DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing
March 15-17, Vienna, Austria
http://www.ci.tuwien.ac.at/Conferences/DSC-2001
K. Hornik & F. Leisch (eds.) *ISSN 1609-395X*

Distributing data to different platforms using XML

Mikko Virtanen*

Abstract

The ATBC Cancer prevention study is large and complex cohort study with long history. The path from the raw data to publications is described. Easiest way would be to standardize the used statistical package and provide methods to access the data for this package only. In practice the skill level of those doing the analyses vary and several packages must be supported. Extra care must be put on ensuring that the versions of the data are same. The XML offers structured way of combining the definition, documentation and the revision history of each data item. These definitions are then transformed using the XSLT to code, meta data (help files) and documentation (HTML and PDF). The current version defines the data structures in a rather restricted way, but the extension of the same idea to analysis results will be discussed briefly.

1 Background: the ATBC study

The ATBC cancer prevention study is a 2×2 factorial, randomized, placebo controlled double blinded cancer prevention study. The active substances were α-tocoferol (AT) and β-carotene (BC). The primary aim was to prevent lung cancer in a high risk population of middle aged male smokers.

For this purpose, 29133 men were recruited from the southern part of Finland between December 1984 and July 1988. The supplementation lasted until the end of April 1993. During the intervention the men visited their local study centers three

*National Public Health Institute (KTL), Finland. E-mail `Mikko.Virtanenktl.fi`

times a year. In addition to questionnaires regarding health behavior (including the diet), some other specimen were collected, namely toenails and serum.

The endpoints were collected from the population registries, mainly from the Population Registry Centre (deaths) and The Finnish Cancer Registry. This inactive followup has been continuing after the intervention.

Thus the amount of *data is large*, lot of subjects with lot of followup and lot of data from each followup point. Also, the *data are complex*, the endpoint validation requires obtaining medical records, death certificates and pathological consultations. And perhaps more importantly, the *amount of data is increasing*. Not only is the follow up continuing, but also more and more information gets extracted from, for instance, the dietary data and the serum samples.

The main results are published in [2]

The data has been stored into several databases. To ease the use of it in the analyses an special version called Analysis Data Base (ADB) was created[1]. The idea was to create a single copy of well defined variables in a format that can be accessed using different packages on different platforms. The format chosen at that time (1994) was NetCDF [2]. The creation and especially documentation of these files was found laboursome and thus a new system resign was required. This presentation describes this design.

2 Problem: Uniqueness of the variables

In epidemiological analyses one constantly encounters concepts such as death and lung cancer. Both are relatively well defined and understood by many. For statistical analyses these are not, however, unique enough.

The information flows to the study database slowly, piece by piece. Decisive information about a cancer might only be available several years after the original diagnosis. Similar problems are present with the diet data: the collection of the composition data takes a lot of time and patience. This slow data flow is the main source of non-uniqueness.

The numbers of lung cancer cases in the first published paper [2] differ from the second [1] (876 vs 894, about 860 common cases) because the two years in between have brought additional information about the cancers. Some suspect cases have turned out to be originated from other sites, the others were not fully diagnosed at the first point of observation. Neither of the definitions are wrong, they both represent our best knowledge at their time. We would not use the first one in new analyses, actually we might not want to use the latter either, as we now have several years of follow-up more.

[1] The original system and the main ideas in the new came from Jaason Haapakoski

[2] `http://www.unidata.ucar.edu/packages/netcdf/`

3 Problem: Heterogenous environment

Additional problems are caused by the number of packages used in analyses (table 1). We would certainly like to have all data available on all platforms without any loss of information. This is difficult because of different ways to provide different functionality.

package	Windows	Linux	Open VMS	Tru64Unix
R	(in future)	**1.X.X**	—	1.X.X
S-Plus	2000	—	—	**3.4**, 5.1
SPSS	**9.X**	—	—	—
SAS	—	—	**6.13**, 8.1	—

Table 1: Statistical packages used in the ATBC study since 1995. Columns are platforms, numbers in cells refer to versions. The main package used on each platform is in **bold**

4 Universal relations

The traditional approach to data management in statistical packages has been based on flat files. These files have rows as observations and columns as variables. For many packages this is the only available format.

In our study, the data are collected into *universal relations*. A universal relation is a group of objects that belong a virtual, usually very wide relation, but which needs not be implemented as one relation. This design comes from the RDBMS world, where the database tables are always in row-specific form making wide tables with lots of attributes (variables) inefficient. In practice, only a fraction of the data are used, so column specific form is more useful. The increase of computing power have made URs a bit obsolete, but in out context they are useful.

The S/R data frames implement universal relations. Their efficiency is limited when the widths become large. In our study, the width of the largest UR is potentially several thousand. This is far too much for the current implementations. Thus on S and R we use on separate vectors (or other objects) that are tied together using a naming scheme. Object `base0.schol` is the serum cholesterol at baseline for the study members, version 0. Using vectors instead of data frames gives the additional benefit that the relation can be extended during the analyses.

The variables are collected into groups that share the same source. The source might be a database table. On S each variable is a persistent object in a library, on R they will be implemented using the `data()` mechanism.

Most other packages do not support UR as well as S and R. To solve the problem of wide matrices, there must be a separate system that creates the needed datasets on the fly from the separate blocks. The naming is sometimes more difficult because of limitations in the lengths of the variable names. The newer versions of SAS have finally broken the 8 char barrier making names like `base0_schol` possible.

5 Distributing data

At first we looked at two possible designs: push and pull. With push the data is transformed to the native format. There are some commercial products for this, such as the DBMS/Copy. With pull the data are stored in a format accessible by all packages. The main example of this is ODBC.

Most systems have some kind of direct connections to databases such as the ODBC. The data could be formed as tables, views and stored procedures and then accessed directly without intermediate format. There are two problems regarding this approach.The ODBC and the like do not have a standard way to move meta-data. For instance, an variable that would be modeled as a factor in S/R would not have proper labels without additional programming. Also, the documentation needs separate programming.

The StatDataML[3] provides another interesting new format. The support for StatDataML is still weak on commercial systems, but this will most probably change in the future. This format includes the whole data into the definitions which in our case would result in very large files.

Our approach is to try to combine the best of push and pull. We push the code to pull the data. Thus the definition files can use ODBC but supplement it. The files are remain small and flexible. Using special transformations called off-lining the data can be pushed to platforms that are unable to access the original source. This is analogous to sending URLs in stead of documents.

6 The basic design

We have defined three goals:

1. Code and documentation must go together
2. Data must be distributable to a wide group of packages
3. Variables must have explicit naming scheme that includes reference to the UR and the version

To achieve this we have designed a XML (eXtensible Mark-up Language [4]) DTD (Document Type Definition) called ADB DTD. It defines a document format that can be used to define URs in two parts. Subset definitions define the row dimension of the UR and variable definitions the column dimension.

The code that distributes the data can then be created from these definitions using XSLT (eXtensible Style sheet Language Transformations [5]).

7 The ADB DTD

Skeletal DTD of the ADB file format is in the table 2.

[3] `http://cran.r-project.org/src/contrib/Devel/StatDataML_0.3-1.tar.gz`
[4] `http://www.w3.org/TR/REC-xml/`
[5] `http://www.w3.org/Style/XSL/`

```
<!ELEMENT adbdata
 (summary, datasource, subset, variables, offline?)>
<!ATTLIST adbdata
  name       CDATA #REQUIRED
  generation CDATA #REQUIRED >
 <!ELEMENT summary
  (label, author, version, description, history, keyword*)>
  <!-- definitions for children -->
 <!ELEMENT datasource
  (sql|cards|netcdf|file)>
  <!-- definitions for children -->
 <!ELEMENT subset
  ((target|prefix),data)>
  <!-- definitions for children -->
 <!ELEMENT variables (variable)*>
  <!ELEMENT variable
   (label, description*, expression, class)>
   <!-- definitions for children -->
   <!ELEMENT class (class-attribute)*>
   <!ATTLIST class name CDATA #REQUIRED>
    <!ELEMENT class-attribute (#PCDATA)>
    <!ATTLIST class-attribute name CDATA #REQUIRED>
 <!ELEMENT offline (file)>
```

Table 2: Skeleton of the ADB DTD

- The root element, `adbdata` has attributes `name` and `generation` that identify the file. The `generation` is also used in naming of the variables. Actual prefix comes from the target subset.

  ```
  <?xml version="1.0"?>
  <!DOCTYPE adbdata SYSTEM "adb.dtd">
  <adbdata name="cancer" generation="0">
  ```

- The element `summary` includes all the textual documentation including short and long descriptions and version and history elements (for use with RCS) etc.

  ```
  <summary>
  <label>Cancers as used in the NEJM94 paper.</label>
  <author>Mikko Virtanen</author>
  <version>$Id$</version>
  <description>...</description>
  <history>$Log$</history>
  </summary>
  ```

- The actual data is defined in the element `datasource`. There are a lot of options for this, including SQL and flat file.

```
<datasource>
<sql><sql-expr>
select id,lungday,deathday,entryday
       from cancers_nejm94
       order by id
</sql-expr></sql>
</datasource>
```

- There are (almost) always two subsets involved with each file. The data subset defines the subjects etc in the data-source, and the target defines the same in the end result. The local data source is then normalized by outer join to the defining subset.

```
<subset>
<target name="setti.base"/>
<data>
<sub-id  type="name">id</sub-id> <!-- subject identifier -->
<sub-vst type="null"></sub-vst>  <!-- visit number: any  -->
<sub-dn  type="null"></sub-dn>   <!-- day number: any    -->
<sub-ag  type="fail"></sub-ag>   <!-- aggregation: fail  -->
</data>
</subset>
```

  Only the subject id is needed, so others are coded as `NULL`. Element `sub-ag` defines the aggregation, `type="fail"` means that it is an error to have duplicate id:s.

  Target definition `setti.base`[6] is actually an partial URI. It refers to a file `setti.base.ss.xml`.

- Variables have their own documentation. The `class` is actually a hook to a function that is called after the variable is otherwise ready. The class attributes, such as the labels for factors can be passed using this mechanism.

```
<variables>
<variable name="lung">
<label>Lung cancer</label>
<description>...</description>
<expression type="name">lungday</expression>
<class name="endpoint">
<class-attribute name="censor">deathday</class-attribute>
<class-attribute name="entry" >entryday</class-attribute>
```

[6] Setti is the name of the study in Finnish

```
<class-attribute name="eof"   >3238</class-attribute>
</class>
</variable>
</variables>
```

This part might require revision, as the class mechanism is missing in SAS and SPSS. Simpler approach is to limited set of predefined classes that have ready counterparts on every package (numeric, character, factor, date)

- Element `offline` (optional) is used to distribute the data to environments that can not access the original data, such as laptops. The original definition is kept for documentation purposes. In practice this means that the data source is evaluated in S or R, an data frame is created and output (using, say, `write.table`) and a relatively standard `offline` tag is embedded.

8 Transformations

The full implementation shall not be discussed here. The source of the transformations can be obtained from the author, allthought the system's general applicability may be weak.

The required transformations are

- XML to HTML to create the backbone of the documentation. A link to each object is stored to a central repository
- XML to R/S. As the resulting code is relatively simple a compatibility API has been designed
- XML to .Rd and .Sd to provide the online documentation for R
- XML to other packages such as SAS
- Off-lining

After these transformations, each set of files is copied to their respective repositories and R and S libraries are be build.

The transformation to R takes three steps:

1. The data source is transformed to code depending on the type of the source. Each type has an API function, such as `adbSQL` for SQL queries. All of the functions return a data frame.
2. The both subset definitions are parsed. If there is element `prefix` we are defining a new UR. Otherwise there must be element `target`, and the datasource is then outer joined to this.
3. As a last step the variables are created. There's another function, `adbVar`, that takes as parameters the data source frame, expression, class name and arguments and some documentation items. As a result all resulting variables

should have at least the attributes `class`, `label` and `subset`. The latter two can be used in reporting and maintaining the traceability of the data thru out the analyses.

For example, the data source on page 6 could be transformed using following template:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output
  method="xt:nxml" xmlns:xt="http://www.jclark.com/xt"/>

<xsl:template match="datasource">
<nxml><data><xsl:apply-templates/></data></nxml>
</xsl:template>

<xsl:template match="sql/sql-expr">
tmp.ds&lt;-adbSQL("<xsl:value-of select="normalize-space(.)"/>")
</xsl:template>

</xsl:stylesheet>
```

This template uses XSL extensions from James Clark's XT[7].
The resulting code after transformation is as simple as

```
tmp.ds<-adbSQL("select id,lungday,lungstatus,deathday,entryday from deaths order by id")
```

Note that the unwanted newlines have been removed by `normalize-space`.

On SAS the steps 2 and 3 can be combined. The use is more difficult. The users must combine separate blocks either by hand or by some helper macros.

9 Further topics

The system is still under work, the full body of the data have not been transformed to it. Increased experience with the system might lead to changes in the design. The variable class issue needs revising, a complete list of needed classes can be made only by going through the data.

The current design uses static batch transformations and distribution. The packages are beginning to support XML input. For instance, the XML package for R[8] could be used to read the definitions and create the variables. This could solve the biggest unresolved problem, security, as R could access the files using authenticated and decrypted methods. Such features are being added to the StatDataML also.

There may be need to document and distribute other kinds of objects also. The basic results from canonical analyses could be used as a basis in further analyses. To allow access from different packages is an enormous challenge. The main task is to

[7] `http://www.jclark.com/xt`
[8] `http://www.Omegahat.org/RSXML`

write XML based model formulae. The differences in generality of model formulas on different packages makes this rather difficult. Otherwise the transformations should be relatively simple.

The advent of web based user interfaces might simplify this issue. If most analyses are done via the web, the choice of the package working in the background is no longer important to the end users and can be made by the maintainers of the systems. Thus the most capable and most suitable system can be used for each purpose.

Acknowledgements

This work is a part of larger documentation project that is a collaboration with Timo Lamminjoki, Heikki Kilpelinen and Jarmo Virtamo. Much of the design work was done by Jaason Haapakoski in 1994, this is just a reimplementation of that system.

References

[1] Demetrius Albanes, Olli P Heinonen, Philip R Taylor, Jarmo Virtamo, Brenda K Edwards, Matti Rautalahti, Anne M Hartman, Juni Palmgren, Laurence S Freedman, Jaason Haapakoski, Michael J Barrett, Pirjo Pietinen, Nea Malila, Eero Tala, K Liippo, E R Salomaa, Joseph A Tangrea, Lyly Teppo, Frederic B Askin, Eero Taskinen, Yener Erozan, P Greenwald, and Jussi K Huttunen. Alpha-tocopherol and beta-carotene supplements and lung cancer incidence in the atbc study: effects of baseline characteristics and study compliance. *Journal of National Cancer Institute*, 88:1560–1570, 1996.

[2] ATBC Cancer Prevention Study Group. The effect of vitamin e and beta-carotene on the incidence of lung cancer and other cancers in male smokers. *New England Journal of Medicine*, 330:1029–35, 1994.