Examples from

SAS Functions by Example

Ron Cody

Herman Lo

Technical Analyst, RBC Capital Markets
Agenda

- Book Structure
- Examples from the Book
  - Character Functions (CATS, CATX)
  - Date and Time Functions (INTCK, INTNX)
  - Descriptive Stats (IQR, SMALLEST/LARGEST)
  - Special Functions (INPUT, PUT)
  - Macro Functions (CALL SYMPUT, CALL SYMPUTX)
- The Verdict
Book Structure

- TOC – List of Chapters
- List of Programs
- At the beginning of each chapter
  - List of Functions
- Inside each chapter
  - For each function
    - Purpose
    - Syntax
    - Examples/Anticipated Outputs
    - A Sample Program
- At the back
  - List of Functions
  - Index (Alphabetical)
Character Functions

- **CATS**\( (\text{string-1}, \text{string-2} \ldots) \)
  - Joins strings, stripping both leading and trailing blanks
- **CATX**\( (\text{string-1}, \text{string-2} \ldots) \)
  - Joins strings, stripping both leading and trailing blanks, and add a space in between them

**Example**

\[
\begin{align*}
A &= \text{“Star”} \\
B &= \text{“Wars”} \\
\text{CATS}(A, B) &= \text{“StarWars”} \\
\text{CATX}(A, B) &= \text{“Star Wars”}
\end{align*}
\]

(See Pg. 57-58)
Date and Time Functions

- INTCK(‘interval<Multiple><.shift>’, date1, date2)

  - Returns number of Intervals between date1 and date2

  - Date1 and Date2 can be date, time, datetime values
    - date: ’01JUN2000’d
    - time: ’9:15:09’T
    - datetime: ’01JUN2000:9:15:09’DT

  - Interval = The unit of the interval
    - Interval(date) = DAY, WEEK, WEEKDAY, ...
    - Interval(time) = SECOND, MINUTE, HOUR
    - Interval(datetime) = DTDAY, DTWEEK, DTWEEKDAY, ...
Date and Time Functions

- INTCK(‘interval<Multiple><.shift>’, date1, date2)
  - Multiple *(optional)* = Multiple of interval unit
    - DAY50 = 50-DAY intervals
  - Shift *(optional)* = starting point of interval
    - Meaning of Shift depends on the Interval
    - Interval=YEAR, SEMIYEAR, QTR, MONTH → Shift = MONTH
      - YEAR4.11 = 4-YEAR intervals starting on November
    - Interval=SEMIMONTH, HOUR, MINUTE, SECOND → Shift=Interval
      - but only Multi-intervals can be shifted (Multiple must be specified)
      - HOUR8.6=8-HOUR intervals starting at 6AM (6AM, 2PM, 10PM)
  - See Pg. 186
Date and Time Functions

- INTCK('interval<Multiple><.shift>', date1,date2)

Examples

INTCK('YEAR.7', '05MAY2002'd, '15JUL2002'd) = 1

INTCK('WEEK', '01JAN1960'd, '04JAN1960'd) = 1

01JAN1960 is a Sunday, so the week counter is triggered because default WEEK starting point is Sunday.

CAREFUL:
Results may surprise you (off-by-one problems). Watch where the starting point is.
Date and Time Functions

- **INTNX('interval',start-date,increment<,'alignment'>)**
  - Interval = same as INTCK
  - Start-date = starting date
  - increment = # of intervals between start date and output date
  - alignment (optional) = BEGINNING, MIDDLE, END of Interval
    - default is BEGINNING

- **Example**
  - `INTNX('WEEK','01JAN1960'd,1,'MIDDLE')='06JAN,1960'd`
Descriptive Stats

- **IQR(<of> numeric-values)**
  - Computes the interquartile range (25th percentile and 75th percentile) in a list of values
  - Use **of** to define a list of values
  - **Examples**
    - \(X1=1, X2=2, X3=3, X4=\)
    - \(\text{IQR(of } X1-X4)=2\)
    - \(\text{IQR}(X1,X2,X3,X4)=2\)
Descriptive Stats

- **SMALLEST/LARGEST(N,<of> numeric-values)**
  - Finds the Nth smallest or largest number in the list of values
  - Returns missing value and writes an error in log if N is larger than number of values
  - Returns missing value and does not write an error in log if N is larger than the number of missing values

- **Examples**
  - X1=1, X2=2, X3=3, X4=., X5=.
  - SMALLEST(3, X2, X3, X4)=.
  - LARGEST(1, X1, X2, X3, X4)=3
Special Functions

- **INPUT(value, informat)**
  - Performs character-to-numeric conversion.
  - Value is a character variable
  - Informat is a SAS informat, defines the input format of Value

- **PUT(value, format)**
  - Performs numeric-to-character conversion.
  - Value is a character variable
  - Format is a SAS format, defines the output format

- Program Example from Pg. 302
***Primary functions: PUT, INPUT;
PROC FORMAT;
VALUE ITEM
  1=’APPLE’
  2=’PEAR’
  3=’GRAPE’
  OTHER=’UNKNOWN’;
VALUE $COST
  ‘A’ – ‘C’ = ’44.45’
  ‘D’ = ’125.’
  OTHER = ‘’;
RUN;

DATA TABLE;
  INPUT ITEM_NO CODE $ @@;
  ITEM_NAME = PUT(ITEM_NO, ITEM.);
  AMOUNT = INPUT(PUT(CODE, $COST.), 9.);
DATALINES;
  1 B 2 D 3 X 4 C
;
PROC PRINT DATA=TABLE NOOBS;
  TITLE “Listing of Data Set TABLE”;
RUN;

<table>
<thead>
<tr>
<th>ITEM_NO</th>
<th>CODE</th>
<th>ITEM_ NAME</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>APPLE</td>
<td>44.45</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>PEAR</td>
<td>125.00</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>GRAPE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>UNKNOWN</td>
<td>44.45</td>
</tr>
</tbody>
</table>
Macro Functions

- CALL SYMPUT(macro-var, character-value)
- CALL SYMPUTX(macro-var, character-value)
  - Assigns a value to a macro variable during execution of DATA step
  - CALL SYMPUTX: Blanks are not removed from value before assignment to macro variable
  - CALL SYMPUTX: Leading & trailing blanks are removed from value before assignment to macro variable
- Equivalent to `%let` statement in macro language
Macro Functions

- Example

```sas
DATA TEST;
  INPUT STRING $CHAR10. ;
  CALL SYMPUT("StringWithBlanks",STRING);
  CALL SYMPUTX("StringWithoutBlanks",STRING);
DATA _NULL_;  
  ABC ;
DATA _NULL_;  
  WITHBLANKS = "----" || "&StringWithBlanks" || "----";
  WITHOUTBLANKS = "----" || "&StringWithoutBlanks" || "----";
  PUT "Value of StringWithBlanks is " WITHBLANKS;
  PUT "Value of StringWithBlanks is " WITHOUTBLANKS;
RUN;
```

**SAS LOG**

```
Value of WITHBLANKS is ---- ABC ----
Value of WITHOUTBLANKS is ----ABC----
```
File I/O Functions

- File I/O functions are used to obtain information about SAS data sets

- \[ \text{dsid} = \text{OPEN('data-set-name' <,'mode'>)} \]
  - Opens a SAS data set with the name `data-set-name` and returns a data set ID `dsid`
  - A Data set ID is necessary for File I/O Functions
  - If data set cannot be opened, OPEN returns a 0

- \[ \text{EXIST(dsid)} \]
  - Returns 1 if Data set exists and a 0 otherwise

- \[ \text{CLOSE(dsid)} \]
  - Closes SAS data set after it has been opened by the OPEN function
File I/O Functions

- **ATTRC(dsid,'attribute')**
  - Returns the **character** value of a character type attribute
  - i.e. ATTRC(DSID, ‘sortedby’) = name of the BY variable. Empty if not sorted.

- **ATTRN(dsid,'attribute')**
  - Returns the **numeric** value of a character type attribute
%MACRO NOBS(DSN);
   IF EXIST("&DSN") THEN DO;
      DSID = OPEN("&DSN");
      NOBS=ATTRN(DSID,"ANY");
      NOBS=ATTRN(DSID,"NLOBS");
      NOBS=ATTRN(DSID,"NVARs");
   END;
   ELSE NOBS=.;
   RC = CLOSE(DSID);
%MEND NOBS;

DATA _NULL_;
   %NOBS(TABLE);
   PUT ANY=;
   PUT NLOBS=;
   PUT NVARs=;
RUN;

SAS LOG

ANY=1   (indicates that dataset has both observations and variables)
NLOBS=4 (dataset has 4 logical observations)
NVARS=4 (dataset has 4 variables)

Listing of Data Set TABLE

<table>
<thead>
<tr>
<th>ITEM_NO</th>
<th>CODE</th>
<th>ITEM_NAME</th>
<th>AMOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>B</td>
<td>APPLE</td>
<td>44.45</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>PEAR</td>
<td>125.00</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>GRAPE</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>UNKNOWN</td>
<td>44.45</td>
</tr>
</tbody>
</table>
The Verdict

- The book is an excellent Handbook on SAS Functions
- New SAS9.1 Functions are demonstrated
- What I wish it has:
  - An associated web resource/software help manual for ease of lookup