization Project

Przemek Powałko, Polish Academy of Sciences

Abstract

This paper discusses experiences in managing data from the Harmonization Project, a large scale data base that is a synthesis of relatively smaller databases that consists of international survey projects. The main questions addressed in this article are: (1) how to manage such a variety of the data files? (2) how to process the data sets in order to make them comparable from survey to survey? (3) how to browse the data in a convenient way? and (4) how to produce a single data file (a master file as we call it) that would contain all data needed for statistical analysis? We advocate exploiting free and open-source software and making an extensive use of programming languages. The skills required to develop one's own environment might be perceived as a drawback, but we think that advantages mentioned so far outweigh the costs.

Acknowledgements

This work is a part of the project "Democratic Values and Protest Behavior: Data Harmonization, Measurement Comparability, and Multi-Level Modeling in Cross-National Perspective", and is funded by the National Science Centre, Poland, under the grant number 2012/06/M/HS6/00322.

Introduction

We have a growing wealth of social survey data that could answer many questions but the data are often not comparable or well documented. Regional specialization of international surveys - such as European Social Survey, Latino Barometer, Asia Europe Survey, among others - severely hinders research pertaining to world-relevant issues.

The Harmonization Project addresses challenges of cross-national analyses of survey data by creating online accessible, comparable measurements of social values, action and demographics with global coverage. For our purposes we use data from 22 cross-national survey projects that cover a total of 142 countries or territories over a time span of almost 50 years (1966-2013). The large amount of data as well as a variety of data formats deployed in so many sources urged us to develop in-house tools for extracting, transforming and loading data into a common database, automating as many procedures as possible. We have created an environment based on freeware and open-source software that constitutes an alternative to statistical packages typically used for such purposes in social science research. The platform allows us for storing and managing data in a single place, and - what is crucial - enables us to easily manipulate data in order to prepare a harmonized data set necessary for further investigations. The paper outlines the process of setting up this programming and database environment, describes technical issues of processing data, and discusses pros and cons of the adopted approach.

Some terms used in the text may seem quite technical (though we use them in a very simplified way), so we highlight them in text and explain them in the section ‘Glossary’ at the end of the article.

Source data

The selected survey projects meet the following criteria: (1) they are non-commercial (mainly academic), (2) designed as cross-national (and preferably multi-wave) enterprises; (3) the samples are intended as representative of the entire adult population of a given country or territory; (4) they contain questions of substantive interest to the project, that is about political attitudes and protest behaviors; (5) the data is freely available in the public domain; and (6) sufficient documentation (study description, codebook and/or questionnaire) is provided in English. For the full list of survey projects refer to Table 1.

-- Table 1 about here --

The sheer number of the sources of data has introduced the first challenge to deal with: the data files' format. We have decided to work with SPSS system (or portable) files because they appear to be the most frequently used in the social science community. In a few cases when SPSS files were not available we had to convert existing files to a format of choice with appropriate tools. The files prepared in such a way were then automatically loaded into the database.

The published data are not given once and for ever and it may happen that errors are found in the original files. Occasionally, we had to implement corrections following announced alerts and errata. Changes were made in three ways: (1) if a new version of the source file were introduced, we were replacing the file in our repository; (2) if an erratum was published in the form of an SPSS patch, we were applying it to the original SPSS file; finally, (3) if instructions were given in other ways - usually as explanatory notes - we were modifying the data in the database itself thus not touching the original files. We kept on updating the data until the end of first quarter of 2014.

The universe of our research consists of approximately 2.3 million cases stored in 81 data files. A single data file may contain from one country in one wave to many countries in many waves so that the number of waves may differ (and it actually does) from the number of data files, and equals 89. The number of data sets, that is national samples in all surveys carried out in all ways and in all projects, equals 1726.

A few questions immediately arise: (1) how to manage such a variety of the data files? (2) how to process the data sets in order to make them comparable from survey to survey? (3) how to browse the data in a convenient way? and (4) how to produce a single data file (a master file as we call it) that would contain all data needed for statistical analysis? These questions are discussed in detail in the following sections.

Concept

Let us start from a simple observation: all data we are interested in is textual. This means that all information, be it expressed in words or numbers, is made of strings of characters and as such can be processed with text processing tools. However, statistical software usually stores data in a **binary format** which is not readable by text editors and cannot be easily manipulated outside **proprietary software**. Moreover, the amount of raw data in our project, totaling 2.3 GB, as well as the need for selecting and extracting ad hoc subsets of the whole database, poses a big challenge even for most ambitious installations of statistical packages and calls for developing alternative ways of facing the problem.

All this encouraged us to develop a custom solution founded on the concept of a relational database.

A **relational database** stores data in structured objects that we call tables. Tables are organized in rows and columns and in a graphical representation they may resemble a matrix of a spreadsheet application. A relational database allows for easy, fast and - at the same time - sophisticated access to data using a high-level language, and enables mechanisms of consistency, integrity and coherence of data. There are many relational database management systems available (for example by Oracle, Microsoft, IBM, and other vendors) and the right choice might be a difficult task, but from the point of view of an average end-user the most important feature of a relational database is its interface, the language.

SQL, Structured Query Language, is the most widely used database language implementing a relational model. It is declarative, meaning it describes *what* should be done, in contrast to procedural languages, which describe *how* things should be done. This approach makes communication with a database very simple. All operations both on data and objects containing data can be performed through SQL statements.

One of the characteristic and user friendly features of SQL is its high readability and similarity to basic English. For example, the following statement:

```
SELECT country_code

FROM country_gdp

WHERE y2012 > 30000

ORDER BY country_code
```

can be read as: select values from the column country_code from the table country_gdp for which the value of the column y2012 is greater than 30000, then order the result set by the country's code. The example is very simple but the capabilities of SQL are much richer and make it a flexible and powerful tool for handling data.

One issue remains yet unsolved: how to extract data from binary data files and load it into a database? The answer is: make it textual, first.

Environment

Before we describe the procedure of processing the data, in this section we will focus on the structure of the development environment, trying to name and shortly characterize all elements involved.

In the world of computer sciences it is widely known that text processing goes along with **Unix**, a family of operating systems. This is because Unix stores system data as text and thus provides an impressive collection of text processing tools. Although we use Microsoft Windows as an operating system, we have enriched it by installation of Cygwin, an effective Unix-like environment that provides native integration of Windows resources with software tools of a Unix provenance. Thanks to this, aside from developing the data processing platform, we can also run tests in proprietary statistical packages in a single workstation.

Cygwin is equipped with command-line interpreters (so-called shells providing interface to the operating system), tools for text processing, and programming languages, that is a means we extensively use in the development: bash, sed, awk, perl - to name a few.

To read SPSS data files and converting them to text files we use PSPP, a free replacement for SPSS. Thanks to its command-line interface we can automate the process by embedding PSPP commands in Cygwin based scripts.

The text files are subsequently loaded into the database. The choice of MySQL, a free and open-source database, was dictated by the large number of variables in some data sets. These variables are to be represented by columns in database tables, which places high demands on a candidate database. Another key factor is the existence of a convenient interface to the database needed for programming the data processing. MySQL meets both requirements allowing for up to 4096 columns in a table and seamlessly integrating with the command-line interface of a Cygwin terminal.

Another advantage of MySQL is the variety of storage engines that can be used for different purposes. We have chosen the ARCHIVE engine, which can significantly compress data in tables therefore making them much smaller and making reads from disk much faster, which in turn translates in much smaller demands on internal memory and leads to much faster data processing.

For writing queries and browsing data in the database we use HeidiSQL, a free SQL editor. For writing programs in scripting languages we use Notepad++, a free text editor. Together with Cygwin, PSPP and MySQL, all these tools are free and/or **open-source software**, and this is exactly what we meant to achieve: to build a dedicated system generating no unnecessary costs.

Data preprocessing

Knowing the structure of the environment we are now ready to learn the flow of the data processing. Let us assume that an SPSS source file has been saved to a working directory in our workstation. The following steps are fully automated and executed by Cygwin scripts:

(1) Extract data from the SPSS source file and export it to a text file. This is done through a PSPP syntax file. The resulting text file has a standardized format of a comma separated values (CSV) file, a very convenient and popular way of storing tabular data.

(2) Prepare the SQL syntax file for table definition according to the values in the CSV file. Scan and analyze all values in all fields of the CSV file and create a corresponding SQL syntax file.

(3) Connect to the database and execute the SQL syntax file to create a table for storing the source data. After this step the table is ready for filling in with data.

(4) Prepare the CSV file for import to the table so that all values comply to the database format. In particular, null values should be coded properly.

(5) Connect to the database and load the CSV file data into the table.

(6) Extract metadata from the SPSS source file and store it in a text file, which we call a dictionary (DIC) file. Metadata is the information about variables: their names, labels, acceptable values, missing values, and data formats.

(7) Prepare the SQL syntax file for dictionary table definition according to the values in the DIC file.

(8) Connect to the database and execute the SQL syntax file to create a dictionary table for storing the metadata. After this step, the dictionary table is ready for filling in with (meta)data.

(9) Connect to the database and load the DIC file data into the dictionary table.

The process is completed. In the workstation that we use for tests, an SPSS data file containing 50000 cases and 300 variables and weighing 45MB takes roughly 40 seconds to get processed and the corresponding tables weigh about 2MB. The automation of the above procedure allows for fast and convenient way of adding new data files and populating the tables.

The data is stored in the database and is ready for use.

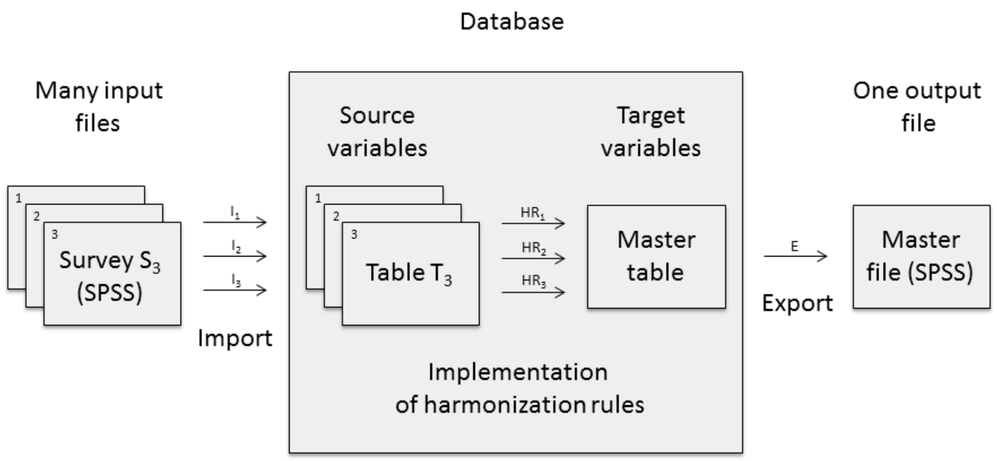

Figure 1. Overview of the harmonization process.

Data processing

According to the procedure described above, we have created 81 data tables and 81 dictionary tables, one for each data file used in the project. The data tables contain source values, usually coded in numbers. At any time they may be recoded to descriptive labels through SQL operations of joining with dictionary tables. For example, the following syntax matches country codes from one table with their full names from another table which are then displayed in the resulting set:

```
SELECT country_name

FROM country_gdp

JOIN country_codes USING (country_code)

WHERE y2012 > 30000

ORDER BY country_name
```

The JOIN ... USING clause means that two tables (country_gdp and country_codes) are related and both have corresponding columns called country_code. This relationship gives access to the content of both tables and allows the query to see the country's name from country_codes table.

The process of harmonization itself, that is discussion of harmonization rules that we have created during an extensive analysis of data, is out of the scope of this article. We want to stress, however, that technically speaking, the implementation of harmonization rules can be easily done with a sequence of simple SELECT statements backed (when necessary) with JOIN clauses. The results of such atomic queries are saved in the master table, a common place for harmonized data, with a series of SQL INSERT statements of the following form:

```
INSERT INTO master

SELECT ...

FROM ...
```

Preparation of these statements is a manual work, and every time the harmonization rules change, this step has to be rerun and the data in the master table has to be reloaded. However, execution of SQL statements is fast and straightforward, and at the end of the process we get a long awaited effect: all data is gathered in a single table.

The structure of the master table may depend on end-user’s expectations and needs. We plan to have at least three types of variables (columns): (1) source variables, that is raw data from the source data tables preserved for reference; (2) target variables resulting from the application of the harmonization rules on the source variables; and (3) control variables which provide additional information about the source variables, for example the context or specific meaning that the source variables were used in.

A great advantage of having data in a relational database is that all data is directly available for browsing and querying without a delay or any additional resources usually associated with opening and reading SPSS files. Executing an SQL query leads to a series of small data gets from relevant tables so that, at all times, only a fraction of data is read from disk and loaded into the internal memory, making the whole process fast and scalable.

Figure 1 outlines the flow of data in the harmonization process. We start with a number of SPSS data files, extract data and save it to text files, which afterwards are imported to the database and stored in corresponding tables. Then, we implement the harmonization rules in the form of SQL statements and save query results in the master table. At this point the data is ready for browsing and preliminary statistical analysis.

In practice, the end-user does not need to be aware of all technical details described above: at the final stage we plan to export the master table back to a text file and to convert it to the master file of SPSS or any other required format. The master file will be available online[1].

Summary

In the article we discussed the structure of the platform for processing data in the data harmonization project. The essence of this solution lies in exploiting free and open-source software and making an extensive use of programming languages. The skills required to develop one's own environment might be perceived as a drawback, but we think that advantages mentioned so far outweigh the costs. An additional value of storing the data in a relational database is that SQL gives opportunity to perform basic statistical analyses of data without the need of leaving the database. Of course, the capabilities of SQL are limited and rather modest if compared to dedicated statistical software, but there are statistical and data mining extensions to databases, which are free or reasonably priced for academic purposes. We plan to experiment with some of them in future.

[1] For more information, see http://dataharmonization.org

Glossary

Some terms used in the article are quite technical and a reader from a purely social sciences background may be not acquainted with them. In this section we briefly explain their meanings making use of short excerpts from Wikipedia:

A **binary file** is a computer file that is not a text file; it may contain any type of data, encoded in binary form for computer storage and processing purposes.

Open-source software is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change and distribute the software to anyone and for any purpose. Open-source software is very often developed in a public, collaborative manner.

Proprietary software or closed source software is computer software licensed under exclusive legal right of the copyright holder with the intent that the licensee is given the right to use the software only under certain conditions, and restricted from other uses, such as modification, sharing, studying, redistribution, or reverse engineering.

A **relational database** is a database that stores information about both the data and how it is related. Each database is a collection of related tables. Each table is a physical representation of an entity or object that is in a tabular format consisting of columns and rows. Columns are the fields of a record or the attributes of an entity. The rows contain the values or data instances; these are also called records.

SQL, Structured Query Language, is a special-purpose programming language designed for managing data held in a relational database management system.

UNIX is a family of operating systems that are known for a use of plain text for storing data thus providing a great number of sophisticated tools for text processing.

References

Here is a list of programs mentioned in the article.

Cygwin: http://www.cygwin.com

MySQL: http://www.mysql.com

PSPP: http://www.gnu.org/software/pspp

HeidiSQL: http://www.heidisql.com

Notepad++: http://notepad-plus-plus.org

SPSS: http://www.ibm.com/software/analytics/spss

Table 1. The list of investigated survey projects

abbrev.	survey project	time span	#waves	#files	#data sets	#cases
AFB	Afrobarometer	1999-2009	4	4	66	98942
AMB	Americas Barometer	2004-2012	5	1	92	151341
ARB	Arab Barometer	2006-2011	2	2	16	19684
ASB	Asian Barometer	2001-2011	3	3	30	43691
ASES	Asia Europe Survey	2000	1	1	18	18253
CB	Caucasus Barometer	2009-2012	4	4	12	24621
CDCEE	Consolidation of Democracy in Central and Eastern Europe	1990-2001	2	1	27	28926
CNEP	Comparative National Elections Project	2004-2006	1	8	9	13978
EB	Eurobarometer	1983-2012	7	7	152	138753
EQLS	European Quality of Life Survey	2003-2012	3	1	93	105527
ESS	European Social Survey	2002-2013	6	2	146	281496
EVS/WVS	European Values Study / World Values Survey	1981-2009	9	1	312	423084
ISJP	International Social Justice Project	1991-1996	2	1	21	25805
ISSP	International Social Survey Programme	1985-2013	13	13	363	493243
LB	Latinobarometro	1995-2010	15	15	260	294965
LITS	Life in Transition Survey	2006-2010	2	2	64	67866
NBB	New Baltic Barometer	1993-2004	6	1	18	21601
PA2	Political Action II	1979-1981	1	1	6	6682
PA8NS	Political Action - An Eight Nation Study	1973-1976	1	1	8	12588
PPE7N	Political Participation and Equality in Seven Nations	1966-1971	1	7	7	16522
VPCPCE	Values and Political Change in Postcommunist Europe	1993	1	5	6	5769
		1966-2013	**89**	**81**	**1726**	**2293337**