

Being a Perfect Programmer: Code Smarter, Review Sharper, and Embrace the Future

Akhil Vijayan

Catalyst Clinical Research

akhil.vijayan@catalystcr.com

Limna Salim

Catalyst Clinical Research

limna.salim@catalystcr.com



Agenda

INTRODUCTION

Macros in Review Process

Code Smarter (Best Practices)

Case Studies

Automation & AI

Questions?



INTRODUCTION

Evolving Role of Clinical Programmers:

- Technical skill alone is not enough; efficiency, adaptability, and forward-thinking are key.

Focus Areas:

- Leveraging SAS/STAT® macros for review efficiency.
- Adopting reusable coding structures for consistency across studies.
- Using AI tools to automate, review, and enhance coding practices.



Macros in Review Process

Purpose of Macros:

- Simplify programming tasks, improve quality, and support data checking.
- Reduce repetitive work and improve accuracy, especially during lead reviews.

%VALIDATION_CHECKS Macro:

- Automates critical review steps:
 - Confirms validation dataset exists for each production dataset.
 - Checks generation order: validation after development.
 - Verifies number of observations and variables match.
 - Confirms consistency of domain labels.
 - Ensures datasets are created after program update dates.

Code Smarter- Handling Numeric Variables

Extract 1

SUBJID	TEST	TESTDT	COMPDT	RESULT
XX-01	Dummy Test1	2025-07-24	2025-08-13	0.0619083639
XX-01	Dummy Test2	2025-07-30	2025-08-13	0.5776866905
XX-01	Dummy Test3	2025-08-05	2025-08-13	0.5273230507
XX-02	Dummy Test1	2025-07-30	2025-08-18	0.5866976243
XX-02	Dummy Test2	2025-08-11	2025-08-18	0.3272008478
XX-02	Dummy Test3	2025-08-23	2025-08-18	0.3184468152
XX-03	Dummy Test1	2025-08-05	2025-08-23	0.7036767279
XX-03	Dummy Test2	2025-08-23	2025-08-23	0.1855538158
XX-03	Dummy Test3	2025-09-10	2025-08-23	0.2803286748
XX-04	Dummy Test1	2025-08-11	2025-08-28	0.3112880845
XX-04	Dummy Test2	2025-09-04	2025-08-28	0.5824215373
XX-04	Dummy Test3	2025-09-28	2025-08-28	0.0470493199

```
data casel;  
  set input;  
  if compdt gt testdt then flag="Y";  
  else flag="N";  
  if flag="Y" then result_1=result+.5;  
run;
```

```
161 data casel;  
162 set input;  
163 if compdt gt testdt then flag="Y";  
164 else flag="N";  
165 if flag="Y" then result_1=result+.5;  
166 run;
```

NOTE: There were 12 observations read from the data set WORK.INPUT.
NOTE: The data set WORK.CASE1 has 12 observations and 7 variables.
NOTE: DATA statement used (Total process time):
real time 0.00 seconds
cpu time 0.01 seconds

SUBJID	TEST	TESTDT	COMPDT	RESULT	FLAG	RESULT_1
XX-01	Dummy Test1	2025-07-24	2025-08-13	0.0619083639	Y	0.5619083639
XX-01	Dummy Test2	2025-07-30	2025-08-13	.	Y	.
XX-01	Dummy Test3	.	2025-08-13	0.5273230507	Y	1.0273230507
XX-02	Dummy Test1	2025-07-30	2025-08-18	0.5866976243	Y	1.0866976243
XX-02	Dummy Test2	2025-08-11	2025-08-18	0.3272008478	Y	0.8272008478
XX-02	Dummy Test3	.	2025-08-18	.	Y	.
XX-03	Dummy Test1	2025-08-05	2025-08-23	0.7036767279	Y	1.2036767279
XX-03	Dummy Test2	.	2025-08-23	0.1855538158	Y	0.6855538158
XX-03	Dummy Test3	2025-09-10	2025-08-23	0.2803286748	N	.
XX-04	Dummy Test1	2025-08-11	2025-08-28	0.3112880845	Y	0.8112880845
XX-04	Dummy Test2	2025-09-04	2025-08-28	.	N	.
XX-04	Dummy Test3	2025-09-28	2025-08-28	0.0470493199	N	.

Extract 2

SUBJID	TEST	TESTDT	COMPDT	RESULT
XX-01	Dummy Test1	2025-07-24	2025-08-13	0.0619083639
XX-01	Dummy Test2	2025-07-30	2025-08-13	.
XX-01	Dummy Test3	.	2025-08-13	0.5273230507
XX-02	Dummy Test1	2025-07-30	2025-08-18	0.5866976243
XX-02	Dummy Test2	2025-08-11	2025-08-18	0.3272008478
XX-02	Dummy Test3	.	2025-08-18	.
XX-03	Dummy Test1	2025-08-05	2025-08-23	0.7036767279
XX-03	Dummy Test2	.	2025-08-23	0.1855538158
XX-03	Dummy Test3	2025-09-10	2025-08-23	0.2803286748
XX-04	Dummy Test1	2025-08-11	2025-08-28	0.3112880845
XX-04	Dummy Test2	2025-09-04	2025-08-28	.
XX-04	Dummy Test3	2025-09-28	2025-08-28	0.0470493199

```
172 data casel;  
173 set input;  
174 if compdt gt testdt then flag="Y";  
175 else flag="N";  
176 if flag="Y" then result_1=result+.5;  
177 run;
```

NOTE: Missing values were generated as a result of performing an operation on missing values.
Each place is given by: (Number of times) at (Line):(Column).
2 at 176:36
NOTE: There were 12 observations read from the data set WORK.INPUT.
NOTE: The data set WORK.CASE1 has 12 observations and 7 variables.
NOTE: DATA statement used (Total process time):
real time 0.01 seconds
cpu time 0.01 seconds

Code Smarter- Adding Checks in User-Defined Formats

Study 1

SUBJID	RESULT
XX-01	Result 1
XX-01	Result 2
XX-01	Result 3
XX-01	Result 4
XX-01	Result 5
XX-02	Result 1
XX-02	Result 2
XX-02	Result 3
XX-02	Result 4
XX-02	Result 5

```
*Formats used for the result;
proc format;
  value $resfmt
    "Result 1"="E1"
    "Result 2"="E1"
    "Result 3"="E2"
    "Result 4"="E2"
    "Result 5"="E3"
    other= "CHECK_VALUE";
run;

data case2;
  set result;
  result_cat=put(result,$resfmt.);
  if result_cat in ("E2" "E3") then anfl="Y";
  if result_cat="CHECK_VALUE" then put
    "WAR" "NING: Check the result value for subject=" subjid " and result=" result;
run;
```

```
256 data case2;
257   set result;
258   result_cat=put(result,$resfmt.);
259   if result_cat in ("E2" "E3") then anfl="Y";
260   if result_cat="CHECK_VALUE" then put
261     "WAR" "NING: Check the result value for subject=" subjid " and result=" result;
262 run;
```

```
NOTE: There were 10 observations read from the data set WORK.RESULT.
NOTE: The data set WORK.CASE2 has 10 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Study 2

SUBJID	RESULT
XX-01	Result 1
XX-01	Result 2
XX-01	Result 3
XX-01	Result 4
XX-01	Result 5
XX-01	Result 6
XX-02	Result 1
XX-02	Result 2
XX-02	Result 3
XX-02	Result 4
XX-02	Result 5
XX-02	Result 6

```
287 data case2;
288   set result;
289   result_cat=put(result,$resfmt.);
290   if result_cat in ("E2" "E3") then anfl="Y";
291   if result_cat="CHECK_VALUE" then put
292     "WAR" "NING: Check the result value for subject=" subjid " and result=" result;
293 run;
```

```
WARNING: Check the result value for subject=XX-01 and result=Result 6
WARNING: Check the result value for subject=XX-02 and result=Result 6
NOTE: There were 12 observations read from the data set WORK.RESULT.
NOTE: The data set WORK.CASE2 has 12 observations and 4 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Code Smarter- Avoiding Truncation

Using delimiters (||) can cause truncation risks when code is reused or rerun on different extracts.

Two datasets give the same output, but the first uses ||, while the second uses cat().

cat() is safer, it issues a warning in the log if the result length is insufficient, helping prevent unnoticed truncation.

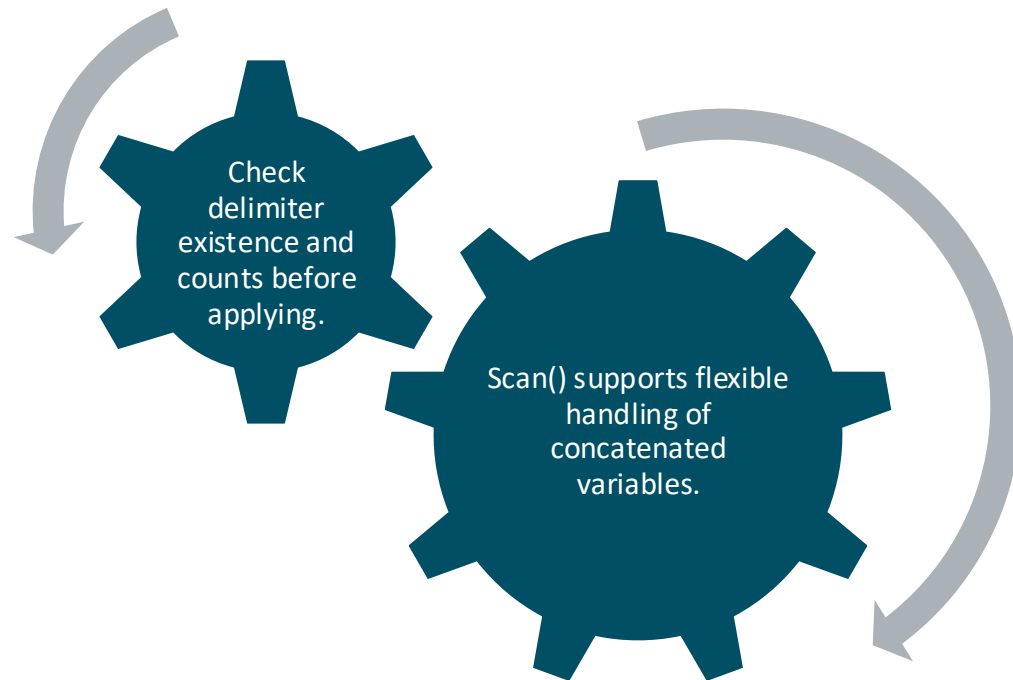
```
125 data not_rec;
126   length conc $10;
127   set sashelp.cars (obs=2);
128   conc=strip(model)||"-"||strip(type);
129 run;

NOTE: There were 2 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.NOT_REC has 2 observations and 16 variables.
NOTE: DATA statement used (Total process time):
      real time           0.02 seconds
      cpu time            0.01 seconds

130
131 data recmnd;
132   length conc $10;
133   set sashelp.cars (obs=2);
134   conc=catx("-",of model,type);
135 run;

WARNING: In a call to the CATX function, the buffer allocated for the result was not long enough to
contain the concatenation of all the arguments. The correct result would contain 20
characters, but the actual result might either be truncated to 10 character(s) or be
completely blank, depending on the calling environment. The following note indicates the
left-most argument that caused truncation.
NOTE: Argument 2 to function CATX('-', RSX Type S '[12 of 40 characters shown],'Sedan ') at line
134 column 8 is invalid.
CONC= MAKE=Acura MODEL=RSX Type S 2dr TYPE=Sedan ORIGIN=Asia DRIVETRAIN=Front MSRP=$23,820
INVOICE=$21,761 ENGINE SIZE=2 CYLINDERS=4 HORSEPOWER=200 MPG_CITY=24 MPG_HIGHWAY=31 WEIGHT=2778
WHEELBASE=101 LENGTH=172 _ERROR_=1 _N_=2
NOTE: There were 2 observations read from the data set SASHELP.CARS.
NOTE: The data set WORK.RECMND has 2 observations and 16 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Code Smarter- Using The SCAN() Function Effectively



```
613 %let rescatdel=,;
614 data example1;
615   drop i;
616   set case4;
617   *Array for checking the existence of a delimiter before concatenation ;
618   array resvar {*} rescat: ;
619   do i=1 to dim(resvar);
620     if find(resvar(i), "&rescatdel") then put "WAR" "NING: Delimiter found in " SUBJID= "
620! Variable=RESCAT" i " Value=" resvar[i];;
621   end;
622   catval=catx("&rescatdel", of rescat:);
623 run;
```

WARNING: Delimiter found in SUBJ ID=SUB01 Variable=RESCAT2 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB01 Variable=RESCAT3 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB02 Variable=RESCAT1 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB02 Variable=RESCAT3 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB02 Variable=RESCAT4 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB02 Variable=RESCAT5 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB04 Variable=RESCAT1 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB05 Variable=RESCAT4 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB05 Variable=RESCAT5 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB06 Variable=RESCAT4 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB07 Variable=RESCAT1 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB07 Variable=RESCAT3 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB07 Variable=RESCAT4 Value=Unknown,
WARNING: Delimiter found in SUBJ ID=SUB08 Variable=RESCAT4 Value=Unknown,
NOTE: There were 8 observations read from the data set WORK.CASE4.
NOTE: The data set WORK.EXAMPLE1 has 8 observations and 7 variables.
NOTE: DATA statement used (Total process time):
real time 0.03 seconds
cpu time 0.03 seconds

Code Smarter- Data Handling-IDVAR In PROC TRANSPOSE

Transpose is widely used (~90%) in SDTM/ADaM programming.

A simple ID statement helps detect and prevent dropout of variables like COL2, COL3.

```
351 proc transpose data=case5 out=case5_t1;
352   var param1 param2 param3;
353   by subjid visit;
354 run;

NOTE: There were 7 observations read from the data set WORK.CASE5.
NOTE: The data set WORK.CASE5_T1 has 18 observations and 5 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
      real time    0.04 seconds
      cpu time     0.01 seconds

355
356 data case5a;
357   set case5;
358   idvar="result";
359 run;

NOTE: There were 7 observations read from the data set WORK.CASE5.
NOTE: The data set WORK.CASE5A has 7 observations and 6 variables.
NOTE: DATA statement used (Total process time):
      real time    0.03 seconds
      cpu time     0.01 seconds

360
361 proc transpose data=case5a out=case5_t2;
362   var param1 param2 param3;
363   by subjid visit;
364   id idvar;
365 run;

ERROR: The ID value "RESULT" occurs twice in the same BY group.
NOTE: The above message was for the following BY group:
      SUBJID=XX-01 VISIT=Visit 3
WARNING: 1 BY groups omitted due to earlier errors.
NOTE: There were 7 observations read from the data set WORK.CASE5A.
NOTE: The data set WORK.CASE5_T2 has 15 observations and 4 variables.
NOTE: PROCEDURE TRANSPOSE used (Total process time):
```

Extract 1

SUBJID	VISIT	PARAM1	PARAM2	PARAM3
XX-01	Visit 1	36.292445351	74.5194713	83.105867302
XX-01	Visit 2	27.627717297	18.382375417	72.88826889
XX-01	Visit 3	7.7892529814	73.431820596	70.725353794
XX-01	Visit 4	76.407951245	48.775378731	18.158753458
XX-01	Visit 5	15.276748228	39.360054647	45.952215905
XX-01	Visit 6	85.948743618	39.253602754	18.129381406

Extract 2

SUBJID	VISIT	PARAM1	PARAM2	PARAM3
XX-01	Visit 1	36.292445351	74.5194713	83.105867302
XX-01	Visit 2	27.627717297	18.382375417	72.88826889
XX-01	Visit 3	7.7892529814	73.431820596	70.725353794
XX-01	Visit 3	.	.	76.407951245
XX-01	Visit 4	48.775378731	18.158753458	15.276748228
XX-01	Visit 5	39.360054647	45.952215905	85.948743618
XX-01	Visit 6	39.253602754	18.129381406	16.222651078



Automation & AI

Automation & Efficiency: AI tools help automate repetitive tasks, such as generating trackers or scripts.

Error Detection & Robust Testing: Synthetic datasets created via AI allow testing of edge cases and derivations without using real study data.

Code Review & Optimization: AI can assist in reviewing code, suggesting improvements, and ensuring consistency across studies.

Learning & Knowledge Support: Programmers can use AI to quickly understand new methods, formats, or programming standards.

Practical Example: Using ChatGPT to create reusable SAS dataset templates or trackers for SDTM/ADaM development.



Thank you



Questions?