Paper 158-31

# SAS® Programs for Extracting Data from LexisNexis Documents

Robert S. Matthews, University of Alabama at Birmingham, Birmingham, Alabama
Thomas J. Bender, PhD, University of Alabama at Birmingham, Birmingham, Alabama

## ABSTRACT

Text files often contain data for multiple variables that are distinguished from one another by consistently used text strings that label and organize the data. Devising a programmatic solution for extracting these data from text files can require a significant time investment that may only be justified when the number of files is quite large. The approach we used to extract address information from over 100,000 text files may serve as an instructive and time-saving example for SAS users who desire a similar solution that ensures faster throughput and greater reliability than manual data extraction.

To develop residential histories for subjects in a retrospective follow-up study of cancer incidence, we sought address information for each subject from the LexisNexis National Group Files database. We downloaded the search results from the LexisNexis website as Rich Text Format (RTF) files. The file for each subject contained one or more documents of several types, each with its own characteristic set of data delimiting text strings. This paper describes two programs that (1) read an entire RTF file into a single character string and (2) extract data for variables identified by text strings into a SAS dataset.

## INTRODUCTION

The National Group Files database provided by LexisNexis has a variety of document types, and a subset of these (e.g. P-SRCH, P-TRAK, DCEASE) are associated with Social Security Number (SSN). In searching for each of our study subjects by SSN, we downloaded the returned documents in RTF files. We divided our study group (over 100,000 individuals) into batches containing several thousand subjects each, and we saved the downloaded RTF files, named by SSN, into separate batch directories.

A LexisNexis search for a given SSN may return one or more documents, and each document typically contained an individual's name, month and year of birth, current address, previous address, and/or telephone number. DCEASE documents, derived from the Social Security Death Master file, contained name, death date, and state of death. Comparing name and birthdate information in the documents with information from employment records, we assigned a matching score to each document indicating the likelihood that it pertained to the study subject of interest. Manual review of documents assigned matching scores employing less stringent matching criteria (e.g. first and last name but not birth year) serves to eliminate those documents that do not in fact describe a study subject.

Resulting data can then be downloaded into one of several different formats; we have found Rich Text Format (RTF) to be the most desirable type and the rest of this paper describes two programs that read the resulting RTF files and extract data fields with their associated values from each file.

## DETAILS

We present a pair of programs to convert our collection of downloaded RTF files into a SAS dataset for further analysis. Program 1 (see Appendix) produces a list of all RTF files located in a given batch directory, reads each file, removes all of the embedded RTF formatting tags, and creates a SAS dataset that has a separate observation for every line of text in each RTF file. Program 1 has two major DATA steps, the first of which takes a directory path as input and reads all of the RTF files contained in the specified directory. Each file is output as a separate observation and stored in a temporary dataset. The second DATA step standardizes the SSN filenames for each observation and removes the embedded RTF formatting tags from the textual data, such that each observation contains a clean line of text stored in a single character string variable.

Program 2 (see Appendix) splits observations for each RTF file (or equivalently each SSN) into separate segments for each document, extracts data for name and birthdate variables identified by text strings, and then assigns a matching score to each segment. The program's matching score algorithm compares identifying information from LexisNexis documents to name and birthdate information from employment records and yields 34 possible nonzero scores, where a higher score reflects the fulfillment of progressively less stringent sets of matching criteria. To aid comparison, we eliminate embedded characters (e.g. hyphens, apostrophes, etc.) from name fields in both LexisNexis documents and employment records. We strive to retain as many potentially matching segments as possible by tailoring the program to a variety of inaccuracies commonly found in LexisNexis records. For example, the matching score algorithm checks for transposed names, considers other formatting problems, and employs the soundex function to accomodate spelling discrepancies. A matching score of zero indicates that the identifying information for a document does not fulfill even the least stringent set of matching criteria.

## CONCLUSION

We have used these programs extensively for several studies. Auxiliary programs facilitate the task of automatically searching lists of SSNs in the LexisNexis database. A collection of SAS/AF® screens link and display all study information for subjects on a single screen and aid research staff review of LexisNexis documents with specific matching scores. We have created additional programs that extract addresses, telephone numbers, and associated dates from matching LexisNexis documents.

## ACKNOWLEDGEMENTS

Robert would like to thank Dr. Elizabeth Delzell for initiating and directing the studies for which these programs were designed, for encouraging me to write and present papers for SAS conferences, and for motivating me to constantly challenge myself with new and innovative tasks.

Thomas is similarly grateful to Dr. Delzell for her insights and guidance. This work constituted part of his doctoral thesis project, which was supported by the International Business Machines Corporation and the Medical Scientist Training Program at the University of Alabama at Birmingham.

**Additional Information**

For more information about these programs, please contact the authors at the addresses listed below.

Robert Matthews
University of Alabama at Birmingham
Department of Epidemiology
1665 University Blvd. RPHB 517C
Birmingham, AL 35294-0022
rsm@uab.edu
www.rsm-photography.com

Thomas John Bender, PhD
University of Alabama at Birmingham
Department of Epidemiology
1665 University Blvd. RPHB 523B
Birmingham, AL 35294-0022
bender@uab.edu

APPENDIX I

```
*************************************************************************
*  Univ. of Alabama at Birmingham - Occupational Epidemiology Group  *
*  Research project                                                  *
*************************************************************************;
options pagesize=74 linesize=150 pageno=1 missing=' ' date ;
* Programmer    : Robert Matthews
footnote2 "%sysfunc(datetime(),datetime14.)";
title1 '--- Research Study ---';
*************************************************************************;

%let filetype=.RTF;  * File type to search for;

* Create a dataset containing a list of all the RTF files in a particular
directory;
data read_directory ;
   path = "g:\research\data\lexis\";
   rc   = filename('mydir',path);
   dsid = dopen('mydir');    Open directory specified in PATH above;
   if dsid = 0 then
      put 'ERROR: Directory could not be opened: ' path=;
   else
      do;
         length filename $80 ;
         memcount  = dnum(dsid);    * # of directory entries;
         filecount = 0;             * Initialize file counter;
         do i=1 to memcount;        * Read all directory members;
            filename = dread(dsid,i);
            if index(upcase(filename), "&filetype") then
               do;
                  filecount+1;
                  fullname = path || filename;
                  output;
            end;
         end;
         rc = dclose(dsid);
         put "Total &filetype files read: " filecount;
      end;

      drop i rc dsid memcount;
run;

* Read specified RTFs, one line at a time, and remove all RTF tags;
data read_files(compress='yes');
   retain prev_line;
   length line bslash prev_line $300 ssn $9 ;
   set read_directory(keep=fullname filename path);

   * if filename is in the form of 451_PRE_2_0000_W_SEG_401_96_.rtf
     then convert it to a 9 digit social security number format;
   if index(filename,'PRE') then
      ssn = substr(filename,1,3) ||
            substr(filename,26,2) ||
            substr(filename,11,4);
   else
```

```
      ssn = substr(compress(filename,'-'), 1,9);

   header = 2;
   prev_line = '';

   infile in end=finished filevar=fullname;

   * Open file, read all records, then close it and get the next one;
   do while (not finished);
      input;
      line = _infile_;
      if header = 2 & index(line,'\header') then header = 1;
      if header = 1 & index(line,'}')       then header = header - 1;
      if header = 0 & index(line,'\header') then header = 2;
      if index(line,'\') then
         do;
            repeatnum = length(line) - length(compress(line,'\'));
            do i = 1 to repeatnum;
               slash_pos = index(line,'\');
               if slash_pos then
                  do;
                     bslash = substr(line, slash_pos);
                     space  = index(bslash,' ');
                     if slash_pos = 1 then
                        line = substr(line, slash_pos + space);
                     else
                        line = substr(line,1, slash_pos - 1) ||
                               substr(line, slash_pos + space);
                  end;
               bslash = '';
               space = 0;
            end;
         end;

      * output individual lines;
      if line = '' then
         if prev_line ne '' & header = 0 then
            do;
               output;
               prev_line = line;
            end;
         else;
      else if index(line,'THIS DATA IS FOR') = 0 &
              indexc(line,'{}') = 0 & header = 0 then do;
            output;
            prev_line = line;
         end;

      if header = 3 then prev_line = '';
   end;

   drop i repeatnum header bslash prev_line space;
run;

data lexis_results(label='1st pass of Lexis RTF files' compress='yes');
   set read_files(where=(line ne ''));
run;
```

APPENDIX II

```
*************************************************************************
*  Univ. of Alabama at Birmingham - Occupational Epidemiology Group  *
*  Research project                                                  *
*************************************************************************;
options pagesize=74 linesize=150 pageno=1 missing=' ' date;
* Programmer    : Robert Matthews
footnote2 "%sysfunc(datetime(),datetime14.)";
title1 '--- Research Study ---';
*************************************************************************;

%macro namesplt;
   x = index(name,',');

   if x=0 then
      put 'WARNING: **** No comma in NAME field!  ' ssn= name= segnum=;
   else
      do;
         last   = compress(substr(name,1,x-1), "'`-,.");
         lname1 = scan(last,1,' ');
         sfx1   = scan(last,2,' ');
         if sfx1 ^in ('JR','SR','III','II','IV','J','DR','RN') then
            do;
               sfx = ' ';
               lname = compress(last,' ');
            end;
         else
            do;
               sfx = sfx1;
               lname = compress(lname1,' ');
            end;

         name2 = substr(name,x+1);
         name3 = compress(name2,"'`-,.");
         fname = scan(name3,1); finit = substr(fname,1,1);
         mname = scan(name3,2); minit = substr(mname,1,1);
         if mname='MALE' then ln_gender=mname;
         if mname='FEMALE' then ln_gender=mname;
         if mname in ('DECEASED','MALE','FEMALE') then mname='';
      end;

   drop x last lname1 name2 name3 sfx1;
%mend;

* Input dataset created from program in Appendix I;
proc sort data=lexis_results out=_lexis; by ssn; run;

data lexis;
   set _lexis;
   retain  segnum segtype;
   length segtype $28 ;

   if index(line,'DOCUMENT') then
      do;
         segnum = input(substr(line,1,2),2.);
```

```
         segtype = '';
      end;

   x= index(line,'PERSON LOCATOR');
   if x then segtype = compbl(line);

   x= index(line,'SOCIAL SECURITY');
   if x then segtype = 'SSA';

   drop path filename x;
run;

data lexis2;
   merge lexis lexis(firstobs=2 keep=ssn line
                     rename=(ssn=next_ssn line=next_line));
run;

data mrg;
   length sfx sfx1 $4. last lname lname1 name2 name3 second_surname
          former_surname spouse_name $20 fname mname $15 other_names
          also_known_as $100 ln_ddate ln_gender $10 ln_ssn $11;

   set lexis2; by ssn;

   retain birth_year lname fname mname sfx finit second_surname
          spouse_name former_surname other_names ln_ddate ln_gender
          ln_ssn also_known_as;

   if first.ssn then
      do;
         lname=''; fname=''; sfx=''; mname=''; finit=''; birth_year=.;
         second_surname=''; former_surname=''; other_names='';
         ln_ddate=''; ln_gender=''; ln_ssn=''; also_known_as='';
      end;

   x   = index(upcase(line),'NAME:');
   xg  = index(line,'(');
   xge = index(line,')');
   ss  = index(line,'Second Surname:');
   ss2 = index(line,'Former Name:');
   ss3 = index(line,'Other Names:');
   ss4 = index(line,'Client ID/Project Name');
   ss5 = index(line,'Spouse Name:');
   ss6 = index(line,'Also Known As:');
   if x and ss=0 and ss2=0 and ss4=0 and ss5=0 and ss6=0 then
      do;
         name = left(substr(line,x+5));
         %namesplt;
      end;

   if x and xg then ln_gender = substr(line,xg+1,(xge-xg-1));
   if ln_gender='DECEASED' then ln_gender=' ';

   if ss  then second_surname = substr(line,ss+16);
   if ss2 then former_surname = next_line;
   if ss3 then other_names    = substr(line,ss3+13);
   if ss5 then spouse_name    = substr(line,ss5+13);
```

```
   if ss6 then also_known_as  = next_line;

   prev_seg = lag(segnum);
   if ^first.ssn and segnum ne prev_seg then
      do;
         lname=''; fname=''; sfx=''; mname=''; finit=''; birth_year=.;
         second_surname='';former_surname='';other_names=''; ln_ssn='';
         spouse_name=''; also_known_as='';ln_ddate=''; ln_gender='';
      end;

   x = index(line,'Gender:');
   if x then ln_gender = substr(line,x+8);

   x = index(line,'Birthyear:');
   if x then birth_year = input(substr(line,x+11,4),4.);
   x = index(line,'Birthdate:');
   if x=0 then x=index(upcase(line),'DATE OF BIRTH:');
   if x then birth_year = input(scan(line,-1),4.);

   x=index(upcase(line),'DATE OF DEATH:');
   if x then
      do;
         ln_dyear = input(scan(line,-1),4.);
         ln_ddate = substr(line,x+15,10);
      end;

   x=index(line,'Social Security Number:');
   if x then ln_ssn=substr(line,x+24);

   drop name prev_seg ss ss2 ss3 ss4 ss5 ss6 x xg xge ;
run;

proc sort data=mrg; by ssn segnum; run;

*COHORT dataset contains NAME and Date of Birth for each study subject;
data mrg_gia;
   merge cohort(in=a keep=ssn der_lname der_fname der_mname der_dob)
         mrg(in=b);
   by ssn;
   if a & b;
   uab_byear = year(der_dob);
run;

* Replace DER_LNAME and DER_FNAME with the name of your cohort variables for
  Last name & first name;

%let lname  = der_lname = lname ;
%let fname  = der_fname = fname ;
%let lnamef = der_lname = fname ;
%let fnamef = der_fname = lname ;

data single;
   length segtype2 $4 lname_i1 fname_i1 minit_a $1;
   set mrg_gia;
   by ssn segnum;
   if last.segnum;
   segtype2 = left(scan(segtype,-1,'-'));
```

```
if der_lname ne '' then lname_i1 = substr(der_lname,1,1);
if der_fname ne '' then fname_i1 = substr(der_fname,1,1);
if der_mname ne '' then minit_a = substr(der_mname,1,1);

uab_lname_soundex1 = soundex(der_lname);
uab_fn_soundex1 = soundex(der_fname);
ln_soundex      = soundex(lname);
ln_fn_soundex   = soundex(fname);
mn_soundex      = soundex(mname);

* create match score;
select;
   when (&lname & &fname & minit_a=mname &
         uab_byear=birth_year)                                  match=1;
   when (&lname & &fname & minit_a=mname & birth_year =.)       match=2;
   when (&lname & &fname & minit_a =: mname)                    match=3;
   when (&lname & &fname & uab_byear = birth_year)              match=4;
   when (&lname & &fname & birth_year =.)                       match=5;
   when (&lname & &fname)                                       match=6;
   when (&lname & fname_i1 = finit & uab_byear = birth_year)    match=7;
   when (&lname & fname_i1 = finit)                             match=8;
   when (&lname & &fname & birth_year ne .)                     match=9;
   when (&lname & uab_byear = birth_year)                       match=10;

   * if no match then try transposing the first, middle and last names;
   when (&lnamef & fname_i1=lname & minit_a=mname &
         uab_byear=birth_year)                                  match=11;
   when (&lnamef & fname_i1=mname & minit_a=fname &
         uab_byear=birth_year)                                  match=12;
   when (&lnamef & &fnamef & minit_a=mname)                     match=13;
   when (&lnamef & fname_i1=mname & minit_a=fname)              match=14;
   when (&lnamef & &fnamef & minit_a=fname)                     match=15;
   when (&lnamef & &fnamef & uab_byear=birth_year)              match=16;
   when (&lnamef & &fnamef)                                     match=17;
   when (&lnamef & fname_i1=substr(lname,1,1) &
         uab_byear=birth_year)                                  match=18;
   when (&lnamef & fname_i1=substr(lname,1,1))                  match=19;
   when (&lnamef & uab_byear=birth_year)                        match=20;

   * if still no match, then try some additional matching possibilities;
   when (index(der_lname, lname) & &fname)                      match=21;
   when (index(der_lname, lname))                               match=22;
   when (&fname & uab_byear = birth_year)                       match=23;
   when (&fname & minit_a = mname)                              match=24;
   when (lname =:der_lname & fname =:der_fname)                  match=25;
   when (uab_lname_soundex1 = ln_soundex & &fname)              match=26;
   when (uab_lname_soundex1 = ln_soundex)                       match=27;
   when (uab_fn_soundex1 = ln_fn_soundex & &lname)              match=28;
   when (uab_lname_soundex1 = mn_soundex & &fnamef)             match=29;
   when (uab_fn_soundex1 = ln_fn_soundex)                       match=30;

   * residual matching;
   when (&lname)                                                match=31;
   when (&lname & birth_year ne .)                              match=32;
   when (&lnamef)                                               match=33;

   otherwise match=0;
```

```
   end;

   rename lname=ln_lname fname=ln_fname mname=ln_mname sfx=ln_sfx
          birth_year=ln_byear;
   drop line uab_lname_soundex1 uab_fn_soundex1 mn_soundex
        ln_fn_soundex ln_soundex minit_a;
run;

data total_matches;
   set single(keep=ssn match segtype2);
   by ssn;
   retain match_count ;
   if first.ssn then match_count=0;
   if match > 0 then match_count=1;
   if last.ssn;
   drop match segtype2;
run;

data final;
   merge total_matches(in=a where=(match_count=0))
         single;
   by ssn;
   FileMatch=1;
   if a then FileMatch=0;
   drop segtype lname_i1 fname_i1 match_count uab_byear finit minit
        next_ssn next_line;
run;

data lexis_result_segments(label='Segmented Lexis results'
                           compress='yes' index=(ssn));
   set final;
   label match    = 'Match code (0-33)'
         segtype2 = 'Segment type (SRCH, TRAK, SEEK, SSA)'
         segnum   = 'Segment number'
         filematch= '0 - if no MATCH codes for all segments > 0'
         ln_lname = 'Lexis last name'
         ln_fname = 'Lexis first name'
         ln_mname = 'Lexis middle name'
         ln_sfx   = 'Lexis name suffix'
         ssn      = 'Cohort SSN'
         ln_byear = 'Lexis birth year';

   rename former_surname = ln_former_surname
          second_surname = ln_second_surname
          other_names    = ln_other_names
          also_known_as  = ln_also_known_as
          spouse_name    = ln_spouse_name;
run;
```