Identifying Invalid Social Security Numbers

Paulette Staum, Paul Waldron Consulting, West Nyack, NY
Sally Dai, MDRC, New York, NY

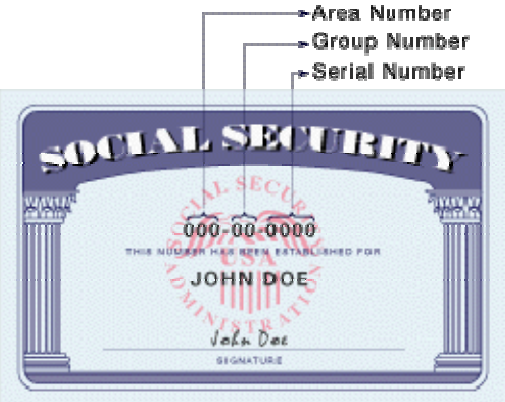
ABSTRACT

Do you need to check whether Social Security numbers (SSNs) are invalid? An SSN consists of a three-digit area number, a two-digit group number and a four-digit serial number. A list of the valid group numbers for each area is available from a Social Security Administration web page that changes monthly. This paper demonstrates automatic downloading of the list of area and group codes, plus testing for out-of-range values and impossible combinations. The paper assumes knowledge of the DATA step and PROC SQL.

INTRODUCTION

In the United States, most individuals are identified by a Social Security number (SSN) assigned by the Social Security Administration (SSA). An SSN can be used as a common key for joining information from different sources. If SSNs are invalid, joining different data sources is much more difficult, if not impossible. Therefore, one of the first steps in processing data with SSNs is to identify invalid SSNs. This paper describes a process for checking SSNs.

An SSN consists of three sets of digits. The first set is the three-digit area number. The second set is the two-digit group number. The third set is the four-digit serial number. Often these sets of digits are shown separated by hyphens, so an SSN looks like this: 999-99-9999.



Obviously, instead of guessing which SSNs are invalid, we would like a systematic process for determining their validity. Validating SSNs is required for many different projects (often for each batch of new data), so it is worth investing some effort to develop a good approach. There are several stages in creating a useful process. First, you need to determine what rules apply to SSNs. The best resource is the SSA web site. (See the References section for more information.) Next, plan the checking process and make decisions about the best techniques for each step in the process. Finally, develop static tests for whole SSNs, static tests for each part of SSNs, and dynamic tests for the area and group numbers. These dynamic tests will adjust as valid values for area and group codes change over time.

SAMPLE DATA

Here are some sample SSNs. So far as we know, these are all fake. Any resemblance to actual SSNs is completely coincidental. Can you guess which SSNs are valid? Some problems are obvious, but other problems are not.

```
123456789   WWW123456   206045678   06-648-1234
MR1234567   567890000   248953456   066-48-1234
666786543   086981234   349021234   0664-81-234
800445678   010901234   130964321   02468135
45678AB90   010921234   007105678   246813579
772123456   011025678   150204321   600981234
987654321   011243456   148171234   078051120
234567890   050982222   163851234
123004567   044131234   163841234
000345678   221083456   735115678
```

STEP 1 – CHECKING THE BASICS

The SSA web site describes some very basic rules for SSNs. If hyphens are present in SSNs, there should be one between the area and group numbers and another between the group and serial numbers. Without hyphens, an SSN should have nine digits and nothing else. None of the individual parts should be equal to zero. The area number should not be 666. The program below does these basic checks and a few additional checks that are described later. It assumes that the SSN is stored in a character variable.

This program relies heavily on character functions. Some of them are relatively new. The PRXPARSE function defines a pattern and the PRXMATCH function looks for the pattern in an SSN. The NOTDIGIT function searches a character string for any character that is not a digit.

```
data Problems1;
  set TestSSN;
  length Message $80;
  retain hyphenre;
  * If hyphens are present, they match the pattern ddd-dd-dddd.;
  if _n_ = 1 then do;
    hyphenre=prxparse('/\d\d\d-\d\d-\d\d\d\d/');
  end;
  if index(SSN,'-') and prxmatch(hyphenre,SSN)^=1 then do;
    message = "Hyphen misplaced";
   output;
  end;
  * Remove any hyphens;
  SSN= compress(SSN,'- ');
  * Basic checks;
  if length(SSN) ^= 9 then do;
    message="SSN not 9 digits long";
    output;
  end;
  else if notdigit(trim(SSN)) > 0 then do;
    message="Non-digit in SSN";
    output;
  end;
  else if substr(SSN,1,3) in ('000','666') or
          substr(SSN,4,2)='00' or
          substr(SSN,6,4)='0000' then do;
    message="Invalid area, group or serial number";
    output;
  end;
  else if SSN in ("078051120","111111111","123456789","219099999") or
    (SSN >= "987654320" and SSN <= "987654329") or
    SSN='999999999' then do;
      message="Dummy SSN";
      output;
  end;
  drop hyphenre;
```

The following Social Security numbers in the sample data were identified as invalid.

```
     SSN         Message

123456789        Dummy SSN
MR1234567        Non-digit in SSN
666786543        Invalid area, group or serial number
45678AB90        Non-digit in SSN
123004567        Invalid area, group or serial number
000345678        Invalid area, group or serial number
WWW123456        Non-digit in SSN
567890000        Invalid area, group or serial number
06-648-1234      Hyphen misplaced
0664-81-234      Hyphen misplaced
02468135         SSN not 9 digits long
078051120        Dummy SSN
```

You might be wondering why a few of these Social Security numbers are flagged as invalid by this program. The SSNs 987-65-4320 to 987-65-4329 are reserved for use in advertisements. The SSA used 219-09-9999 in a promotional pamphlet in 1940. In 1938, a wallet manufacturer inserted a sample Social Security card in each of its wallets as part of a marketing effort. The sample was a copy of Hilda Schrader Whitcher's card, with the SSN 078-05-1120. Hilda (pictured at the right) was secretary to the Vice President of Marketing of the firm. (Note that the Vice President did not copy his own card.) According to the SSA, over 40,000 people have used this SSN.

While this first step is a good start in identifying invalid SSNs, there is additional information from the SSA web site that can be used to do more checking of SSNs.

STEP 2 - DOWNLOADING INFORMATION FROM THE SSA WEB SITE

The Social Security Administration web site has a text file of information about the currently valid ranges for SSN area and group codes. SAS ® makes it easy to download files from a web site by using a FILENAME statement with the URL access method. It works like magic.

```
filename ssn url 'http://www.socialsecurity.gov/employer/highgroup.txt';
```

The beginning of the web page looks like this.

```
       HIGHEST GROUP ISSUED AS OF 6/01/07


Anything with an asterisk (*) is a change effective 6/01/07.
This list shows the SSN area and group numbers that are in
the process of being issued as of the date at the top of this page.

NOTE:  INDICATES GROUP CHANGE SINCE LAST MONTH.
001 06  002 04      003 04 004 08 005 08 006 06
007 06       008 90 009 90  010 90  011 90  012 90
013 90 014 90 015 90 016 90 017 90 018 90
019 90 020 90 021 90 022 90 023 90 024 90*
025 88 026 88 027 88 028 88 029 88 030 88
031 88 032 88        033 88 034 88 035 72 036 72
037 72  038 70       039 70 040 11 041 11  042 11
043 11 044 11 045 11 046 11*       047 08 048 08
049 08 050 96 051 96 052 96 053 96 054 96
055 96 056 96 057 96 058 96 059 96  060 96
…
```

Once you have downloaded the file, you will discover that the formatting of the file is a little bit messy. Sometimes tabs are used as delimiters and sometimes spaces are used as delimiters. Also, the asterisks indicating recent changes are not necessary for our purposes. The DATA step below reads the text file from the web site. It ignores the header lines and converts the asterisks and tabs to spaces. It outputs a text file with a more regular structure.

```
data _null_;
  infile ssn length=len truncover;
  input a $varying200. len;
  file 'ssncurrgrp.txt';
  b=translate(a,' ','09'x);
  b=translate(b,' ','*');
  if _N_ > 9 then put b;      * ignore header;
run;
```

The next DATA step creates a data set with one observation for each area number. Each observation has an area number and the group number that is currently being assigned for that area. Some observations from the middle of the output data set are displayed at the right.

```
data work.CurrentGroup;
  length Area $3 Group $2;
  infile "ssncurrgrp.txt";
  input Area $ Group $ @@;
run;
```

```
Area  Current Group
643       06
644       06
645       06
646       94
647       92
648       44
649       42
```

Now you have the information that will allow you to check the validity of the combination of an area number and a group number in an SSN. There are two ways to use that information – to check area numbers and to check group numbers.

STEP 3 – CHECKING AREA NUMBERS

Before looking at the current group numbers, let's do something simpler and look at the area numbers by themselves. The data set that was created from the web site contains a list of valid area numbers. It can be used to check for invalid area numbers. This is a standard lookup problem, and there are several standard solutions to it:

- Use a DATA step merge
- Use PROC SQL
- Generate a format with PROC FORMAT and use it in a DATA step to flag valid and invalid codes
- Load the valid codes into a hash object in a DATA step

Based on previous experience, we know that the SQL solution is reasonably efficient. Since SAS programmers are usually more familiar with SQL than with hash objects, we'll show a SQL subquery approach. The invalid SSNs in our sample data are shown to the right. Note that some of these problems were also found by the static tests.

```
proc sql;
  create table Problems2 as
  select *, "Invalid area number" as Message
    from TestSSN
   where substr(ssn,1,3) not in
   (select area from CurrentGroup)
    order by ssn;
quit;
```

SSN	Message
000345678	Invalid area number
06-648-1234	Invalid area number
666786543	Invalid area number
735115678	Invalid area number
800445678	Invalid area number
987654321	Invalid area number
MR1234567	Invalid area number
WWW123456	Invalid area number

Now consider the problem of determining whether a group number is valid for an area. This next check is a bit more complicated and will require some preparation.

STEP 4 – UNDERSTANDING THE ORDER OF ASSIGNMENT FOR GROUP NUMBERS

At first glance, it looks as if you can interpret the group number associated with each area number as the highest valid group number for the area. Alas, life is not so simple. The SSA web site explains that group numbers are not assigned in sequential order. Instead, they are assigned (one number at a time) in the following, somewhat unusual order:

Odd Numbers	01 – 09
Even Numbers	10 - 98
Even Numbers	02 - 08
Odd Numbers	11 – 99

Thus, the group numbers are used in the order 01,03,05,07,09,10,12,14,16,…,96,98,02,04,06,08,11,13,15…,97,99. The simplest way to think about this is to give a sequential "group rank" to each group number as it assigned. Although group numbers are not assigned in consecutive order, the group ranks will be in order. You can expect that a group with rank 1 will always be assigned before a group with rank 2, etc. The code below produces the ranks displayed at the right.

```
data GroupRank;
  rank=1;
  do group = 1 to 9 by 2;
    output;
    rank = rank + 1;
  end;
  do group = 10 to 98 by 2;
    output;
    rank = rank + 1;
  end;
  do group = 2 to 8 by 2;
    output;
    rank = rank + 1;
  end;
  do group = 11 to 99 by 2;
    output;
    rank = rank + 1;
  end;
  format group z2.;
```

Group Number	Rank
01	1
03	2
05	3
07	4
09	5
10	6
12	7
…	…
98	50
02	51
04	52
06	53
08	54
11	55
13	56
…	…

What is a good way to use these group ranks to check group numbers? Let's start by adding the group ranks to the Current-Group data set, with the current group numbers for each area. Note that the generated rank and group variables are numeric, in contrast with the character group number from the SSN.

```
proc sql;
  create table CurrentRank as
  select curr.area, curr.group as CurrGroup,
         rank.Rank as CurrRank
    from CurrentGroup curr, GroupRank rank
    where input(curr.group,2.) = rank.group;
quit;
```

Area	Current Group	Current Rank
643	06	53
644	06	53
645	06	53
646	94	48
647	92	47
648	44	23
649	42	22

STEP 5 – CHECKING GROUP NUMBER RANKS

Finally, we need to check that the group rank in each SSN is not higher than the current maximum group rank for its area. This can be done using any of the lookup methods that were described for the area number lookup task.

The SQL step below is deceptively short, but it is remarkably powerful. It joins three data sets. The first data set provides the SSNs to test. The second data set provides the group ranks for each group number, so that each SSN's group number can be converted to a group rank. The third data set provides the currently assigned group rank for each area number. This is compared with the SSN group number's rank.

The WHERE clause has three parts:

- First, it links the SSN area number with the group rank that is currently being assigned for that area.
  substr(test.ssn,1,3) = currRank.area
- Next, it also links the SSN group number with the rank for that group number.
  substr(test.ssn,4,2) = put(grpRank.group,Z2.)
- Finally, it limits the output to cases where the SSN group number rank is higher than the SSN area's currently assigned group rank.
  grpRank.rank > currRank.currRank

```
proc sql;
  create table Problems3 as
  select test.ssn, grpRank.group, grpRank.rank,
         currRank.currRank, currRank.currGroup,
         "Group number not assigned" as Message
    from TestSSN test, CurrentRank currRank, GroupRank grpRank
   where substr(test.ssn,1,3) = currRank.area
     and substr(test.ssn,4,2) = put(grpRank.group,z2.)
     and grpRank.rank > currRank.currRank
   order by ssn;
quit;
```

These Social Security numbers in the sample data were identified as invalid.

SSN	Group	Rank	Curr Rank	Curr Group	Message
010921234	92	47	46	90	Group number not assigned
011025678	02	51	46	90	Group number not assigned
044131234	13	56	55	11	Group number not assigned
050982222	98	50	49	96	Group number not assigned
086981234	98	50	49	96	Group number not assigned
123456789	45	72	48	94	Group number not assigned
130964321	96	49	48	94	Group number not assigned
163851234	85	92	43	84	Group number not assigned
206045678	04	52	42	82	Group number not assigned
221083456	08	54	53	06	Group number not assigned

SUMMARIZING RESULTS

For reporting purposes, we can combine the separate data sets of problems. Here is code for this purpose, followed by the output for the sample data.

```
proc sql;
  create table AllProblems as
  select ssn, message from problems1
    union select ssn, message from problems2
    union select ssn, message from problems3
  order by ssn;
quit;

proc report data=AllProblems nowd;
  column ssn message;
  define ssn / order;
run;
```

```
SSN             Message
000345678       Invalid area number
                Invalid area, group or serial number
010921234       Group number not assigned
011025678       Group number not assigned
02468135        SSN not 9 digits long
044131234       Group number not assigned
050982222       Group number not assigned
06-648-1234     Hyphen misplaced
                Invalid area number
0664-81-234     Hyphen misplaced
078051120       Dummy SSN
086981234       Group number not assigned
123004567       Invalid area, group or serial number
123456789       Dummy SSN
                Group number not assigned
130964321       Group number not assigned
163851234       Group number not assigned
206045678       Group number not assigned
221083456       Group number not assigned
45678AB90       Non-digit in SSN
567890000       Invalid area, group or serial number
666786543       Invalid area number
                Invalid area, group or serial number
735115678       Invalid area number
800445678       Invalid area number
987654321       Dummy SSN
                Invalid area number
MR1234567       Invalid area number
                Non-digit in SSN
WWW123456       Invalid area number
                Non-digit in SSN
```

There is one final question. Which of the SSNs in the test SSNs were NOT identified as invalid? Here is the code to answer this question and the output for the sample data.

```
proc sql;
  create table OKSSN as
  select testssn.SSN
    from testssn
    where testssn.ssn not in
      (select ssn from AllProblems)
    order by ssn;
quit;
```

```
007105678
010901234
011243456
066-48-1234
148171234
150204321
163841234
234567890
246813579
248953456
349021234
600981234
772123456
```

UNDETECTED PROBLEMS

Inevitably, there will be undetected errors, such as miskeying of single digits or transpositions of digits. When two data sets are joined using an SSN as the key, some matches will not occur due to these errors. Evaluate whether an omitted match is more problematic than a false match. If it is important to avoid omitted matches, consider a flexible and adventurous matching criterion suggested in an article by Ian Whitlock (2001). This criterion is based on characteristic patterns in data entry mistakes. If a single digit in an SSN is mistyped, then 8 of 9 digits would still match. If two digits are transposed, then 7 of 9 digits would match. The WHERE clause below will match SSNs despite errors in one or two digits. Since there is a risk that some incorrect matches will also be made, consider using other fields to check the accuracy of these approximate matches.

```
…
where sum ( substr(a.SSN,1,1) = substr(b.SSN,1,1) ,
substr(a.SSN,2,1) = substr(b.SSN,2,1) ,
....
substr(a.SSN,9,1) = substr(b.SSN,9,1)
            ) >= 7
…
```

NON-SAS METHODS

All of this checking still does not assure that an SSN is valid for a specific individual. Other non-SAS methods are available for verification of SSNs. Several online services verify individual SSNs. Also, the SSA has a Verification Service that will process a file of SSNs and determine whether each SSN is valid or invalid. Unfortunately, its intended use is limited to verifying Social Security numbers for new employees. See the References section for further information about these approaches. There is also a SSA Death Master File that can be used to identify SSNs for people who have died since 1962. It can be ordered from the SSA. Also, online access to the Death Master File for individual SSNs is available through genealogical web sites.

CONCLUSION

SAS provides useful tools for accessing Social Security Administration resources that facilitate identifying invalid Social Security numbers. This project has used:

- String processing functions, including old friends like SUBSTR and TRANSLATE and new ones like NOTDIGIT, PRXPARSE and PRXMATCH
- FILENAME URL access method for web resources
- INFILE and INPUT statements for reading text files
- PROC SQL joins and subqueries for implementing logical set operations to combine information and identify exceptions

We hope that you will find some of these methods useful, both for this task and for other tasks.

REFERENCES

The Social Security Administration web site is an excellent source of information.

General information	http://www.ssa.gov/
Numbering scheme	http://www.ssa.gov/history/ssn/geocard.html
History of high groups for each area	http://www.socialsecurity.gov/employer/ssnvhighgroup.htm
Current high group for each area	http://www.socialsecurity.gov/employer/highgroup.txt
States and areas	http://www.ssa.gov/employer/stateweb.htm
Order of issuance of numbers	http://www.ssa.gov/employer/ssnweb.htm
Common invalid numbers	http://www.ssa.gov/history/ssn/misused.html
Verification service	http://www.ssa.gov/employer/ssnv.htm
Restrictions on using verification service	http://www.ssa.gov/employer/ssnvrestrict.htm
SSA Death Master File (since 1962)	http://www.ntis.gov/products/ssa-dmf.asp

The genealogical community provides some access to the SSA death master file.

http://ssdi.rootsweb.com/ Social Security Administration Death Index

There are two SAS User Group conference papers that have some helpful ideas:

Whitlock, Ian. 2001, “PROC SQL - Is it a Required Tool for Good SAS® Programming?” Proceedings of the 26th Annual SAS Users Group International Conference

Winn, Thomas J. Jr. 2006 “Fraud Detection – A Primer for SAS® Programmers” Proceedings of the 31st Annual SAS Users Group International Conference

ACKNOWLEDGMENTS

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Images are from the Social Security Administration web site.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Many other methods could be used for this task. If you have suggestions, please contact us at:

Paulette W. Staum
Paul Waldron Consulting
2 Tupper Lane
West Nyack, NY 10994
staump@optonline.net

Sally Dai
MDRC
16 East 34th Street
New York, NY 10016
sally.dai@mdrc.org