Highlighting Inconsistent Record Entries in EXCEL: Possible with SAS® ODS, Optimized in Microsoft® DDE

Perry Watts, Independent Consultant Fox Chase Cancer Center

Abstract

While ODS-HTML output to EXCEL easily accommodates column-based color highlighting with format invocations in PROC REPORT, the method does not work when color assignments for record inconsistencies depend on relative values in multiple columns. If DATE1 is supposed to come before DATE2, for example, the following DEFINE statements will not work as intended:

```
define date1/ display style=[background=$ColorF.];
define date2/ display style=[background=$ColorF.];
```

Inputs to these formats are the dates supplied by DATE1 and DATE2, not the required row and column identifiers needed for assigning colors based on the relationship between two variables in a single observation.

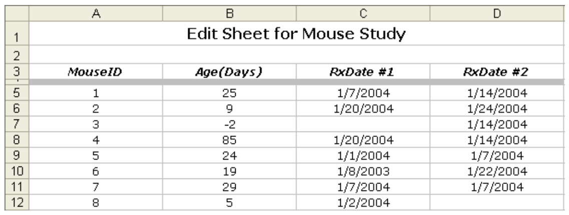
This paper describes two methods for highlighting inconsistent record entries in an EXCEL spreadsheet. The first method uses PROC REPORT to generate ODS-HTML output to EXCEL. However, the column restrictions in the DEFINE statement described above are circumvented by removing the formats and invoking a macro that assigns background colors from a COMPUTE block. Unfortunately PROC REPORT becomes prohibitively slow when the formats are removed. An alternative solution using Dynamic Data Exchange (DDE) with EXCEL version 4 Macro Language (X4ML) is described. Not only is DDE faster than ODS-HTML, but the formatted output is superior.

Mouse Data Set: Before and After Color Highlighting with ODS-HTML and DDE

A very simple data set, MOUSEDAT, containing 4 columns of data for eight mice is sufficient for explaining the concepts described in this paper. Age is expected to be between 11 and 29 days, and RxDate#1 comes before RxDate#2. Both treatments occur in 2004. Figure 1 shows the data in plain and highlighted formats.

Figure 1. Data entry errors are more easily detected when MOUSEDAT is highlighted in EXCEL with ODS-HTML or DDE.

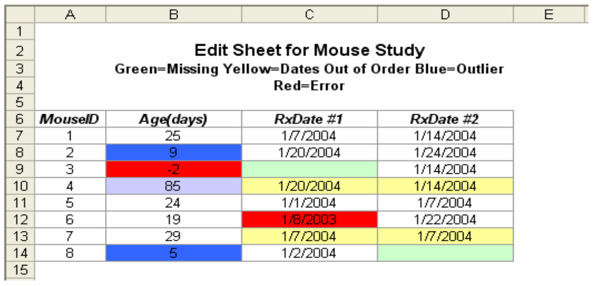
Before Highlighting: Superficially, the data look fine with the exception of a couple of missing values and a negative AGE for Mouse #3.



Edit Sheet for Mouse Study

MouseID	Age(Days)	RxDate #1	RxDate #2
1	25	1/7/2004	1/14/2004
2	9	1/20/2004	1/24/2004
3	-2		1/14/2004
4	85	1/20/2004	1/14/2004
5	24	1/1/2004	1/7/2004
6	19	1/8/2003	1/22/2004
7	29	1/7/2004	1/7/2004
8	5	1/2/2004	

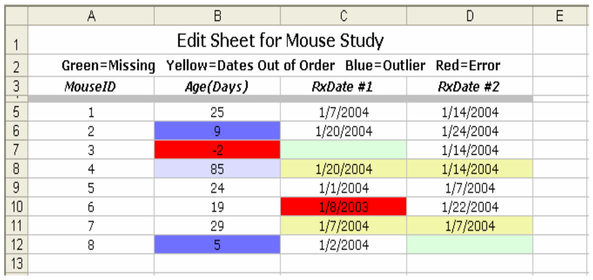
After with ODS-HTML: The data reveal additional problems. Some mice are either too young or too old. Others have treatment dates that are out of order. Note too that treatment dates for mouse #6, age 19 days, are over a year apart.



Edit Sheet for Mouse Study
Green=Missing Yellow=Dates Out of Order Blue=Outlier Red=Error

MouseID	Age(days)	RxDate #1	RxDate #2
1	25	1/7/2004	1/14/2004
2	9	1/20/2004	1/24/2004
3	-2		1/14/2004
4	85	1/20/2004	1/14/2004
5	24	1/1/2004	1/7/2004
6	19	1/8/2003	1/22/2004
7	29	1/7/2004	1/7/2004
8	5	1/2/2004	

After with DDE: The same output is generated, but the titles are properly centered over columns A-D, and all cells in the spreadsheet are outlined. DDE and ODS-HTML colors also match perfectly.



Edit Sheet for Mouse Study
Green=Missing Yellow=Dates Out of Order Blue=Outlier Red=Error

MouseID	Age(Days)	RxDate #1	RxDate #2
1	25	1/7/2004	1/14/2004
2	9	1/20/2004	1/24/2004
3	-2		1/14/2004
4	85	1/20/2004	1/14/2004
5	24	1/1/2004	1/7/2004
6	19	1/8/2003	1/22/2004
7	29	1/7/2004	1/7/2004
8	5	1/2/2004	

Out-of-range values for AGE in MOUSEDAT can easily be handled by a format associated with a column in ODS-HTML. However, formats cannot be used to determine when individual treatment dates are out of order. Format *ranges* are always column-bound, but the highlights in the AFTER panels in Figure 1 are based on the relationship *between* two dates. Note, for example, that the RXDATE#2 with the same value is highlighted for Mouse #4, but it is not for Mice #1 and #3. Again, what is being evaluated is the relationship between dates not their absolute values. Relationship type errors such as the ones displayed in Figure 1 are likely to occur when data are manually entered from paper forms into EXCEL workbooks lacking validity checks.

Working with ODS-HTML in Version 8.2

Chevell Parker describes how to use PHTML and HTML *tagsets* to generate an EXCEL spreadsheet in his paper *Generating Custom Excel Spreadsheets using ODS* **[9],p.1**. PHTML, undefined in the paper and "experimental" in SAS 8.2, fails to transfer assigned background colors to the EXCEL file. This failure may be replicated in SAS 9.1, since PHTML has been described elsewhere as a limited *tagset* producing "a basic HTML that uses twelve style elements and no class attributes" **[10],p.6**. Fortunately, the HTML *tagset* is sufficient for producing the highlighted output shown in the lower-left panel of Figure 1:

```
ods html
  file="c:\N04\BgHighlight\XLS\MouseChkODSnewest.xls" ❶
  STYLE=styles.Withborder; ❷
title1 "<td COLSPAN=3 align=center><font size=3>
  <b>Edit Sheet for Mouse Study  </b></font></td>"; ❸
title2 "<td align=Center COLSPAN=3><font size=2>
  <b>Green=Missing   Yellow=Dates Out of Order   Blue=Outlier   Red=Error
  </b></font></td>"; ❹
```

❶ The only requirement for redirecting output to an EXCEL file is to specify an XLS extension.

❷ WITHBORDER inherits all properties from the MINIMAL style. Only BORDER is changed from thick to thin and from black to gray. The goal here is to make the output blend seamlessly into EXCEL. WITHBORDER almost succeeds except that borders are only defined for the output data set and headers, not the entire spreadsheet. Below is the source code for the WITHBORDER style:

```
proc template;
  define style styles.Withborder;
  parent=styles.minimal;
    style table /
      borderwidth=1
      bordercolor=#909090
      rules=all;
  end;
run;
```

Initial background colors are assigned properly in the MINIMAL style and its derivative, WITHBORDER. This means that the ODS-HTML generated spreadsheet inherits color settings from the DISPLAY device in the WINDOWS Control-panel. Some Windows users substitute light gray for white backgrounds in order to reduce screen glare.

❸ Columns B-D are spanned for the titles, not A-D as intended. If COLSPAN were set to 4, the output would even be more askew.

❹ Regardless of the value assigned to FONT SIZE, two lines of output are generated instead of one.

Highlighting Inconsistent Record Entries in ODS-HTML

Now that the ODS destination commands have been issued, erroneous, missing and inconsistent values for RXDATE1 and RXDATE2 need to be color-coded for easy detection. The three steps required for completing this task are similar to those used for constructing the color charts shown in the NESUG-15 paper *Working with RGB and HLS Color Coding Systems in SAS® Software* **[15]Watts,pp.2-3**.

1) Create the MOUSECOLORS data set that assigns background colors to RXDATE1 and RXDATE2 by MOUSEID. MOUSEID ranges from 1 to *n*, the number of mice in a study.
2) Derive macro variables from MOUSECOLORS that are resolved in a SELECT macro issued from a compute-block in PROC REPORT.
3) Invoke PROC REPORT to create the spreadsheet.

Step #1: Create the MOUSECOLORS Data Set

The background colors assigned in MOUSECOLORS are derived from the relative values for TEST1DT and TEST2DT in MOUSEDAT:

```
data MouseColors(keep=MouseID t1Colr t2Colr);
  length green default yellow red $8;
  retain default "&defaultColor"; ❶
  retain green    "CXDCFFDC";     *-- MISSING VALUES;
  retain Yellow   "CXF2F6A8";     *-- Test2dt < Test1dt;
  retain Red      "CXFF0000";     *-- ERRONEOUS VALUES;
  array NV{*} test1dt test2dt;    *-- Numeric Variable;
  array NVColr{*} $8 t1Colr t2Colr;
  set mouseDat;

  *-- COLOR MISSING AND ERRONEOUS VALUES WITH ARRAYS;
  do i=1 to dim(NV);
    if NV[i] eq . then NVColr[i]=green;
    else if NV[i] lt '31DEC2003'd then NVColr[i]=Red;
    else NVColr[i]=default; ❷
  end;
  *-- IDENTIFY LATERAL INCONSISTENCIES WITH CROSS-VARIABLE CHECKING;
  if Test2dt le Test1dt and Test1dt gt '31DEC2003'd and Test2dt gt '31DEC2003'd then
    do;
      t1Colr=Yellow; t2Colr=Yellow;
    end;
run;
```

❶ While the WITHBORDER style will assign default background colors properly, an assignment of MISSING to either T1COLR or T2COLR in the MOUSECOLORS data set will generate an ERROR message in the LOG. This bug is difficult to detect. Despite the error messages, the output looks fine! The macro variable DEFAULTCOLOR is set at the top of the program. It is also used in the format for highlighting AGE in PROC REPORT.

❷ Here is where the default color is assigned. Unlike ODS-HTML, DDE does not have a problem with default colors. Color values only need to be assigned to cells requiring highlighting.

Step #2: the SELECT Macro

The SELECT macro invoked from a compute block in PROC REPORT assigns correct background colors to Columns #3 (TEST1DT) and #4 (TEST2DT) with a SELECT statement:

```
%macro select;
 select (MouseId); ❶
  %do i = 1 %to &nobs;
    when (&i.)
      do;
        CALL DEFINE("test1Dt" ,"STYLE", "STYLE=[ BACKGROUND=&&Column3_&i. ]"); ❶ ❷
        CALL DEFINE("test2Dt" ,"STYLE", "STYLE=[ BACKGROUND=&&Column4_&i. ]"); ❶ ❷
      end;
  %end;
  otherwise;
 end;
%mend select;
```

❶ CALL DEFINE used within a COMPUTE block "sets the value of an attribute for a particular column in the current row" **[12]SAS Institute,p.899**. The syntax for the command is:

CALL DEFINE (*column-id, 'attribute-name', value*);

In this application, there is a one-to-one correspondence between MOUSEID and the macro variable ***i***. This means that each time a data record is processed in PROC REPORT, a second poll of the same data set must be taken inside the COMPUTE block to match MOUSEID with ***i.*** Processing can get bogged down when thousands of records are involved. One way to save time is to assign colors for both variables in a single loop. This is done by defining *column-ids* in the CALL DEFINEs as quoted constant literals pointing to the variables that are being highlighted. The automatic variable _COL_ does not work as a *column-id*, because it would reference MOUSEID not TEST1DT or TEST2DT. For additional information about COMPUTE blocks in PROC REPORT, see **[1]Buchanan,pp.298-300** and **[7]Pass,pp.5-6**.

❷ Macro variables must be used for assigning background colors in a COMPUTE block, because the STYLE= *value* in CALL DEFINE calls for unquoted constant text, not a SAS variable.

Step #3: Invoking PROC Report

Here is how the background colors in the first *after* panel in Figure 1 are derived from PROC REPORT in ODS:

```
ODS HTML … ;
proc report data=mouseDat nowindows
  style(Header)=[font_weight=BOLD font_style=Italic]; ❶

  columns mouseID age test1Dt test2Dt;
    define mouseID  / display "MouseID"    center;
    define age      / display "Age(days)" center
                     style(column)=[font_weight=Medium background=AgecFmt.]; ❶ ❷
    define test1Dt  / display "RxDate #1"  center;
    define test2Dt  / display "RxDate #2"  center;
  compute MouseId; ❸
    %select
  endcomp;
run;
ODS HTML Close;
```

❶ *Locations* such as HEADER and COLUMN should be specified in the STYLE statements or the output won't be formatted properly. See chapter 32, *The Report Procedure* in the *SAS Procedures Guide* for a complete list of style *locations* **[12],p.890**. Sue Cross demonstrates their use in her paper *Create a Living Report from Various Parts* **[2],pp.367-369**.

❷ Since age anomalies can be determined by the value for AGE alone, a format is sufficient for making background color assignments. Probably it would have been more efficient to include AGE in the COMPUTE block along with TEST1DT and TEST2DT. That way, the binary search required for formatting could have been eliminated.

❸ MOUSEID is needed for determining the correctness of the relative values for TEST1DT and TEST2DT. MOUSEID is the row-identifier, and TEST1DT and TEST2Dt in the SELECT macro are the column-identifiers. Row and Column identifiers, as mentioned in the abstract, are needed for assigning colors based on the relationship between two variables in a single observation.

Communication between SAS and EXCEL in ODS-HTML

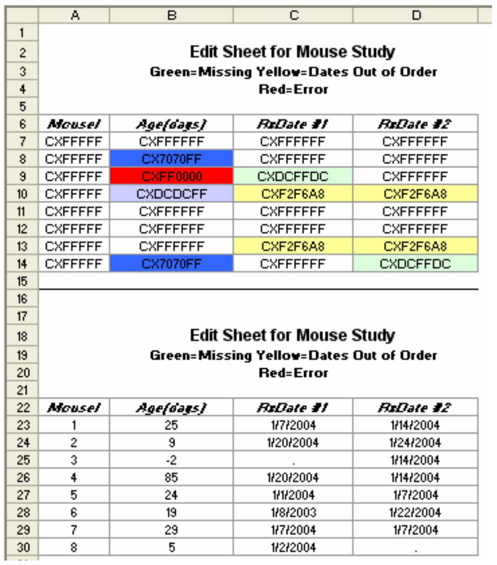
Communication between SAS and EXCEL is non-interactive in ODS-HTML. This means that EXCEL must be closed in order to receive all directives in a single file transfer from SAS. SAS provides a RESULTS VIEWER to compensate for the break in communication. Furthermore, each time a SAS program is re-executed the output XLS file is deleted and recreated afresh, unless EXCEL is left opened by mistake. In those instances an ERROR message is generated and the XLS file is locked.

The ODS-HTML single-pass method just described is not as flexible as DDE where formats and data can be applied in layers to an opened spreadsheet file. Even when MOUSEDAT and MOUSECOLORS are sent to an EXCEL spreadsheet in the *same* ODS-HTML session, they occupy different blocks of cells in the output. Layering has to be achieved manually. To get the layered output in the right panel of Figure 2, the following steps must be completed in EXCEL:

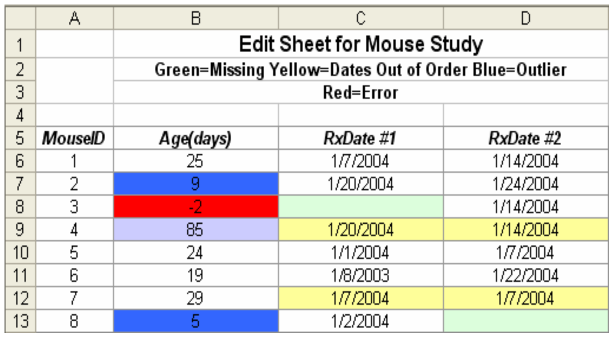
1) Copy and *paste* MOUSEDAT in the lower part of the left-panel to a separate sheet.
2) Copy and *paste-special (formats)* to transfer background colors and formats from MOUSECOLORS to the same location in the target spreadsheet. The HEX color codes in the source spreadsheet are left behind. Only their format that includes *font color* is transferred.
3) Reformat TEST1DT and TEST2Dt as dates in the target spreadsheet. Replace dots (.) with space characters to denote missing values.
4) Re-size the headers so that the titles are fully visible.

Using ODS in the data step to bind data to a table definition created in PROC TEMPLATE doesn't work for layering output either. CELLSTYLE in the Template Procedure's DEFINE COLUMN statement has the same limitation as using a format in a DEFINE statement in PROC REPORT; namely, other columns cannot be referenced when style elements in a given column are defined **[11]SAS Institute,p. 171**.

Figure 2. MOUSECOLORS and MOUSEDAT pictured in the left panel are transferred to EXCEL in a single ODS-HTML session. Once inside EXCEL, spreadsheets are manipulated by hand to produce the layered output shown on the right.



	A	B	C	D
1				
2	Edit Sheet for Mouse Study			
3	Green=Missing Yellow=Dates Out of Order			
4	Red=Error			
5				
6	*MouseI*	*Age(days)*	*RxDate #1*	*RxDate #2*
7	CXFFFFF	CXFFFFFF	CXFFFFFF	CXFFFFFF
8	CXFFFFF	CX7070FF	CXFFFFFF	CXFFFFFF
9	CXFFFFF	CXFF0000	CXDCFFDC	CXFFFFFF
10	CXFFFFF	CXDCDCFF	CXF2F6A8	CXF2F6A8
11	CXFFFFF	CXFFFFFF	CXFFFFFF	CXFFFFFF
12	CXFFFFF	CXFFFFFF	CXFFFFFF	CXFFFFFF
13	CXFFFFF	CXFFFFFF	CXF2F6A8	CXF2F6A8
14	CXFFFFF	CX7070FF	CXFFFFFF	CXDCFFDC
15				
16				
17				
18	Edit Sheet for Mouse Study			
19	Green=Missing Yellow=Dates Out of Order			
20	Red=Error			
21				
22	*MouseI*	*Age(days)*	*RxDate #1*	*RxDate #2*
23	1	25	1/7/2004	1/14/2004
24	2	9	1/20/2004	1/24/2004
25	3	-2	.	1/14/2004
26	4	85	1/20/2004	1/14/2004
27	5	24	1/1/2004	1/7/2004
28	6	19	1/8/2003	1/22/2004
29	7	29	1/7/2004	1/7/2004
30	8	5	1/2/2004	.



	A	B	C	D
1		Edit Sheet for Mouse Study		
2		Green=Missing Yellow=Dates Out of Order Blue=Outlier		
3		Red=Error		
4				
5	*MouseID*	*Age(days)*	*RxDate #1*	*RxDate #2*
6	1	25	1/7/2004	1/14/2004
7	2	9	1/20/2004	1/24/2004
8	3	-2		1/14/2004
9	4	85	1/20/2004	1/14/2004
10	5	24	1/1/2004	1/7/2004
11	6	19	1/8/2003	1/22/2004
12	7	29	1/7/2004	1/7/2004
13	8	5	1/2/2004	

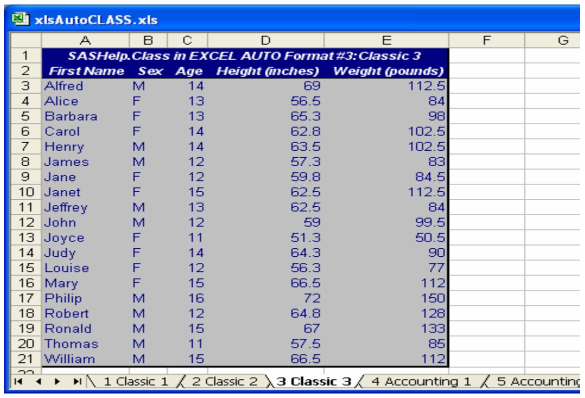
Additional Problems with ODS-HTML and ODS-XML

The ODS-HTML protocol also lacks versatility. Chevell Parker points out that XML rather than HTML must be used to modify *page setup*, generate and name multiple worksheets in a workbook, add formulas, select cells, generate charts, and scale printed output; all very basic operations **[9],p 4.**

However, XML also has its limitations. Vincent DelGobbo in *From SAS to Excel via XML* mentions that graphics transfers to EXCEL via XML are not supported by Microsoft **[4],p.3**, and that there are limitations to using the new Version 9 ExcelXP *tagset*. For example, while it is no longer necessary to work directly with XML in Version 9, DelGobbo cautions that the new *tagset* cannot be used to resize text cells **[4],p.6** or to configure non-rectangular data **[4],p.11**. Also the user does not have full latitude in naming columns or selecting worksheet names. Instead, column headers are derived from variable labels or column numbers **[4],p11**, and EXCEL sheet names must reflect SAS data set names in their displays **[4],p.9**. With these requirements, a serious workaround would have to be developed to accommodate the 18-sheet DDE display of SASHELP.CLASS shown in Figure 3. While XML-compatible *labels* are used as column headers in Figure 3, variable *names* would work just as well in a DDE display. The SASHELP.CLASS workbook can be downloaded from the file attachment associated with the companion paper, *Using Single-Purpose SAS® Macros to Format EXCEL Spreadsheets with DDE* **[14] Watts.**

Figure 3. DDE allows the user full latitude in naming spreadsheets. In this display 18 different EXCEL auto-formats are applied to the SASHELP.CLASS data set by exercising DDE commands inside a SAS program.

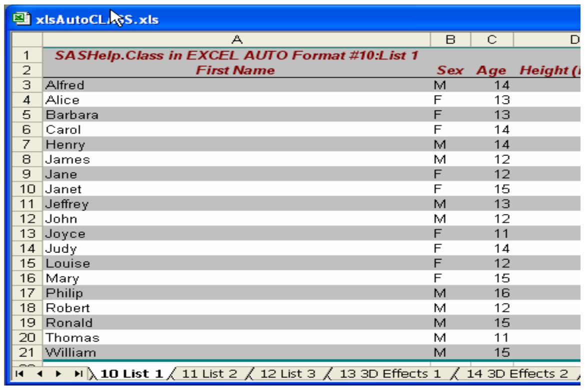
DDE number 3 references the *Classic 3* EXCEL auto-format.



xlsAutoCLASS.xls

	A	B	C	D	E	F	G
1	*SASHelp.Class in EXCEL AUTO Format #3:Classic 3*						
2	*First Name*	*Sex*	*Age*	*Height (inches)*	*Weight (pounds)*		
3	Alfred	M	14	69	112.5		
4	Alice	F	13	56.5	84		
5	Barbara	F	13	65.3	98		
6	Carol	F	14	62.8	102.5		
7	Henry	M	14	63.5	102.5		
8	James	M	12	57.3	83		
9	Jane	F	12	59.8	84.5		
10	Janet	F	15	62.5	112.5		
11	Jeffrey	M	13	62.5	84		
12	John	M	12	59	99.5		
13	Joyce	F	11	51.3	50.5		
14	Judy	F	14	64.3	90		
15	Louise	F	12	56.3	77		
16	Mary	F	15	66.5	112		
17	Philip	M	16	72	150		
18	Robert	M	12	64.8	128		
19	Ronald	M	15	67	133		
20	Thomas	M	11	57.5	85		
21	William	M	15	66.5	112		

DDE number 10 references the *List 1* EXCEL auto-format.



xlsAutoCLASS.xls

	A	B	C	D
1	*SASHelp.Class in EXCEL AUTO Format #10:List 1*			
2	*First Name*	*Sex*	*Age*	*Height (*
3	Alfred	M	14	
4	Alice	F	13	
5	Barbara	F	13	
6	Carol	F	14	
7	Henry	M	14	
8	James	M	12	
9	Jane	F	12	
10	Janet	F	15	
11	Jeffrey	M	13	
12	John	M	12	
13	Joyce	F	11	
14	Judy	F	14	
15	Louise	F	12	
16	Mary	F	15	
17	Philip	M	16	
18	Robert	M	12	
19	Ronald	M	15	
20	Thomas	M	11	
21	William	M	15	

The DDE Solution for Highlighting Inconsistent Records Entries

Layering is an integral part of the DDE solution. In fact Dynamic Data Exchange (DDE) is defined as a "mechanism that permits two applications to talk to each other by continuously and automatically exchanging data" **[5]Microsoft,p.209**. EXCEL is opened on the desktop so that data and formatting instructions can be transferred serially via DDE *triplets* and *doublets.* Additional details about these constructs along with a description of the EXCEL 4 Macro Language (X4ML) used in DDE transfers to EXCEL can be found in the companion paper, *Using Single-Purpose SAS® Macros to Format EXCEL Spreadsheets with DDE* **[14] Watts.** It is advisable to read the companion paper first, since the single-purpose macros used in this application are described there. In Table 1 the necessary steps for generating a DDE highlighted spreadsheet are listed along with references showing where additional information can be found.

Table 1: Algorithm for Highlighting Inconsistent Record Entries in an EXCEL Spreadsheet

Step	Task	Sub-Task	More Information
1	Create MOUSEDAT		Source code (cards statement) in Appendix. Identical for both ODS-HTML and DDE.
2	Define New Colors for DDE		See below
3	Create MOUSECOLORS		See ODS-HTML discussion above. The only difference between ODS-HTML and DDE is that there is no need for defining default background colors in DDE.
4	Create BGCOLORS that provide cell locations for highlighted colors		See below
5	Transfer data to EXCEL	Mouse Data via *triplet*	Companion paper (macro `WriteDStoXLS`)
		Titles via *triplet*	Companion paper and source code.
		Data Set headers via *triplet*	Companion paper (macro **HdrWvblLabel)**
		BGCOLORS via *doublet*	See below
6	Format Spreadsheet	Title	Companion paper (macro **formatTitle)**
		Headers	Companion paper (macro **formatHeaders)**
		Body	Companion paper (macro **formatBody)**
		Resize Body cells	See source code and companion paper for X4ML syntax
		Insert diving line after headers.	Companion paper (macro **insertLineDivider)**

Using Colors in DDE

DDE can take advantage of EXCEL's versatility when it comes to assigning background colors to a spreadsheet. Contrary to Vyverman (only a 16-color palette) **[13],p.3** and DelGobbo (support for only particular colors) **[3],p.3** EXCEL is capable of displaying 256^3 or 16+ million colors. However, only 56 of them can be viewed in a session. SAS/GRAPH software works identically, except that the maximum is increased to 256 colors per graphics display.

Working with color in DDE is not so straight-forward, however. Default colors are identified by ordinal number (1-56) and not by RGB code as is the case for ODS-HTML. Therefore, to get a match in DDE, the following code must be executed:

```
data _null_;
  file DDECmds; ❶
  put '[edit.color(16,170,170,170)]';                         %*GRAY FOR BORDERS;
  put %unquote(%bquote('[edit.color(35,%RGBDEC(CXDCFFDC))]')); %* light green; ❷ ❸
  put %unquote(%bquote('[edit.color(36,%RGBDEC(CXF2F6A8))]')); %* yellow;
  put %unquote(%bquote('[edit.color(22,%RGBDEC(CXFF0000))]')); %* red;
  put %unquote(%bquote('[edit.color(37,%RGBDEC(CX7070FF))]')); %* dark blue;
  put %unquote(%bquote('[edit.color(41,%RGBDEC(CXDCDCFF))]')); %* light blue;
run;
```

❶ See the companion paper for a discussion about DDECMDS as a *doublet* in DDE.

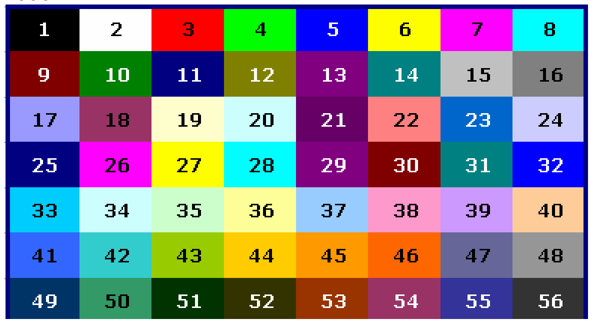
❷ EDIT.COLOR is the X4ML command used for assigning an RGB color code to an EXCEL color number. Panel 2 in Figure 4 below shows how the X4ML palette is affected.

❸ EXCEL uses decimals to represent color codes whereas SAS uses hexadecimals. The RGBDEC macro function converts from hexadecimal to decimal. While colors #36 and #37 haven't changed much in Figure 4, a perfect match to ODS-HTML colors is guaranteed, because the RGB coding system is universal.

In Figure 4, EXCEL X4ML and 2002 color mapping systems are displayed together with an X4ML altered list that shows how the EDIT.COLOR command works inside EXCEL. X4ML developed in 1990 is based on EXCEL version #4 in use at the time. The color mappings are completely different in EXCEL 2002. Note, for example, that the first eight colors in Panel 1 are primary or secondary colors whereas black is the only primary color found in the first row of the EXCEL 2002 palette.

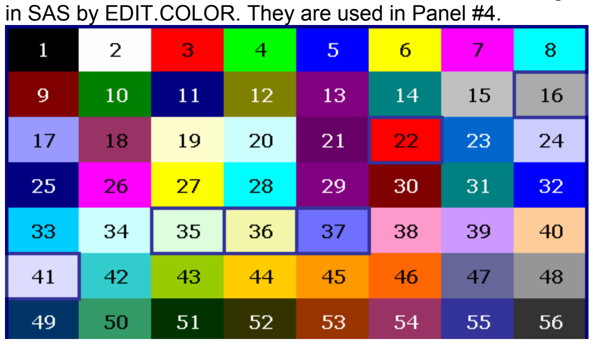
Figure 4. EXCEL color mappings have changed over the years. Panel #1 shows the X4ML default colors in vogue during the 1990s. Selected color codes are changed in Panel #2 for the Mouse Study, and EXCEL 2002 default colors are displayed in Panel #3. Output from the Mouse Study is copied and pasted for convenience from Figure 1 into Panel #4.

Panel 1: EXCEL was capable of working with 56 colors in 1990.



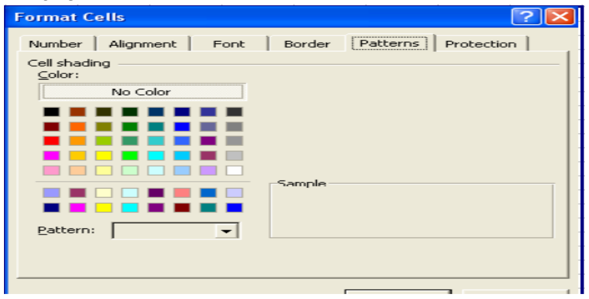
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Panel 2: Colors #16, #22, 35-37, and 41 have been changed in SAS by EDIT.COLOR. They are used in Panel #4.

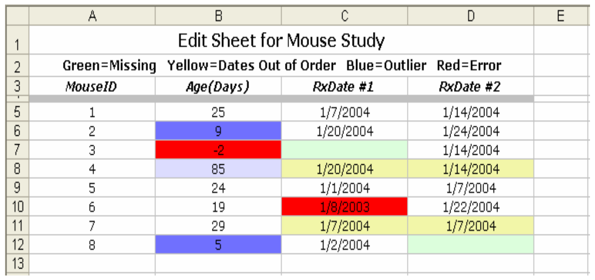


1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56

Panel 3: Default colors from the Patterns window in FORMAT CELLS from EXCEL 2002 are ordered differently from those in Panel 1.



Panel 4: Mouse Study spreadsheet from Figure 1 uses custom colors outlined above in Panel 2.



	A	B	C	D	E
1	Edit Sheet for Mouse Study				
2	Green=Missing	Yellow=Dates Out of Order	Blue=Outlier	Red=Error	
3	*MouseID*	*Age(Days)*	*RxDate #1*	*RxDate #2*	
5	1	25	1/7/2004	1/14/2004	
6	2	9	1/20/2004	1/24/2004	
7	3	-2		1/14/2004	
8	4	85	1/20/2004	1/14/2004	
9	5	24	1/1/2004	1/7/2004	
10	6	19	1/8/2003	1/22/2004	
11	7	29	1/7/2004	1/7/2004	
12	8	5	1/2/2004		
13					

Highlighted cells are non-rectangular in EXCEL

Macro variables that feed the COMPUTE block in ODS-HTML are replaced by data values from the BGCOLORS data set in DDE. Both configurations use MOUSECOLORS for input, but ODS-HTML links MOUSEID and COLUMN to a color, whereas a color in DDE is assigned to a cell address such as "R6C2". Below is SAS code for creating BGCOLORS:

```
data BGColors(keep=RowCol Color); ❶
  retain rowOffset &rowOffset.; ❷
  retain colOffset &colOffset.;
  array ColorV{*} ageColr t1Colr t2Colr;
  set mouseColors;
  do i=1 to dim(ColorV);
    if ColorV[i] ne . then do; ❸
      color=ColorV[i];
      row= _n_ + rowOffset;
      col= i + colOffset;
      RowCol=compress('R'||(left(row)||'C'||left(col))); ❹
```

```
          output;
        end;
      end;
    run;
```

❶ ROWCOL stands for row-***column*** not row-*color*. COLOR stands for background color.

❷ Offsets are implemented as macro variables. ROWOFFSET for MOUSEDAT would be set to 3 to accommodate two titles and a row of headers. COLOFFSET would be set to 2, because MOUSEID is never assigned a background color.

❸ Here is where un-highlighted cells are skipped over. Background colors are non-rectangular in composition because with so many missing values, no observable pattern for row or column display is detected in the output.

❹ A value is assigned here to ROWCOL. X4ML syntax treats internal spaces as errors. Therefore, the COMPRESS function must be used to correctly generate a cell address in row-column format.

Values from BGCOLORS are inserted into the target spreadsheet by exercising an implicit loop in the DATA _NULL_ step below.

```
    filename ddeCMDS dde 'excel|system';
    data _null_;
      file ddeCmds;
      set BGColors ; ❶
      if _n_ eq 1 then do;
       put '[workbook.activate("sheet1")]';
       put %unquote(%bquote('[select("r&stRow.c1:r&nrows.c&nvar")]')); ❷
       put '[clear(1)]';
       put '[border(,1,1,1,1,,,15,15,15,15)]';
      end;
      put '[select("' rowCol +(-1) '")]'; ❸
      put '[patterns(1,0,' color +(-1) ')]'; ❹
    run;
```

❶ Looping is implicit in the SET statement.

❷ From Figure 4 Panel 4, **r4c1:r11c4** would be defined as the parameter in the SELECT function with the gray dividing line between headers and data being added later. In the next PUT statement cell contents and formats from previous runs are CLEARED to eliminate the possibility of unwanted side-effects caused by layering output, and finally in the last PUT statement associated with the IF condition, thin gray cell borders are drawn around cells in the data region so that highlighted cells in the region will be in conformance with all cells in the spreadsheet. Otherwise, cell borders would automatically be erased when background colors are inserted.

❸ One cell is selected at a time from the BGCOLORS data set.

❹ A background color is inserted into the selected cell.

DDE is more efficient than ODS-HTML when it comes to assigning background colors. First, there is no need to process default colors in DDE, and secondly COLOR in the BGCOLORS data set is a *variable,* not a *constant literal* requiring a second pass at a data set to resolve an associated macro variable.

The method used for assigning background colors in DDE is also similar to the one described by William C. Murphy in *Give Your Clients What They Want: Fill Report Tables with the SAS® System and DDE*. **[6]**. Instead of background colors, data values are assigned to unordered individual cells in a non-rectangular spreadsheet template with data _NULL_ and a SET statement. Murphy uses a DDE *triplet* rather than a *doublet* to handle the data transfer, however.

Identified *Layers* in the DDE Solution

From Table 1 and the discussion about color above, DDE layers become programming steps in SAS. The layer-steps are identified as:

1) Write out title lines.
2) Format the titles.
3) Write out the header line.
4) Format the header line.
5) Assign background colors.
6) Write out table values (body).
7) Format the body.
8) Align the body.

9) Insert gray line to separate headers from body.

When layering is employed, the code becomes modular resulting in an application that is both transparent and modifiable.

Summary and Conclusions

Two methods for highlighting inconsistent record entries in an EXCEL spreadsheet have been described in this paper. ODS-HTML uses a COMPUTE block in PROC REPORT to assign background colors by row and column *name* whereas DDE uses a *doublet* inside data _NULL_ to assign a background color directly by cell *address*. DDE is more efficient than ODS-HTML, requiring only one pass at the data. Default color values can also be ignored in DDE. Titles and headers are formatted better in DDE, and color selection is not limited as previously assumed to the 56 colors in the X4ML palette. Finally, for this application, it is doubtful that DDE can be replaced by the newest ODS-XML configuration, since the Version 9 ExcelXP *tagset* doesn't work with a non-rectangular data.

Copyright Statement

Trademark Citation

Contact Information

The author welcomes feedback via email at perryWatts@comcast.net

References

[1]Buchanan, Laurie. *Using Compute Blocks to Enhance PROC REPORT Output.* Proceedings of the 10th Annual Northeast SAS Users Group Conference, Baltimore, MD, 2001, pp.298-300.

[2]Cross,Sue. *Create a Living Report from Various Parts.* Proceedings of the 14th Annual Northeast SAS Users Group Conference, Baltimore, MD, 2001, paper # ET5001.

[3]DelGobbo, Vincent. *A Beginner's Guide to Incorporating SAS® Output in Microsoft® Office Applications*. Proceedings of the Twenty-Eighth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2002, paper #52.

[4]DelGobbo, Vincent. *From SAS to Excel via XML*. http://support.sas.com/rnd/papers/sugi29/ExcelXml.pdf.

[5]Microsoft Corporation. *Visual Basic User's Guide: Automating, Customizing, and Programming in Microsoft Excel with the Microsoft Visual Basic Programming System, Applications Edition*. Document XL57927-0694. Printed in the United States of America, 1993.

[6]Murphy, William C. *Give Your Clients What they Want: Fill Report Tables with the SAS® System and DDE.* Proceedings of the 15th Annual Northeast SAS Users Group Conference, Buffalo, NY, 2002, paper # PS025.

[7] Pass, Ray and Sandy McNeill. *PROC REPORT: Doin' It in Style*. Proceedings of the 15th Annual Northeast SAS Users Group Conference, Buffalo, NY, 2002, paper # AT004.

[8]Pass, Ray and Sandy McNeill. *PROC TABULATE: Doin' It in Style*. Proceedings of the Twenty-Ninth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2004, paper #085.

[9]Parker, Chevell. *Generating Custom Excel Spreadsheets using ODS*. Proceedings of the Twenty-Eighth SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2003, paper #12.

[10]http://support.sas.com/rnd/base/topics/odsmarkup/markup_stmt.html. *Using ODS to Export Output in a Markup Language.* Accessed 5/25/2004. Last updated 9/5/2003.

[11]SAS Institute Inc. *The Complete Guide to the SAS® Output Delivery System, Version 8.* Cary, NC: SAS Institute Inc., 1999.

[12]SAS Institute Inc. *SAS® Procedures Guide, Version 8.* Cary, NC: SAS Institute Inc., 1999.

[13]Vyverman, Koen. *Creating Custom Excel Workbooks from Base SAS® with Dynamic Data Exchange: A Complete Walkthrough*. Proceedings of the Twenty-Seventh SAS® User Group International Conference, Cary, NC: SAS Institute Inc., 2002, paper #190.

[14]Watts, Perry. *Using Single-Purpose SAS® Macros to Format EXCEL Spreadsheets with DDE*. Proceedings of the 17th Annual Northeast SAS Users Group Conference. Baltimore, MD, 2004, paper #ap06.

[15]Watts, Perry. *Working with RGB and HLS Color Coding Systems in SAS® Software*. Proceedings of the 15th Annual Northeast SAS Users Group Conference. Buffalo, NY 2002, paper #ps013.

Appendix

Source code for the ODS-HTML and DDE SAS-to-EXCEL highlighting applications are included below.

```
/*  ---------------------------------------------------------------------------
    Program  :  RowTrafficLightingODS.sas

    Author   :  Perry Watts
    Date     :  01Apr2004 06:19

    Project  :  N04 Row Traffic Lighting
    Path     :

    Purpose  :  Do lateral (row-based) error checking by color coding cell background
                colors in an EXCEL spreadsheet where green means MISSING, yellow means
                attributes out-of-order, blue means attribute out-of-bounds, and red
                means ERROR. Use PROC REPORT in ODS output to an EXCEL file via HTML.

    Notes    :  **MouseChkODS.xls must NOT be opened on the desktop when this
                  program is executed.
                **See pages 198-208 in ODS manual for STYLE attributes.
                **See See chapter 32, page 890 The Report Procedure in v8 Procedures
                  Guide for STYLE locations.
                **Can leave DEFAULT color values (white or light gray) set to
                  missing but if you do, the following ERROR messages will be
                  written to the LOG:
                ERROR 22-322: Syntax error, expecting one of the following: a name,
                              a format name, DMBLACK, DMBLUE, DMBROWN,
                              DMCYAN, DMGRAY, DMGREEN, DMMAGENT, DMORANGE, DMPINK,
                              DMRED, DMSBLACK, DMSBLUE, DMSBROWN,
                              DMSCYAN, DMSGRAY, DMSGREEN, DMSMAGENTA, DMSORANGE,
                              DMSPINK, DMSRED, DMSWHITE, DMSYELLOW,
                              DMWHITE, DMYELLOW, SYSBACK, SYSFORE, SYSSECB, _UNDEFINE_,
                             _UNDEF_, _UND_.
               ERROR 76-322: Syntax error, statement will be ignored.
               Hence create a macro variable DEFAULTCOLOR to take WINDOWS default
               color.

    Input    :  CARDS

    Output   :  MouseChkODS.xls
    Copyright:  Copyright(C) 2004 Perry Watts
    ----------------------------------------------------------------------  */
    options date number pageno=1;
    options nomprint;
    options noxsync noxwait;
    options missing=' ';
    footnote1;

    %let defaultColor=CXFFFFFF; *-- white;

    *-- GET INPUT DATA;
    data mouseDat(keep=mouseID age test1dt test2dt);
      attrib test1dt test2dt informat=mmddyy10. format=mmddyy10.;
      infile cards missover;
      input mouseID age test1dt test2dt;
      cards;
      1 25 01/07/2004  01/14/2004
      2 9  01/20/2004  01/24/2004
      3 -2 .    01/14/2004
      4 85 01/20/2004  01/14/2004
      5 24 01/01/2004  01/07/2004
      6 19 01/08/2003  01/22/2004
      7 29 01/07/2004  01/07/2004
      8 5  01/02/2004 .
    run;

    *-- ASSIGN ROW COLORS TO TEST1 AND TEST2. EXPECTED: TEST1 >= TEST2;
    data MouseColors(keep=MouseID t1Colr t2Colr);
      length green default yellow red $8;
      retain default "&defaultColor";
      retain green   "CXDCFFDC";     *-- MISSING VALUES;
      retain Yellow  "CXF2F6A8";     *-- Test2dt < Test1dt;
      retain Red     "CXFF0000";     *-- ERRONEOUS VALUES;
      array NV{*} test1dt test2dt;   *-- Numeric Variable;
      array NVColr{*} $8 t1Colr t2Colr;
      set mouseDat;

      *-- COLOR MISSING AND ERRONEOUS VALUES WITH ARRAYS;
      do i=1 to dim(NV);
        if NV[i] eq . then NVColr[i]=green;
        else if NV[i] lt '31DEC2003'd then NVColr[i]=Red;
        else NVColr[i]=default; /*IF OMIT, WILL GET ERR MSG LISTED IN HEADER. */
      end;
      *-- IDENTIFY LATERAL INCONSISTENCIES WITH CROSS-VARIABLE CHECKING;
      if Test2dt le Test1dt and Test1dt gt '31DEC2003'd and Test2dt gt '31DEC2003'd then
        do;
          t1Colr=Yellow; t2Colr=Yellow;
        end;
    run;

  *-- ROW|COLUMN-BASED MACRO VARIABLES ARE CREATED WHERE MOUSEID MAPS TO
      COLUMN COLORS BASED ON VALUES IN THE MOUSECOLORS DATA SET.;
    data _null_;
    set mouseColors end=last;
      call symput('Column3_'||left(put(MouseID,1.)),t1Colr);
      call symput('Column4_'||left(put(MouseID,1.)),t2Colr);
      if last then
        call symput('nobs',left(put(_n_,3.)));
    run;

  *-- INVOKED FROM PROC REPORT;
   %macro select;
     select (MouseId);
      %do i = 1 %to &nobs;
        when (&i.)
          do;
            CALL DEFINE("test1Dt" ,"STYLE", "STYLE=[ BACKGROUND=&&Column3_&i.. ]");
            CALL DEFINE("test2Dt" ,"STYLE", "STYLE=[ BACKGROUND=&&Column4_&i.. ]");
          end;
      %end;
      otherwise;
     end;
   %mend select;

   *-- ATTRIBUTE-BASED OR COLUMN OUTLIER VALUES CAN BE IDENTIFIED BY FORMAT WHERE START
       VALUES REPRESENT MOUSE AGE IN DAYS. <0=ERROR, 0-9=TOO YOUNG, 10-29=OK, 30+=TOO OLD
       (FABRICATED);
    proc format;
      value AgeCfmt low - <0="CXFF0000" 0-<10="CX7070FF"
                    10-<30="&defaultColor" 30-high="CXDCDCFF";
    run;

   *-- WRITE COLORS OUT TO AN EXCEL SPREADSHEET WITH ODS. EXCEL MUST NOT BE OPENED.;
   *-- IN-LINE COMMENT;
   ods html
     file="c:\N04\BgHighlight\XLS\MouseChkODSnewest.xls"
     STYLE=styles.Withborder;
     title1 "<td colspan=3 align=center><font size=3>
        <b>Edit Sheet for Mouse Study  </b></font></td>";
     title2 "<td align=Center colspan=3><font size=2>
        <b>Green=Missing   Yellow=Dates Out of Order   Blue=Outlier   Red=Error</b></font></td>";
  proc report data=mouseDat nowindows style(Header)=[font_weight=BOLD font_style=Italic];
    columns mouseID age test1Dt test2Dt;
    define mouseID  / display "MouseID"    center;
    define age      / display "Age(days)"  center
                      style(column)=[font_weight=Medium background=AgecFmt.];
    define test1Dt  / display "RxDate #1"  center;
    define test2Dt  / display "RxDate #2"  center;
    compute MouseId;
      %select
    endcomp;
  run;

 ods html close;
 ods listing;
==========================================================================================
/*  -----------------------------------------------------------------------
    Program  :  RowTrafficLightingDDE.sas

    Author   :  Perry Watts
    Date     :  21Feb2004 07:24

    Project  :  N04 Row Traffic Lighting

    Purpose  :  Write data set variables to a background-color coded Excel
                spreadsheet that highlights lateral data errors.

    Notes    :  **Compare program to RowTrafficLightingODS.sas. This program uses
                 DDE instead of ODS output to HTML. MouseChkDDE.xls, therefore
                 must be opened on the desktop when this program is executed.
    DDEcmds  :
                EDIT.COLOR(color_num, red_value, green_value, blue_value)
                 Color_num is a number from 1 to 56 specifying one of the 56
                    color palette boxes for which you want to set the color.
                 red_value, green_value, and blue_value are DECIMAL numbers from
                    0 to 255.

    Input    :  CARDS

    Output   :  MouseChkDDE.xls
    Copyright:  Copyright(C) 2004 Perry Watts
    -----------------------------------------------------------------------  */
    options mprint mautosource;
    options sasautos=(sasautos 'c:\N04\DDEwSASMac\sasauto');
    options date number pageno=1 source2;
    options noxsync noxwait;
    options missing=' ';

    Footnote1;
    title1;

  /* CONVERT RGBHEX THAT SAS UNDERSTANDS TO RGBDEC FOR MICROSOFT*/
    %macro RGBDec(CXrrggbb);
        %local rr gg bb;
        %let rr=%substr(&CXrrggbb,3,2);
        %let gg=%substr(&CXrrggbb,5,2);
        %let bb=%substr(&CXrrggbb,7);
        %let DecCode= %sysfunc(compress(%sysfunc(putn(%sysfunc(inputn(&rr,hex2.)),z3.))
                      %sysfunc(putn(%sysfunc(inputn(&gg,hex2.)),z3.))
                      %sysfunc(putn(%sysfunc(inputn(&bb,hex2.)),z3.))));
        %substr(&DecCode,1,3),%substr(&DecCode,4,3),%substr(&DecCode,7,3)
    %mend RGBDEC;

    *-- TO ISSUE X4ML (EXCEL-VSN4-MACRO LANGUAGE) COMMANDS;
    filename ddeCMDS dde 'excel|system';

    *-- MATCH DDE AND ODS OUTPUT COLORS;
    data _null_;
      file DDECmds;
      put '[edit.color(16,170,170,170)]';  *-- CHANGE COLOR #16 TO A LIGHTER GRAY FOR BORDERS;
      put %unquote(%bquote('[edit.color(35,%RGBDEC(CXDCFFDC))]'));  %* light green;
      put %unquote(%bquote('[edit.color(36,%RGBDEC(CXF2F6A8))]'));  %* yellow;
      put %unquote(%bquote('[edit.color(22,%RGBDEC(CXFF0000))]'));  %* red;
      put %unquote(%bquote('[edit.color(37,%RGBDEC(CX7070FF))]'));  %* dark blue;
      put %unquote(%bquote('[edit.color(41,%RGBDEC(CXDCDCFF))]'));  %* light blue;
  run;

  *-- GET INPUT DATA;
  data mouseDat(keep=mouseID age test1Dt test2Dt);
    retain mouseID age test1Dt test2Dt;
    attrib test1dt test2dt informat=mmddyy10. format=mmddyy10.;
    infile cards missover;
    input mouseID age test1Dt test2Dt;
    label MouseID="MouseID"
              age="Age(Days)"
          test1Dt="RxDate #1"
          test2Dt="RxDate #2";
    cards;
    1 25 01/07/2004  01/14/2004
    2 9  01/20/2004  01/24/2004
    3 -2 .    01/14/2004
    4 85 01/20/2004  01/14/2004
    5 24 01/01/2004  01/07/2004
    6 19 01/08/2003  01/22/2004
    7 29 01/07/2004  01/07/2004
    8 5  01/02/2004 .
  run;

  %let rowOffset=3;  *-- TO ACCOMMODATE TWO TITLES AND HEADERS;
  %let colOffset=1;  *-- TO ACCOMMODATE UNCHECKED MOUSEID (PK): IN COLORING DATASET ONLY;
  %let rc=%sysfunc(open(mouseDat));
  %let nVar = %sysfunc(attrn(&rc,NVARS));
  %let nobs = %sysfunc(attrn(&rc,NOBS));
  %let nrows=%eval(&nobs + &rowOffset);
  %let startRow=%eval(&rowOffset +1);
  %let nvar=%left(&nvar);
  %let rc=%sysfunc(close(&rc.));

  *-- ASSIGN ROW COLORS TO TEST1 AND TEST2. EXPECTED: TEST1 >= TEST2;
  data mouseColors(keep=mouseID ageColr t1Colr t2Colr);
    retain green  35;  *-- MISSING VALUES;
    retain Yellow 36;  *-- Test2Dt <= Test1Dt;
    retain Red    22;  *-- ERRONEOUS VALUES;
    retain DkBlue 37;  *-- OUT-OF-BOUNDS (OUTLIER TOO LOW);
    retain LtBlue 41;  *-- OUT-OF-BOUNDS (OUTLIER TOO HIGH);
    array NV{*} test1Dt test2Dt;  *-- Numeric Variable;
    array NVColr{*} t1Colr t2Colr;
    set mouseDat;

 /* COLUMN-BASED COLOR ASSIGNMENTS -- BASED ON ONE VARIABLE ONLY*/
    *-- IDENTIFY AGE ERRORS;
    if Age lt 0 then AgeColr=Red;
    else if Age ge 0 and age lt 10 then AgeColr=DkBlue;
    else if Age ge 30 then AgeColr=LtBlue;

    *-- COLOR MISSING AND ERRONEOUS VALUES WITH ARRAYS;
    do i=1 to dim(NV);
      if NV[i] eq . then NVColr[i]=green;
      else if NV[i] lt '31DEC2003'd then NVColr[i]=Red;
    end;

  /* ROW-BASED COLOR ASSIGNMENTS -- BASED ON MORE THAN ONE VARIABLE*/
    if Test2dt le Test1dt and Test1dt gt '31DEC2003'd and Test2dt gt '31DEC2003'd then
      do;
        t1Colr=Yellow;
        t2Colr=Yellow;
      end;
  run;

  data BGColors(keep=RowCol Color);
    retain rowOffset &rowOffset.; *-- FOR TITLE AND HEADERS;
    retain colOffset &colOffset.; *-- FOR CASENUM WHICH IS NOT CHECKED;
    array ColorV{*} ageColr t1Colr t2Colr;
    set mouseColors;
    do i=1 to dim(ColorV);
      if ColorV[i] ne . then do;
        color=ColorV[i];
        row= _n_ + rowOffset;
        col= i + colOffset;
        RowCol=compress('R'||(left(row)||'C'||left(col)));
        output;
      end;
    end;
  run;
  proc print data=BGColors;
  run cancel;
/* *****
    Some OUTPUT:
                    Row
   Obs    color     Col
    1      36     R3C4
    2      37     R4C2
    9      37     R10C2
   10      35     R10C4
  ***** */

 /* PRINT TITLE LINES*/
  filename title dde "excel|[MouseChkDDE.xls]sheet1!r1c1:r2c&nvar" notab;
  data _null_;
    file title;
    put "Edit Sheet for Mouse Study";
    put "Green=Missing   Yellow=Dates Out of Order   Blue=Outlier   Red=Error";
  run;
  filename title clear;
/* FORMAT TITLES*/
 %formatTitle(MaxCol=&nvar, Fontsize=12, Row=1, RowHeight=20)
 %formatTitle(MaxCol=&nvar, Fontsize=10, Row=2, RowHeight=15)

 /* PRINT HEADER */
  filename HEADERS dde "excel|[MouseChkDDE.xls]sheet1!r3c1:r3c&nvar" notab;
  data _null_;
    file HEADERS;
    put %HdrWvblLabel(dsName=mouseDat)
  run;
  filename HEADERS clear;
  /* FORMAT HEADER */
  %formatHeaders(fontSize=9, Row=3, MaxCol=&nVar)

/* ASSIGN BACKGROUND COLORS (PROCESS DATA SET: BGCOL)*/
filename ddeCMDS dde 'excel|system';
data _null_;
  file ddeCmds;
  set BGColors;
  if _n_ eq 1 then do;
     put '[workbook.activate("sheet1")]';
     put %unquote(%bquote('[select("r&startRow.c1:r&nrows.c&nvar")]')); *-- SELECT DATA MATRIX;
     put '[clear(1)]';                                                *-- FROM PREVIOUS RUNS;
     put '[border(,1,1,1,1,,,15,15,15,15)]';                          *-- BORDER EACH CELL;
  end;
  put '[select("' rowCol +(-1) '")]';                                 *-- SELECT A CELL;
  put '[patterns(1,0,' color +(-1) ')]';                              *-- APPLY BACKGROUND COLOR;
run;

/* WRITE BODY*/
filename ddedata dde
  "excel|[MouseChkDDE.xls]sheet1!r&startRow.c1:r&nrows.c&nvar" notab;

data _null_;
   set mouseDat;
   file ddedata;
   put %WriteDStoXLS(dsName=mouseDat)
run;

/* FORMAT BODY*/
%formatBody(CmdFileName=ddeCmds, fontSize=9, MinRow=&startRow, Maxrow=&nrows, MinCol=1, MaxCol=&nVar);

/* ALIGN BODY*/
data _null_;
  file ddeCmds;
  put %unquote(%bquote('[select("r&startRow.c1:r&startRow.c&nVar")]'));
  put '[alignment(3,2)]';                                    *-- CENTER VERTICALLY AND HORIZONTALLY;
  put %unquote(%bquote('[column.width(20,"C1:C&nVar.")]'));
  put %unquote(%bquote('[row.height(0,"r&startRow.:r&nvar",false,3)]'));    *-- AUTO FIT ROW HEIGHT;
run;
```

```
/* INSERT THIN GRAY LINE UNDER THE HEADERS. THIS MUST BE THE LAST COMMAND */
%insertLineDivider(row=4, maxCol=&nvar)

quit;
```