Paper CI-02-2013

Introducing PROC PSYCHIC

David J. Corliss

Magnify Analytic Solutions, Detroit, MI

ABSTRACT

Here at MWSUG, we are pleased to present some of the functionality of the upcoming procedure PSYCHIC. While this procedure has been somewhat delayed and a release date is not set at this time, much of the functionality of PROC PSYCHIC is available in the most recent release of SAS®; this paper will focus on currently available capabilities, processes and techniques. Features of this data integration master procedure include interrogation of data to determine errors and the design of data models best suited to the data itself despite the best intentions of helpful managers. While the brain wave text mining feature remains in beta in the upcoming release, recommendations are made to replicate this functionality using voice-based communication to better identify customer needs and wants in the usual absence of clear direction.

Keywords: Data Integration, Best Practices

INTRODUCTION

Data integration best practices in SAS have been developed by many different contributors; a recommended reading is given at the end of this paper. The development of a database through the integration of data from multiple sources demands the user to establish project requirements, develop a dictionary and data model, create match keys to connect different data sets and clean the data. PROC PSYCHIC has been developed to address these important considerations by combining features from multiple strategies in to a single procedure.

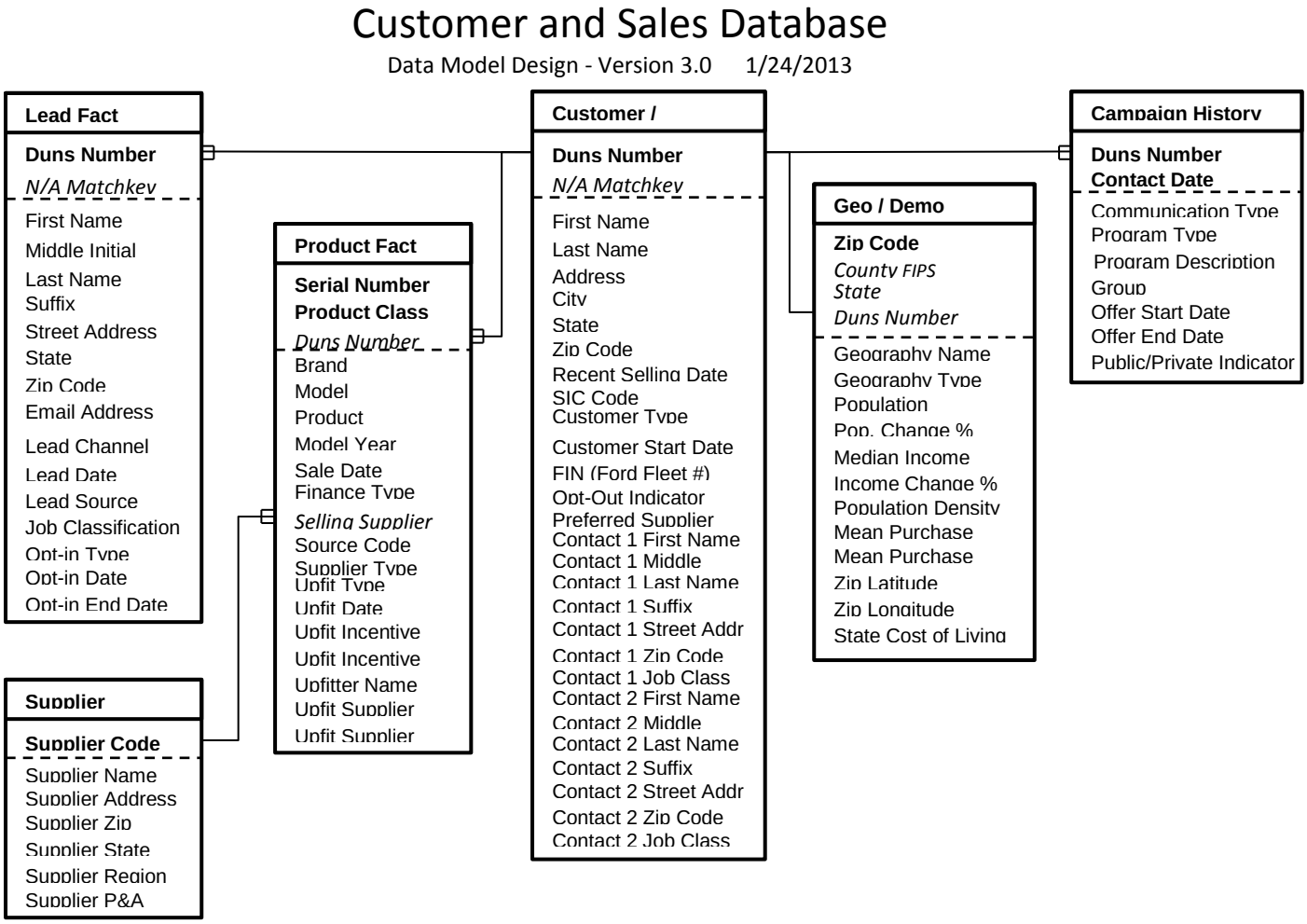
PROJECT REQUIREMENTS

The first step in data integration is the one least commonly performed: the establishment of concise project requirements. At the time of publication of this paper, the brain-wave text mining feature developed for PROC PSYCHIC remains classified by the NSA; if this option is not functional in the initial release of the procedure, it will continue to be necessary to ask a customer or manager the following standard questions:

- Has a written data model been established? In this context, “written data model” refers to a diagram with table names connected to each other by named match keys. The typical answer to this question is no; if so, data sources should be listed.
- Is there a data dictionary? The answer distribution function is similar to the data model question.
- What match keys recommended to connect different types of data to one another?
- Text in numeric fields and numbers in text fields: what fields look like numbers but should be treated as text? Zip codes are an example, with leading zeros required and distinct intelligence in different parts of the field.
- What are the known problems in specific data fields?
- Age and volatility: how long ago was the data last updated? How often should new updates be implemented?

DATA DICTIONARY AND DATA MODEL

Example of a Data Model plot output by PROC PSYCHIC, created by plotting the results of SAS dataset interrogation performed by the procedure



The DATAMODEL option of PROC PSYCHIC operates by interrogating all of the SAS datasets in the library specified in the LIB statement, adding the plot to the list of all fields in all library data members given today by PROC CONTENTS. This listing also provides a basic data dictionary, which can then be augmented as needed.

```
proc contents data=libname._ALL_ memtype=data out=lib_contents noprint;
run;
```

MATCH KEYS

Match keys normally will have the same or similar names in multiple tables. Before PROC PSYCHIC, it was necessary to identify such fields by performing a fuzzy merge on the data dictionary by finding the cross-product of the variable names in each table and using the COMPGED text function to find fields with similar names. This SQL and COMPGED process is discussed in detail by Staum (2007). For example, the macro below creates a draft data dictionary from the output of PROC CONTENTS with the _ALL_ option and then searches for possible match keys by identifying fields found on multiple SAS datasets with the same or similar names. Candidate match keys are required to have a LENGTH of at least 4, eliminating flags and indicators from the candidate list. When PROC PSYCHIC is released, this will all be automatic.

```
%macro libinfo(lib);

proc contents data=&lib.._ALL_ memtype=data out=&lib..dictionary noprint;
run;

proc sort data=&lib..dictionary out=work.mem_list (keep=memname) nodupkey;
   by memname;
run;

proc sql;
   create table match_keys as
   select upper(a.name) as name1, upper(b.name) as matchkey_name,
   compged(a.name,b.name) as text_distance
   from &lib..dictionary a, &lib..dictionary b
   where a.length ge 4 and calculated text_distance le 150
   order by text_distance;
quit;

proc sort data=match_keys (keep=name1) nodupkey;
   by name1;
run;

%mend libinfo;
```

DATA REVIEW

To facilitate the data review process, samples are created using the same functionality now available in PROC SURVEYSELECT. A default of 1,000 records are pulled for testing each data set, allowing for faster testing of large data sets. A loop construction captures the name of each SAS data set from the data dictionary and a SURVEYSELECT is run on each member of the SAS library.

```
%macro sample(lib);

**** Count the number of SAS data sets in the library ****;


data _null_;
   set work.mem_list end=lastobs;
   if lastobs = 1 then call symput('mem_count',_n_);
run;


%do i=1 %to &mem_count.;

**** Identify the name of a library member ****;
```

```
   data _null_;
      set work.mem_list;
         if &i. = _n_ then do;
            call symput('mem',memname);
         end;
   run;

**** SURVEYSELECT performs a Simple Random Select (method=SRS) of 1,000 records ****;

   proc surveyselect data=&lib..&mem. method=SRS sampsize=1000 seed=12345
out=sample_&mem.;
     id _all_;
   run;

%end;


%mend sample;
```

The sample from each data set is tested for fields that are largely missing. The names of variables with a default value of 80% or more missing are written to a SAS data set. This is done using the same loop structure as before, operating at the level of individual variables.

```
%macro miss(lib);

data missing;
   set &lib..dictionary (obs=0);
   keep name;
run;

data _null_;
   set &lib..dictionary end=lastobs;
   if lastobs = 1 then call symput('var_count',_n_);
run;

%do i=1 %to &var_count.;

   data _null_;
      set &lib..dictionary;
      if &i. = _n_ then do;
         call symput('mem',memname);
         call symput('var',name);
         call symput('var_type',type);
      end;
   run;
   %put &mem. &var.;

   %if &var_type. = 1 %then %do;
      data test;
         set work.sample_&mem.;
         count = _n_;
         if _n_ = 1 then miss_count = 0;
         if &var. = . then miss_count + 1;
         retain miss_count;
      run;
   %end;
```

```
    %if &var_type. = 2 %then %do;
       data test;
          set work.sample_&mem.;
          if _n_ = 1 then miss_count = 0;
          if &var. = '' then miss_count + 1;
          retain miss_count;
       run;
    %end;

    data test;
       set test end=lastobs;
       if lastobs = 1;
         length memname $32;
         memname = "&mem.";
         name = "&var.";
       miss_rate = miss_count / _n_;
         if miss_rate ge 0.6;
         keep memname name miss_rate;
    run;

    data missing;
       set missing test;
    run;

%end;


%mend miss;
```

DATA REVIEW

A number of effective data cleansing routines are available, each with their own strengths. Accordingly, PSYCHIC does not attempt to recreate them, leaving the particular choice to the user. Cody (1999) and Fehd (2004) have published data cleansing macros using PROC FORMAT with the `other = invalid` option. Cody (1999), Solak (2009) and others offer routines for identifying outliers in numeric data. Experience with the challenges of the particular data in hand will best guide with routines to use.

SUMMARY

While the final release of PROC PSYCHIC has release has been delayed and a date is not established at this time, additional functionality will become available in subsequent releases. Until such time that SAS capabilities become psychic, established data integration best practices will provide much of the desired functionality:

- Ask questions, such as those given in the Project Requirements section of this paper, and get answers before moving forward
- Identify data sources, variables and match keys. Compile a list of known problems in specific fields
- Create a written data model and review it with all stakeholders to find out what they didn't tell you in the first place
- Create a Data Dictionary, including formats, invalid values, how null values are represented and the update cycle of the variables

- Test all variables for missing values, invalid formats and outliers and correct any problems
- Use the match rates between different tables to test the performance of match keys

READING LIST

Cody, R, “Cody’s Data Cleaning Techniques Using SAS Software Cary, NC: SAS Institute Inc.”, 1999. 226 pp.

Fehd, R., ‘Invalid: a Data Review Macro - Using PROC Format Option Other=Invalid to Identify and List Outliers’, PharmaSUG 2004 http://www.lexjansen.com/pharmasug/2004/datamanagement/dm06.pdf

Solak, M., ‘Detection of Multiple Outliers in Univariate Data Sets’, PharmaSUG 2009 http://www.lexjansen.com/pharmasug/2009/sp/sp06.pdf

Staum, P., ‘Fuzzy Matching using the COMPGED Function’, NESUG 2007 http://www.nesug.org/proceedings/nesug07/ap/ap23.pdf

ACKNOWLEDGMENTS

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David J Corliss
Magnify Analytic Solutions
1 Kennedy Square, Suite 500
Detroit, MI 48224
Phone: 313.202.6323
Email: dcorliss@magnifyas.com