Using SAS for Simple Calculations

Jayson Shurgold
VanSUG – Nov 4th, 2015
SAS – Modern, Reliable, Accurate:

support.sas.com...

- Access data in almost any format (SAS tables, Excel, and others).
- Manage and manipulate your data (data subsets, data combinations, or new data columns).
- Data analysis using statistical techniques (descriptive measures, correlations, logistic regression, modern model selection, or Bayesian hierarchical models).
- Present results and generate reports.

- Calculations!
Comparison: Calculator vs PC (SAS)
Example 1: \( X = 50 + 6 \)
Comparing simple calculations

Example 1: \( X = 50 + 6 \)

Regular Calculator

PC with SAS

```sas
data calculation;
  x=50+6;
  put 'The answer is:' x;
run;
```

```
68     data calculation;
69     x=50+6;
70     put 'The answer is:' x;
71     run;
```

The answer is: 56
Comparing simple calculations

Example 2: $X = \left(\frac{9}{45}\right)(5*8) + 9(42)$

Regular Calculator

PC with SAS
Comparing simple calculations

Example 2: \( X = \frac{9}{45}(5 \times 8) + 9(42) \)

Regular Calculator

PC with SAS

```
data calculation;
x=(9/45)*(5*8)+9*(42);
put 'The answer is:' x;
run;
```

The answer is: 386
Comparing simple calculations

Example 3: \( X = (3 \times 0.1) - 0.3 \)

Regular Calculator

PC with SAS
Comparing simple calculations

Example 3: \( X = (3 \times 0.1) - 0.3 \)

Regular Calculator

PC with SAS

\[
\begin{align*}
data\ calculation; \\
x &= (3 \times 0.1) - 0.3; \\
p\ put 'The answer is:' x; \\
run;
\end{align*}
\]

The answer is: \( 5.551115E-17 \)
Numerical precision in general

### Everyday Numerical Errors:

#### Example 1: Base 10

<table>
<thead>
<tr>
<th>Fraction</th>
<th>Decimal (base 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/1</td>
<td>2</td>
</tr>
<tr>
<td>1/1</td>
<td>1</td>
</tr>
<tr>
<td>1/2</td>
<td>0.5</td>
</tr>
<tr>
<td>1/3</td>
<td>0.33333</td>
</tr>
<tr>
<td>1/4</td>
<td>0.25</td>
</tr>
</tbody>
</table>

#### Example 2: ???

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.57</td>
<td>1.55</td>
<td></td>
</tr>
<tr>
<td>60.0</td>
<td>60.0</td>
<td></td>
</tr>
<tr>
<td>2.35</td>
<td>2.35</td>
<td></td>
</tr>
<tr>
<td>8.88</td>
<td>8.90</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>1.00</td>
<td></td>
</tr>
</tbody>
</table>
Numerical precision in general

A global programming problem:

Most programming languages do not understand recursion:

$$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$$

$$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} \neq 1$$
Recall: $(3 \times 0.1) - 0.3$

Using decimal arithmetic, the value 0.1 has an exact representation.
Using binary arithmetic, the value 0.1 does not have an exact representation.

$$0.0001100110011001100110011001100110011001100110011001$$

At some point, the value represented is truncated or rounded, leading to error.
How does SAS store numbers:

Consider the number 987, which can be also be expressed as: $0.987 \times 10^3$

\[
(+) \ 0.987 \quad \times \quad 10^3
\]

- **Sign**: Positive or Negative
- **Mantissa**: Represents the number to be multiplied by the Base
- **Base**: The number being raised to a power
- **Exponent**: The power to which the base is raised
How does SAS store numbers:

Consider the number 987, which can be also be expressed as: $0.987 \times 10^3$

\[ (+) 0.987 \times 10^3 \]

Floating Point Representation

Another Day
8 or 9 step process
Bias of 127, implicit digits
How does SAS store numbers:

Remember that the default length of a numerical variable in SAS is 8 bytes.

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6</th>
<th>Byte 7</th>
<th>Byte 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
</tbody>
</table>

- **Exp**: Sign bit (1 for positive, 0 for negative), Exponent bits
- **Sign**: Sign bit (1 for negative, 0 for positive), Exponent bits
- **Mantissa**: Other bits

**987 =**

```
01000100 01110110 11000000 00000000 00000000 00000000 00000000 00000000
```

**0.1 =**

```
00111101 11001100 11001100 11001100 11001100 11001100 11001100 11001101
```

Error
Why Floating Point Representation?

Space and Time.

Floating point representation allows the efficient calculation of very large and/or very small numbers using the same predictable 8 bytes.

In essence, Floating Point Representation is Scientific Notation in Base 2
Example 1: Iterations.

```sas
data _null_
  do i=-1 to 1 by .1;
    put i=
    if i=0 then put 'AT ZERO';
  end;
run;
```

- \( i = -1 \)
- \( i = -0.9 \)
- \( i = -0.8 \)
- \( i = -0.7 \)
- \( i = -0.6 \)
- \( i = -0.5 \)
- \( i = -0.4 \)
- \( i = -0.3 \)
- \( i = -0.2 \)
- \( i = -0.1 \)
- \( i = -1.38778E-16 \)
- \( i = 0.1 \)
- \( i = 0.2 \)
- \( i = 0.3 \)
- \( i = 0.4 \)
- \( i = 0.5 \)
- \( i = 0.6 \)
- \( i = 0.7 \)
- \( i = 0.8 \)
- \( i = 0.9 \)
- \( i = 1 \)
Example 2: Manipulations

Example 2: Data Manipulations:

```plaintext
data a;
x=15.7;
y=-11.9;
z=x+y;
if z=3.8 then put 'eligible';
else put 'not eligible';
run;
```

```plaintext
data a;
x=15.7;
y=-11.9;
z=x+y;
if z=3.8 then put 'eligible';
else put 'not eligible';
run;
```

not eligible
How this relates to you

One well-rounded solution:

```plaintext
data _null_;  
do i=-1 to 1 by .1;  
I = round(I, .1);  
put i=;  
if i=0 then put 'AT ZERO';  
end;  
run;

data a;  
x=15.7;  
y=-11.9;  
z=round(x+y,0.1);  
if z=3.8 then put 'eligible';  
else put 'not eligible';  
run;
```

```plaintext
119 data a;  
120 x=15.7;  
121 y=-11.9;  
122 z=round(x+y,0.1);  
123 if z=3.8 then put 'eligible';  
124 else put 'not eligible';  
125 run;

eligible
```
And finally... \( X = (3 \times 0.1) - 0.3 \)
Thank you

Resources:

http://support.sas.com/documentation/cdl/en/lrcon/68089/HTML/default/viewer.htm#p0ji1unv6thm0dn1gp4t01a1u0g6.htm

https://www.youtube.com/watch?v=PZRI1fStYO