NISS

NISS WebSwap: A Web Service for Data Swapping

Ashish P. Sanil, Shanti Gomatam , Alan F. Karr
and Chunhua "Charlie" Liu

Technical Report Number 126
June, 2002 (revised: February, 2003)

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

NISS WebSwap: A Web Service for Data Swapping

Ashish P. Sanil, Shanti Gomatam, Alan F. Karr and Chunhua "Charlie" Liu[1]
National Institute of Statistical Sciences, Research Triangle Park, NC 27709-4006, USA
{ashish,sgomatam,karr,cliu}@niss.org

Abstract

Web Services are an exciting new form of distributed computing that allow users to invoke remote applications nearly transparently. National Institute of Statistical Sciences (NISS) has recently started hosting NISS Web Services as a service and example to the statistical sciences community. In this paper, we describe and provide usage information for NISS WebSwap, the initial NISS Web Service, which swaps one or more attributes (fields) between user-specified records in a microdata file, uploading the original data file from the user's computer and downloading the file containing the swapped records.

1 Introduction

NISS WebSwap is a Web service (see §3) that swaps one or more attributes (fields) between user-specified records in a microdata file, uploading the original data file from the user's computer and downloading the file containing the swapped records.

Data swapping [7, 8] is a strategy for protecting confidentiality in released microdata records. In the simplest case, values of a single attribute are swapped between randomly selected pairs of records. The purpose of data swapping is to introduce uncertainty into the mind of any data user or intruder as to whether records correspond to real data elements. In the example in Figures 3 and 6, only seven of the ten records shown from the post-swap database are "real." Swapping does not change the marginal distribution of any attribute. Nor does it change either the joint distribution of the unswapped attributes or—if more than one attribute is swapped[2]—the joint distribution of the swapped attributes. It *does*, however, distort joint distributions involving both swapped and unswapped attributes.

Not all attributes are swapped, of course. We call the subset of attributes that are swapped the *swapped attributes* or *swap attributes*. The fraction of records in the microdata that are initially marked to be swapped will be called the *swap rate*, and is denoted by r. We allow only true swaps, which result in different pre- and post-swap values for the records being swapped.

In some situations there may be conditions on pairs of records, defined by attributes other than swap attributes, in order for a pair of records to be feasible swap candidates for each other. For instance, in the microdata presented in Figure 3, we may allow swaps of Age (the first attribute following the identifier) only between those records with the same value of Employer Type (the second attribute), or allow swaps of Age only between those records that have different values of Education (the third attribute). Such attributes whose values define the feasibility of swap candidates are called *constraining attributes*.

When swapping is used for purposes of statistical disclosure limitation (SDL), several choices are necessary: the swap attributes, the swap rate, and constraints (if any). We conceptualize these choices as taking place in a risk-utility framework [2, 4]. In virtually all cases, there is a tradeoff: higher utility implies higher risk. NISS WebSwap is usable, for example, to generate swapped databases under a range of choices,

[1]The authors thank Adrian Dobra for his input. Support for the research was provided by National Science Foundation grant EIA–9876619 to NISS and by the National Center for Education Statistics (NCES).

[2]In our framework, multiple attributes are always swapped simultaneously.

and to compute risk–utility (R–U) frontiers that quantify tradeoffs and inform the selection process [4]. Although risk and utility are not explicitly part of NISS WebSwap, we mention illustrative examples.

A widely held viewpoint in the field of SDL is that records with rare attribute values are susceptible to identity disclosure. Well-established rules such as the n-rule—records that fall in any cell with a (non-zero) count less than or equal to n (in practice, n is often 2) are considered to be at risk—embody this viewpoint and are often used to characterize risk. Based on this reasoning, one measure of risk is the number of *unswapped records* that fall in cells with low counts in a cross-tabulation of the post-swap data.

Because data swapping may change the joint probability distribution of sets of attributes containing the swap attributes, one can measure the (dis)utility or distortion resulting from swapping by means of the distance between the pre- and post-swap distributions of the data. Hellinger distance, for example, is used in [3, 4] to study systematically the effect of different choices of swap attributes and swap rate for Current Population Survey (CPS) data.

The remainder of the paper summarizes NISS WebSwap functionality and the data swapping algorithm (§2), treats implementation of NISS WebSwap as a Web service (§3) and describes the NISS WebSwap client software (§4).

2 Functionality

The principal functionality of NISS WebSwap is simultaneously to swap prescribed attributes in a specified fraction of the records in a database. The user specifies the data file containing the microdata records (see §4.4) and the swap rate r—the fraction of records for which the specified attributes will be swapped, typically on the order of 1–10%. At most $2r$% of the records will actually be swapped; fewer will be swapped if records marked to be swapped are swapped with one another. Details of how pairs of records to be swapped are selected appear below.

NISS WebSwap allows the user to specify that two records will be swapped only if designated constraining attributes are either equal (Example: swapping may be allowed only for record pairs both drawn from the same state) or different (Example, records may be swapped only if they come from different counties). Constraints are optional, and may be used in any combination. For the purposes of constraints, all attributes are treated as categorical.

The NISS WebSwap swapping algorithm operates in the following manner.[3]

1. Initially, mark all records as unswapped, and set RS, the number of swapped records, to zero.
2. Randomly select a user-specified fraction (the swap rate r) of the records and mark them as records to be swapped. Let RTS be the number of records to be swapped.
3. Select a record R_1 at random from the current set of marked, unswapped records.
4. Select a second record R_2 at random from the current set of *all* unswapped records (marked or not).
5. Determine whether the swap is a *true* swap: $R_1 \neq R_2$. If not, return to Step 4.
6. Determine whether equality and inequality constraints are satisfied. If not, return to Step 4.
7. Interchange the swapped attribute(s) between R_1 and R_2, mark *both* as swapped, and set

$$\mathrm{RS} = \begin{cases} \mathrm{RS}+1 & \text{if only } R_1 \text{ had been marked for swapping} \\ \mathrm{RS}+2 & \text{if both } R_1 \text{ and } R_2 \text{ had been marked for swapping} \end{cases}$$

[3] This is a general overview; fine details of the implementation are omitted.

(If R_2 had also been marked for swapping, then no record replaces it, which is why fewer than $2r\%$ of the records may actually be swapped.)

8. If $\mathrm{RS} < \mathrm{RTS}$, return to Step 3.

3 Web Services Implementation

Web services [1] is a technology that started taking shape in early 2002, and has evolved and expanded rapidly since then. A Web service is an application that exists in a distributed environment, such as the Internet or an organization's intranet. A Web service accepts a request, performs its function based on the request, and returns a response. Communications to and from a Web service are encoded using extensible markup language (XML) [9]. For example, a client invokes a Web service by sending an XML message, then waits for the XML response. Because all communication is in XML, Web services are not tied to any specific operating system or programming language. Java, Perl and Python programs can all communicate, and Windows applications can interact with Unix applications.

A Web service has a public interface, defined in a common XML grammar. The interface describes the methods available to clients and specifies the signature for each method. Currently, interface definition is accomplished via the Web Service Description Language (WSDL). This opens dramatic possibilities for software re-use: to employ a particular Web Service, we can use its WSDL to determine how to access from within our applications the services it provides!

NISS WebSwap is implemented as a free Web service, and its WSDL description is available (Appendix A). This description enables interested users to access the swapping services using XML-based requests (which does require some expertise and effort, though). Since NISS WebSwap is largely a research tool to perform data swapping, we provide a GUI-based *client* (see §4) to access the service and perform the swapping.

The entire NISS WebSwap application consists of the client, which communicates with a Java *Servlet* [6] running on our server, which in turn passes requests to and from the Web Service, also running on our server. Technically, the Servlet is the client of the Web service, and what we call the client is a client of the Servlet, whereas a "true" Web service client would access the service directly. However, we felt that for general users, the GUI client should be as lightweight and portable as possible, which is accomplished by inserting the Servlet layer to handle the XML messaging.

The client is a Java application that collects user's swapping specifications and transmits the specifications and data to the Servlet using Java's remote method invocation (RMI) protocol. The Servlet is a (server-side) Java application that runs within a "servlet container" on the NISS Web Services server. The Servlet, using libraries that accompany Sun's Web services toolkit, bundles the data and specifications into an XML request that it transmits, using the simple object access protocol (SOAP), to the Web service. The Web service receives the request from the Servlet, extracts the data, carries out the requested swapping, and rewraps the swapped data and auxiliary information as an XML response that is passed back to the Servlet. Finally, the Servlet unbundles the XML response and transmits the swapped data back to the client on the user's machine. The process is depicted in Figure 1.

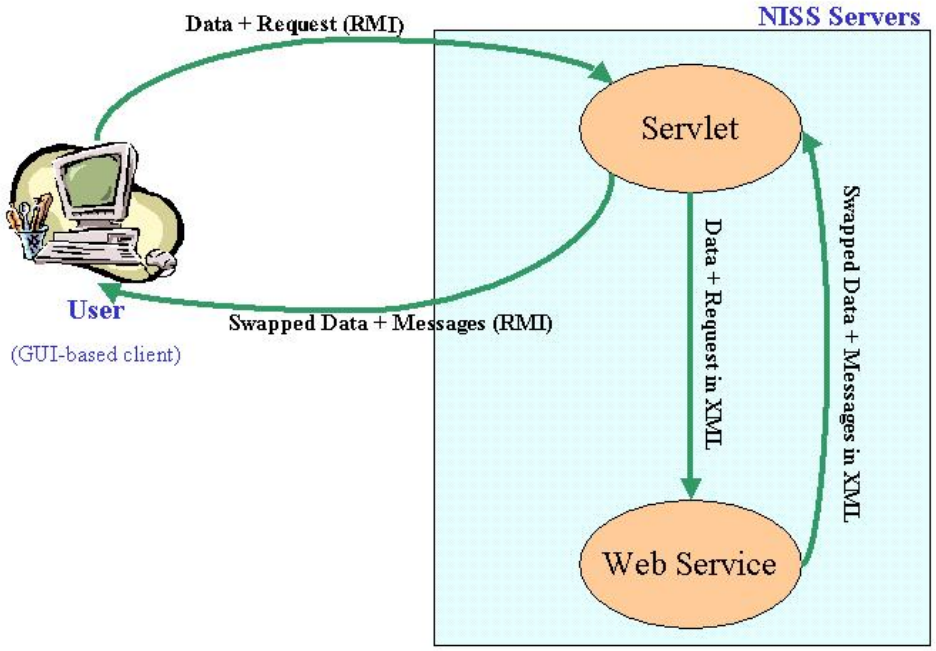

Figure 1: NISS WebSwap components.

4 The NISS WebSwap Client

Here we describe the NISS WebSwap client software. Figure 2 shows both the NISS WebSwap system and the associate files (on the user's machine), consisting of two input files (§4.4), an automatically produced specifications file (§4.5), and two output files (§4.6).

4.1 System Requirements

The NISS WebSwap program requires that a Java 2 runtime environment compliant with the Java 2 Platform, Standard Edition (J2SE) version 1.3.1 or 1.4.x be installed on the user's computer. This can be downloaded from `http://java.sun.com` or from the Web sites of operating system vendors. An Internet connection is required to access the NISS Web services server.

4.2 Software Installation

First, download the NISS WebSwap distribution NISSWebSwap_v11.zip from the NISS Web site, at `www.niss.org/WebServices/dg/WebSwap.html`. Unzip the distribution into separate directory, for example, named *NISSWebServices*. Finally, read the README.webswap file bundled with the distribution. Data, description and specifications files (see below) may reside in any directory, but must all be in

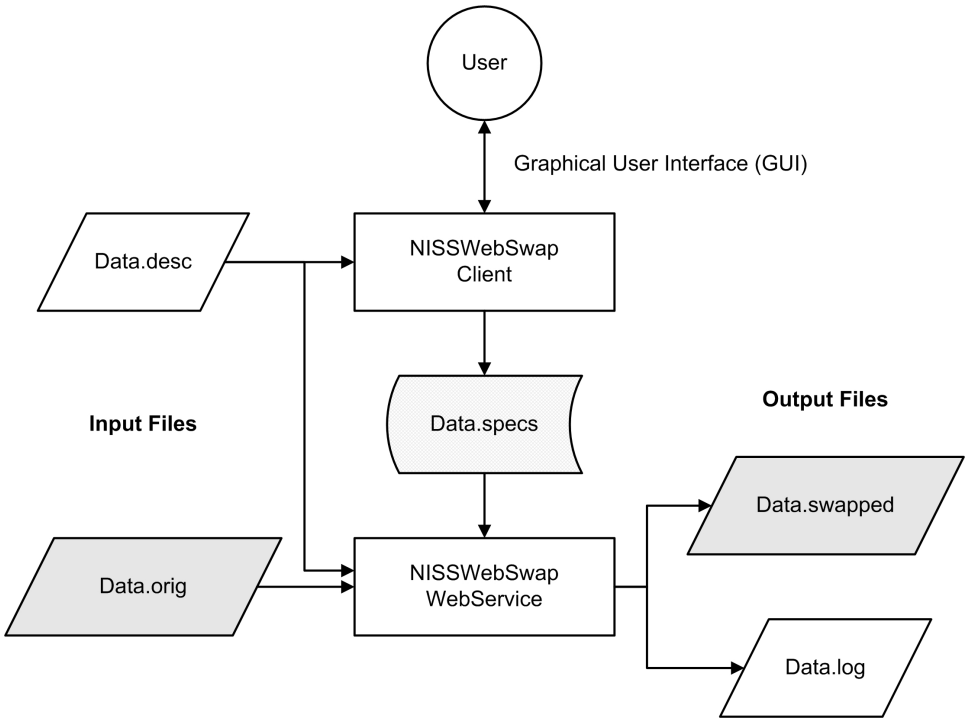

Figure 2: The NISSWebSwap System. Rectangles are the software components and parallelograms are the input and output files; data files are in gray. The NISS WebSwap–produced specifications file is in the center of the diagram.

the same directory. By default output and log files are placed in that same directory.

4.3 Starting NISS WebSwap

NISS WebSwap is distributed in JAR (Java Archive) format: the file NISSWebSwap.jar contains all necessary executable code. There are two ways to launch the NISS WebSwap application. For *direct start*, which is available only under Windows 9x/NT/2000/XP, double–click on NISSWebSwap.jar in Windows Explorer or any other file browser. If the proper Java 2 runtime environment has been installed, this will start NISS WebSwap. *Command-line invocation* requires that the command java be in the system "path," and that it invoke the correct version of the Java 2 runtime environment. Then, execute the command

```
java -jar NISSWebSwap.jar
```

from the command prompt of a Unix/Linux or Windows system.

4.4 Input Files

Two input files are required by NISS WebSwap: a *data file* containing the microdata records and a *description file* containing corresponding metadata. Samples of both are contained in the NISS WebSwap distribution.

```
1,25_55,Gov,Bach,UM,W,M,40,<50
2,25_55,SE,Bach,M,W,M,<40,<50
3,25_55,Pvt,HS,UM,W,M,40,<50
4,25_55,Pvt,<HS,M,NW,M,40,<50
5,25_55,Pvt,Bach,M,NW,F,40,<50
6,25_55,Pvt,Bach+,M,W,F,40,<50
7,25_55,Pvt,<HS,M,NW,F,<40,<50
8,25_55,SE,HS,M,W,M,40+,50+
9,25_55,Pvt,Bach+,UM,W,F,40+,50+
10,25_55,Pvt,Bach,M,W,M,40,50+
```

Figure 3: Sample CSV data file webswapdemo.orig included in the NISS WebSwap distribution. The file contains an 8–attribute data set derived from the CPS [5]. The first entry in each record is the ID; the "real attributes" are Age, Employer Type, Education, Marital Status, Race, Sex, Hours Worked, and Salary.

```
ID,K
Age,C
EmplType,C
Educ,C
MarStatus,C
Race,C
Sex,C
AveHours,C
Salary,C
```

Figure 4: Sample description file webswapdemo.desc included in the NISS WebSwap distribution, corresponding to the data file webswapdemo.orig.

Data File. The file containing the microdata must be in comma-separated value (CSV) format, with one record per line, as illustrated in Figure 3. CSV files can be produced by most database management systems and statistical packages (such as SAS), as well as Microsoft Excel. Both Microsoft/Excel CSV (MS) and ISO standard CSV (ISO) are supported. *The first entry in each record must be a unique identifier*. The simplest way to create identifiers, as in Figure 3, is to number the records.

Description File. The description file is an ASCII metadata file containing the names and types of the attributes in the data file. Figure 4 illustrates a description file. Each line in the description file corresponds to one attribute, and has the form

`AttributeName, AttributeType`.

Allowable attribute types are: K = record identifier, C = categorical and R = real. The order of attribute names in the description file must match the order in which the attributes appear in the data file. Some inconsistencies between the data file and description file (for instance, unequal numbers of attributes) are detected by the NISS WebSwap client.

```
1024
webswapdemo.orig
webswapdemo.desc
webswapdemo.log
webswapdemo.swapped
webswapdemo.specs
25.0
S,O,O,O,O,O,O,O
MS
```

Figure 5: Sample specifications file webswapdemo.specs produced by the NISSWebSwap client from the description file webswapdemo.desc and swapping specification shown in Figure 9. The swap rate is (an unrealistically high) 25%; Age is to be swapped, and all other attributes are unconstrained.

4.5 The Specifications File

The specifications file is an ASCII text file produced by the NISS WebSwap client, containing the following items:

- Number of records in the data file;
- Names of the data file, description file, log file, output file and specifications file;
- The swap rate;
- The swapping specification: for each attribute, in the order specified by the description file, whether it is to be Swapped, must remain Fixed, must Differ between any pair of records that are swapped, or is not constrained—Other.
- The CSV type of the data file.

Existing specifications files may be loaded directly by the NISS WebSwap graphical user interface (GUI): see §4.7.

The specifications file is required by the NISS WebSwap Web service. Because it is produced automatically with the proper structure by the NISS WebSwap GUI, *it should not be edited by hand*.

4.6 Output Files

As shown in Figure 2, NISS WebSwap produces and downloads to the user's computer two files: an *output file* containing the swapped microdata and a *log file* containing details of the swapping process.

The output file is in the same CSV format as the data file. An example is shown in Figure 6. Record identifiers are retained.

The log file, illustrated in Figure 7, contains both details of the user input (file names and swapping specification) and results such as the number of swaps performed.

4.7 NISS WebSwap User Interface

The NISS WebSwap GUI is used to create or edit the specifications file required by the NISS WebSwap Web service, by allowing the user to specify files and construct the swapping specification. The main

```
1,25_55,Gov,Bach,UM,W,M,40,<50
2,25_55,SE,Bach,M,W,M,<40,<50
3,25_55,Pvt,HS,UM,W,M,40,<50
4,<25,Pvt,<HS,M,NW,M,40,<50
5,25_55,Pvt,Bach,M,NW,F,40,<50
6,25_55,Pvt,Bach+,M,W,F,40,<50
7,25_55,Pvt,<HS,M,NW,F,<40,<50
8,<25,SE,HS,M,W,M,40+,50+
9,<25,Pvt,Bach+,UM,W,F,40+,50+
10,25_55,Pvt,Bach,M,W,M,40,50+
```

Figure 6: Sample CSV NISS WebSwap output file webswapdemo.swapped arising from the data file, description file and specifications file appearing in Figures 3, 4 and 5. Comparison with Figure 3 shows that "Age" has been swapped for records 4, 8, and 9.

```
Swapping Log:Thu Oct 24 17:22:16 EDT 2002
Number of risky records = 1024
Number of records marked for swapping = 256
Number of swaps performed:  223
-- listing properties --
desc.file=webswapdemo.desc
spec.file=webswapdemo.specs
attribute.specs=S,O,O,O,O,O,O,O
output.file=webswapdemo.swapped
num.records=1024
csv.type=MS
log.file=webswapdemo.log
data.file=webswapdemo.orig
swap.percentage=25.0
```

Figure 7: Sample NISS WebSwap log file webswapdemo.log corresponding to the demonstration data file, description file and specifications file appearing in Figures 3, 4 and 5. Because only 223 swaps were performed, in 33 of these, both records swapped must have been marked for swapping. In the other 190 swaps, only one of the records had been marked.

window (see Figure 8) serves as the means of communication to the user, recording loading of files, contents of specifications files and the results of swapping, as well as warnings (black type) and errors (red type). Its entire contents may be saved to a session file (§4.8).

The basic unit for NISS WebSwap is the *project*, which is defined by its specifications file and associated data and description files. Existing projects are loaded via the `Open` item on the `Project` menu, and new ones are created with the `New` item.

`Open` loads an existing specifications file (*.specs), and provided that the data and description files that it requires exist and be compatible, allows the user, by choosing the `Edit` item on the `Project` menu, to edit the output file names and swapping specification, using the window shown in Figure 9. `New` creates a project by selecting the underlying data file (*.orig), loads and verifies the associated description file (*.desc), and provides a window, essentially identical to the one shown in Figure 9, that allows the user to enter output file names and the swapping specification.

The `New` and `Edit` windows allow the user to specify or modify two principal items:

Output File Names: The default names for data file data.orig are data.specs for the specifications file (in the case of a new project), data.swapped for the output file and data.log for the log file. Any of these may be changed by the user. If the name of the specifications file is changed, the names of the output and log files are changed automatically to match it, but these may also be changed independently. Similarly, if the name of the output file is changed, that of the log file changes to match it. Changing the name of the log file alone is possible, but keeping the names of the output and log files identical preserves their association for future reference.

The Swapping Specification. As outlined in §2, the swapping specification consists of the swap rate and attribute specifications. The swap rate is set by the slider in the `Edit` (Figure 9) or `New` project windows. Allowable values are 1%,. . .,50%. Although the slider allows the swap rate to be set to zero, NISS WebSwap will not save the specifications file in this case, and will produce an error message. Attribute specifications specify, for each attribute, whether it is to be *Swapped*, must remain *Fix*ed, must *Differ*, or is unconstrained (*Other*). The selections are made by the checkboxes shown in Figure 9. For a new project, the default value is *Other* for all attributes. If no attribute is chosen to be swapped, the specifications file will not be saved, and an error message will result.

When the selections are complete, clicking on the "Save" button saves them to the specifications file. The contents of the specifications file are then displayed in the NISS WebSwap main window.

Once a specifications file has been opened (and possibly edited) or created, the user simply clicks on the "Swap" button (Figure 8) to perform the swapping. Figure 8 shows the messages resulting from a successful swap, including the names of the output files. The NISS WebSwap GUI does not provide direct capability to view these files, but they may be examined with any file viewer. The CSV output file may also be opened in applications such as Excel or statistical packages.

Figure 10 summarizes the major steps in using the NISS WebSwap GUI.

4.8 Session Files

Upon exit from NISS WebSwap, the user has the option to save the cumulative contents of the main window to a session file (*.session), or to append the contents to an existing session file.

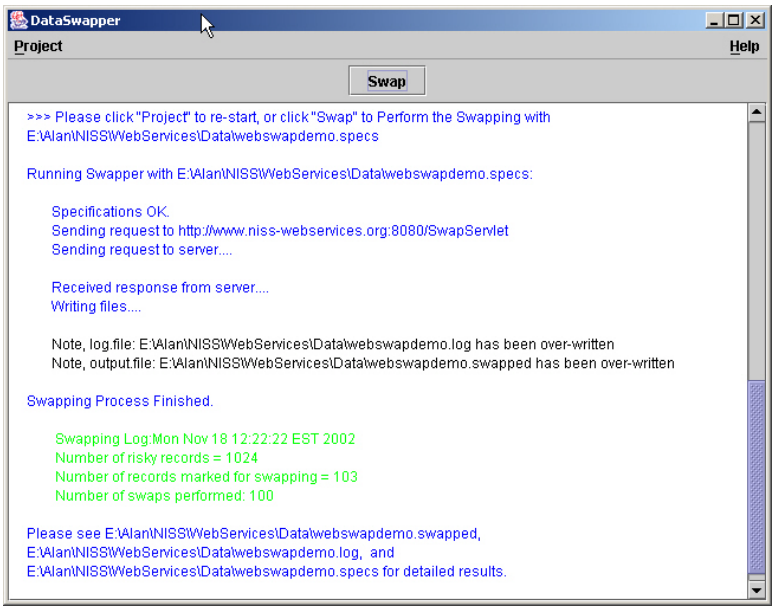

Figure 8: The NISSWebSwap main window, showing the messages produced when the swapping is performed successfully.

References

[1] E. Cerami. *Web Services Essentials*. O'Reilly & Associates, Sebastopol, CA, 2002.

[2] G. T. Duncan, S. A. Keller-McNulty, and S. L. Stokes. Disclosure risk vs. data utility: The R-U confidentiality map. *Management Sci.*, 2001. Under review.

[3] S. Gomatam and A. F. Karr. Distortion measures for categorical data swapping. *J. Official Statist.*, 2003. Under review.

[4] S. Gomatam, A. F. Karr, and A. P. Sanil. A risk-utility formulation for categorical data swapping. *J. Official Statist.*, 2003. Submitted for publication.

[5] S. Hettich and S. D. Bay. The UCI KDD Archive. Irvine, CA: University of California, Department of Information and Computer Science, 1999. Available on-line at kdd.ics.uci.edu.

[6] Sun Microsystems, Inc. Java Servlet Technology. Information available on-line at java.sun.com/products/servlet/.

[7] L. C. R. J. Willenborg and T. de Waal. *Statistical Disclosure Control in Practice*. Springer–Verlag, New York, 1996.

[8] L. C. R. J. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*. Springer–Verlag, New York, 2001.

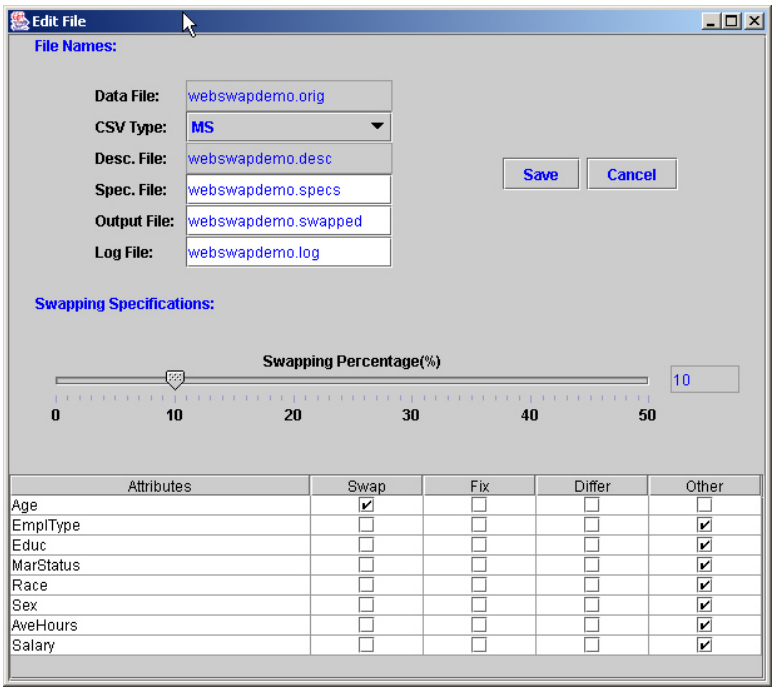

Figure 9: The NISSWebSwap `Edit` project window used to edit the specifications file shown in Figure 5. As desired, the user may change the names for the specifications, output and log files and the swapping specification.

[9] World Wide Web Consortium. Extensible Markup Language (XML). Information available on-line at www.w3.org/TR/REC-xml.

A The NISS WebSwap WSDL

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="Swap_dataService"
targetNamespace="http://WebSwap_swap.org/wsdl"
xmlns:tns="http://WebSwap_swap.org/wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ns2="http://WebSwap_swap.org/types"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
    <schema targetNamespace="http://WebSwap_swap.org/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:tns="http://WebSwap_swap.org/types"
        xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns="http://www.w3.org/2001/XMLSchema">
```

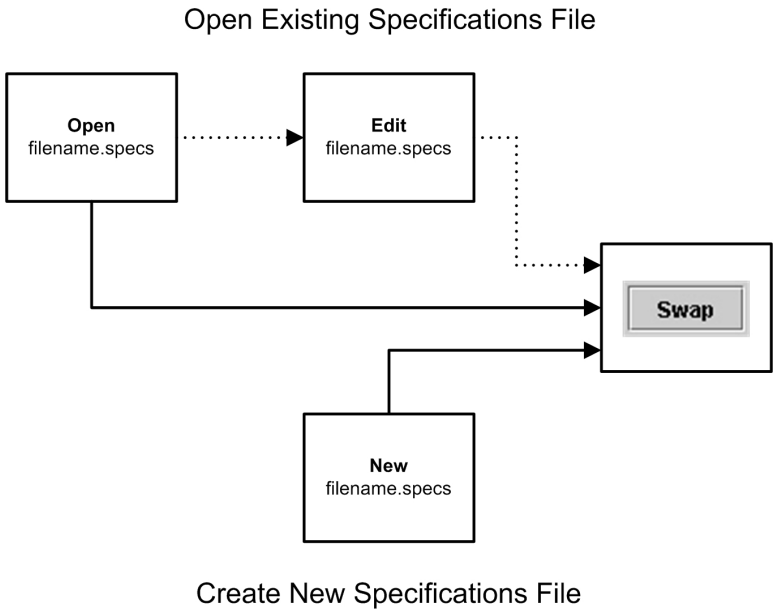

Figure 10: Summary of steps in using NISS WebSwap. *Top*: using NISS WebSwap with an existing specifications file; the edit step is optional. *Bottom*: using NISS WebSwap by creating a new specifications file.

```
<complexType name="SwapData">
  <sequence>
    <element name="numFields" type="int"/>
    <element name="outputFile" type="string"/>
    <element name="numRecords" type="int"/>
    <element name="riskCutoff" type="double"/>
    <element name="data" type="tns:ArrayOfArrayOfstring"/>
    <element name="dataFile" type="string"/>
    <element name="constraints" type="base64Binary"/>
    <element name="log" type="tns:ArrayOfstring"/>
    <element name="swapRate" type="double"/>
    <element name="riskFraction" type="double"/>
    <element name="logFile" type="string"/>
    <element name="csvType" type="string"/></sequence></complex Type>
<complexType name="ArrayOfArrayOfstring">
  <complexContent>
    <restriction base="soap-enc:Array">
      <attribute ref="soap-enc:arrayType"
          wsdl:arrayType="tns:ArrayOfstring[]"/>
          </restriction></complexContent></complexType>
<complexType name="ArrayOfstring">
  <complexContent>
    <restriction base="soap-enc:Array">
      <attribute ref="soap-enc:arrayType"
          wsdl:arrayType="string[]"/></restriction>
```

```
             </complexContent></complexType></schema></types>
<message name="doSwap">
  <part name="SwapData_1" type="ns2:SwapData"/></message>
<message name="doSwapResponse">
  <part name="result" type="ns2:SwapData"/></message>
<portType name="SwapIF">
  <operation name="doSwap">
    <input message="tns:doSwap"/>
    <output message="tns:doSwapResponse"/></operation></portType>
<binding name="SwapIFBinding" type="tns:SwapIF">
  <operation name="doSwap">
    <input>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://WebSwap_swap.org/wsdl"/></input>
    <output>
      <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          use="encoded" namespace="http://WebSwap_swap.org/wsdl"/></output>
    <soap:operation soapAction=""/></operation>
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/></binding>
<service name="Swap_data">
  <port name="SwapIFPort" binding="tns:SwapIFBinding">
    <soap:address location="http://www.niss.web-services:8080/WebSwap/SwapIF"/>
      </port></service></definitions>
```