

Importing Complicated Excel® Files into SAS® Drug Development (SDD)

Heather L. Murphy, Eli Lilly and Company, Indianapolis, IN

Gregory C. Steffens, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

With the rise of new programming environments come new programming challenges. This paper illustrates a method to convert Microsoft® Excel® worksheets into SAS® data sets, using code that runs in SDD (SAS® Drug Development). SDD is a newer programming environment that provides control and compliance. SDD has been seen as an attractive application to pharmaceutical companies to perform programming tasks while tracking modifications and remaining compliant with both regulatory and company standards. The down side to this new environment is that converting data sets that reside in an Excel worksheet is no longer as easy as using PROC IMPORT. As most statisticians know, Excel data rarely comes in a pretty format that can easily be converted into other environments, i.e., SAS. This paper will describe methods to convert Excel worksheets into SAS data sets using code that executes in SDD. Most of the code will be applicable to both SDD and Unix® SAS users since SDD executes code on a Unix server. Solutions will be offered for addressing complications that arise with Excel worksheets that are not well-designed for programmatic access.

INTRODUCTION

SAS Drug Development (SDD) is designed to provide an environment that supports the execution of SAS code and the storage of data that meets the FDA requirements of a controlled environment. SDD includes a set of several servers (Unix, Oracle®, and MS® Windows®) and runs your SAS code on a Unix server. So, for that code to convert an Excel worksheet to a SAS data set, it is necessary to copy the Excel worksheet to a SDD server running the MS Windows operating system and run the conversion code there too. This paper will give code and instructions to convert complex Excel files into SAS data sets in SDD. The SDD Advanced Loader can convert simple Excel worksheets to SAS data sets, but it has a limited scope of application because it makes many assumptions about the Excel file. Some of these assumptions are: 1) variable/column names in row one of the worksheet, 2) variable values in row two of the worksheet, 3) non-mixed column values, and 4) only one worksheet in a workbook. We will offer solutions to frequent problems incurred by current SDD users when the Excel worksheets do not conform to the assumptions made by the Advanced Loader in SDD.

WHAT DO YOU DO WHEN YOU CAN NOT USE THE SDD ADVANCED LOADER

Working for a company which has used SDD for the past year, we have become quite familiar with the differences in the SDD environment as compared to an environment consisting of PC SAS and mainframe SAS. For us, one of the problematic differences has been converting complicated Excel worksheets into SDD data sets, without first using desktop PC SAS. This paper is designed to address issues that exist when converting Excel worksheets with unusual variable names, unusual variable values, unusual worksheet titles, and with multiple data sets within one worksheet. Now, before we begin, we should note that the term 'unusual' will be defined as any aberration from the assumptions of the SDD Advanced Loader.

Back in the day of PC SAS we used LIBNAME statements and/or PROC IMPORT to convert our data. In SDD we are provided with the Advanced Loader and a Unix execution environment to convert our data. Since SAS code, in SDD, executes on a Unix server it can be difficult to convert Excel worksheets. To successfully convert Excel worksheets requires programmatic access to the worksheets be executed on a MS Windows, i.e., Microsoft remote, server. Therefore, the SDD Advanced Loader must access a MS Windows server to convert Excel worksheets into SAS data sets. As mentioned above, the SDD Advanced Loader makes certain assumptions about the structure of the Excel worksheets you are trying to convert. If the Excel worksheet you are trying to convert is normal (meaning it conforms to all the Advanced Loader assumptions), then the Advanced Loader will convert the worksheet without a problem. However, if your Excel workbook is unusual in some way you will be unable to successfully use the Advanced Loader.

VARIABLE NAMES

Variable names, or column names, can be made up of letters, numbers, special characters, and/or spaces. This combination of text can present problems when trying to convert Excel worksheets into SAS data sets in SDD. This section will address problems importing variable/column names that contain trailing spaces and/or special characters.

The code shown below is used to convert an Excel worksheet to SDD using a LIBNAME statement. The following code is designed to allow the user to only need to specify: 1) the old and new data set names in the macro call of the program, 2) the input data set location in the parameter section, and 3) the output data set location in the parameter section. If you look at Appendix A, Example 1, you will see the before (Excel) and after (SDD) result of executing the following code.

```
libname out "&outlib";
*** This macro is where the input and output data sets are defined;
%macro Excel_test(data=,out=);
data _null_;
    call symput('now',compress(put(datetime()),datetime18.0),': ');
    stop;
run;
*** The following code starts a SAS session on the Microsoft remote server;
signon;
*** Syslput statements are necessary to create macro variables on the Microsoft
remote server;
%syslput Excel_file_sddxpath = %sysfunc(pathname(&Excel_file));
%syslput Excel_file_pcpth = &_sddusr_._&now..xls;
%syslput data="&data.$"n;
%syslput out=&out;
*** The following code executes on the Microsoft remote server;
rsubmit;
    options noxwait xsync;
*** The following step is necessary to copy the Excel workbook to the Microsoft
remote server;
    %nrstr(
        proc upload infile="&Excel_file_sddxpath" outfile="&Excel_file_pcpth"
binary connectstatus=no;
        run;
** This step is necessary when using the LIBNAME statement to convert the file.
The statement allows the previous PROC UPLOAD time to complete the step;
data null;
rc=sleep (5,1);
stop;
run;

*** This is the LIBNAME statement that may or may not be changed depending on
what LIBNAME options are needed to import your data set;
libname in Excel "&Excel_file_pcpth" header=yes;

*** This step is necessary to convert the Excel workbook to SAS and is needed
when using the LIBNAME statement;
data &out;
set in.&data;
run;
    )
*** This step will copy the SAS data set from the Microsoft remote server to
the SDD Unix session;
proc download inlib=work outlib=out connectstatus=no;
```

```

        select &out;
run;
*** This step will clear the LIBNAME and delete the copy of the file from the
Microsoft remote server;
libname in clear;
filename _esrmxls "%nrstr(&Excel_file_pcpath)";
        data _null_;
            rc = fdelete('_esrmxls');
            sysmsg = sysmsg();
            put rc= / sysmsg;
            stop;
        run;
        filename _esrmxls clear;
endrsubmit;
signoff;
%mend Excel_test;

*** This step calls the macro to execute the code. This is where you specify
the Excel worksheet you wish to convert and what you want your new SDD data set
to be named;
%Excel_test(data=Sheet1,out=ex1_data);

```

The SDD parameters for the above code are defined as follows:

#	Var Name	Type	Label	Enabled	Required	Default	Tabname
1	*LOG*	SAS log	---	☐	☐	SAS Log File	Sys. Files
2	*LST*	SAS output	---	☐	☐	SAS Output File	Sys. Files
3	SDDPARMS	Process para...	---	☐	☐	SAS v9 Data...	Sys. Files
4	*PGM*	SAS program	---	☐	☐	SAS Program..	Sys. Files
5	NOW	Text field	---	☒	☒		Parameters
6	EXCEL_FILE	<i>Input file</i>	<i>(File Name)</i>	☐	☐	(name of Excel file).xls	Parameters
7	_SDDUSR_	Text field	---	☒	☒		Parameters
8	EXCEL_FILE_SDDXPATH	Text field	---	☒	☒		Parameters
9	OUTLIB	<i>Folder</i>	<i>Output Folder</i>	☐	☐	(name of output folder)...	Parameters
10	EXCEL_FILE_PCPATH	Text field	---	☒	☒		Parameters
11	DATA	Text field	---	☒	☒		Parameters
12	OUT	Text field	---	☒	☒		Parameters

The code above in red italics is the code that you may need to change when working with this program. The macro parameters listed at the beginning of the code (%macro Excel_test(*data=,out=*);), and used in the macro call (*%Excel_test(data=Sheet1,out=ex1_data)*);), are defined as: *data=* the name of the worksheet in the Excel data set that you are trying to convert, i.e., sheet 1, and *out=* the name you are specifying for the new SAS data set to be stored in SDD.

In the SDD parameter section, the **EXCEL_FILE** parameter is the Excel workbook you are trying to convert. Please note that when converting a workbook, you need to identify the Parameter 'Type' as an '*Input file*'. Then, customize the Parameter 'Type' by choosing the 'File Type' to be a '*Generic Text File*'. Finally, you need to browse for your Excel workbook location. As a last note make sure to change the 'Files of Type' from 'Generic Text File (*.txt)' to '*All Files*' since your original input file is in *.xls, not *.txt.

The **OUTLIB** parameter is the output folder, or location you plan to store your new SAS data set in SDD. Please note that when converting a workbook, you need to identify the Parameter 'Type' as a 'Folder'. Then, you need to customize the Parameter 'Type' by choosing 'Folder Type' equal to 'Output'. Finally, remember to browse for your SDD data set output location.

In the SDD parameter section, in addition to the previous adjustments you need to make, you also need to disable the macro variables used in the analysis, i.e., those that are blacked out (5, 7, 8, 10, 11, & 12).

As stated before, the above code is designed to import Excel worksheets with unusual variable/column names. By adding **header=yes** in the above LIBNAME statement, the first row of the Excel worksheet will be used to define the names of the variables in the SAS data set. You will also be able to import variable names that include special characters not allowed in SAS; however, please note that every special character will import as an underscore. If you prefer special characters in your variable names, you will need to go back and use the 'rename' statement to rename your variables after they are converted into SDD.

VARIABLE VALUES

There are two types of SAS variables, character and numeric. Often we will have a variable, i.e., column, with data that has a mixture of these types of values. By default, SDD is set up to scan the first 8-rows of a column which will determine the length of the variable value and whether the variable is defined as character or numeric. This can be problematic if your first 10 values are 'No' followed by a value of 'Yes' in a subsequent row or if your first 15 values are 'No' followed by a row that contains a value of '8'. By default, SAS will truncate the longer variable values and assign missing values to numeric variables in rows where character values exist.

When you have mixed variable value types, what is needed to correct this problem is to add the code: **mixed=yes**, to the code shown in the 'Variable Names' section after the code **header=yes** in the LIBNAME statement, as shown below:

```
libname in Excel "&Excel_file_pspath" header=yes mixed=yes;
```

By adding **mixed=yes** to your code, this will allow your program to import columns containing both character and numeric values by defining the SAS variable as a character type. If you look at Appendix A, Example 2, you will see the before (Excel) and after (SDD) result of executing the program.

Next, there is still the problem of variable value truncation when longer values exist after the first 8-rows as compared to values on or before row 8. To correct this problem you will need to contact your SDD support staff to reconfigure the Microsoft remote server. Currently, our company has corrected this problem and SDD will scan all rows of data before determining the length and type of the variable. This correction is not something that can be individually corrected on a desktop PC. This correction needs to be changed on the PC, server by your SDD support staff. In order to implement this change, you can contact your support staff and refer them to the registry URLs in the Recommended Reading section of this paper.

In addition to variable names and values, your Excel workbook may also have multiple worksheets. Using the same code shown in the 'Variable Names' section, you can convert multiple Excel worksheets using whatever data set name you specify to the **out=** macro parameter. In the above code, when you call the macro, you can specify the Excel worksheet (data=) and the SAS output data set name (out=). An example of this code is:

```
%Excel_test(data=Sheet1,out=ex2_data1);  
%Excel_test(data=Sheet2,out=ex2_data2);
```

Next, we will show another way to import Excel worksheets using PROC IMPORT. Both methods are useful, and which method you chose should be determined by the user's preference.

WORKSHEET NAMES

Worksheet names can become complicated too. In the following code, we demonstrate how to convert an Excel workbook with complicated worksheet names into SDD using PROC IMPORT. As you will see, the majority of the code is the same whether you use PROC IMPORT or a LIBNAME statement. However, if you are already familiar with the PROC IMPORT options for converting Excel workbooks into PC SAS, you may be more comfortable using the following code. The code below shows what to do when you have trouble with special characters, missing spaces, and trailing spaces in worksheet names.

```
libname out "&outlib";  
  
%macro Excel_test(data=,out=);  
data _null_;
```

```

    call symput('now',compress(put(datetime()),datetime18.0),': ');
    stop;
run;
signon;

%syslput Excel_file_sddxpath = %sysfunc(pathname(&Excel_file));
%syslput Excel_file_pcpth = &_sddusr_._&now..xls;
%syslput data=&data;
%syslput out=&out;

rsubmit;
    options noxwait xsync;
    %nrstr(
        proc upload infile="&Excel_file_sddxpath" outfile="&Excel_file_pcpth"
binary connectstatus=no;
        run;
    )

*** This is where the code changes from the previous examples. In this
example, we are using PROC IMPORT to perform the task. We have removed the
'data null', 'libname in', and 'data &out' command string from the previous
text and replaced them with the following PROC IMPORT command;
    PROC IMPORT OUT = &out DATAFILE= "&Excel_file_pcpth" DBMS=EXCEL REPLACE;
    GETNAMES=YES;
    MIXED=YES;
    SHEET="&data.$";
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
    RUN;

proc download inlib=work outlib=out connectstatus=no;
    select &out;
run;

filename _esrmxls "%nrstr(&Excel_file_pcpth)";
data _null_;
    rc = fdelete('_esrmxls');
    sysmsg = sysmsg();
    put rc= / sysmsg;
    stop;
run;
filename _esrmxls clear;
endrsubmit;
signoff;
%mend Excel_test;

```

```

*** In this code, %str is required to convert the special character and/or
trailing spaces at the end of the worksheet name;
%Excel_test(data=%str(Day-1),out=ex3_Day01);
%Excel_test(data=Day 2,out=ex3_Day02);
%Excel_test(data=%str(Day3 ),out=ex3_Day03);

```

The SDD parameters for the above code are the same as what was shown before in the 'Variable Names' section of the text. The code above in red italics is the code that you may need to change when you are working with this program.

As mentioned before, the above code converts unusual Excel worksheet names (Day-1, Day 2, and Day3) into SAS data sets within SDD. While most of this new code is the same as the code in the above 'Variable Names' section, an important difference is the macro call. In the first and third macro calls in the 'Worksheet Names' section, you will see that special characters and trailing spaces are shown in the original worksheet name. This problem is corrected by using the string command (*%str*) which allows the code to read worksheet names with both special characters and/or trailing spaces. If you look at Appendix A, Example 3, you will see the before (Excel) and after (SDD) result of executing this code. As a side note, one should be aware that not all special characters in worksheet names can be converted into SAS datasets. This topic will be discussed in more detail later in this paper.

MULTIPLE DATA SETS WITHIN ONE WORKSHEET... AND VARIABLE VALUES THAT DO NOT START ON ROW 2

Our final example for converting Excel worksheets to SAS data sets in SDD involves having multiple cell ranges within a single worksheet and/or converting variable values that do not start on row 2 in the Excel worksheet. The way to resolve these two issues is to use a range statement. Using the above code from the 'Worksheet Names' section, you will need to make a few changes. To begin with, the new macro will include a range parameter:

```
%macro Excel_test(data=,range=,out=);
```

This new macro parameter is used to specify the range of the data set within the Excel worksheet that you are trying to convert to SDD. The other parameters (data=, out=) remain the same as in previous examples. The next piece of code that will change is the addition of another syslput statement in the code (see below).

```
%syslput range=&range;
```

This code should be placed with the other syslput statements (data, out). Next, you need to change the PROC IMPORT section of the code to:

```

PROC IMPORT OUT = &out DATAFILE= "&Excel_file_pspath" DBMS=EXCEL REPLACE;
    GETNAMES=YES;
    MIXED=NO;
    RANGE="&data.&range";
    SCANTEXT=YES;
    USEDATE=YES;
    SCANTIME=YES;
RUN;

```

As you can see, the code has replaced the SHEET="&data.\$" (from the Worksheet Name section) with RANGE="&data.&range". You should also observe that MIXED=YES has been changed to MIXED=NO; this is because the variables in these data sets do not contain any mixed variable values, meaning each column's values are either all character or all numeric. Finally, the last change is the macro call (see below).

```

%Excel_test(data=Sheet1,range=$B3:G14,out=ex4_data1);
%Excel_test(data=Sheet1,range=$F18:J29,out=ex4_data2);

```

As you can see, *data=* is still the Excel worksheet name and *out=* is still the name of your new SAS data set in SDD. The new addition, *range=*, is the range of the data from the beginning variable name to the last variable value for the data set you are converting. The data in the example above have two ranges within one worksheet; hence the above two macro calls.

Finally, since you have a new parameter in the code, you also have a new parameter to define in the 'Parameter Section' of SDD (see below). As with the other macro parameters (data & out), you will also need to disable the parameter for range. If you look at Appendix A, Example 4, you will see the before (Excel) and after (SDD) result of executing the program involving multiple data sets within a single worksheet.

#	Var Name	Type	Label	Enabled	Required	Default	Tabname
11	DATA	Text field	---	☒	☒		Parameters
12	RANGE	Text field	---	☒	☒		Parameters
13	OUT	Text field	---	☒	☒		Parameters

The ability to resolve issues pertaining to a specific range of data can be very useful. For example, you may have multiple data sets within one Excel worksheet or you may have an Excel worksheet with variable labels in row 1, variable names in row 2, and variable values in row 3. Many issues develop out of an inability to specify exactly what you need to convert. In the above sections we have presented solutions.

ADDITIONAL COMPLICATIONS IN SDD

In the above sections, we have shown ways to resolve issues dealing with variable names, variable values, worksheet names, and multiple data sets within a single worksheet. In this section we will address data issues that, at this time, continue to be problematic when trying to convert Excel worksheets to SAS data sets in SDD.

EXCEL 2007

At this time, you cannot convert Microsoft Excel 2007 worksheets to SAS data sets in SDD.

SDD FILE TYPES

When trying to convert Excel worksheets to SAS data sets within SDD, you need to specify the Parameter Type as 'Input file' in the parameter section of SDD. Once you choose *Input file*, you must then customize the file by determining the 'File Type' that you are trying to import. As you can see below, an Excel file (.xls) is not an option under 'File Type' (see below).

The screenshot shows the 'Customizing Input file - EXCEL_FILE' dialog box. The 'File Type' dropdown is highlighted with a red circle, showing 'Generic Text File' selected. Other options in the dropdown include XML File, SAS Program File, Generic Binary File, Tab-Delimited File, Delimited File, and Comma-Separated Value File. The 'Path Option' is set to 'SAS Program File' and the 'Base Path' is 'Generic Binary File'. The 'Default' is 'Tab-Delimited File'. The 'Update' checkbox is unchecked. The 'Clear' button is visible on the right. The 'OK' and 'Cancel' buttons are at the bottom.

Thus, at this time, the best choice is *Generic Text File*. If an Excel file was an option, we may be able to convert our Excel worksheets to SAS data sets in SDD with greater ease and fewer complications.

PROBLEMS WITH SPECIAL CHARACTERS

Special characters can be tricky in SDD. Sometimes they work and sometimes they do not. As mentioned earlier, not all special characters in Excel worksheet names can be converted into SAS data sets in SDD. Listed below are examples of what does and does not work when referencing worksheet names:

1. The special characters/data anomalies that **are** recognized in Excel to SDD data transfers are: spaces within names (Day 2), trailing spaces (Day3), at signs (@), dollar signs (\$), carrots (^), ampersands (&), and dashes (-).
2. The special characters that **are not** recognized in Excel to SDD data transfers are: pound signs (#) and exclamation points (!).
3. The special characters that will **prevent your code from running** in Excel to SDD data transfers are: single or double parentheses () and percent signs (%).

Further, SAS will not allow you to use special characters in your new SAS data set names within SDD. The only allowable special character is an underscore.

Variable names that contain special characters can be converted from Excel worksheets to SAS data sets in SDD. However, all special characters will be changed to an underscore which, if you prefer, can be renamed after the data is converted to SAS data sets in SDD.

Variable values that contain special characters can also be converted from Excel worksheets to SAS data sets in SDD. However, you need to make sure to use the MIXED=YES command to ensure all your data points (i.e., special characters, numbers, and characters) are converted into SAS/SDD.

CONCLUSION

We have offered you solutions to converting complicated Excel worksheets into SAS data sets using SAS code that executes on SDD servers. We have demonstrated ways to resolve problems with variable names, variable values, worksheet names, and multiple data sets within a single Excel worksheet. We have offered you examples of problems still waiting to be resolved in hopes that you will not spend needless hours trying to make something work that cannot work.

In conclusion, Microsoft Excel is a useful tool for end-users that gives them a great deal of flexibility. This flexible aspect makes Excel inappropriate to use as a database for the exchange of data, especially data that is as important as clinical trial data. Microsoft Excel is not a database nor is it a file format designed for programmatic access. Instead, Excel is a Microsoft product that is programmatically accessible only on a Microsoft Windows server using the Microsoft API, i.e., MS Jet. This interface is the underlying cause of the problems of converting Excel worksheets to SAS data sets. Solving the problems of converting Excel worksheets to SAS data sets is solving the wrong problem. The real solution is to stop using Excel in a way that it was not designed to be used, e.g., as a database. There are many other formats to consider that are designed for data exchange. The industry standards evolving in the pharmaceutical industry are based on xml schemas, e.g., CDISC ODM. Even the use of Microsoft Access[®] is preferable to Excel, since MS Access types its columns. But, whatever your preference of file format is, it is critical that the recipients of the data set define the format of that data set to the supplier of the data points, prescriptively before the data set is sent from supplier to recipient.

We have to evolve beyond the state of affairs where data files show up on a programmer's desk one day, with no prior planning or knowledge. The programmer is then asked to deal with the file, and insert the data into the analysis of a clinical trial. This data leads to critical decisions about drug safety and efficacy. Thus, this clinical trial data requires better planning and specification so that the problems described in this paper do not exist. We hope that this paper helps that programmer deal with the unexpected Excel file, but we hope, more so, that an unplanned file in a difficult to use format never appears in a database as critical as the database that contains clinical trial data.

ACKNOWLEDGMENTS

We would like to acknowledge Jeffery T. Heath for his expert graphic skills. Thank you for your work on the Appendix, we really appreciate it!

RECOMMENDED READING

Heaton, Ed (2006). *So, Your Data are in Excel!* Retrieved June 24, 2009, from SUGI 31 Proceedings: Applications Development Web site: <http://www2.sas.com/proceedings/sugi31/020-31.pdf>

Registry sites:

<http://support.sas.com/kb/6/123.html>

<http://support.sas.com/kb/31/765.html>

<http://support.sas.com/kb/19/409.html>

Other useful locations to learn more about Excel & SDD:

<http://blogs.msdn.com/Excel/archive/2006/07/20/671995.aspx>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/PDF/default/acpcref.pdf>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/HTML/default/a003103761.htm#a003112210>

<http://www2.sas.com/proceedings/sugi31/020-31.pdf>

<http://support.sas.com/kb/19/550.html>

<http://support.sas.com/kb/16/812.html>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/HTML/default/a002644676.htm>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/HTML/default/a003102095.htm>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/HTML/default/acpcrefwhatsnew902.htm>

<http://support.sas.com/documentation/cdl/en/acpcref/61891/HTML/default/a003293791.htm>.

CONTACT INFORMATION

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

Heather L. Murphy
Eli Lilly and Company
Lilly Corporate Center, DC 6025
Indianapolis, Indiana 46285
Work Phone: (317) 277-2845
E-mail: murphyhe@lilly.com

Gregory C. Steffens
Eli Lilly and Company
Lilly Corporate Center, DC 6158
Indianapolis, Indiana 46285
Work Phone: (317) 651-4857
E-mail: steffensgc@lilly.com

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.

Excel®, Microsoft®, MS®, Windows®, and Access® are registered trademarks of the Microsoft Corporation, in the United States and other countries.

UNIX® is a registered trademark of The Open Group.

Oracle® is a registered trademark of the Oracle Corporation and/or its affiliates.

Appendix A:

MS Excel 2003:

	A	B	C	D	E	F
1	Auburn	is	awe-some	&	#1	
2	10	9	32	87	65	
3	5	4	98	75	12	
4	10	9	32	87	65	
5	5	4	98	75	12	
6	10	9	32	87	65	
7	5	4	98	75	12	
8						

Example 1: Variable Names

SDD:

#	AUBURN	IS__	AWE_SOME	_	_1
1	10	9	32	87	65
2	5	4	98	75	12
3	10	9	32	87	65
4	5	4	98	75	12
5	10	9	32	87	65
6	5	4	98	75	12

MS Excel 2003:

	A	B	C	D	E	F	G
1	time1	time2	time3	time4	date	Handedness	
2	38	95	206	427	09-Mar-09	left	
3	23	77	242	404	12-Mar-09	left	
4	6	95	224	424	12-Mar-09	left	
5	47	79	221	404	14-Mar-09	left	
6	11	82	222	421	24-Mar-09	left	
7	21	53	222	419	23-Mar-09	left	
8	30	55	221	415	21-Mar-09	left	
9	39	78	204	411	09-Mar-09	left	
10	22	71	248	422	08-Mar-09	left	
11	35	82	242	403	04-Mar-09	left	
12	19	58	232	409	03-Mar-09	left	
13	47	71	244	430	01-Mar-09	left	
14	unk	88	213	429	15-Mar-09	left	
15	10	51	238	413	01-Apr-09	left	
16	7	59	204	422	24-Apr-09	left	
17	28	95	225	419	12-Apr-09	left	
18	17	68	245	407	16-Apr-09	left	
19	6	-63	249	416	12-Jun-09	left	
20	36	71	250	#421	23-Jul-09	left	
21	3	83	216	411	21-Mar-09	left	
22	31	85	243	#306	16-Feb-09	left	
23	-32.08	79	unk	402	xx-Mar-09	right	
24	35	94	226	408	23-Mar-09	right	
25	1	650001	unkn	411	xx-xxx-09	left	
26	15.33333	58	245	417,218	21-Jul-09	right-handed	
27	52	976	230	123	x-x-72	left	
28							

Example 2: Variable Values

SDD:

#	TIME1	TIME2	TIME3	TIME4	DATE	HANDED...
1	38	95	206	427	9-Mar-09	left
2	23	77	242	404	12-Mar-09	left
3	6	95	224	424	12-Mar-09	left
4	47	79	221	404	14-Mar-09	left
5	11	82	222	421	24-Mar-09	left
6	21	53	222	419	23-Mar-09	left
7	30	55	221	415	21-Mar-09	left
8	39	78	204	411	09-Mar-09	left
9	22	71	248	422	08-Mar-09	left
10	35	82	242	403	04-Mar-09	left
11	19	58	232	409	03-Mar-09	left
12	47	71	244	430	01-Mar-09	left
13	unk	88	213	429	15-Mar-09	left
14	10	51	238	413	01-Apr-09	left
15	7	59	204	422	24-Apr-09	left
16	28	95	225	419	12-Apr-09	left
17	17	68	245	407	16-Apr-09	left
18	6	-63	249	416	12-Jun-09	left
19	36	71	250	#421	23-Jul-09	left
20	3	83	216	411	21-Mar-09	left
21	31	85	243	#306	16-Feb-09	left
22	-32.08	79	unk	402	xx-Mar-09	right
23	35	94	226	408	23-Mar-09	right
24	1	650001	unkn	411	xx-xxx-09	left
25	15.33333...	58	245	417,218	21-Jul-09	right-handed
26	52	976	230	123	x-x-72	left

MS Excel 2003:

	A	B	C	D
1	day	month	code	
2		1	8 a	
3		2	7 r	
4		3	6 s	
5		4	5 d	
6		5	4 e	
7		6	3 erw	
8		7	2 we	
	Day-1	Day 2	Day3	

Example 3: Worksheet Names

SDD:

Run of New Job on Thu Jun 25 09:44:2

- excel2sdd_sesug_ex3p.sas
 - Input Files
 - Output Files
 - Output Folder
 - ex3_day01.sas7bdat-<
 - ex3_day02.sas7bdat-<
 - ex3_day03.sas7bdat-<
 - System Files
 - excel2sdd_sesug_ex3p.sas-<

* Process Editor - excel2sdd_sesug_ex3p

```

%mend excel test;

%excel_test(data=%str(Day-1),out=ex3_Day01);
%excel_test(data=Day 2,out=ex3_Day02);
%excel_test(data=%str(Day3 ),out=ex3_Day03);
  
```

MS Excel 2003:

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3		Name	Subjid	Age	Gender	Race	DMType				
4		Jeff	1	35	M	Caucasion	1				
5		Pete	2	42	M	Black	1				
6		Sarah	3	31	F	Hispanic	1				
7		Peter	16	43	M	Asian	2				
8		Jan	17	39	F	Caucasion	1				
9		Kim	18	47	F	Black	1				
10		Curtis	19	59	M	Hispanic	2				
11		Mavis	20	57	Male	Asian	2				
12		Melissa	21	32	F	Caucasion	1				
13		Jefferson	22	34	Female	Black	2				
14		Scott	23	53	M	Hispanic	1				
15											
16											
17											
18						Score1	Score2	Score3	Score4	Score5	
19						23	54	123	87	567	
20						43	541	-64	876	345	
21						53	345	-8766	12	23	
22						234	76	45.001	34	756	
23						6	3546	867	45	4	
24						467	98	4	867	54	
25						234	2354	78	34	876	
26						113	76	8	45	3	
27						65	2	34	897	456	
28						0.33333333	6	1	56	85	
29						75	-7	678	345	34	
30											
31											

Example 4: Multiple Data Sets Within One Worksheet

SDD:

#	SCORE1	SCORE2	SCORE3	SCORE4	SCORE5
1	23	54	123	87	567
2	43	541	-64	876	345
3	53	345	-8766	12	23
4	234	76	45.001	34	756
5	6	3546	867	45	4
6	467	98	4	867	54
7	234	2354	78	34	876
8	113	76	8	45	3
9	65	2	34	897	456
10	0.33333333...	6	1	56	85
11	75	-7	678	345	34

#	NAME	SUBJID	AGE	GENDER	RACE	DMTYPE
1	Jeff	1	35	M	Caucasion	1
2	Pete	2	42	M	Black	1
3	Sarah	3	31	F	Hispanic	1
4	Peter	16	43	M	Asian	2
5	Jan	17	39	F	Caucasion	1
6	Kim	18	47	F	Black	1
7	Curtis	19	59	M	Hispanic	2
8	Mavis	20	57	Male	Asian	2
9	Melissa	21	32	F	Caucasion	1
10	Jefferson	22	34	Female	Black	2
11	Scott	23	53	M	Hispanic	1