PSID A national study of socioeconomics and health over lifetimes and across generations

Technical Series Paper #16-02

Data-out Experience and Challenges in Blaise 5

Mohammad Mushtaq
Institute for Social Research, University of Michigan

April Beaule
Institute for Social Research, University of Michigan

Funding for this research was provided by the National Science Foundation (SES 1157698).

Data-out Experience and Challenges in Blaise 5

Mohammad Mushtaq and April Beaulé

Abstract: Blaise 5 web interface was used to collect data from Opt-In Pilot 2015. At the time of data collection, a limited number of data-out utilities were available. The data extraction from Blaise 5 became a challenging task. The available data-out options were flat file and xml data format and previously developed tool for Blaise 4 were not working successfully with these options. Using available data-out options in Blaise 5, the data were extracted into Blaise 4 compatible flat file. Then it was imported into SAS, the wide tables had 11664 and 11970 variables from two survey instruments V1 and V2 respectively.

Using knowledge of core interview instruments from prior years, Blaise 4 extraction and reverse engineering method of data transformation, the wide data files were transformed in relational data structure. First, identify interview sections (Housing, Employment, etc.) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop instance into level-one tables from each section (parent table), remove loop-one variables from parent table. Third, some sections had nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this relational transformation, the table structure is matching with PSID 2013. Using same method, the data from All Stars 2016 can be compared with data from PSID (CATI) 2015.
The data-out task would have been much simpler if ASCII-Relational output option in Manipula can be made available sooner so that projects like the PSID can benefit from existing data processing tools.

A second major hurdle that we have to find a solution for with Blaise 5 is identifying questions that are ‘on the route’ but not answered versus questions ‘not on the route’. For self-administered web questionnaires where respondents may just skip a question, it is imperative for data processing that we can easily identify which questions were presented and not answered so that we can properly document the universe of each question in the codebook and generate the appropriate frequencies for valid responses.

1. Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 9,000 U.S. families. Since 2001, the PSID has used Blaise as its main software for its data collection. Due to the size and complexity of the instrument, PSID staff work with tables extracted in looped blocks rather than a single flat file. As the PSID moves to more web-based or mixed mode data collections we have begun to run pilot projects using Blaise 5.

This paper will discuss the challenges we encountered in extracting data from Blaise 5. We also discuss potential future challenges in data handling and documentation for web-based interviews, for which respondents' ability to skip questions creates challenges for variable documentation.

We begin by giving you an overview of specific challenges of Blaise 5 extraction where are goal was to extract in the same format as Blaise 4.8 CATI. Secondly, we will review the issues surrounding determining an acceptable partial interview in the web self interviewing environment. Finally we will discuss the data processing challenges that Blaise 5 self-administered web projects present for final variable handling and documentation.

2. Background

In the early years of the study, the data collection instrument was a paper and pencil questionnaire. In 1992, the PSID automated the survey instrument using SurveyCraft software. By 2001, the Survey Research Center (SRO) at the University of Michigan began using Blaise and the PSID was re-programmed at that time based on the new software. Just as the sample continues to grow, so has the instrument. In 1968, the first year of the PSID, the interview took approximately 20 minutes to administer and the staff released 447 variables. By 2015, the average interview length was 78 minutes and we plan to release 5,493 variables on the Family File.

The PSID is a family level instrument meaning that one respondent reports details about all the individuals in the family. In order to capture individual-level information, we have to set the threshold for array sizes higher than the average family size to allow data collection for larger families. These large array sizes as well as the need for many embedded arrays makes flat extraction unwieldy and cumbersome given the large number of blank positions in most interviews.

In all previous versions of Blaise, we extracted 'relational blocks' using an extraction tool based on Manipula that was designed 'in-house' at the University of Michigan. This has allowed us to handle the data efficiently and keep only rows which contain data while dropping blank rows. However, with our Blaise 5 pilot projects that have launched over the last year we have faced several challenges extracting data in the same format given the limited number of extraction tools available in Blaise 5 at the time of data collection.

3. Blaise 5 Data Extraction and Relational Transformation

At the time of Opt-In Pilot 2015, Blaise 5 Manipula extraction with ASCII-relational output was not available. As previously developed tools did not work with Blaise 5data out options, therefore, data extraction became a challenging task.

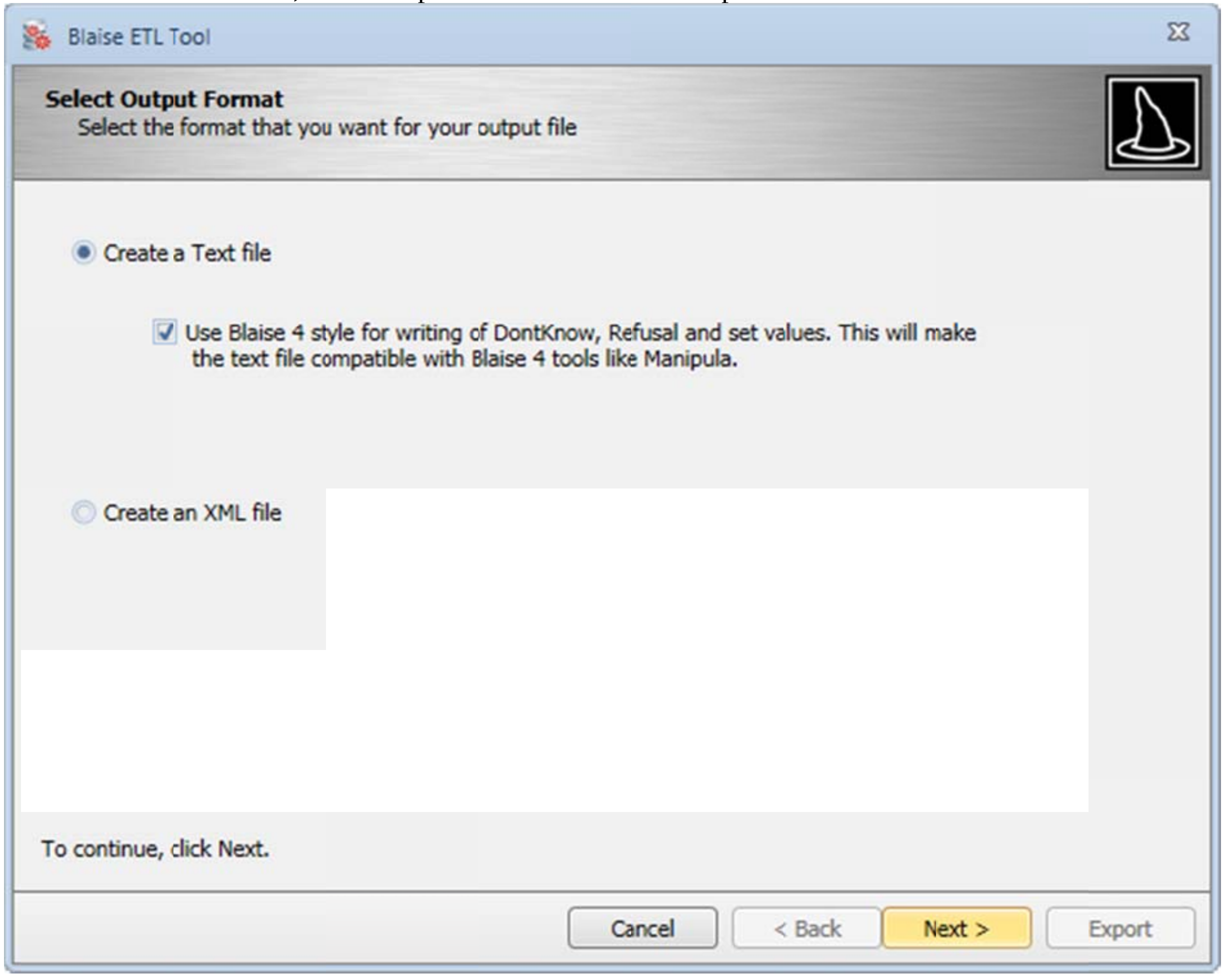
The goal of this task was to create SAS data files which are relational and have same structure as PSID 2015 so that data can easily be compared between two waves of data collection. In order to achieve this goal, wide format extraction and reverse engineering approach were used to transform data into desired relational structure.

Blaise ETL tools (export functions) are available from "Home" and "Data Viewer" tabs. ETL Export function from "Home" tab does not export all cases. It only keeps 400 cases from the database – the first 400, the last 400 or a random set of 400 cases, we don't know.

3.1 Data Extraction: After careful examination of all tabs and available options, below are steps that we have used for successful extraction.

1. Open Blaise 5 database by "Browse Database" option from "Home" tab. With database opened, it also opens "Data Viewer" tab.

2. From "Data Viewer" tab, click "Export Data" and Select Output Format as below.



Select Output Format
Select the format that you want for your output file

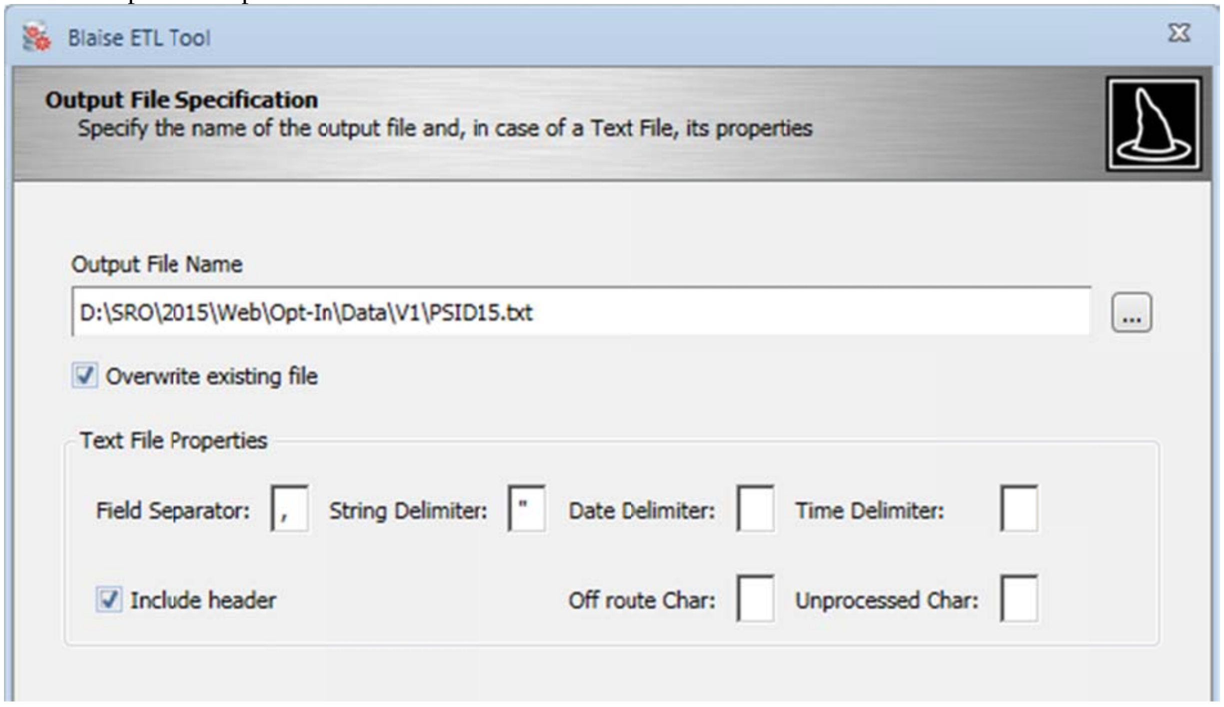
[● Create a Text file]

[✓] Use Blaise 4 style for writing of DontKnow, Refusal and set values. This will make the text file compatible with Blaise 4 tools like Manipula.

○ Create an XML file

To continue, click Next.

3. Select Output File Specification as below and click Next



Blaise ETL Tool

Output File Specification
Specify the name of the output file and, in case of a Text File, its properties

Output File Name
D:\SRO\2015\Web\Opt-In\Data\V1\PSID15.txt

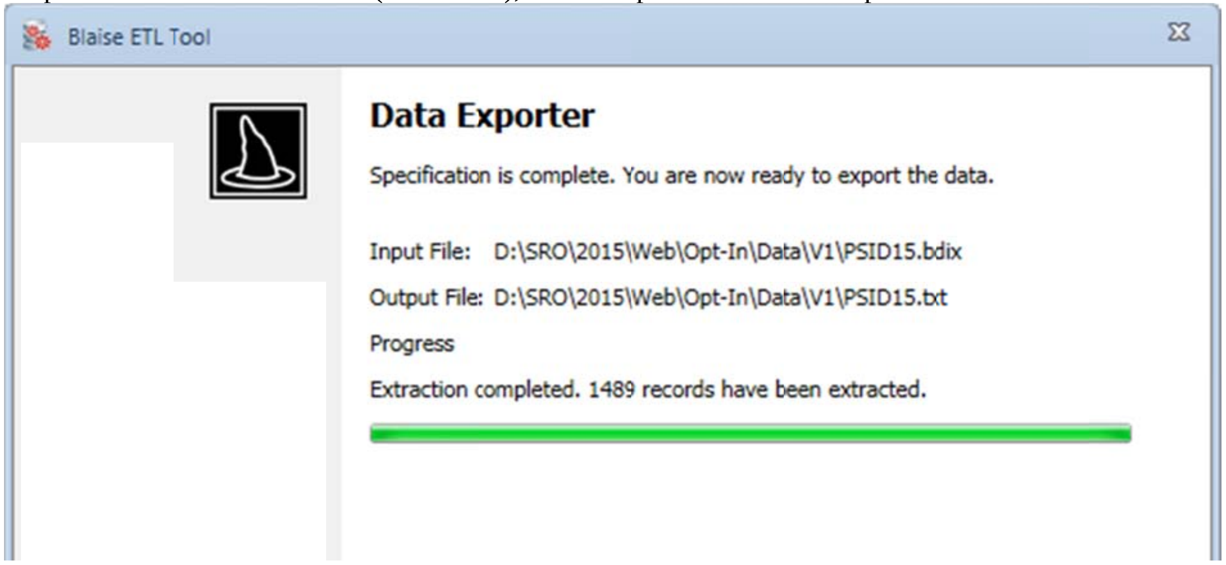
[✓] Overwrite existing file

Text File Properties

Field Separator: , String Delimiter: " Date Delimiter: Time Delimiter:

[✓] Include header Off route Char: Unprocessed Char:

4. Skip Blaise Data Interface File (click Next), select Export from “Data Exporter” tab



Blaise ETL Tool

Data Exporter

Specification is complete. You are now ready to export the data.

Input File: D:\SRO\2015\Web\Opt-In\Data\V1\PSID15.bdix

Output File: D:\SRO\2015\Web\Opt-In\Data\V1\PSID15.txt

Progress

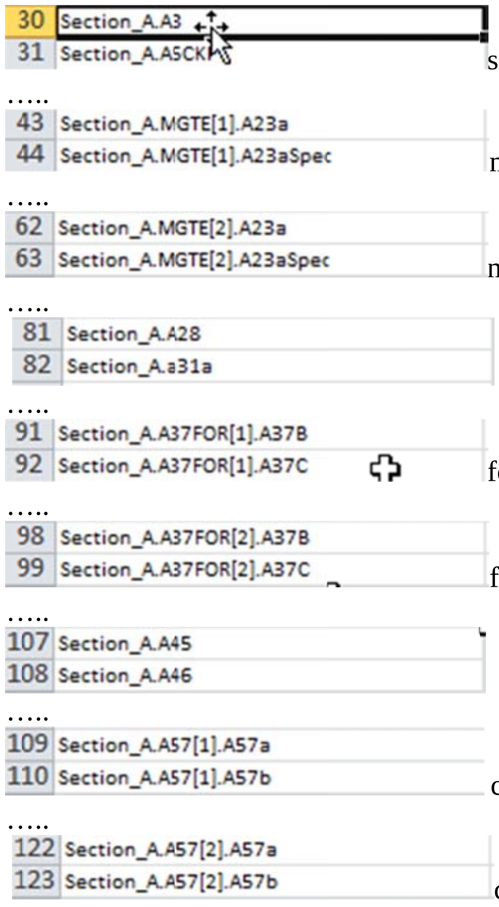
Extraction completed. 1489 records have been extracted.

With these steps, data and open text have been extracted in two files. The wide data file PSID15.txt has 1489 data rows (observations) and 11664 columns (variables). The delimiter is as specified in step #3 above. The first row is column header, which will be processed further to create sections and sub-section from the wide file.

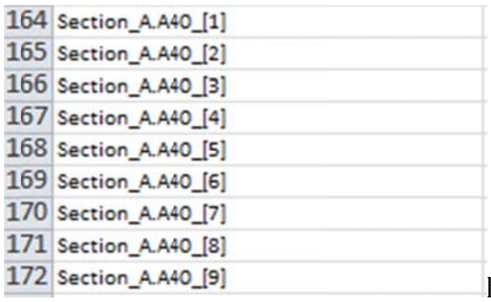
3.2 SAS Transformation: Using knowledge of previous year core interview instruments and reverse engineering method, wide data file was transformed in relational data structure. First, identify interview sections (A, BC, F, G, R, W, P, H, M, KL and J1/J2) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop-one instance into child level tables from parent table, remove loop-one variables from parent table. Third, sections P, H and J has nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this transformation, V1 database has 52 tables and 2554 variables. The resulting table structure is matching with PSID 2015 core, therefore, the relational structure can be used to compare data between CATI 2015. Below are details of transformation process:

1. Create SAS table from wide file, the table has 1489 rows and 11664 columns

2. Identify sections and sub-section from column headers extracted by the extraction process. In wide file, section A has 149 variables, it has three sub-sections – mortgage, foreclosure and computer usage –, each sub-section has two loops.



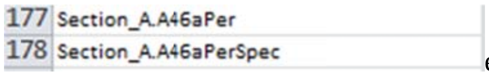
```
30  Section_A.A3
31  Section_A.ASCK                 section A
…..
43  Section_A.MGTE[1].A23a
44  Section_A.MGTE[1].A23aSpec     mortgage instance #1
…..
62  Section_A.MGTE[2].A23a
63  Section_A.MGTE[2].A23aSpec     mortgage instance #2
…..
81  Section_A.A28
82  Section_A.a31a
…..
91  Section_A.A37FOR[1].A37B
92  Section_A.A37FOR[1].A37C       foreclosure instance #1
…..
98  Section_A.A37FOR[2].A37B
99  Section_A.A37FOR[2].A37C       foreclosure instance #2
…..
107 Section_A.A45
108 Section_A.A46
…..
109 Section_A.A57[1].A57a
110 Section_A.A57[1].A57b          computer usage instance #1
…..
122 Section_A.A57[2].A57a
123 Section_A.A57[2].A57b          computer usage instance #2
…..
```



164	Section_A.A40_[1]
165	Section_A.A40_[2]
166	Section_A.A40_[3]
167	Section_A.A40_[4]
168	Section_A.A40_[5]
169	Section_A.A40_[6]
170	Section_A.A40_[7]
171	Section_A.A40_[8]
172	Section_A.A40_[9]

how heated

…..



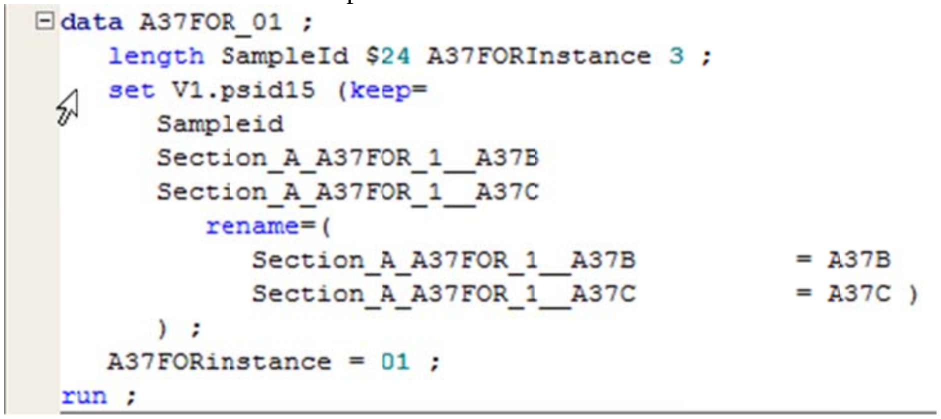
177	Section_A.A46aPer
178	Section_A.A46aPerSpec

end of section A

Section BC has 1040 variables from wide file and several sub-sections, and each sub-section has 10 loops. Sections P, H and J have second level nesting which makes relational conversion more complicated.

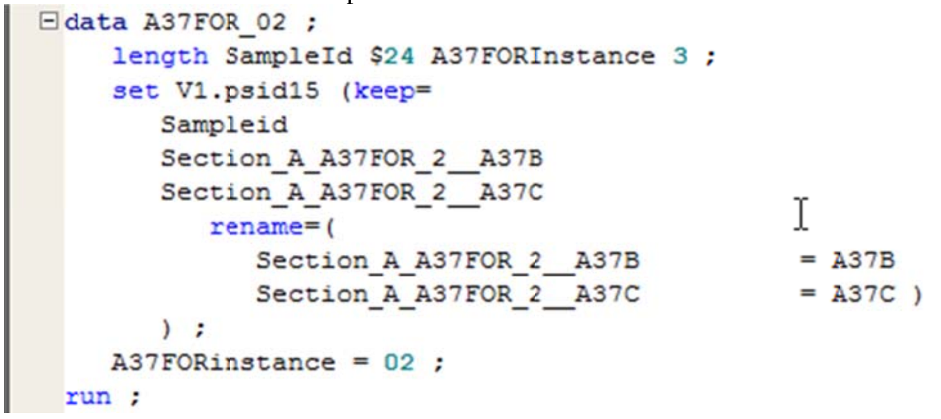
3. From our experience with PSID data, Section A (parent table) has one row per sampleid (primary key), and mortgage, foreclosure and computer usage (child tables) are sub-sections where each row is unique by sampleid and instance number. We used SAS data step programing to write SAS code. It created tables for all loops within each sub-section and then all loops were stacked.

   a. SAS code for foreclosure loop #1



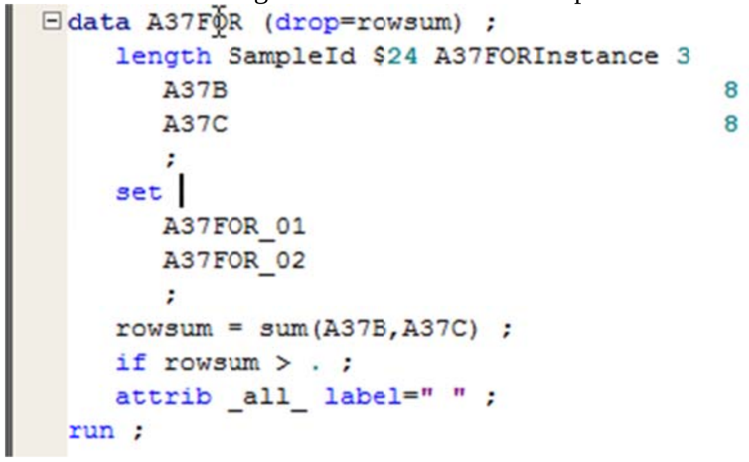
```
data A37FOR_01 ;
   length SampleId $24 A37FORInstance 3 ;
   set V1.psid15 (keep=
      Sampleid
      Section_A_A37FOR_1__A37B
      Section_A_A37FOR_1__A37C
         rename=(
            Section_A_A37FOR_1__A37B          = A37B
            Section_A_A37FOR_1__A37C          = A37C )
      ) ;
   A37FORinstance = 01 ;
run ;
```

   b. SAS code for foreclosure loop #2



```
data A37FOR_02 ;
   length SampleId $24 A37FORInstance 3 ;
   set V1.psid15 (keep=
      Sampleid
      Section_A_A37FOR_2__A37B
      Section_A_A37FOR_2__A37C
         rename=(
            Section_A_A37FOR_2__A37B          = A37B
            Section_A_A37FOR_2__A37C          = A37C )
      ) ;
   A37FORinstance = 02 ;
run ;
```

c. SAS code for stacking data from foreclosure loops



```
data A37FOR (drop=rowsum) ;
   length SampleId $24 A37FORInstance 3
      A37B                              8
      A37C                              8
      ;
   set
      A37FOR_01
      A37FOR_02
      ;
   rowsum = sum(A37B,A37C) ;
   if rowsum > . ;
   attrib _all_ label=" " ;
run ;
```

4. The above example is from two loops and two variables and output table is one level below the parent table. The process becomes more complicated when output tables are two steps deeper and/or loops are significantly large number.

3.3 Missing data:

1. Numeric variables from loops as specified in Blaise data-model, can assume numeric or characters data type in SAS, when delimited file is read into SAS using data step and “dsd” option. In wide file, each instance is separate set of variables. If a numeric variable has missing data on the first row then its data type is set as character. Therefore, same variable from different instances have mixed data types and the stacking process can’t stack instance tables successfully.
2. Characters variables from different instances can be of different length. If instance-1 variable has smaller length from other variables, then stacking process is going to need additional adjustment, else data will be truncated in the stacking process.

In the absence of ASCIIRelational output options from Blaise5 and Manipula, the data extraction task from Blaise 5 database needed lots of resources, good understanding of PSID questionnaire from previous waves and excellent data management skills in SAS.

4. Determining a completed interview

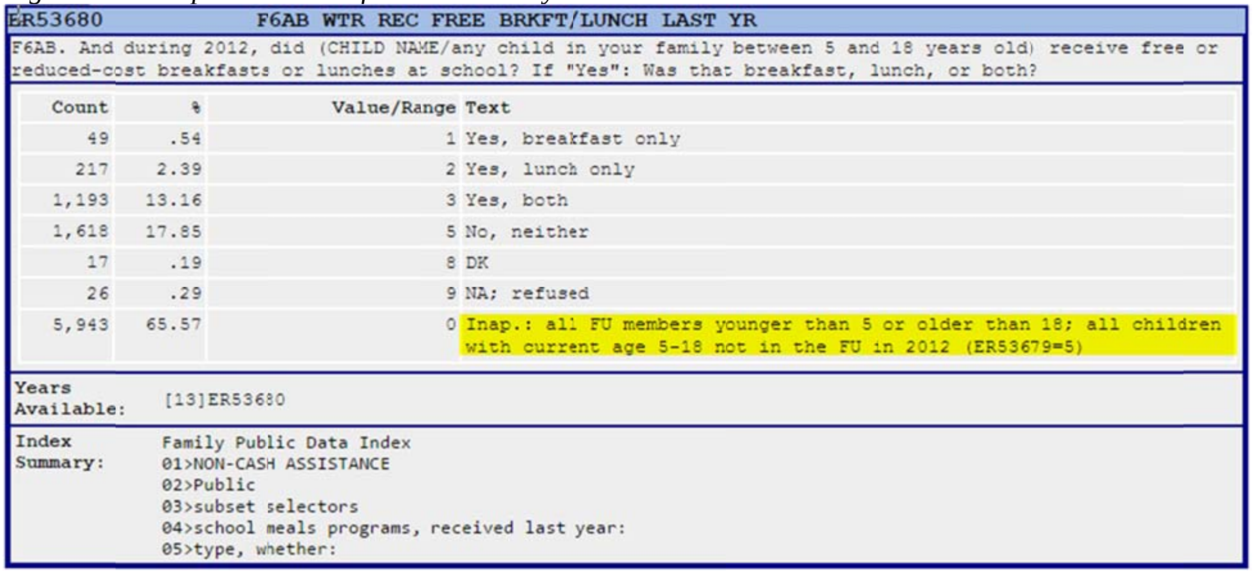
The PSID has started completing supplemental interviews to the main study by offering respondents mixed mode data collection using primarily a Web/PAPI protocol. The web collected records pose several challenges for the data processing team. The first main challenge is the handling of partially completed interviews. In CATI interviews the interviewer codes an interim or final disposition based on whether an interview has gone far enough to be considered a partial. In contrast, with self-administered web interviews we do know when a form has been submitted by the respondent but we do not have a sense of its ‘completeness’. For those cases where an interview has been started but not yet submitted we are also unsure about its status and about whether the respondent will go back and make further attempts to complete it. These ambiguities make it more difficult to monitor data collection while it is in the field and add extra burden for data processing staff to adjudicate which cases we will accept as completed interviews for the final dataset.

In CATI mode, the interviewer develops a sense of item-level refusals and may in fact suspend the interview or code the case out a refusal if the respondent refuses to answer a large proportion of questions. Alternatively, interviewers alert the processing team of problematic cases by triggering a flag in the observation section of the interview. These interviewer-initiated triggers for further scrutiny are not possible in self-administered web surveys and alternative approaches are required to deem an interview an acceptable completed case: For web self-administered surveys, the Data Processing (DP) team works closely with the lead researcher to determine an acceptable level of item nonresponse based on the forms submitted by the respondent that are automatically coded as complete by the sample management system. Generally we may look at the entire sample and determine the average number of variables with non-missing values to determine the acceptable threshold. This approach presents challenges as instruments increase in complexity and the number of variables ‘on-route’ or ‘off-route’ (see below) varies significantly between interviews. In many situations, cases with questionable ‘completeness’ have to be reviewed on a case by case basis, which is often time-consuming.

5. Problem of determining questions ‘on-route’ vs ‘off-route’

One of the most basic fundamentals in data documentation is identifying the universe of each variable. In the PSID, we have long documented the inverse of the universe in our codebooks *(Figure A).* The INAP (Inappropriate code) lists the criteria the respondent met in order to skip this particular question.

Figure A: Example Codebook for PSID Family Level Variable in CATI



ER53680 F6AB WTR REC FREE BRKFT/LUNCH LAST YR

F6AB. And during 2012, did (CHILD NAME/any child in your family between 5 and 18 years old) receive free or reduced-cost breakfasts or lunches at school? If "Yes": Was that breakfast, lunch, or both?

Count	%	Value/Range	Text
49	.54	1	Yes, breakfast only
217	2.39	2	Yes, lunch only
1,193	13.16	3	Yes, both
1,618	17.85	5	No, neither
17	.19	8	DK
26	.29	9	NA; refused
5,943	65.57	0	Inap.: all FU members younger than 5 or older than 18; all children with current age 5-18 not in the FU in 2012 (ER53679=5)

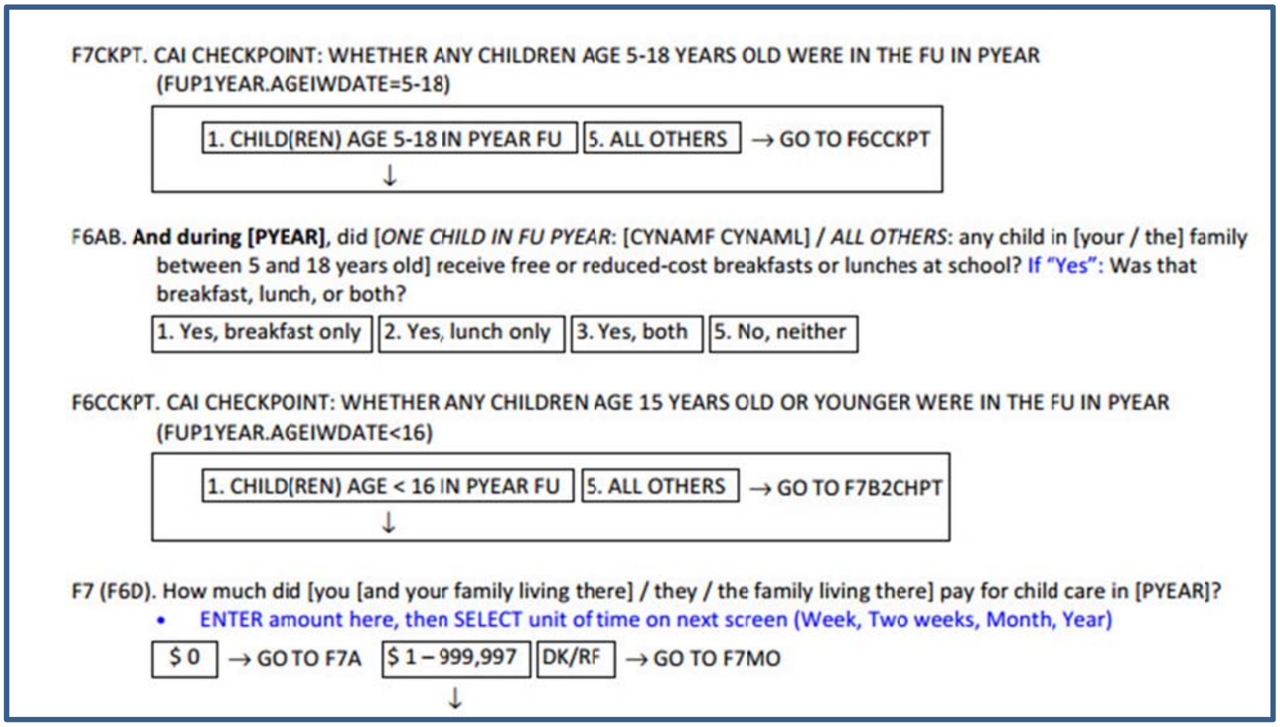
Years Available: [13]ER53680

Index Summary: Family Public Data Index
01>NON-CASH ASSISTANCE
02>Public
03>subset selectors
04>school meals programs, received last year:
05>type, whether:

In a CATI interview, we can determine which questions are skipped easily because all questions that are ‘on-route’ (i.e. those that are presented based on pathing) must be answered by the interviewer even if the response is ‘Don’t Know’ or ‘Refused’. In contrast, in self-administered web interviewers the general norm is to simply let the respondent pass a question without an answer. Similarly, ‘Don’t Know’ or ‘Refused’ are not explicit options available to the respondent at all as the aim is to encourage the respondent to provide a valid (non-missing response).

For web data-out in Blaise 5 there is no distinction between variables with a system missing code because that variable was ‘off-route’ (i.e. those variables that are never presented based on pathing) versus a system missing code because the variable was ‘on-route’ but the respondent did not provide a response but instead just skipped the question. Determining the difference between these two scenarios is imperative once we move to the documentation stage that describes the universe of those who did not receive this question because of routing. Because this critical distinction is not currently a feature that we have, the Data Processing (DP) team must essentially re-write all the skip patterns in the Blaise instrument using cleaning/analysis software like SAS. The DP team members use the ‘Box and Arrow’ *(Figure B)* or visual documentation of the questionnaire often written in a Word document to determine all the possible skip patterns and assign a unique code to questions that are empty because they were ‘off-route’.

Figure B: Example PSID Box and Arrow Documentation of CATI Questionnaire



F7CKPT. CAI CHECKPOINT: WHETHER ANY CHILDREN AGE 5-18 YEARS OLD WERE IN THE FU IN PYEAR (FUP1YEAR.AGEIWDATE=5-18)

1. CHILD(REN) AGE 5-18 IN PYEAR FU ↓ | 5. ALL OTHERS → GO TO F6CCKPT

F6AB. **And during [PYEAR]**, did [*ONE CHILD IN FU PYEAR*: [CYNAMF CYNAML] / *ALL OTHERS*: any child in [your / the] family between 5 and 18 years old] receive free or reduced-cost breakfasts or lunches at school? If "Yes": Was that breakfast, lunch, or both?

1. Yes, breakfast only | 2. Yes, lunch only | 3. Yes, both | 5. No, neither

F6CCKPT. CAI CHECKPOINT: WHETHER ANY CHILDREN AGE 15 YEARS OLD OR YOUNGER WERE IN THE FU IN PYEAR (FUP1YEAR.AGEIWDATE<16)

1. CHILD(REN) AGE < 16 IN PYEAR FU ↓ | 5. ALL OTHERS → GO TO F7B2CHPT

F7 (F6D). How much did [you [and your family living there] / they / the family living there] pay for child care in [PYEAR]?

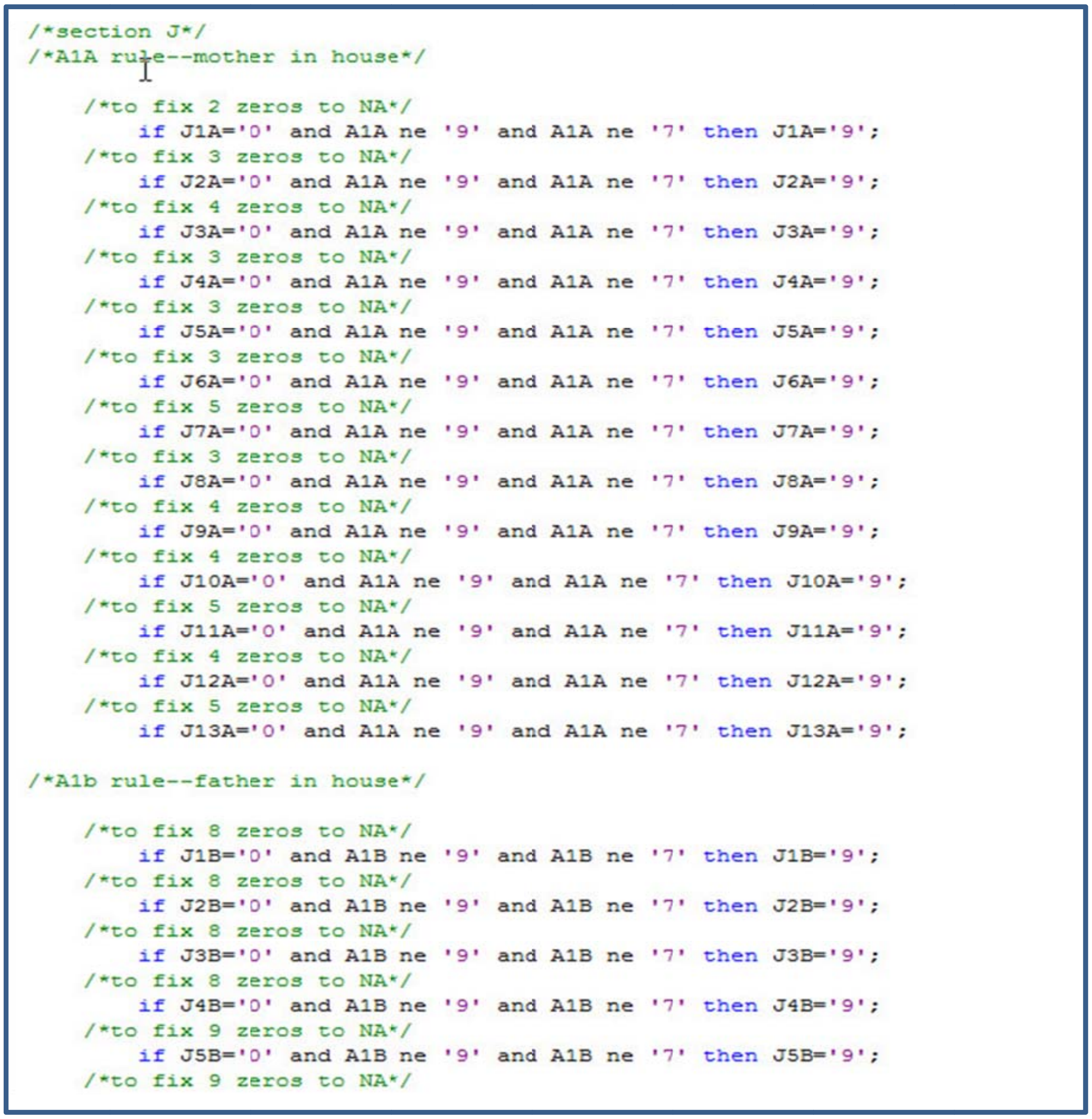
- ENTER amount here, then SELECT unit of time on next screen (Week, Two weeks, Month, Year)

$ 0 → GO TO F7A | $ 1 – 999,997 ↓ | DK/RF → GO TO F7MO

For basic questionnaires with few skip patterns, this is not unduly burdensome. However, if we plan to attempt data collection on the web with more complex instruments containing many loops, arrays, and embedded arrays this becomes ever more challenging for the SAS programmer and error prone. Unlike the Blaise programming phase, which includes many rounds of testing by several testers, the SAS programmers do not have the same amount of time or testing resources to check their programs and output for integrity.

Writing the logic of the entire questionnaire by both the Blaise programmer and then again by the SAS programmer is also extremely inefficient and may affect data integrity if the SAS programmer makes an error in determining the correct skip flow *(Figure C)*. It also may offset some of the presumed cost savings of having the respondent rather than the interviewer complete the questionnaire. Since the Blaise system knows which questions were 'on-route' vs 'off-route' at the time of the interview based on the data model version, it appears there should be a way to communicate those differences in the storage and extraction of data from the bdbx automatically.

Figure C: Example SAS code for Recoding Skipped Variables to Not Ascertained (NA) = 9



```
/*section J*/
/*A1A rule--mother in house*/

    /*to fix 2 zeros to NA*/
        if J1A='0' and A1A ne '9' and A1A ne '7' then J1A='9';
    /*to fix 3 zeros to NA*/
        if J2A='0' and A1A ne '9' and A1A ne '7' then J2A='9';
    /*to fix 4 zeros to NA*/
        if J3A='0' and A1A ne '9' and A1A ne '7' then J3A='9';
    /*to fix 3 zeros to NA*/
        if J4A='0' and A1A ne '9' and A1A ne '7' then J4A='9';
    /*to fix 3 zeros to NA*/
        if J5A='0' and A1A ne '9' and A1A ne '7' then J5A='9';
    /*to fix 3 zeros to NA*/
        if J6A='0' and A1A ne '9' and A1A ne '7' then J6A='9';
    /*to fix 5 zeros to NA*/
        if J7A='0' and A1A ne '9' and A1A ne '7' then J7A='9';
    /*to fix 3 zeros to NA*/
        if J8A='0' and A1A ne '9' and A1A ne '7' then J8A='9';
    /*to fix 4 zeros to NA*/
        if J9A='0' and A1A ne '9' and A1A ne '7' then J9A='9';
    /*to fix 4 zeros to NA*/
        if J10A='0' and A1A ne '9' and A1A ne '7' then J10A='9';
    /*to fix 5 zeros to NA*/
        if J11A='0' and A1A ne '9' and A1A ne '7' then J11A='9';
    /*to fix 4 zeros to NA*/
        if J12A='0' and A1A ne '9' and A1A ne '7' then J12A='9';
    /*to fix 5 zeros to NA*/
        if J13A='0' and A1A ne '9' and A1A ne '7' then J13A='9';

/*A1b rule--father in house*/

    /*to fix 8 zeros to NA*/
        if J1B='0' and A1B ne '9' and A1B ne '7' then J1B='9';
    /*to fix 8 zeros to NA*/
        if J2B='0' and A1B ne '9' and A1B ne '7' then J2B='9';
    /*to fix 8 zeros to NA*/
        if J3B='0' and A1B ne '9' and A1B ne '7' then J3B='9';
    /*to fix 8 zeros to NA*/
        if J4B='0' and A1B ne '9' and A1B ne '7' then J4B='9';
    /*to fix 9 zeros to NA*/
        if J5B='0' and A1B ne '9' and A1B ne '7' then J5B='9';
    /*to fix 9 zeros to NA*/
```

6. Conclusion and Summary

Using knowledge of core interview instruments from prior years, Blaise 4 extraction and reverse engineering method of data transformation, the wide data files from Blaise 5 were transformed into relational data structure. First, identify interview sections (Housing, Employment, etc.) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop instance into level-one tables from each section (parent table), remove loop-one variables from parent table. Third, some sections had nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this relational transformation, the table structure is matching with PSID 2015.

It is likely that many large surveys like the PSID will continue to explore data collection via the web given cost-cutting pressures from funders as well as increasing respondent coverage as more and more families gain access to computers and the internet. The expectation from both funders and the user community is that data collected via the web will have the same standards of cleaning and documentation as data collected via CATI.

The current tools offered in Blaise 5 do not allow us to easily check and assign appropriate codes for ‘on-route’ but not answered versus ‘off-route’. This is a significant drawback for data processing and documentation. If tools could be provided to automatically assign fields with this distinction then we could also build on that information to help us come up with algorithms for determining ‘completeness’ of a web interview. Instead of case-by-case checking of completeness we could flag outliers that have more than the average share of missing responses by dividing the number of presented but skipped questions by the number of overall questions that were ‘on-route’.