

WINDOW & DISPLAY Statements:

Creating a Crude User Interface

-- Peter Kilpatrick

Two SAS Statements In One Data Step

- WINDOW Statement – Defines the layout and formatting of the window that gets displayed
 - Control the size of window, location of window, location of text in window, background color, font color
- DISPLAY Statement – displays the information
 - Use multiple DISPLAY statements to organize when information gets displayed

The WINDOW Statement

- Made up of *field definitions*, or *groups of field definitions*
- Groups – used to organize *field definitions*
 - I usually make groups by question
- Field Definitions – either *character fields* or *variable fields*
 1. Specify row and column (location in window)
 2. Type of field (*character* or *variable*)
 3. Options for field (color, brightness)

Example 1: Basic WINDOW/DISPLAY step

```
data _null_;  
  window start color=BLACK  
    #5 @26 'WINDOW & DISPLAY'  
      color=WHITE  
    #7 @26 'Creating an interface'  
      color=WHITE  
    #12 @26 'Press Enter to Continue'  
      color=WHITE  
  ;  
  display start;  
  stop;  
run;
```

Example 1

- Made up of *character fields*
 - No options for data entry or variable display
- When the ENTER key is pressed:
 - The window closes
 - SAS moves to the next data step
- Use this basic data step for:
 - Basic information about the program
 - Friendly Reminders

Example 1

```
data _null_;  
  window start color=BLACK
```

- `_null_` is used to avoid creating a dataset that is not used
- `window` statement begins and it is named "start"
- the color of the window is BLACK

Example 1

```
#5  @26 'WINDOW & DISPLAY'  
      color=WHITE  
#7  @26 'Creating an interface'  
      color=WHITE  
#12 @26 'Press Enter to '  
      color=WHITE  
      ;
```

- Things look messy, but gets familiar
- Row number X at Column Y
 - Row # X @ Column Y

Example 1

```
display start;  
stop;  
run;
```

- `display` the window named "start"
- `stop` displaying the window
 - User must press ENTER to get to the stop statement and close the window

Example 2: Two Windows, One Step

```
window start1 color=BLACK
      rows=20 columns=50
      irow=1 icolumn=1
#5 @5 '2 Windows in One Data Step';
window start2 color=WHITE
      rows=20 columns=50
      irow=21 icolumn=1
#5 @5 '2 Windows in One Data Step';
display start1;
display start2;
stop;
```

Example 2

- By using 2 WINDOW statements and 2 DISPLAY statements we can have display windows (pages) for presenting information.
- The size and location of windows can be controlled by utilizing:
 - rows=X columns=Y [size of window]
 - irow=X icolumn=Y [initial location]
 - These options follow the name of the window

Example 3: Storing Keyed Data

- Instead of `_null_` create a new dataset
- Add *variable* fields
 - *Variable* fields are similar to *character* fields
 - Need a row and column location
 - You can specify similar options
 - *Variable* fields differ from *character* fields
 - Requires a format
 - More options

Example 3

- Introduction of *groups*
 - I use *groups* for individual questions
 - *Groups* contain *field definitions*
 - group=group_name
field definitions
 - group=next_group
field definitions

Example 3

```
group=name
#10 @22 'NAME:' color=WHITE
+1 NAME_VAR $CHAR15.
color = YELLOW
attr = underline
required = yes
```

- Change the pointer/columns the eye
- NAME_VAR is my variable
- \$CHAR15. (Format of my variable)
- required= Must I fill it in?

Example 3

- Displaying multiple *groups*
 - Use multiple DISPLAY statements
 - display window_name.group_name;
- Output statement gets the data into the dataset
- Creative ways to stop displaying the window

```
display store_page.name BLANK;
display store_page.season;
output;
if _n_ eq 10 then STOP;
```

Example 3

- Display statement options
 - BLANK clears the window
 - NOINPUT, BELL, DELETE

Example 4: Retrieving Data From a Dataset

- Similar to previous example
 - Use a set statement locating the data set you want information from
- For display, you must be creative:

```
data retrieve_data;
set store_data;
length NUM_VARd $4.;
NUM_VAR = _n_;
NUM_VARd=compress(NUM_VAR | '.)');
```

Example 4

```
window retrieve_page color=BLACK
#(NUM_VAR+6) @22 NUM_VARd
color = WHITE
persist = yes
protect = yes
```

- #(NUM_VAR+6)
- persist= Will the display of this field still be visible after moving to the next observation?

Example 4

```
window retrieve_page color=BLACK
#(NUM_VAR+6) @22 NUM_VARd
color = WHITE
persist = yes
protect = yes
```

protect= Do you want to make this variable unable to be edited?

Example 4

```
+1 NAME_VAR $CHAR15.
color=GREEN
persist=yes
protect=yes
attr=(underline,highlight)
```

attr=(underline,highlight)
[Multiple attributes]

Benefits

- Instructions/Warnings can be provided in the program itself
- A User-interface simplifies end-user interaction with SAS
- This system can be used to create an adequate data entry interface
- Based on user responses, routes can be created to avoid running unnecessary steps

Thank You

- Any questions?