Confidentiality Audit Procedures

Michael A. Nolte, Senior Research Associate
Health and Retirement Study

Goal: To identify variables in any given dataset that might compromise respondent confidentiality if released to the general public.

Step 1. Identify possible risk factors inherent in releasing a particular data file to the public in unmodified form. Variables that present such risks include, but are not limited to, the following content types:[1]

- Name(s)
- Address (lot/street, city, state, Zip Code)
- Exact dates (date of birth, date of death, date of interview date married/divorced, military service dates, disability periods)
- Identifying sample information (PSU, segment, CMSA, census tract)
- Geographic identifiers other than Census Region in any context (state, county, census tract, congressional district, MSA, centroid information)
- Numeric identifiers (SSN, Medicare ID, Medicaid ID, employer ID, badge number, telephone number, driver's license or vehicle registration information).
- Internet identifiers (IP address, email address, MAC, login identifier, username)
- Biometric information (blood type, DNA information; fingerprints; retina patterns; weight/height)
- Indirect identifiers (employer name and/or address, detailed industry and/or occupation codes; religious affiliation; race/ethnicity; education level) that, when used together, could expose a respondent's identity.
- Any collection of respondent information that could potentially be matched in a commercial database (earnings records as reported to federal, state or local authorities; housing assessments; medical conditions)
- Detailed medical information as defined in HIPAA
- Interviewer comments

Step 2. Determine actions to be taken to remove risk factors when problem variables are found.

- Remove from dataset
- Remove from dataset and release in separate restricted dataset
- Recode/bracket/collapse
- Top/bottom code, blur
- Mask

[1] See Appendix A for a list of possible search terms for use in a programmatic audit of machine readable documentation. This list should be changed as necessary in order to match the properties of the data file(s) under review.

Steps 1 and 2 should be considered in determining which fields should be extracted from Surveycraft or Blaise, in creating the Survey Item Disposition List (a.k.a. *qnumlist*), and during the final audit process.

Step 3. Obtain access to all components of the data set under review, including associated meta-data files and processing narratives. If the dataset under review has already been released (e.g., the cross-year RAND files), retrieve the complete distribution set, being sure to use same data and documentation file(s) that the public is accessing

Step 4. Identify key documentation items:

Input
- Questionnaire
- Programming specifications for data collection instrument
- Interview flow documentation

Processing
- Processing notes
- Intermediate data files and accompanying data definition statements
- File transformation source code

Output
- Data files
- Data definition statements (SAS, SPSS, Stata, etc.)
- Question text matching the data definition
- Code frame
- Descriptive statistics (MIN, MAX, MEAN, SD) and frequencies
- User notes

Step 5. If possible, build Documentation Database for use by automated audit procedures.
- QTABLE – data definition statement components
- QTEXT – question text
- CTABLE – code frame
- Optional elements – questionnaire flow, frequencies

Step 6. Testing Process
- Scan all data files for alphanumeric content
- Generate descriptive statistics for all variables with a width > 8; review for inappropriate content (see Step 1)
- If Documentation Database exists, apply automated procedures (see Appendix B) to retrieve variables with possible problems
  - Review label and question text of variables (from Step 1) flagged by audit program

  - If necessary, review questionnaire/data collection program specifications as well as processing files and notes to supplement audit program results
  - Examine frequency counts and descriptive statistics for variables flagged by audit program
  - If necessary, examine data file content for problem variables
- If no Documentation Database exists, review "paper trail" for file processing
  - Review paper questionnaire or data collection program specifications for questions that present disclosure risk (from Step 1). This review should cover both question text and code frame
  - Review processing files and notes
  - Examine frequency counts for problem variables
  - Determine data file content for problem variables

Step 7. Apply confidentiality rules as determined from steps 1 and 2. If problems are found:

- Develop variable/record change procedures
- Make recommendations for file modifications
- Prepare draft of revised documentation
- Notify processing staff
- Notify HRS Senior Staff

If no problems are found, proceed to Step 8.

Step 8. Submit audit process results to HRS Senior Staff for final review and action.

- Release: Project Director signs off on release of file.
- Rework
- (If previously released) Recall

Step 9. If Step 8 requires Rework or Recall

- If file is currently in distribution, implement recall process
- Implement variable/record change procedures necessary to create revised file
- Publish revised documentation
- Create distribution package (data and documentation)
- If appropriate, forward distribution package to outside parties for final review.
  - Data Confidentiality Committee
  - IRB
  - External oversight entities (e.g., SSA)
  - Other_____

Appendix A: Typical Search Terms

These terms are loosely based on a list developed by Sheila Deskins in consultation with Bill Rodgers and Dan Hill. If the reader has any suggestions for additions or changes, please contact the author of this paper via e-mail: `manolte@isr.umich.edu`.

1. Personal (name, birth, death, month/year)
    NAME (in any context)
    YEAR, MONTH, DAY, DATE
    BORN, BIRTH
    DEATH, DIE, DEAD, DECEASED

2. Geography (city, state, zip, address, phone)
    LOC{+}, ADDR{+}, LIVE, ZIP{+}, PHONE
    CITY, STATE, COUNTY, COUNTRY, *FOREIGN*[2]
    SAMPLE, PSU, MSA, CENSUS, TRACT,*AREA*

3. Occupation and Industry/Education/Income
    IND{+}, OCC{+}, EMP{+}, JOB{...}TITLE, JOB
    EDUC{+}, ED, *DEG{+}*
    INCOME, SALARY, WAGE

4. Race
    RACE, HISPAN{+}, SPANISH, ETHNIC

5. SSN/Medicare/Medicaid
    SSN, SOC{...}SECURITY, MEDICARE, MEDICAID

Notes:
1. **{+}** indicates stemming; for example, IND+ will retrieve IND, INDUS, INDUSTRY. If too many false hits show up, change the search pattern to somthing more specific, i.e. INDUS*
2. **{...}** indicates a match on any intervening category

[2] Additions to the search list suggested after the original version of this document was produced are in italics

Appendix B: A Sample Perl script for Documentation Database Audits.

```
#!/opt/local/bin/perl
#======================================================================
# audit.pl -- scan text strings for certain phrases and/or words
#             input:  ACCESS database via ODBC link
#             output: ASCII text file
#
# This script can be used to scan question text and variable labels
# for phrases that might compromise respondent confidentiality.
# Classes of phrases include:
#   1. personal (name, birth, death, month/year)
#   2. geography (city, state, zip, address, phone)
#   3. occ-ind/education/income
#   4. race
#   5. SSN/Medicare/Medicaid
#
# v1.0    initial implementation                 man     aug 99
#=======================================================================
$script_name = "audit.pl";
$ver = "v1.0";
$date = "Aug 1999";

# set up link to ACCESS DBMS via ODBC
use Win32::ODBC;
$DSN = "hrs96audit";
$db = new Win32::ODBC($DSN);
die ("Cannot open ODBC database\n") if ! $db;


# get the output file name
if ($ARGV[0] eq "") {
   die "\n...You must specify an output file\n"
   ."Usage: [perl] $script_name {outfile} {search class}\n";
} else {
   $outfi = ">".$ARGV[0];
   unless (open(OUTFILE, $outfi))
   {die ("...cannot open outfile $outfi\n");}
}

if ($ARGV[1] eq "") {
   die "\n...Specify the search class:\n"
       ."   0=a11, 1=personal, 2=geography,"
       ." 3=occ-ind/education/income,"
       ." 4=race, 5=SSN/Medic*)\n";
} else {
   $class = $ARGV[1];
}

# error file
   unless (open(ERRFILE, ">errtest1.txt"))
   {die ("...cannot open error file\n");}

# setup for output
# format to display character string exactly as is
format OUTLINE =
@*
$olin
.

# question text formatting
format QTREC =
~~        ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$olin
.

#define some variables
$StudyName = "HEALTH AND RETIREMENT STUDY WAVE 3 (1996)";
$FileContent = "Possible Problem Cases";
```

```
#read in QTABLE contents
$QTABLE = "PrelimCodebook_qid";

print ("\n...reading $QTABLE.\n");
$tot= 0;
$stmt = "SELECT * FROM " . $QTABLE . " ORDER BY " . $QTABLE . ".SORT1";
$rc = $db->Sql($stmt);
if ($rc) {
   print ERRFILE ("\nSELECT failed: $stmt --");
   die ("\nSELECT failed: $stmt --");
}

# do the header
select(OUTFILE);
$~ = "OUTLINE";
$olin = "     " . $StudyName;
write;
$olin = "     " . $FileContent;
write;
if ($class == 0) {
     $olin = "     " . "All audit classes selected"; write;
     $olin = "     " . "  1=personal (name, birth, death, month/year)"; write;
     $olin = "     " . "  2=geography (city, state, zip, address, phone)"; write;
     $olin = "     " . "  3=occ-ind/education/income"; write;
     $olin = "     " . "  4=race"; write;
     $olin = "     " . "  5=SSN/Medicare/Medicaid"; write;
}
if ($class == 1) {
     $olin = "     " . "Audit class selected:"; write;
     $olin = "     " . "  1=personal (name, birth, death, month/year)"; write;
}
if ($class == 2) {
     $olin = "     " . "Audit class selected:"; write;
     $olin = "     " . "  2=geography (city, state, zip, address, phone)"; write;
}
if ($class == 3) {
     $olin = "     " . "Audit class selected:"; write;
     $olin = "     " . "  3=occ-ind/education/income"; write;
}
if ($class == 4) {
     $olin = "     " . "Audit class selected:"; write;
     $olin = "     " . "  4=race, 5=SSN/Medicare/Medicaid"; write;
}
if ($class == 5) {
     $olin = "     " . "Audit class selected:"; write;
     $olin = "     " . "  5=SSN/Medicare/Medicaid"; write;
}

$ndx=0;
while ($db->FetchRow()) {
   #                  0        1      2      3      4        5       6     7     8
9     10    11      12
   @qvec  = $db-
>Data("section","name","qid","label","multis","vtype","loc","wid","ndec","md1","md2","qte
xt","sort1");
   $tot++;
   select (STDOUT);
   print "*" if (($tot % 200) == 0);
   #last  if ($tot > 50);

   #save the variables
   $qname = $qvec[1];
   $qid   = $qvec[2];
   $qlabel= $qvec[3];
   $qtext = $qvec[11];

   $saveqtxt = $qtext . qlabel;
   $saveqtxt =~ tr/[a-z]/[A-Z]/ ; #upcase

   $printthis = 0;
   $ptclass   = "Audit Classes:";
```

```
   # add problem phrases to the appropriate category below
   # 1. personal (name, birth, death, month/year)
   if ($class == 0 || $class == 1) {
      if ($saveqtxt =~ / NAME | NAM/ or
          $saveqtxt =~ / YEAR| MONTH| DAY| DATE/ or
          $saveqtxt =~ / BORN| BIRTH/ or
          $saveqtxt =~ / DEATH| DIE| DEAD| DECEASED/) {$printthis = 1;}
      if ($printthis == 1) {$ptclass = $ptclass . " 1";}
   }

   # 2. geography (city, state, zip, address, phone)
   if ($class == 0 || $class == 2) {
      if ($saveqtxt =~ / LOC| ADDR| LIVE| ZIP| PHONE/) {$printthis = 2;}
      if ($saveqtxt =~ / CITY| STATE[\.\?\,\:\;]| COUNTY| COUNTRY/) {$printthis = 2;}
      if ($saveqtxt =~ / SAMPLE|PSU| MSA| CENSUS| TRACT/) {$printthis = 2;}
      if ($printthis == 2) {$ptclass = $ptclass . " 2";}
   }

   # 3. occ-ind/education/income
   if ($class == 0 || $class == 3) {
      if ($saveqtxt =~ / IND| OCC| EMP| JOB.*TITLE| JOB/) {$printthis = 3;}
      if ($saveqtxt =~ / EDUCATION| EDUC| ED| DEGREE/) {$printthis = 3;}
      if ($saveqtxt =~ / INCOME | SALARY| WAGE/) {$printthis = 3;}
      if ($printthis == 3) {$ptclass = $ptclass . " 3";}
   }

   # 4. race
   if ($class == 0 || $class == 4) {
      if ($saveqtxt =~ / RACE| HISPANIC| HISPAN| SPANISH| ETHNIC/) {$printthis = 4;}
      if ($printthis == 4) {$ptclass = $ptclass . " 4";}
   }

   # 5. SSN/Medicare/Medicaid
   if ($class == 0 || $class == 5) {
      if ($saveqtxt =~ / SSN| SOC.*SECURITY| MEDICARE| MEDICAID/) {$printthis = 5;}
      if ($printthis == 5) {$ptclass = $ptclass . " 5";}
   }

   # display the question text
   if ($printthis != 0) {
      select(OUTFILE);
      $~ = "OUTLINE";
      $olin = "
================================================================================";
      write;
      $olin = "     " . $qid . " " . $qname . " " . $qlabel;
      write;
      $olin = "          " . $ptclass;
      write;
      # question text (if any)
      if ($qtext ne "") {
         select(OUTFILE);$~ = "OUTLINE";$olin = " "; write;
         $qtext =~ s/\r//g;          # get rid of CRs
         @holdq = split (/\n/,$qtext);
         $bcount = @holdq;
         for ($iq=0; $iq < $bcount; $iq++) {
            $lineout = $holdq[$iq];
            if (length($lineout) > $qlim) {
               select(OUTFILE);
               $~ = "QTREC";
            } else {
               select(OUTFILE);
               $~ = "OUTLINE";
               $lineout = $space10 . $lineout;
               #chop($lineout);
            }
            $olin = $lineout;
            write;
         }
      }
```

```
      $olin = "";
      write;
   }
}

# close up and end
select (STDOUT);
print ("\n...$QTABLE complete; $tot processed. \n");
$db->Close();
exit(0);
```