

The SQUEEZE Macro

SAS Health User Group
23 April 2009

Getting rid of unnecessary weight

- Drop unnecessary variables ASAP (use the keep and drop options on the data step)
- Get rid of unnecessary observations ASAP (use the “where” statement on input to avoid wasting CPU time processing observations you don’t need to process)
data females; set all (where = (sex = ‘F’));

Don't pass through data more often than needed

- Avoid unnecessary sorts

 - Use *class* rather than *by* for *proc means* or *proc summary*

 - Consider the *notsorted* option if data are grouped but not sorted

- Use *proc append* to add a small dataset to a large one

 - Warning: by default, structure of base data is retained

Don't pass through data more often than needed

- Use the modify statement rather than reading in a dataset

```
proc datasets library = mylib;  
  modify mydata;  
  rename sex = gender;  
  label edu = "education";  
  format birthdate date9.;  
quit;
```

Avoid wasting space

- Using necessary space to store data
 - Wastes disk space
 - Wastes processing time

**Warning: this presentation
contains gross
oversimplifications**

Bits and Bytes

Smallest unit of computer storage is a bit

Bits are collected into groups of 8.

8 bits = 1 byte

Each bit has a value of either 0 or 1.

00110010001010100111000000111111

Storing Character Variables

- Every character in a string requires one byte**.
- How some characters are represented
 - Capital letters: A = 65, B = 66, ...
 - Lower case letters: a = 97, b = 98, ...
 - Digits: 0 = 48, 2 = 49, ...
 - Space = 32
- In binary: A = 01000001

** for everyday applications in “American” English

How SAS determines the length of a character string

- With no other information, SAS allocates 8 bytes. Longer strings are truncated.
`input word1 $ word2 $;`
- Character variables created within a dataset are assigned the length of the *first* string SAS finds.
`if sex = 1 then gender = "male";`
`else if sex = 2 then gender = "female";`
- Explicit assignment of length:
`input word $ 1-10;`
`length my_char $10;`
- Substring command: substring will be as long as original variable.
`one_char = substr(sex, 1, 1);`

How SAS determines the length of a number

- By default, all numbers are assigned 8 bytes (64 bits)
- All numbers are stored as floating point numbers

Floating point numbers

Original	Normalized	Mantissa	Exponent
60.54	$+.654 \times 10^2$	+654	+2
-1.693	$-.1693 \times 10^1$	+1693	+1
-0.000182	$-.182 \times 10^{-3}$	-182	-3
0.0245	$+.245 \times 10^{-1}$	+245	-1
18	$+.18 \times 10^2$	+18	+2
0.5	$.5 \times 10^0$	+5	0

Floating point numbers

- IEEE standard for floating point representation
 - Institute of Electrical and Electronics Engineers
- Each number will be stored using 8 bytes (64 bits).
- First 11 bits always used for the **exponent**.
- Remaining 53 bits always used for the **mantissa**.

Floating point numbers

- IEEE standard for floating point representation
- Each number will be stored using 8 bytes (64 bits).
- First 11 bits always used for the **exponent**.
- Remaining 53 bits always used for the **mantissa**.
- $60.54 \rightarrow +.654 \times 10^2 \rightarrow$ **+654** **+2**
 \rightarrow **00000000002** **06540000000000000000...**
- Only need all 53 bits for the mantissa for
 - Very precise numbers
 - Very large numbers (9×10^{15})

What happens if we specify a smaller length?

00000000002 065400000000000000...

- The **exponent** always uses 11 bits
- The **mantissa** is shortened to fit in the remaining space
- E.g. `length my_number 3;`

00000000002 0654000000000000

- Numbers are always expanded back to 8 bytes (by adding 0's to the end of the mantissa) before they're used in calculations.

What happens if we specify a smaller length?

- What if you specify a length that is too small to hold a number?

Loss of precision

- Akin to storing 123,456 as 123,000

Length in bytes	Largest integer represented exactly	Exponential notation
3	8,192	2^{13}
4	2,097,152	2^{21}
5	536,870,912	2^{29}
6	137,438,953,472	2^{37}
7	35,184,372,088,832	2^{45}
8	9,007,119,254,740,992	2^{53}

Compressing

- Doesn't work well (in my experience)
 - Only compresses character strings
 - Works on one observation at a time
 - Adds a tag to each observation, to indicate how to uncompress the observation
- Any compression technique has to be balanced against the CPU time required to uncompress the data before use

Compression

Dataset	Original size (bytes)	UNIX 'compress'	SAS compression
Short records	7,266,304	-70.3%	+16.4%
Long records	273,162,240	-91.7%	-73.1%
Inter-mediate records	12,918,884	-72.3%	-1.4%

The %squeeze macro

Numbers

- Repeatedly remove 1 byte from each numeric variable until value stored in (n-1) bytes \neq value stored in (n) bytes.

```
data test ;
```

```
    a = 2001 ;
```

```
    if trunc( a, 7 ) ne a then length_a = 8 ;
```

```
    else if trunc( a, 6 ) ne a then length_a = 7 ;
```

```
    else if trunc( a, 5 ) ne a then length_a = 6 ;
```

```
    else if trunc( a, 4 ) ne a then length_a = 5 ;
```

```
    else if trunc( a, 3 ) ne a then length_a = 4 ;
```

```
    else length_a = 3 ;
```

```
run ;
```

The %squeeze macro character variables

- Determines number of bytes needed to keep rightmost character in the string.

The %squeeze macro

<http://support.sas.com/kb/24/804.html>