SASifying CAPI Audit Trails for Analysis

Phuc Ho, Lucien Smith, William Chan (BAE Systems/BLS), Sang Y. Kang, Lucilla Tan
chan.william@bls.gov, tan.lucilla@bls.gov
Bureau of Labor Statistics, Division of Consumer Expenditure Surveys, 2 Massachusetts Ave,Suite 3985, Washington DC, 20212[1]

Abstract
With the advance in portable computing resources in recent years, many data collection agencies have moved from a paper-and-pencil instrument to Computer Assisted Personal Interview (CAPI). With CAPI came the ability to capture timing and the sequence of keystrokes associated with an interview, via an audit trail. The audit trails contain a wealth of information for instrument diagnostics and survey methodologists. However, analyses of audit trails have been limited because of the overhead in processing these detailed, semi-structured text files.

This paper reports on the development of a prototype to analyze CAPI audit trails for the *Consumer Expenditure Survey* at the Bureau of Labor Statistics. The transformation of the audit trails into SAS data sets expands the accessibility of information contained in the audit trails to a wider range of users, as programming queries on SAS data sets is simpler than programming queries on the audit trails in their native format. Techniques using SAS to parse the audit trail files into a more structured format, and examples of information gleaned from analyzing the audit trails, are presented here.

Introduction

The *Consumer Expenditure Survey (CE)* is a national survey sponsored by the U.S. Bureau of Labor Statistics (BLS) to collect information on spending by the U.S. civilian, non-institutional population. The CE is the basic source of data for revising the items and weights for the Consumer Price Index; it is also used to construct statistical measures of consumption and for market and economic research of expenditure patterns by demographic characteristics. The *Interview Survey* is one of two survey components of the CE. It is a detailed survey on a broad range of expenditure categories, administered by an interviewer who records the respondent's answers to the survey. The U.S. Census Bureau, under contract with BLS, is responsible for data collection and removing confidential respondent information before transferring the data to BLS.

In April 2003, data collection for the *CE Interview survey* was changed from paper-and-pencil questionnaire to Computer Assisted Personal Interview (CAPI), so the questionnaire now resides in a portable computer, and the interviewer records the respondent's answers directly into the computer. Blaise, the software used to program the CE CAPI, has an audit trail facility, which produces an *audit trail* for each respondent interviewed.

Each month, about 5,000 audit trails for the *CE Interview survey* are generated, totaling about 1 gigabyte in size. Manually reviewing audit trails is tedious and time consuming. Analysts were writing their own SAS programs to read and sequentially process the audit trail text files each time an analysis was performed. A more efficient way of culling information from these detailed, semi-structured text files was needed. A preliminary exploration of how to do this was the motivation for this project. Given time, budget and staff resource constraints and the availability of Base SAS version 8.2, a decision was made to develop a prototype for storing and analyzing the audit trails in SAS, to determine if further investment in resources was justified. This paper highlights how the audit trails were transformed and loaded into a simple hierarchical data base structure, and provide a few examples of insights that can be obtained from analyzing the audit trails.

CAPI Audit Trails

An audit trail shows the sequence of keystrokes the interviewer entered into the computer over the course of an interview (or case); it thus provides a detailed history of the sequence and timing of the interview flow, as well as how the instrument is used by the interviewer. This type of information is of interest to survey managers, survey methodologists, and data analysts by offering insights to the following issues:

1. the data collection process - by providing data on how interviewers are using the instrument
2. data quality - at the initial phase of data processing at BLS, data analysts who find discrepancies in a respondent's records look at the case's associated audit trail as the first step of their investigation to determine if the source of discrepancy is in the instrument, or in the processing algorithms.

[1] Opinions expressed in this paper are those of the authors and do not reflect official policy of the U.S. Bureau of Labor Statistics.

3. alleviate respondent burden - by flagging potential problem areas in the questionnaire, it may help identify effective ways to improve the questionnaire

Structurally, an audit trail is a comma delimited text file, where each line in the audit trail (except for header information) shows the date and time when a keystroke was entered. An example of an audit trail for a case appears in Figure 1**.**

Figure 1: Excerpt from an audit trail file

```
"1/11/2004 9:15:50 AM","Enter Form:1","Key:XXXXXXXX "
"1/11/2004 9:15:50 AM","Metafile name:C:\WINCM\DATA\STUDIES\CEQ_BA01\e-inst\inst.bmi"
"1/11/2004 9:15:50 AM","Metafile timestamp:Wednesday, December 03, 2003 8:47:42 AM"
"1/11/2004 9:15:50 AM","WinUserName:FR"
"1/11/2004 9:15:50 AM","Enter Field:Front.Start","Status:Normal","Value:"
"1/11/2004 9:16:13 AM","Leave Field:Front.Start","Cause:Next Field","Status:Normal","Value:5"
……..
"2/11/2004 5:52:21 PM","Enter Field:Sect03.ANYRENT","Status:Normal","Value:"
"2/11/2004 5:52:24 PM","Leave Field:Sect03.ANYRENT","Cause:Next Field","Status:Normal","Value:2"
……

"1/11/2004 6:16:42 PM","Enter Field:Back.Appt.verify_info","Status:Normal","Value:"
"1/11/2004 6:16:43 PM","Leave Field:Back.Appt.verify_info","Cause:Next Field","Status:Normal","Value:1"
"1/11/2004 6:16:44 AM","Leave Form:1","Key:XXXXXXXX "
```

Keystrokes entered into the CAPI instrument are identified by keywords in the audit trails; examples of keywords appear in Figure 2.

Figure 2: Example of Audit Trail Keywords

KEYWORD	DESCRIPTION	KEYWORD	DESCRIPTION
Key	Case identifier	Enter Field	Identifies the name of the field (question) entered
Enter Form	Enter case	Leave Field	Identifies the name of the field (question) left
Leave Form	Leave case	Value	The value entered for the field

Transforming the audit trails into a more structured format

The text format of the audit trails necessitated sequential scanning of these files to access file contents. To perform complex analysis and to optimize queries, the audit trails were transformed into a more structured format for faster random access to information in the audit trails.

Hierarchical database structure for SAS data sets

We came up with a hierarchical data base structure to store the audit trails audit based by the following characteristics:

1. One audit trail represents one case (an interview with a respondent)
2. A case could take one or more sessions to complete. Within each case, the start of a session is delineated with the audit trail keyword "ENTER FORM", and the end of a session by the keyword "LEAVE FORM"
3. Within each session, the interviewer enters questionnaire responses and administrative information as the survey is administered. The relevant keywords that identify these keystrokes are "ENTER FIELD", "LEAVE FIELD", and "ACTION".

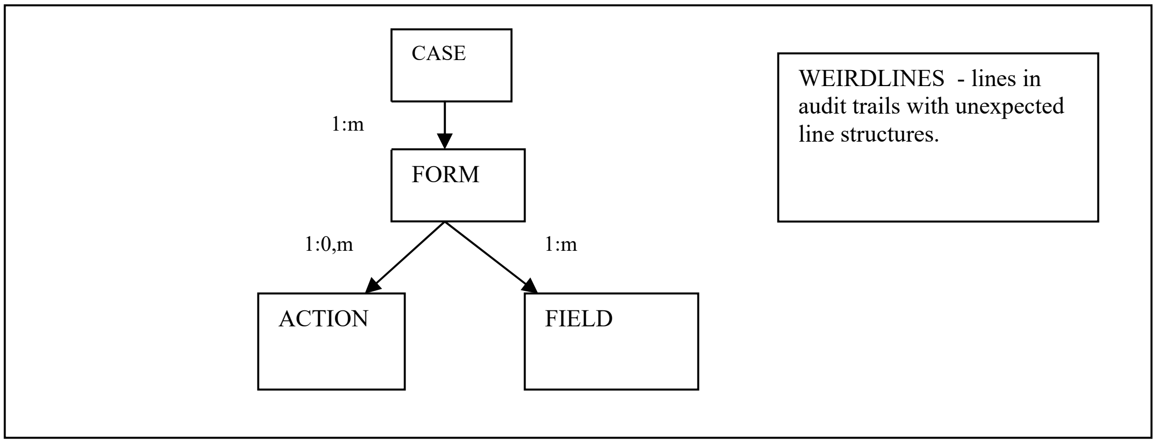
SAS was used to read and parse the audit trails into the following 5 tables (SAS data sets):

- CASE: Each record represents one case, and contains the line number and date-time stamp of start of first entry into the case and last exit from the case.
- FORM: Each record represents one session in a case; so there are as many records for a case as there were sessions to complete the case. Each record shows the line numbers and date-time stamps for the start and end of that session.

- ACTION: Each record represents ACTION keystrokes in a case. Each record shows the date-time stamp, line number, field on which the ACTION keystroke was triggered, the type of action, and resulting value of the action (if any).
- FIELD: In general, each record represents a pair of matching ENTER FIELD - LEAVE FIELD lines in the audit trail. Each record shows the line numbers and date-time stamps for when a field was entered and left, the name of the field, the values upon entering and leaving the field. If there is no matching ENTER - LEAVE for a field, a flag is indicated for the record.
- WEIRDLINES: this table identifies lines in the audit trails that have unexpected structures. Records in an audit have the following expected line structure:
  - each line begins with a date-time stamp
  - there is only one date-time stamp per line
  - within a line, each data field is within a pair of quotes, separated by commas

The relationship between these tables is illustrated in Figure 3.

Figure 3: Database structure for storing audit trails


CASE
1:m
FORM
1:0,m
1:m
ACTION
FIELD
WEIRDLINES - lines in audit trails with unexpected line structures.

The primary key linking the CASE, FORM, ACTION, and FIELD tables is the case identifier. When an audit trail is parsed, the line numbers of the audit trail entries are also captured. Thus, the original audit trails can be reconstructed by merging the tables using the case identifier, and then sorted by the line numbers within each case.

Checking line structures of the Audit Trails

Since parsing the audit trails depended on consistency in the line structures, discrepancies from the expected line structures had to be identified, and corrected where possible.

Each record in the audit trail is read into a variable called READLINE. Next, READLINE is scanned for the occurrence of the date time pattern:

```
%LET DATEPATTERN = ($'0-9' + '/' $'0-9' + '/' $D$D$D$D ' ' $'0-9' +
             ':' $'0-9' + ':' $'0-9' + ' ' ('am' | 'pm'));
```

Using the RXPARSE function, the following problems with date-time stamps were detected:

- Missing Time-stamp

e.g**. "2/10/2004",**"Leave Field:Sect04.TPropt.RowScr1[8].UTC_ITEM","Cause:Next Field","Status:Normal","Value:99"

```
Detection:      ISFIRSTDATE = RXPARSE("&DATEPATTERN ");

                IF (ISFIRSTDATE) THEN
                   CALL RXSUBSTR(ISFIRSTDATE, READLINE, POSITION1, LENGTH1);
```

Action: If no date time value is found the record is flagged, error messages and records are written. No correction to the record is made, since we have no information on what the correct time-stamp should be.

- Double date-time stamps on a line

e.g. "**2/17/2004 9:41:33 AM**","WinUserN"**2/17/2004 10:14:05 AM**","Enter Form:1","Key:XXXXXXXX "

Detection: MULTIPLEDATETEST = RXPARSE("&DATEPATTERN : &DATEPATTERN / ");

```
IF (MULTIPLEDATETEST) THEN
CALL RXSUBSTR(MULTIPLEDATETEST, READLINE, POSITION3, LENGTH3);
```

Action: Split into a new line when 2nd date time-stamp encountered, and the record is flagged.

- Missing date time: if no date time value is found the record is flagged and error messages and records are written.

Once the audit trails have been parsed into the tables, they are ready for analysis.

Analysis

In this section, examples of insights from analyzing the "SASified" audit trails are presented. The largest payoff to storing the audit trails in a more structured format is that it is much simpler to code queries on the audit trails. Unless stated otherwise, analyses was based on 18 months' of audit trails, April 2003 - September 2004 (there are about 5,300 cases each month). We look at how field interviewers are using the CAPI Instrument , and ways to improve the instrument. Due to space limitations, only partial SAS code and results are shown.

a. How Field Interviewers are using the CAPI Instrument

- **Interviewers access FAQ help**

Query: The interviewers can access prepared answers to 5 "Frequently Asked Question" (FAQs) about the survey. How often do interviewers access this reference material?

SAS Code: The audit trail keyword corresponding to entering the FAQ help *and* viewing a specific FAQ is FIELD=faq.h_purpose# and VALUE_L=#. This is a FIELD line, so we work off the FIELD Figure, use the appropriate subsetting IF clause, and do a PROC FREQ on the resulting data set to get the counts. See Figure 4.

Figure 4. Excerpt of SAS code to analyze frequency of interviewers accessing Frequently Asked Questions

```
DATA FAQ (KEEP = SOURCEYR SOURCEMO FIELD VALUE_L);
  SET ATDB.FIELD
    (WHERE = (CENSID NE '' AND
          FIELD IN ("FAQ.H_PURPOSE1"
                "FAQ.H_PURPOSE2"
                "FAQ.H_PURPOSE3"
                "FAQ.H_PURPOSE4"
                "FAQ.H_PURPOSE5")));
RUN;

PROC FREQ DATA = FAQ;
  FIGURE FIELD / MISSING LIST;
  TITLE "FREQUENCY OF FAQ ACCESS - 200304 TO 200409";
RUN;
```

Findings: FAQs are not frequently accessed. Among FAQs accessed, 49% are for #1 (which provides a general description of the survey). See Figure 5.

Figure 5: Frequency of interviewers accessing FAQ help

```
                                                        Cumulative    Cumulative
                    FIELD            Frequency     Percent     Frequency      Percent
                    ------------------------------------------------------------------
                    faq.h_purpose1          160       48.93           160        65.79
                    faq.h_purpose2           86       26.30           246        86.84
                    faq.h_purpose3           21        6.42           267        89.47
                    faq.h_purpose4           27        8.26           294        92.11
                    faq.h_purpose5           33       10.09           327       100.00
```

- **Navigation through the questionnaire**

Query: What percentage of cases flow straight through the questionnaire? Using the audit trails, we can learn what is the most frequent pattern of flowing through the sections in the questionnaire.
SAS Code: due to the more complex code and space limitations, none is shown here.
Findings: Figure 6 shows the most frequent pattern of flowing through the questionnaire sections, by wave of interview.

Figure 6: Most frequently occurring pattern of flowing through the questionnaire sections from the audit trails

Interview Wave	Section Flow
Wave 1 7.7% (n=12,011)	1-> 2-> 3-> 4-> 5-> 6-> 7-> 8-> 9-> 10-> 11-> 12-> 13-> 14-> 15-> 16-> 17-> 18-> 19-> 24
Wave 2 27.1 % (n= 12,154)	3-> 4-> 5-> 6-> 7-> 8-> 9-> 10-> 11-> 12-> 13-> 14-> 15-> 16-> 17-> 18-> 19-> 20-> 21-> 22
Wave 3 16.5% (n=12,010)	3-> 4-> 5-> 6-> 7-> 8-> 9-> 10-> 11-> 12-> 13-> 14-> 15-> 16-> 17-> 18-> 19-> 20-> 22
Wave 4 15.7% (n=11,987)	3-> 4-> 5-> 6-> 7-> 8-> 9-> 10-> 11-> 12-> 13-> 14-> 15-> 16-> 17-> 18-> 19-> 20-> 22
Wave 5 30.0% (n=12,200)	3-> 4-> 5-> 6-> 7-> 8-> 9-> 10-> 11-> 12-> 13-> 14-> 15-> 16-> 17-> 18-> 19-> 20-> 21-> 22

b. Identifying ways to improve the instrument and survey design

- **Time jump**

Query: Since the audit trail is supposed to capture the historical sequence of keystrokes for a case, it is expected that the date-time stamps should be monotonically increasing. However, a case of an audit trail with time jumping backwards was discovered - this could suggest an instrument defect, or manipulation of the computer clock.
SAS Code: Check for time progression for each case, in each of the 4 tables. The example given here applies to the FIELD table for month 03. See Figure 7.
Findings: 27 cases in month 03 were identified to have the problem of the trail files showing time jumping backwards. A listing to identify these cases for further investigation is also generated. See Figure 8.

Figure 7. Excerpt of SAS code to analyze time jump

```
DATA TIMEJUMP;
  SET IN.FIELD (KEEP = CENSID SOURCEYR SOURCEMO FPRIMARY LINE_E
                DATETIME_E FIELD);
  IF (DATETIME_E NE .);
  IF (SOURCEMO IN ("03")); /* processing month 3 audit trail files */
RUN;

PROC SORT DATA = TIMEJUMP;
  BY FPRIMARY LINE_E;
RUN;

DATA TIMETRAVEL;
```

```
  RETAIN TIMETRAVEL LINETIMETRAV 0;
  SET TIMEJUMP;
  BY FPRIMARY;
  LAG_DTE = LAG(DATETIME_E);
  LAG_FPRIMARY = LAG(FPRIMARY); /* this is the CASE id */
  * INITIALIZE FOR NEW CASE;
  IF (FIRST.FPRIMARY) THEN
    DO;
      TIMETRAVEL = 0;
      LINETIMETRAV = 0;
      LAG_DTE = . ;
    END;
  * COMPARE DATETIME STAMP ONLY FOR ENTRIES OF THE SAME CASE;
  * CREATE INDICATOR WHERE TIMEJUMP BACKWARDS ENCOUNTERED THE FIRST TIME;
  ELSE IF ((LAG_FPRIMARY = FPRIMARY) & (LAG_DTE > DATETIME_E)) THEN
    DO;
      TIMETRAVEL = 1;
      LINETIMETRAV = LINE_E;
      OUTPUT TIMETRAVEL;
    END;
RUN;

PROC FREQ DATA = TIMETRAVEL;
  FIGURE SOURCEMO*TIMETRAVEL / MISSING LIST;
  TITLE "FIELD Figure - Cases with Time Jump backwards - 2004m03";
RUN;
```

Figure 8: Identifying cases with "jump back in time"

```
FIELD table - Cases with Time Jump backwards

                        MONTH   TIMETRAVEL   Frequency
                          03         1            27

FIELD table - Cases with Time Jump backwards - 2004m03 - partial listing

 FPRIMARY   MONTH      LINE NUMBER        DATETIME_ENTER              LAG_DATETIME

AAAAAAAA     03            697           05MAR2004:20:04:46         05MAR2004:20:37:07
BBBBBBBB     03            107           06MAR2004:15:46:13         08MAR2004:13:08:04
CCCCCCCC     03             39           10MAR2004:18:04:30         23MAR2004:17:14:18
```

- **Fields where "Soft Edit" message were triggered and the actions taken.**

Query: When interviewers key in a value that is outside a built-in range edit check for that field, or is inconsistent with data in related fields, a pop up "soft edit" is triggered. It prompts the interviewer to see if the keyed value should be changed. The interviewer has the option to accept the original keyed value, suppressing the "soft edit" or to change the value previously entered. At what sections and fields are these soft edit messages being triggered, and what are the actions being taken?

SAS Code: the audit trail keyword corresponding to the soft edits are "ACTION: ERROR SUPPRESS" and “ACTION: ERROR JUMP”. This is an ACTION line, so we can work off the ACTION table, use the appropriate subsetting SELECT statement, and do a PROC FREQ on the resulting data sets to get the counts. See Figure 9.

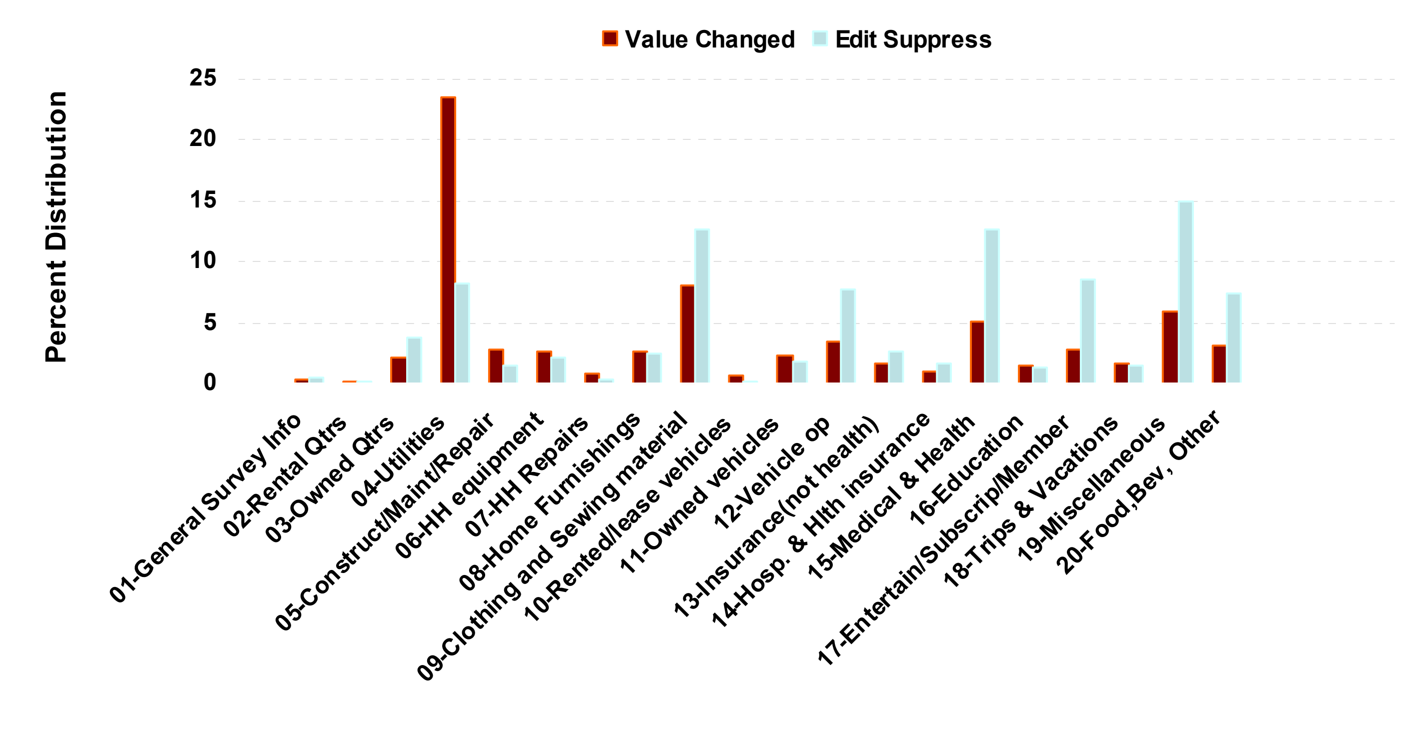
Figure 9. Excerpt of SAS code to analyze actions taken when 'soft edit' triggered

```
DATA SUPPRESSEDIT (DROP = FIELD1 FIELD2)
       ERRORJUMP (DROP = FIELD1 FIELD2);
  LENGTH FIELD1 FIELD2 $ 15;
  SET ATDB.ACTION (WHERE = (CENSID NE '' AND
    ("&FYEAR&FMONTH"<=SOURCEYR||SOURCEMO<="&LYEAR&LMONTH")));
  FIELD1 = SCAN(FIELD,2);
  FIELD2 = REVERSE(SCAN(REVERSE(FIELD),1));
  SECTFIELD2 = SECTION || ' ' || TRIM(FIELD1) || ' ' || TRIM(FIELD2);
  SELECT (ACTION);
    WHEN ("ERROR SUPPRESS") OUTPUT SUPPRESSEDIT;
    WHEN ("ERROR JUMP") OUTPUT ERRORJUMP;
    OTHERWISE;
  END;
RUN;

PROC FREQ DATA = SUPPRESSEDIT NOPRINT;
  FIGURE SECTION / LIST OUT = FREQSECT;
  FIGURE SECTFIELD2 / LIST OUT = FREQFIELD;
RUN;
```

Findings: Suppressions occur most frequently in Sections 19 (Miscellaneous), 09 (Clothing and Sewing materials), and 15 (Medical and Health Expenditures). See Figure 10. The findings from the analysis will help identify possible areas of improvement to the survey such as re-evaluating some of the built-in range edits for section 19, 09, and 15. Values changed in response to the 'soft edit' pop-up confirm that the edit ranges are effective. The audit trails show that value changes occur most in Section 4 (Utilities); this could be partly explained by the unique structure of Section 4 in the CAPI, where responses to utility expenditures from the previous quarter's interview are loaded to the current interview by default, and thus maybe subject to more value changes.

Figure 10: Percent distribution of actions taken on soft edits



Summary

The audit trails contain a wealth of information, but culling information from them had been limited in their native text format. In order to maximize accessibility to the information captured in the audit trails, they were converted into SAS data sets related in a hierarchical structure. The initial investment of resources and time to “SASify” the audit trails into a more structured format have made them more accessible for analyses.