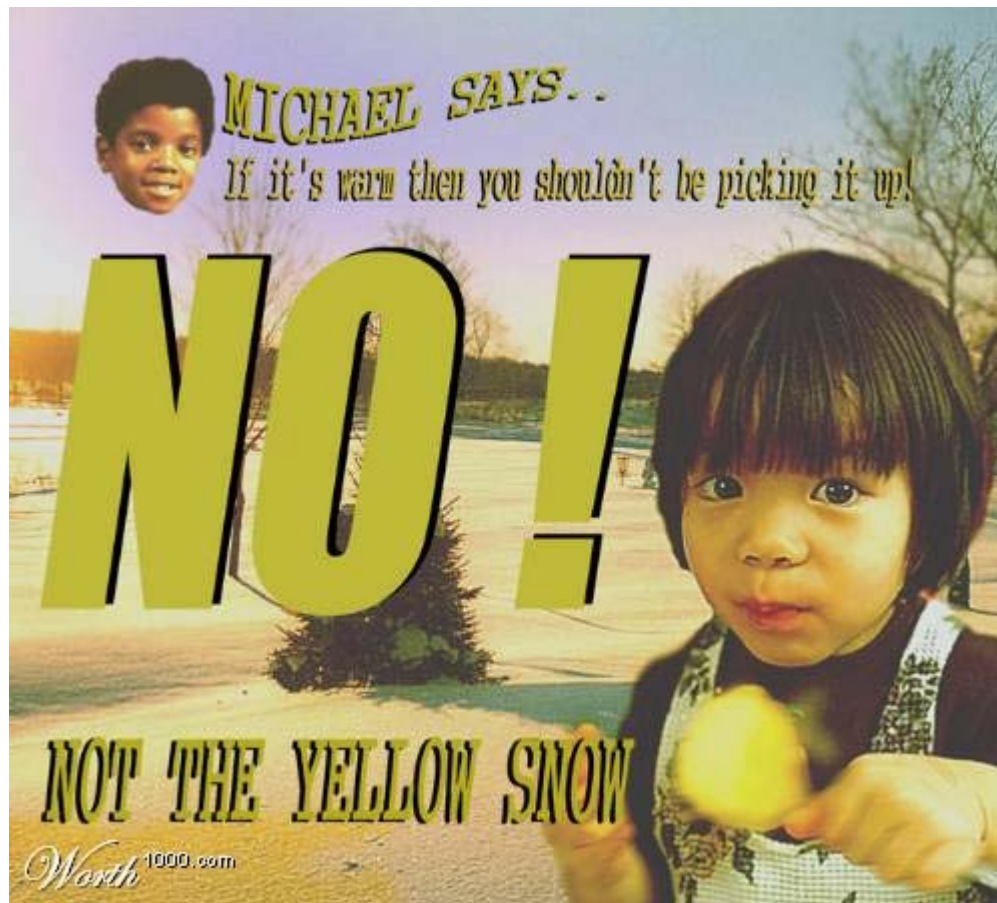




Tips and Tricks

Inaugural
Calgary SAS Users Group Meeting
David Yarmchuk

Tip 1: Don't eat Yellow Snow





Formatting your cares away

- Use lots of empty space
 - Use Tabs to indent logic groups
- Default on Enhanced editor is 4 spaces but you can change this to your own preference.
 - Use blank lines to separate data steps and procedures
 - Use blank lines to separate different logic blocks within a data step or proc step



Formatting your cares away

(continued)

```
* Sample Code Demonstrating Spacing *;
* Create dataset with converted codes *;
Data convert;
  set test;
  attrib
    hc_sex length=$1 label = 'Sex of Person'
  ;
  * Convert sex data into one character abbreviation ;
  if sex = 1 then do;
    hc_sex = 'M';
  end;
  else if sex = 2 then do;
    hc_sex = 'F';
  end;
  else do;
    hc_sex = 'E';
  end;
run;
```



The joy of white space

- Maximum of one program line per physical line of code
 - Split one SAS program line across several physical lines for readability. SAS continues to read until it reads a semi-colon
'`;`'
- Don't be afraid to add 'extra' lines of code
 - Self-documenting
 - Easier to debug
- Use more physical lines not less not less



The joy of white space (continued)

Version A - packed on one line

```
proc format; value cvt_sexN 1= 'M' 2= 'F' other
= 'E'; run;
```

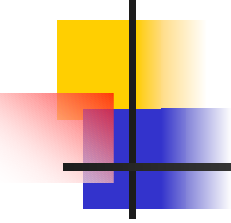
Version B - lots of spaces

```
proc format;
    value cvt_sexN
        1      = 'M'
        2      = 'F'
        other  = 'E'
;
run;
```



Make your Code work for you whether it wants to or not

- Use comments and self-documenting features
 - `/* This is valid syntax for a comment */`
 - `* This is valid syntax for a comment ;`
 - Meaningful names
 - Variable names can be 32 characters long
 - Datasets names can be 32 characters long
 - Use underscore to separate names
 - e.g. `first_name =`
- Use Labels
 - `variable_name Label = 'Variable Label'`



Make your Code work for you whether it wants to or not (continued)

```
* Sample Code Demonstrating Documenting and comments *;  
* Create dataset with work phone list *;  
Data Phone_Numbers  
  (label='List of Health Canada Phone Numbers');  
  
  * List of variables to be created;  
  attrib first_name length=$1 label = 'First Name of Person'  
         last_name length=$1 label = 'Surname of Person'  
         work_phone_number length=10 label = 'Work Phone No'  
  ;  
  infile input_data; /* This data comes from HC phone book */  
run;
```



Banging something out of Access

```
PROC SQL;
CONNECT TO ODBC (PROMPT) ;

%put &sysdbmsg;

CREATE TABLE SETOFF.SETOFFSRAW
  AS SELECT * FROM CONNECTION TO ODBC
  (
    SELECT *
    from [SETOFF])
;

CREATE TABLE SETOFF.STATION
  AS SELECT * FROM CONNECTION TO ODBC
  (
    SELECT *
    from [STATIONS])
;

disconnect from ODBC;
```



Banging something out of Access

(continued)

- PROMPT – Requires user input to process
 - Unable to “batch run” or schedule job
- Depends on DSN entries
 - Code may not run on another machine
 - Will require DSN entries be placed
 - Non-intuitive



Banging something out of Access

(continued)

When using SAS/ACCESS Interface to ODBC, through PROC SQL pass-through, you can specify connection parameters in the CONNECT statement rather than using (PROMPT). To do this, first place the following code, after the CONNECT TO ODBC statement, in your existing PROC SQL pass-through code:

```
%put &sqlxmsg;
```

This line will display (in the SAS log), the exact parameters generated when using PROMPT. For example:

```
dsn=MS Access 7.0 Database; DBQ=C:\MyDocuments\db1.mdb;  
DefaultDir=C:\My Documents;
```

Copy this information from your Log and insert it in the parenthesis in place of the 'prompt'. Be sure and add double quotes ("dsn=...;dbq=...") around the entire string and do not remove the semicolons that separate the parameters. This string is case sensitive, so copying it ensures accuracy. Then remove the %put &sqlxmsg; statement from your code. NOTE: For SAS V7 & V8 use %put &sysdbmsg; instead.



Banging something out of Access

(continued)

```
1     PROC SQL;
2     CONNECT TO ODBC (PROMPT);

3
4     %put &sqlxmsg;
DSN=setoff;DBQ=Y:\setoff\Setoff2000fe.mdb;DriverId=25;FIL=MS
Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;
5     %put &sysdbmsg;
ODBC:
    DSN=setoff;DBQ=Y:\setoff\Setoff2000fe.mdb;DriverId=25;FIL=M
    S
Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;
6
7     disconnect from ODBC;
```



Banging something out of Access

(continued)

```
PROC SQL;
```

```
CONNECT TO ODBC ("DSN=MS Access  
Database;DBQ=R:\CGY_GCS\CAR_PLANNING\setoff\Setoff2000fe.mdb;Default  
Dir=R:\CGY_GCS\CAR_PLANNING\setoff;DriverId=25;FIL=MS  
Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;");
```

```
CREATE TABLE SETOFF.SETOFFSRAW  
AS SELECT * FROM CONNECTION TO ODBC  
(  
    SELECT *  
    from [SETOFF])  
;
```

```
CREATE TABLE SETOFF.STATION  
AS SELECT * FROM CONNECTION TO ODBC  
(  
    SELECT *  
    from [STATIONS])  
;
```

```
disconnect from ODBC;
```



But I want to do further Analysis

- Many of our clients/customers want to take the data we provide and modify it further.
 - Business Intelligence Server
- Excel seems to be a preferred format
 - PROC DBLOAD
 - ODS output to Excel



But I want to do further Analysis (ODS to Excel)

- The easiest way to output to Excel is to output to HTML and name the file with an XLS extension to force it to Excel.

```
ods html body='temp.xls';  
  proc print data=sashelp.class;  
  run;  
ods html close;
```

But I want to do further Analysis

(ODS to Excel)

The SAS System

Obs	Name	Sex	Age	Height	Weight
1	Alfred	M	14	69	112.5
2	Alice	F	13	56.5	84
3	Barbara	F	13	65.3	98
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84
10	John	M	12	59	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90
13	Louise	F	12	56.3	77
14	Mary	F	15	66.5	112
15	Philip	M	16	72	150
16	Robert	M	12	64.8	128
17	Ronald	M	15	67	133
18	Thomas	M	11	57.5	85
19	William	M	15	66.5	112



But I want to do further Analysis (PROC EXPORT)

- The following code gets the data out into a format that is easily readable by Excel

```
proc export
  data= sashelp.class
  outfile= "C:\temp.csv"
  dbms=csv replace
;
run;
```

But I want to do further Analysis (PROC EXPORT)

Name	Sex	Age	Height	Weight
Alfred	M	14	69	112.5
Alice	F	13	56.5	84
Barbara	F	13	65.3	98
Carol	F	14	62.8	102.5
Henry	M	14	63.5	102.5
James	M	12	57.3	83
Jane	F	12	59.8	84.5
Janet	F	15	62.5	112.5
Jeffrey	M	13	62.5	84
John	M	12	59	99.5
Joyce	F	11	51.3	50.5
Judy	F	14	64.3	90
Louise	F	12	56.3	77
Mary	F	15	66.5	112
Philip	M	16	72	150
Robert	M	12	64.8	128
Ronald	M	15	67	133
Thomas	M	11	57.5	85
William	M	15	66.5	112



But I want to do further Analysis (PROC DBLOAD)

```
OPTIONS NOXWAIT; /* Existing files cannot be overwritten by DBLOAD, must  
    be deleted first */  
X 'DEL R:\CGY_GCS\ms_budpm\Reports\Setoffs\SETOFFS_OUTPUT.xls';  
RUN;
```

```
PROC DBLOAD DBMS=XLS DATA=SETOFF.SETOFFS_OUTPUT;  
    PATH='R:\CGY_GCS\ms_budpm\Reports\Setoffs\SETOFFS_OUTPUT.xls';  
    VERSION=7;  
    LABEL;  
    LIMIT=16000; *** Sets the maximum number of rows to output, I  
        believe the default is 5000 ;  
    PUTNAMES YES; *** Puts the variable names in the first row delete  
        this line if you do not want the names ;  
    LOAD;  
RUN;
```

But I want to do further Analysis (PROC DBLOAD)

Area	Parent	_TYPE_	_FREQ_	AVG_REPAIR_HOURS	AVG_HOURS_TO_LIFT
		92.00	18.00	22.27777778	31.11111111
Alberta		94.00	2.00	19	39
BC Interior		94.00	2.00	18	12
Chicago		94.00	1.00	26	26
Manitoba		94.00	1.00	9	23
Northern Ontario		94.00	3.00	19.66666667	25.33333333
SLH ON		94.00	5.00	27.4	28.8
SLH PQ		94.00	3.00	28.66666667	55
St. Paul		94.00	1.00	10	24
Alberta	ALYTH	95.00	1.00	18	64
Alberta	LETHBRIDGE	95.00	1.00	20	14
BC Interior	GOLDEN AREA	95.00	2.00	18	12
Chicago	MILWAUKEE	95.00	1.00	26	26