VALIDATION OF THE UPLOAD AND DOWNLOAD PROCEDURES
IN THE SAS™ SYSTEM

Richard Markus, Allergan Inc.
Jeff Kouba, Allergan Inc.
Curt Snyder, Allergan Inc.
Louise Murphy, Allergan Inc.

Abstract

SAS provides the Upload and Download procedures for transferring data sets between a microcomputer and a host. The integrity of transferred data is critical. Ideally, the transferred data sets would be identical to the original data sets. Unfortunately, this is not likely since there are differences in machine precision. Different operating systems store values in different formats, such as EBCDIC and ASCII.

This validation examines the magnitude and implications of the differences created by transferring SAS data sets between a microcomputer and a host.

Methods

Test a large clinical data set:

A large clinical data set was downloaded from the mainframe to the personal computer. Subsequently, the data downloaded was uploaded back to the mainframe. This twice-transferred data set was then verified against the original data set using PROC COMPARE with the method=relative option.

Test specific types of variables and values:

UPNDOS.SAS and UPDNCMS.SAS are two SAS programs created which utilize 'cards' to generate three specific test data sets. These two programs have identical data steps; therefore, they produce identical data sets in their respective operating environments. The two programs only differ in their 'libname' designations. UPNDOS.SAS is presented in the appendix.

UPDNCMS.SAS created three data sets on the mainframe: updn1cms, updn2cms, and updn3cms. These data sets were considered the 'source' data on the mainframe. The program also created a listing of their values in hex16 format and F20.10 format. The
three data sets were then downloaded to the personal computer.

The download facility, PROC DOWNWAD, was verified by comparing the values of these downloaded data sets to the 'source' values created by UPDNDOS.SAS. The data integrity was verified by one hundred percent visual inspection.

Similarly, UPDNDOS.SAS created three data sets on the personal computer: updn1dos, updn2dos, and updn3dos. The program also created a listing of their values in hex16 format and F20.10 format. The three data sets were then uploaded to the mainframe.

The upload facility, PROC UPLOAD, was verified by comparing the values of these uploaded data sets to the 'source' values created by UPDNCMS.SAS. The data integrity was verified by one hundred percent visual inspection.

Additional Tests:

Finally, some additional tests were conducted to determine the effects of uploading and downloading a data set, and vice versa. The concern was whether or not the altered values produced by transferring a data set were returned to their original values when transferred back to their original platform.

The three data sets created by UPDNDOS.SAS on the personal computer were uploaded to the mainframe and then downloaded back to the personal computer. The data integrity of these twice-transferred personal computer data sets was verified against the original personal computer data sets.

Similarly, the three data sets created by UPDNCMS.SAS on the mainframe were downloaded to the personal computer and then uploaded back to the mainframe. The data integrity of these twice-transferred mainframe data sets was verified against the original mainframe data sets.

Results

Test a large clinical data set:

PROC COMPARE with the method = relative option determined that the values of the large clinical data set which was downloaded to the personal computer and then uploaded back to the mainframe had values within 1.6488E-16 of the original values. This is an acceptable difference across the different operating platforms.

Test specific types of variables and values:

The downloaded values of test data set 1, numeric variables, represented in F20.10 format appear identical to their source values. However, the values of the real numbers differed when compared in hex16 format. For example, a value of 0.2 created on the personal computer had a hexadecimal value of 3FC999999999999A. The same variable, when downloaded from the mainframe, had a value of 3FC9999999999999. This is a hexadecimal decrease of 1. Not all real variables were changed when downloaded. However, those which did were decreased by 1. The implications of the differing hexadecimal values are discussed in the Conclusion.

The downloaded values of test data set 2, alphanumeric variables, were identical in both standard numerical representation and hex16 format.

The downloaded values of test data set 3, formatted variables, were identical when represented with a format of F20.10. However, differences in hex16 format were observed for the real numbered values in variables numb1 through numb4. These differences are the same as those described for test data set 1, above. For example 6.666 in numb4 was represented by 401AA9FBE76C8B44 in the personal computer source data and by 401AA9FBE76C8B43 in the downloaded data. This again is a decrease of 1 in hexadecimal. The values which did change upon downloading were also decreased by 1.
The date and time formatted variables were consistent and identical to the source values. The implications of the differing hexadecimal values are discussed in the Conclusion.

Similarly, the uploaded values of test data set 1, numeric variables, represented in F20.10 format appear identical; however, the values of the real numbers differed when compared in hex16 format. For example, a value of 0.2 created on the mainframe had a hexadecimal value of 4033333333333333. When this value was created on the personal computer and uploaded to the mainframe, its value was 4033333333333334. The greatest change in the hexadecimal representation witnessed was an increase of four. Some values decreased by one. There was no apparent consistency in the change of the values; however, the values which changed were increased more often than decreased. The implications of the changes is discussed in the Conclusion.

The uploaded values of test data set 2, alphanumeric variables, were identical in both standard numerical representation and hex16 format.

The uploaded values of test data set 3, formatted variables, were identical when represented with a format of F20.10. However, differences in hex16 format were observed for the real numbered values, variables numb1 through numb4. These differences were the same as those described for test data set 1, above. For example 6.666 in numb4 was represented by 416AA7EF9DB2D0E in the mainframe source data and by 416AA7EF9DB2D10 in the uploaded data. This is an increase of 2 in the hexadecimal system. The greatest hexadecimal increase observed was 3 and the greatest decrease observed was 3. Again, there was no obvious pattern in the increases or decreases. The date and time formatted variables were consistent and identical to the source values. The implications of the differing hexadecimal values are discussed in the Conclusion.

Additional Tests:

The three data sets which were created by UPNDNOS.SAS were uploaded to the mainframe and then downloaded back to the personal computer. The upload and download facilities appeared complimentary when initiated on the personal computer; however, they did not appear complimentary when initiated on the mainframe.

Specifically, the values of all three test data sets which were created on the personal computer, uploaded, and then downloaded, were consistent when observed with a format of F20.10 and a format of hex16. Therefore, the change in the hexadecimal representation of a value which occurred while uploading was altered back to its original value when downloaded.

In contrast, the first and third data sets originating on the mainframe, downloaded to the personal computer, and then uploaded back to the mainframe, differed from their original values. Therefore, the download and upload facilities are not complimentary upon initiation on the mainframe.

In detail, the mainframe values of test data set 1 were consistent when observed with a format of F20.10. However, the change in the hexadecimal representation which occurred while downloading was not altered back to its original hexadecimal value when uploaded. For example, the hexadecimal representation of 0.2 originating on the mainframe was 4033333333333333. After this value is downloaded and then uploaded back to the mainframe, it was represented by 4033333333333332. This is a hexadecimal decrease of 1. All values which changed appeared to be decreased by a maximum of 5 in the hexadecimal system.

The mainframe values of test data 2 were consistent in both standard numerical representation and hex16 format.

Finally, the real numbered values, variables numb1 to numb4, in the mainframe test data set 3 had the same problem as those in test data set 1. Specifically, the hexadecimal values decreased for some numbers by as
much as 5. The F20.10 formatted values all appeared the same.

Conclusion

Changes due to uploading or downloading data sets are small but present. They are caused by transferring values between platforms which store the numbers in different formats. Since the differences are not apparent with a format of F20.10, numerical computations on transferred data are accurate within the precision of our statistical analyses which often require precision to the hundred-thousandth (fifth digit past the decimal). Therefore, these utilities are valid when properly executed and programs utilizing the data are appropriately modified as described below.

The changes in data values may have a profound impact on the program response from transferred data sets. Specifically, any SAS program statements which make comparisons on real values are likely to produce the response of the values being unequal, even when they were originally equal. The very small numerical difference produced is enough to cause real-valued comparisons to fail. One solution, and our recommended practice, is that all comparisons made on real values be made against values rounded to their necessary level of significance. For example, instead of using:

\[
\text{IF VISIT} = 2.1 \text{ THEN DO;}
\]

use:

\[
\text{IF ROUND(VISIT,0.1) = ROUND(2.1,0.1) THEN DO;}
\]

The SAS Language manual, version 6, describes many relevant considerations in chapter three. The representation of floating point values is discussed on page 86, precision versus magnitude is described on pages 89-90, computational considerations of fractions are described on page 90, and numeric comparison considerations are outlined on page 91. Examples of the need for the \text{round} function and the use of this function are presented on pages 90 and 91.

Notices

SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries. \text{TM} indicates USA registration.

IBM is a registered trademark or trademark of International Business Machines Corporation.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Appendix

```
*===================================================================*
* PROJECT: Validation of proc upload and download                  *
* PROGRAM: UPONDOS SAS                                             *
* PROGRAMMER: Richard Markus                                       *
* DATE: October 15, 1990                                           *
* PURPOSE: This program creates three data sets to be used in      *
* validating PROC UPLOAD and PROC DOWNLOAD.                        *
* SAS VERSION: PC/SAS Version 6.04                                *
* OUTPUT DATA SETS: UPDN100S.ssd                                   *
*                  UPDN200S.ssd                                       *
*                  UPDN300S.ssd                                       *
*===================================================================*
OPTIONS CENTER NOODATE NONUMBER PS=60 LS=132;
LIBNAME tstdata 'D:\VALIDATE';
```
*---FORMATTED DATA: testing dates, times, and floating points---*

DATA testdata.UPDN3DOS;
  INPUT @1 date1 date7.
  @9 date2 mmdyy6.
  @16 date3 ddmmyy8.
  @25 date4 monyy5.
  @31 time1 time8.
  @40 time2 time5.
  @46 numb1 3.0 /*46 - 48*/
  @50 numb2 4.1 /*50 - 53*/
  @55 numb3 5.2 /*55 - 59*/
  @61 numb4 5.3 /*61 - 65*/;
CARDS;
  O1JAN89 01:01:01 01:01 111 1111 11111 1111
  02FEB89 02:02:02 02:02 222 2222 22222 22222
  03MAR89 03:03:03 03:03 333 3333 33333 3333
  04APR89 04:04:04 04:04 444 4444 44444 4444
  05MAY89 05:05:05 05:05 555 5555 55555 55.55
  06JUN89 06:06:06 06:06 666 6666 66666 6.666
  07JUL89 07:07:07 07:07 777 7777 77777 7777
  08AUG89 08:08:08 08:08 888 8888 88888 8888
  09SEP89 09:09:09 09:09 999 9999 99999 9.999
  10OCT89 10:10:10 10:10 111 1111 11111 11.11
  12DEC89 12:12:12 12:12 121 1212 12121 1212.
RUN;

PROC CONTENTS;
  TITLEI 'UPDN3DOS: Formatted Values';
RUN;

PROC PRINT;
  TITLEI 'UPDN3DOS: Formatted Values as F20.10';
  FORMAT date1--numb4 F20.10;
RUN;

PROC PRINT;
  TITLEI 'UPDN3DOS: Formatted Values as HEX16.';
  FORMAT date1--numb4 hex16.;
RUN;