2026 Korea CDISC Day

29 October 2026 · Seoul, Korea

CDISC Implementation - Turning CDISC Rules into Functions

Sanggi Lee
Manager, STAT Programming
C&R Research

29 October 2026

Agenda

1. **The problem**
   Same CDISC rules, re-coded in every program
2. **Why functions**
   PROC FCMP vs. macros
3. **Building the library**
   Structure, examples, and before / after
4. **QC and Validation**
   Independent QC, function testing, change control
5. **Next steps**
   How to get started and functions to build next

Speaker

Sanggi Lee

Manager, STAT Programming
C&R Research

Sanggi Lee is a Manager of Statistical Programming at C&R Research. He holds a master's degree in statistics.

He began his career at SAS Korea, where he worked for about four years on projects such as quality control, executive information systems (EIS), and churn management. He also supported pre-sales, proposing how clients could use SAS for their business needs.

He then spent more than 20 years in the banking industry, leading customer value analysis and cross-selling modeling, as well as CRM, data warehouse, and big data analytics.

For the past five years at C&R Research, he has led statistical programming across a wide range of studies, including with and without CDISC standards, PMS, and OS.

The problem

Same CDISC rules, re-coded in every program

One rule, three implementations

Study A

```
aedy = aestdt - rfstdt + 1;
```

Day 0 before reference date

Study B

```
if aestdt >= rfstdt then
   aedy = aestdt - rfstdt + 1;
else aedy = aestdt - rfstdt;
```

Correct

Study C

```
if aestdt > rfstdt then
   aedy = aestdt - rfstdt + 1;
else aedy = aestdt - rfstdt;
```

Day 0 on reference date

Each program passes its own QC, but results can differ across studies, and the same logic is reviewed again every time. These differences are often found late, when data from several studies are combined for ISS/ISE.

CDISC rules we re-implement in every study

DTC

ISO 8601 dates

Raw date/time to --DTC, including partial dates

DY

Study day

--DY relative to RFSTDTC, no Day 0

IMP

Date imputation

Partial dates to ADT/ASTDT with imputation flags

AGE

Age & durations

AGE, ADURN and similar derived intervals

VIS

Visit windowing

Assigning records to analysis visits (AVISIT)

CT

Lookups & units

Controlled terminology and unit conversions

Why functions

PROC FCMP vs. macros

Macros vs. FCMP functions

	SAS macro	PROC FCMP function
Unit of work	Whole dataset or step	One value or one record
How it is called	%macro(...) around steps	Inline: aedy = studyday(...)
Where it fits	Program flow, repeated steps	Inside DATA step expressions
What it returns	Generated SAS code	A value

Not a replacement: we keep macros for dataset-level work and use functions for the rules inside them.

PROC FCMP in three steps

1 Define

Compile functions into a library

```
proc fcmp
  outlib=funclib.cdisc.dates;
  function ...;
  endsub;
run;
```

2 Set path

Tell SAS where to find them

```
options
  cmplib=funclib.cdisc;
```

3 Call

Use them like built-in functions

```
data ae;
  set ae;
  aedy = studyday(
    aestdt, rfstdt);
run;
```

Building the library

Structure, examples, and before / after

Example 1: Study day without Day 0

```
proc fcmp outlib=funclib.cdisc.dates;
  function studyday(dt, refdt);
    /* missing in, missing out */
    if missing(dt) or missing(refdt)
      then return(.);
    /* on/after reference: +1 */
    if dt >= refdt
      then return(dt - refdt + 1);
    /* before reference: no Day 0 */
    else return(dt - refdt);
  endsub;
run;
```

Test cases (RFSTDT = 2024-03-10)

DT	Expected --DY
2024-03-09	-1
2024-03-10	1
2024-03-11	2
2024-02-29	-10
(missing)	(missing)

In SDTM, there is no Day 0. Forgetting this rule is the most common mistake we see in hand-written code.

Example 2: ISO 8601 dates, including partial

```
proc fcmp outlib=funclib.cdisc.dates;
     /* declare the return length */
     function iso_date(yy, mm, dd) $ 10;
          length dtc $10;
          if missing(yy)
               then dtc = '';
          else if missing(mm)
               then dtc = put(yy, 4.);
          else if missing(dd)
               then dtc = catx('-', put(yy,4.), put(mm,z2.));
          else dtc = catx('-', put(yy,4.),
          put(mm,z2.), put(dd,z2.));
          return(dtc);
     endsub;
run;
```

Collected components to --DTC

YY / MM / DD	--DTC
2023 / 5 / 14	2023-05-14
2023 / 5 / .	2023-05
2023 / . / .	2023
. / . / .	(blank)

Simplified: A known day with an unknown month (e.g. 2023---14) is rare, so it is not handled here. Add a rule if your data needs one.

Example 3: Date imputation with flags

```
proc fcmp outlib=funclib.cdisc.dates;
  subroutine imp_start(dtc $, adt, dtf $);
    outargs adt, dtf;
    adt = .; dtf = '';
    if length(dtc) >= 10 then
      adt = input(substr(dtc,1,10), e8601da.);
    else if length(dtc) = 7 then do;
      adt = input(cats(dtc,'-01'), e8601da.);
      dtf = 'D'; end;
    else if length(dtc) = 4 then do;
      adt = input(cats(dtc,'-01-01'),e8601da.);
      dtf = 'M'; end;
  endsub;
run;
```

```
length astdtf $1; astdt = .;
call imp_start(aestdtc, astdt, astdtf);
```

One call, two results

AESTDTC	ASTDT	ASTDTF
2023-05-14	2023-05-14	
2023-05	2023-05-01	D
2023	2023-01-01	M
(blank)	(missing)	

With SUBROUTINE and OUTARGS, one CALL imputes the date and creates its imputation flag at the same time. The rules here are examples only; follow your study SAP.

Before and after in a study program

BEFORE: inline in every ADaM program

```
_y = substr(aestdtc,1,4); _m = substr(aestdtc,6,2);
_d = substr(aestdtc,9,2);
if _d ne '' then astdt = input(aestdtc, yymmdd10.);
else if _m ne '' then do;
     if year(trtsdt)=input(_y,4.) and
        month(trtsdt)=input(_m,2.) then astdt = trtsdt;
     else astdt = mdy(input(_m,2.), 1, input(_y,4.));
     astdtf = 'D';
end;
else if _y ne '' then do;
     if year(trtsdt)=input(_y,4.) then astdt = trtsdt;
     else astdt = mdy(1, 1, input(_y,4.));
     astdtf = 'M';
end;
if astdtf ne '' and astdt > aendt > . then astdt = aendt;
drop _y _m _d;
/* ...and the same again for AENDT, CMSTDT, ... */
```

AFTER: one Call per variable

```
call imp_start(aestdtc, astdt, astdtf);
```

Multiple lines replaced by one CALL for each derived date. Invalid dates and date limits are handled once, inside the tested function.

Pitfalls we ran into

$

Character lengths

Declare return and local lengths explicitly. Undeclared lengths can silently truncate values such as --DTC.

.

Missing values

Decide and document what every argument does when missing, then put those cases in the test data.

?

Debugging

Errors surface at the call site. Test each function on its own in PROC FCMP before using it in study code.

lib

Library paths

One controlled CMPLIB location for production. Mixed or stale paths mean different code runs.

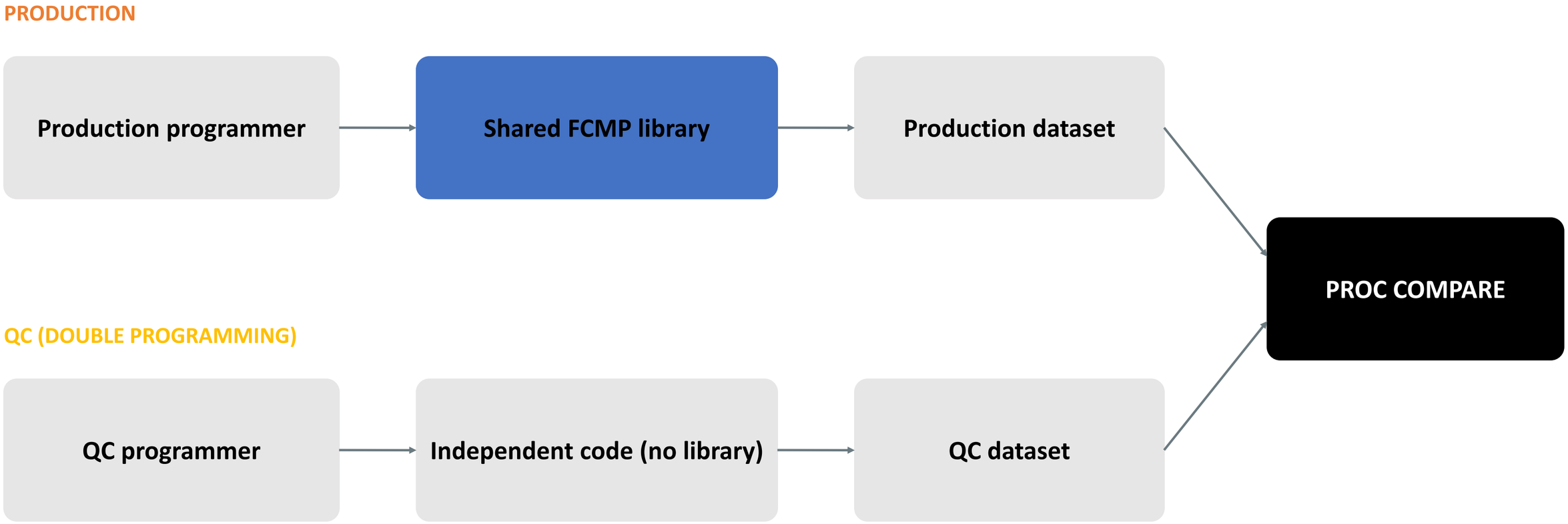
QC and Validation

Independent QC, function testing, change control

Three layers of checking

1	**Developer self-check** *Does it work as I intended?*	The function author runs basic tests while building the function.	**Function developer**
2	**Independent test and release** *Is the rule implemented correctly?*	A second programmer writes test cases from the rule list, runs them, and approves the release.	**Independent tester**
3	**Double programming in each study** *Is this study's dataset correct?*	QC programmer writes own code without the library. PROC COMPARE against production.	**Study QC programmer**

Three layers of checking (cont.)



Testing the function: every rule has a test

No.	Condition	Expected result	Test case ID
1	Complete date	Same date, no flag	S01 S02 S14
2	Day missing, same month as TRTSDT	TRTSDT, flag D	S03
3	Day missing, other month	First day of month, flag D	S04 S05
4	Invalid date (e.g. 2023-02-30)	Missing, warning in log	S12 S13
…	…	…	…

Write the rule list before the program, then write the program to meet every rule. Make test cases from the rule list, not from the code.

* The rules here are examples only. In real work, the rule list must cover all possible cases in your data.

Changing a function safely

1. **Request**
   Reason and affected rule recorded
2. **Update**
   Change source and rule list together
3. **Test**
   Add cases, re-run the full test program
4. **Release**
   Increase the version number
5. **Notify**
   Tell ongoing studies; assess re-runs

Make the version visible in every study run

```
function funclib_version() $ 20;
  return('1.2.0 (2026-09-01)');
endsub;

/* first step of every program */
data _null_; v = funclib_version(); put 'NOTE: funclib ' v; run;
```

Why it matters
The log of every production run shows which library version produced the dataset, so the impact of a function fix can be traced study by study.

Next steps

How to get started and functions to build next

Getting started in your team

1

Start small

Pick 2 or 3 simple, high-reuse rules such as studyday and iso_date.

2

Tests first

Write the rule list and test cases before the function itself.

3

Pilot one study

Use the library in production only; leave QC unchanged.

4

Track and grow

Log discrepancy causes, add version control, then expand.

Writing the function is easy. The hard part is agreeing on the rule and maintaining its tests.

Functions you can build next

Function	What it does	CDISC variables
`iso_dtc(date, time)`	SAS date and time to ISO 8601 (YYYY-MM-DDThh:mm:ss)	--DTC
`is_iso(dtc)`	Flags values that are not valid ISO 8601	--DTC
`ref_period(dt, rfstdt, rfendt)`	BEFORE / DURING / AFTER the reference period	--STRF, --ENRF
`nrind(val, lo, hi)`	LOW / NORMAL / HIGH against the reference range	--NRIND, ANRIND
`txtseg(text, n)`	n-th 200-character piece, split at a word boundary	SUPP--, CO
`chg(aval, base) / pchg(aval, base)`	Change and percent change; missing or zero BASE handled	CHG, PCHG
`trtem(astdt, trtsdt, trtedt, lag)`	Treatment-emergent flag, with a lag after treatment end	TRTEMFL

Key Takeaways

Standardize the rule, keep QC independent

1. Create one FCMP function for each CDISC rule
2. Test every function against the rule requirements
3. Keep QC independent from the shared functions

Thank You

Sanggi Lee · sanggi.lee@cnrres.com

2026 Korea CDISC Day · 29 October 2026



Questions?

