



User-Written DATA Step Functions

Jason.Secosky@sas.com

**THE
POWER
TO KNOW®**

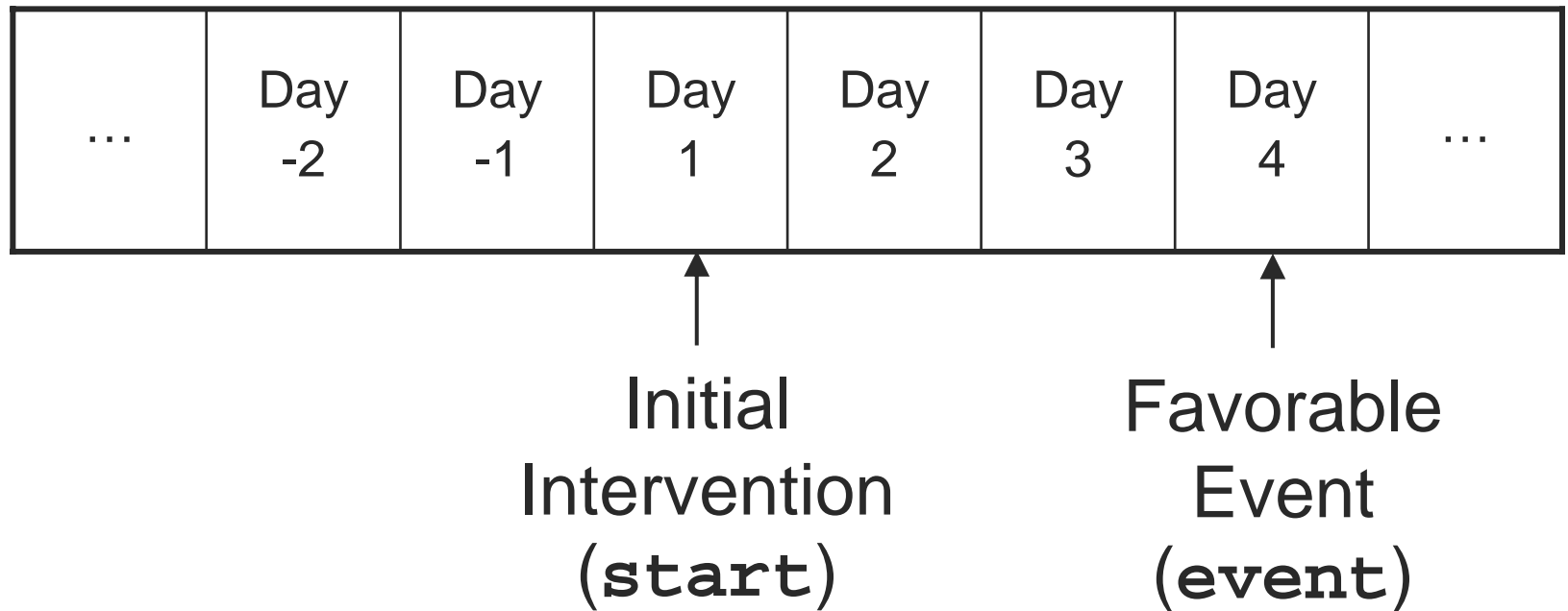
The Power to Know®

- Code abstraction & sharing
can be cumbersome
- Code your own functions
in SAS 9.2
- We'll get:
 - Simpler code
 - Easier to understand
 - More protections
- Jason is a great speaker.
Invite to company mtg. in Hawaii

Preview

- Write a "study day" routine
 - LINK and RETURN
 - Macros
 - PROC FCMP
- Structure, Scope, and Recursion
- Extended Example: Directory Walker

Compute Study Day



Compute Study Day

```
if event < start then
```

```
    study_day = event - start;
```

```
else
```

```
    study_day = event - start + 1;
```

Or, Use LINK and RETURN

```
event = subj_event;
```

```
link study_day;
```

```
...
```

```
study_day:
```

```
    if event < start then
```

```
        study_day = event - start;
```

```
    else
```

```
        study_day = event - start + 1;
```

```
return;
```

Wrap in a Macro

```
%macro study_day(start, event);  
  if &event < &start then  
    study_day = &event - &start;  
  else  
    study_day = &event - &start + 1;  
%mend;
```

Macro Usage

```
%study_day(subj1_start, subj1_ae, subj1_day);
```

```
%study_day(subj2_start, subj2_ae, subj2_day);
```

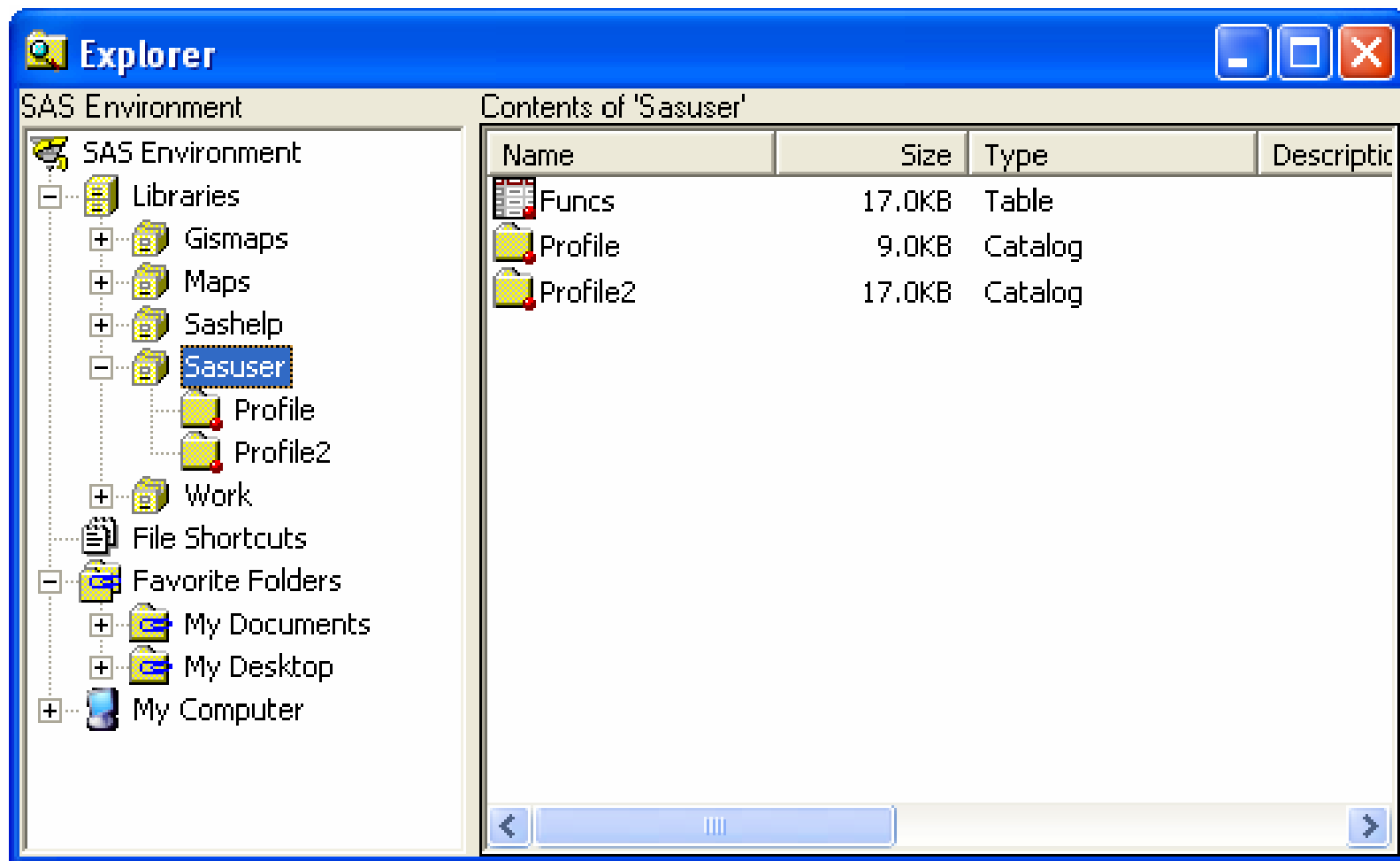
```
if subj1_day < subj2_day then ...
```

Function Usage

```
if study_day(subj1_start, event) <  
    study_day(subj2_start, event)  
then ...
```

Study Day Function

```
proc fcmp outlib=sasuser.funcs.trial;  
  function study_day(start, event);  
    if event < start then  
      return(event - start);  
    else  
      return(event - start + 1);  
  endsub;
```




SAS Function Viewer [minimize] [maximize] [close]

File Edit

Open Functions: SASUSER.FUNCS.trial.study_day [v]

Function Sources

Name
GISMAPS
OLD_WORK
SASHELP
SLKWXL
+ f() finance
+ f() math
+ f() stat
SASUSER
FUNCS
f() trial
f() study_day
WORK

SAS® Function Viewer *The Power to Know.* 

SASHELP.SLKWXL.finance.yield_slk [minimize] [maximize] [close]

```
function yield_slk(settlement,maturity,...
```

SASUSER.FUNCS.trial.study_day [minimize] [maximize] [close]

```
function study_day(start, event);
  if event < start then
    return(event - start);
  else
    return(event - start + 1);
endsub;
```

Function study_day Loaded. [v] [user icon]

localhost : 6591

Calling Study_Day

```
options cmplib = (sasuser.funcs shared.funcs);  
data _null_;  
    start = '15Feb2006'd;  
    today = '27Mar2006'd;  
    sd = study_day(start, today);  
    put sd=;  
run;
```

Advantage of Functions

- Safeguards:
 - No side-effects
 - Local variables
 - Parameters
- Use result in an expression
- The natural thing to do

Preview

- ~~■ Write a "study day" routine~~
 - ~~● LINK and RETURN~~
 - ~~● Macros~~
 - ~~● PROC FCMP~~
- Structure, Scope, and Recursion
- Extended Example: Directory Walker

Function Structure

```
function <name>( <parameters> );
```

```
    <declaration section>
```

```
    <statements section>
```

```
endsub;
```

```
subroutine <name>( <parameters> );
```

```
    <declaration section>
```

```
    <statements section>
```

```
endsub;
```

Scope

```
subroutine subA();  
  x = 5;  
  call subB();  
  put 'In subA:' x=;  
endsub;  
  
subroutine subB();  
  x = 'subB';  
  put 'In subB:' x=;  
endsub;  
  
data _null_;  
  x = 99;  
  call subA();  
  put 'In DATA Step: ' x=;  
run;
```

Scope

```
subroutine subA();  
  x = 5;  
  call subB();  
  put 'In subA:' x=;  
endsub;  
  
subroutine subB();  
  x = 'subB';  
  put 'In subB:' x=;  
endsub;  
  
data _null_;  
  x = 99;  
  call subA();  
  put 'In DATA Step: ' x=;  
run;
```

SAS Log

```
In subB: x=subB  
In subA: x=5  
In DATA Step: x=99
```

Preview

- ~~■ Write a "study day" routine~~
 - ~~● LINK and RETURN~~
 - ~~● Macros~~
 - ~~● PROC FCMP~~
- ~~■ Structure, Scope, and Recursion~~
 - Extended Example: Directory Walker

Directory Walker Output

C:\logs

\2004

\qtr1.log

\qtr2.log

\qtr3.log

\qtr4.log

\2005

\qtr1.log

\qtr2.log

\qtr3.log

\qtr4.log

\2006

\qtr1.log

\qtr2.log

SAS Log

```

c:\logs\2004\qtr1.log
c:\logs\2004\qtr2.log
c:\logs\2004\qtr3.log
c:\logs\2004\qtr4.log
c:\logs\2005\qtr1.log
c:\logs\2005\qtr2.log
c:\logs\2005\qtr3.log
c:\logs\2005\qtr4.log
c:\logs\2006\qtr1.log
c:\logs\2006\qtr2.log

```



Directory Walker

- Input: Starting directory
- Output:
 - Array filled with filenames
 - Number of filenames in array
 - Whether array was too small
- `call dir_entries("c:\logs", files,
n, trunc);`



Directory Walker

```
subroutine dir_entries(dir $, files[*] $,  
                    n, trunc);  
  
    outargs n, trunc;  
    length dir $ 256;  
  
    open directory  
    for each entry in directory  
        entry = dir || '\ ' || entry  
        if entry is a file  
            n = n + 1  
            files[n] = entry  
        else  
            call dir_entries(entry, files, n, trunc)  
    close directory  
endsub;
```

Directory Walker

```
subroutine dir_entries(dir $, files[*] $,  
                    n, trunc);  
  
    outargs n, trunc;  
    length dir entry $ 256;  
  
    if trunc then return;  
    did = diropen(dir);  
    if did <= 0 then return;  
  
    dnum = dnum(did);  
    do i = 1 to dnum;  
        entry = dread(did, i);  
        fid = mopen(did, entry);  
        entry = trim(dir) || "\" || entry;
```

Directory Walker

```
    if fid > 0 then do;
        rc = fclose(fid);
        if n < dim(files) then do;
            trunc = 0;
            n = n + 1;
            files[n] = entry;
        end;
    else do;
        trunc = 1;
        return;
    end;
else
    call dir_entries(entry, files, n, trunc);
end;
```

Directory Walker

```
    call dirclose(did);  
    return;  
endsub;
```

Directory Walker

```
data _null_;  
  array files[1000] $ 256 _temporary_;  
  num = 0;  
  trunc = 0;  
  call dir_entries("c:\logs", files,  
                  num, trunc);  
  if trunc then put 'ERROR: ...';  
  do i = 1 to num;  
    put files[i];  
  end;  
run;
```

Directory Walker Output

C:\logs

\2004

\qtr1.log

\qtr2.log

\qtr3.log

\qtr4.log

\2005

\qtr1.log

\qtr2.log

\qtr3.log

\qtr4.log

\2006

\qtr1.log

\qtr2.log

SAS Log

```

c:\logs\2004\qtr1.log
c:\logs\2004\qtr2.log
c:\logs\2004\qtr3.log
c:\logs\2004\qtr4.log
c:\logs\2005\qtr1.log
c:\logs\2005\qtr2.log
c:\logs\2005\qtr3.log
c:\logs\2005\qtr4.log
c:\logs\2006\qtr1.log
c:\logs\2006\qtr2.log
    
```

Directory Walker Review

- Similarity to DATA step syntax
- Using recursion to ease programming
- Passing `_temporary_` arrays
- Using OUTARGS to return values

Syntax Differences

- PUT Statement
- IF Expressions
 - `x = if y < 100 then 1 else 0;`
- Data set and file I/O
- DATA step debugger

What I've Not Shown

- Dynamic arrays
- Use from PROC REPORT
- FCMP Package Viewer
- Calling C and C++ functions
- SOLVE function
- Microsoft Excel clones

The Power to Know®

- Code abstraction & sharing
can be cumbersome
- Code your own functions
in SAS 9.2
- We'll get:
 - Simpler code
 - Easier to understand
 - More protections
- Jason is a great speaker.
Invite to company mtg. in Hawaii

A Wise Old Owl



A wise old owl sat in an oak

The more he saw, the less he spoke

The less he spoke the more he heard

*Why can't Jason be like that wise old
bird?*

-- Anonymous