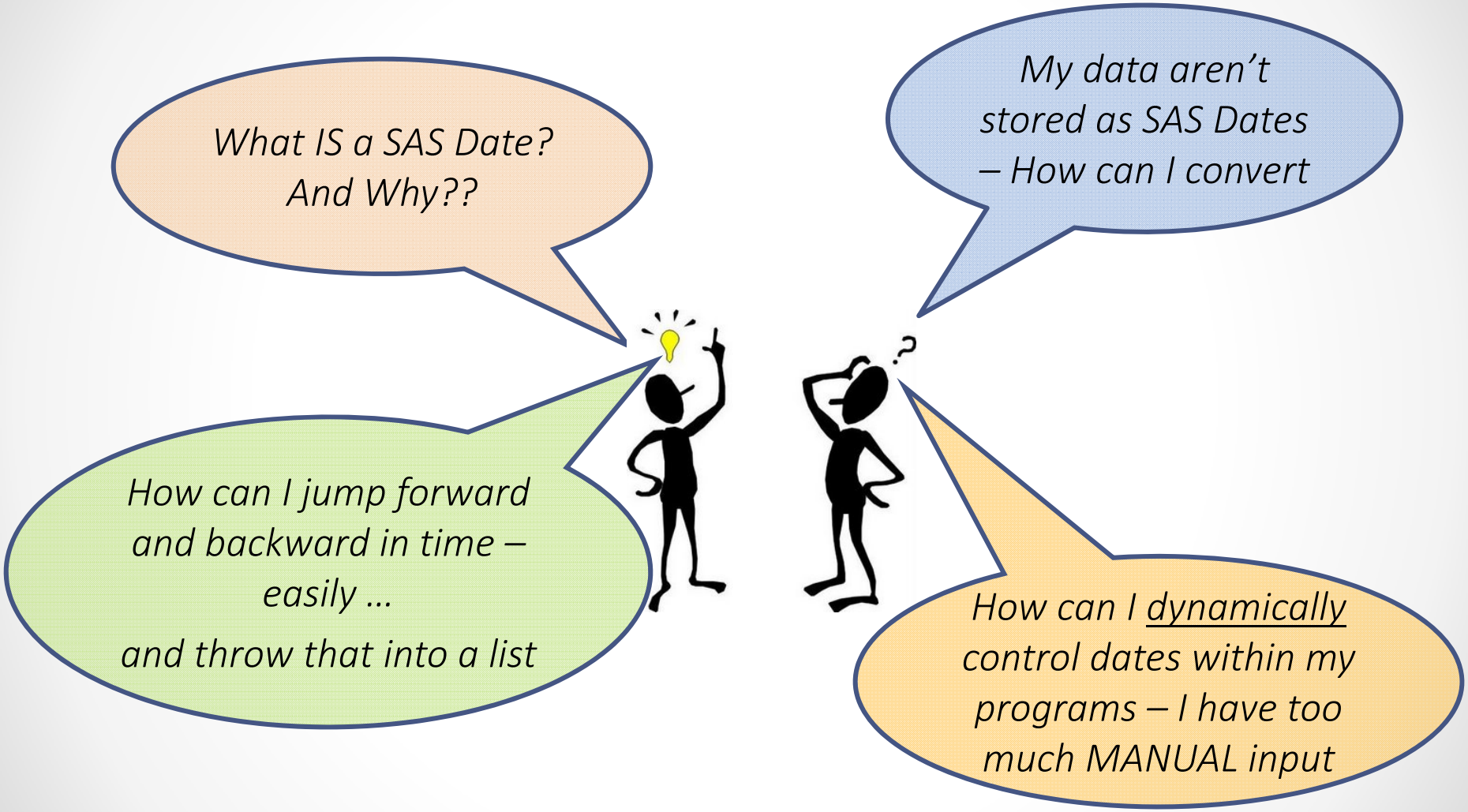
All About SAS Dates

Marje Fecht

Senior Partner, Prowerk Consulting

SAS Dates



Getting the RIGHT Date can be Tricky

This presentation focuses on working with dates in **SAS**.

- Converting Text values to SAS Dates
- Comparing Dates
- Extracting Date Components
- Moving Forward and Backward in Time
- Outputting Dates in Desired Formats
- Generalizing Code to Accommodate Dates

SAS Dates… The Origin

A SAS Date is stored as # of days since January 1, 1960

Limitation: Valid from 1582 A.D. to 19900 A.D.

Origin: See Rick Langston's historical note
https://web.archive.org/web/20080706004217/http:/support.sas.com/community/newsletters/news/insider/dates.html

```
Today_date = today();    /* # days today is since 1/1/1960 */

Today_date = Date();     /* # days today is since 1/1/1960 */


/* Arithmetic works!! */
EndDate   = StartDate + 14;
Yesterday = today() - 1;
```

Subset Data Based on Dates

Goal:

Subset data on a date stored as a SAS date

Two approaches:

- Static date
- Dynamic based on today's date

```
Data SpecificDate;
  set alldata;
  /* use a DATE CONSTANT */
  where SaleDate = '23Mar2017'd;
run;


Data CurrentMonth;
 set alldata;
 where month(SaleDate) = month( date() )
  and  year(SaleDate)  = year( date() );
  . . .
```

Warning! Warning!

How many times should you execute

`today()`

in your program / process?

ONE Time

```
%let Today_dt = %sysfunc( today() );
```

- Efficiency (reduce function calls)
- Programs / processes: run time that crosses midnight

Alternate Solution – Fewer Function Calls

Current Month

```
/**** Extract all data for current month ***/

  if me_dt =
         INTNX ( 'MONTH'     /*increment = month*/
                , today()    /* start at today */
                , 0          /* move ZERO months*/
                , 'E')       /* return END of mth */
            ;
```

Result on 15Nov2017 is ME_DT = 21153

the SAS Date value corresponding to 30Nov2017

INTNX = *move in intervals*

INTNX - handy to dynamically create different variations of dates.

- *increments dates by intervals*

INTNX (*interval*, *from*, *n* < , *alignment* >) ;

- ***interval*** **-** interval name eg: 'MONTH', 'DAY', 'YEAR' , etc
- ***from*** **-** a SAS date value (for date intervals) or datetime value (for datetime intervals)
- ***n*** **-** number of intervals to increment from the interval that contains the *from* value
- ***alignment*** **-** alignment of resulting SAS date, within the interval.

  Eg: **B**EGINNING, **M**IDDLE, **E**ND.

Locate TWO Months ago

```
/* macro variable with yyyymm for TWO months ago */

%let M2 = %sysfunc(
            intnx( MONTH , %sysfunc( today() ) , -2)
            , yymmN6.);
```

- Notice that a fourth INTNX argument is not needed
- ***Result on 15Nov2017 is 201709***

Note:

- ***when use INTNX in %SYSFUNC, do not use quotes for the arguments of INTNX.***
- ***SYSFUNC uses the second argument to format the result***

Dynamic Date values

2 macro variables for 1st and last day of previous month

```
%let M1_beg = %sysfunc( intnx( MONTH
                    , %sysfunc( today() ) , -1 , B)
                    , date9.);
%let M1_end = %sysfunc( intnx( MONTH
                    , %sysfunc( today() ) , -1 , E)
                    , date9.);
```

Note:
Use whatever date format is appropriate for your data.

Dynamic Date Generation

Create a **SET** of **3** macro variables per month for 1st and last day and label (yyyymm)

```
/*  This is INSIDE of a macro definition  */
%do Num = 0 %to &MaxMonth;
/*** use %global if macro variables used outside macro ***/
  %global M&Num. M&Num._beg M&Num._end;
  %let M&Num. = %sysfunc( intnx( MONTH ,%sysfunc( today() )
                          , -&Num.) , yymmN6.);
  %let M&Num._beg = %sysfunc( intnx( MONTH , %sysfunc( today() )
                          , -&Num. , B) , date9.);
  %let M&Num._end = %sysfunc( intnx( MONTH , %sysfunc( today() )
                          , -&Num. , E) , date9.);
%end;
```

Creates:

M0, M0_beg, M0_end ← Current Month

M1, M1_beg, M1_end ← Previous Month

etc

Using Dates to Control Program Flow

```
*** Run monthly report on first of each month;

%macro dayone;
   %if %sysfunc(day( %sysfunc(today()) )) = 1
      %then %do;
         %include 'monthly_report.sas';
      %end;
%mend;

%dayone

*** Run daily report every day;
%include 'daily_report.sas';
```

Changing Character Dates to SAS Dates

```
data Change_Char_To_Date(DROP CharDate);
  CharDate = '1957-03-15';
  putlog 'Char Value: ' chardate;
  /**convert to a SAS date **/
  SAS_date = input(chardate , yymmdd10.);

  putlog 'Stored value of SAS_Date:'  SAS_date;
  putlog 'Format with ddmmyyS10.: '
                                    SAS_Date ddmmyyS10.;
run;

   Char Value: 1957-03-15
   Stored value of SAS_Date:-1022
   Format with ddmmyyS10.: 15/03/1957
```

Dynamic Dates in Filenames

Goal: **Store the current date and time in a macro variable to use for Version Control in file names**

Output: *yyyymmdd_hhmm*

```
%let DateTime =
        %sysfunc(
                  compress(
                           %sysfunc(today(),yymmddN8.)_
                           %sysfunc(time(),hhmm6.), ': '
                           )
                 );
    /* result looks like    20171115_1426    */
/* Note: N in yymmddN8 requests NO separators( Dash, Slash, etc)*/

** route Log to permanent location and version control**;
proc printto  log =
  "&Filepath_Out.\logs\&stage.\&Strategy_ID._&datetime..log"
;run;
```

Changing how Dates are Displayed

Goal: **Use SAS Date Formats to change how dates are displayed**

```
data dates;
   SystemDate = today();
   putlog 'Unformatted System Date is: ' SystemDate  ;
   putlog 'formatted with date9.      : ' SystemDate date9.    ;
   putlog 'formatted with worddate.   : ' SystemDate worddate. ;
   putlog 'formatted with weekdate.   : ' SystemDate weekdate. ;
   putlog 'formatted with weekdate9.  : ' SystemDate weekdate9.;
run;
```

```
Unformatted System Date is: 21138
formatted with date9.      : 15NOV2017
formatted with worddate.   :   November 15, 2017
formatted with weekdate.   :   Wednesday, November 15, 2017
formatted with weekdate9.  : Wednesday
```

Determine the Fiscal Year of a Date

```
/**** Create a Macro to compute fiscal year
          start = beginning month for FY          ***/
%macro fy (date , start=7);
   year(&date) + /* BELOW returns a 0 (F) or a 1 (T) */
          (month(&date) ge &start and &start ne 1)
%mend;

/*** example of macro usage ***/
data try_FY;
  Txn_date = '21nov2017'd;
  FiscalYear = %FY( Txn_Date , start = 11 );
  putlog FiscalYear =  Txn_Date=date9.;
run;
```

FiscalYear=2018 Txn_date=21NOV2017

Difference between two Dates

DATDIF and **YRDIF** functions - number of days(years) between two dates

DATDIF (*sdate , edate , basis*)

YRDIF (*sdate , edate , basis*)

- *sdate* – starting date
- *edate* – ending date
- *Basis* – a character constant that describes HOW to calculate the date difference
  - '30/360' or '360' = 30 day month and 360 day year
  - 'ACT/ACT' or 'ACTUAL' = Actual number of days (years)

To explore

There are so many functions and formats available for manipulating dates! Spend time reviewing:

- Online documentation
- Conference Proceedings
- Communities

Thank You !

Marje Fecht

marje.fecht@prowerk.com