

How Macro Design and Program Structure Impacts GPP (Good Programming Practice) in TLF Coding

Galyna Repetatska, Kyiv, Ukraine
PhUSE 2016, Barcelona



CHILTERN
Designed Around You®

- Number of operations for SAS processor: between multiplicative and additive
- Tools and factors helpful to minimize programming and data dependency
- Keys to universal open-code programming
- TLF-conventional variables #1: groups, categories and analysis data
- Alignment with GPP
- TLF-conventional variables #2: control decimal alignment
- One-Proc calculation with BY and OUTPUT for Adverse Events by Severity
- Different types of analysis for Demographics and Baseline Characteristics
- Useful tricks of PROC SQL to generalize study-specific programming
- From open code to macro design

Number of operations for SAS processor: between multiplicative and additive

- Calculation of each block individually gives the maximum of program steps:

$$N_{operations} \sim N_a * N_{var} * N_{grp} * N_{par} * N_{tpt};$$

- BDS structure helps to reduce program (but not for pooled categories yet):

$$N_{operations} \sim N_a * N_{var} * N_{grp};$$

- Reasonable minimum of operations (Data/Proc steps used to provide result) will be number of statements in specification or shell used to *describe* task:

$$N_{operations} \leq N_a + N_{var} + N_{grp} + N_{par} + N_{tpt};$$

The only non-vanishing component is type of analysis:

$$N_{operations} \sim N_a$$

Table 14.3.x.x
Summary of Change from Baseline in Vital Sign Results
Safety Population

Parameter: xxx (units) **ADVS.param**

TRT (N=xx) PBO (N=xx)

Number of parameters: N_{par} Treatment groups: $N_{grp}=2$ Analysis Variables: $N_{var}=3$

Timepoint	Baseline ADVS.base	At Timepoint ADVS.aval	Change ADVS.chg	Baseline	At Timepoint	Change
ADVS.avisit, atpt						
Baseline						
n	xxx	xxx	xxx	xxx	xxx	xxx
Mean	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x
SD	xxx.xx	xxx.xx	xxx.xx	xxx.xx	xxx.xx	xxx.xx
Median	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x
Min, Max	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx
Q1, Q3	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x
Post-Treatment Assessment 1						
n	xxx	xxx	xxx	xxx	xxx	xxx
Mean	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x
SD	xxx.xx	xxx.xx	xxx.xx	xxx.xx	xxx.xx	xxx.xx
Median	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x	xxx.x
Min, Max	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx	xxx.x, xxx
Q1, Q3	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x	Xxx.x, xxx.x

Note: Only subjects with both baseline and timepoint values are summarized at a given timepoint.

Time points: N_{tpt}

Types of analysis: $N_a=1$

Tools and Factors helpful to minimize Programming and Data Dependency

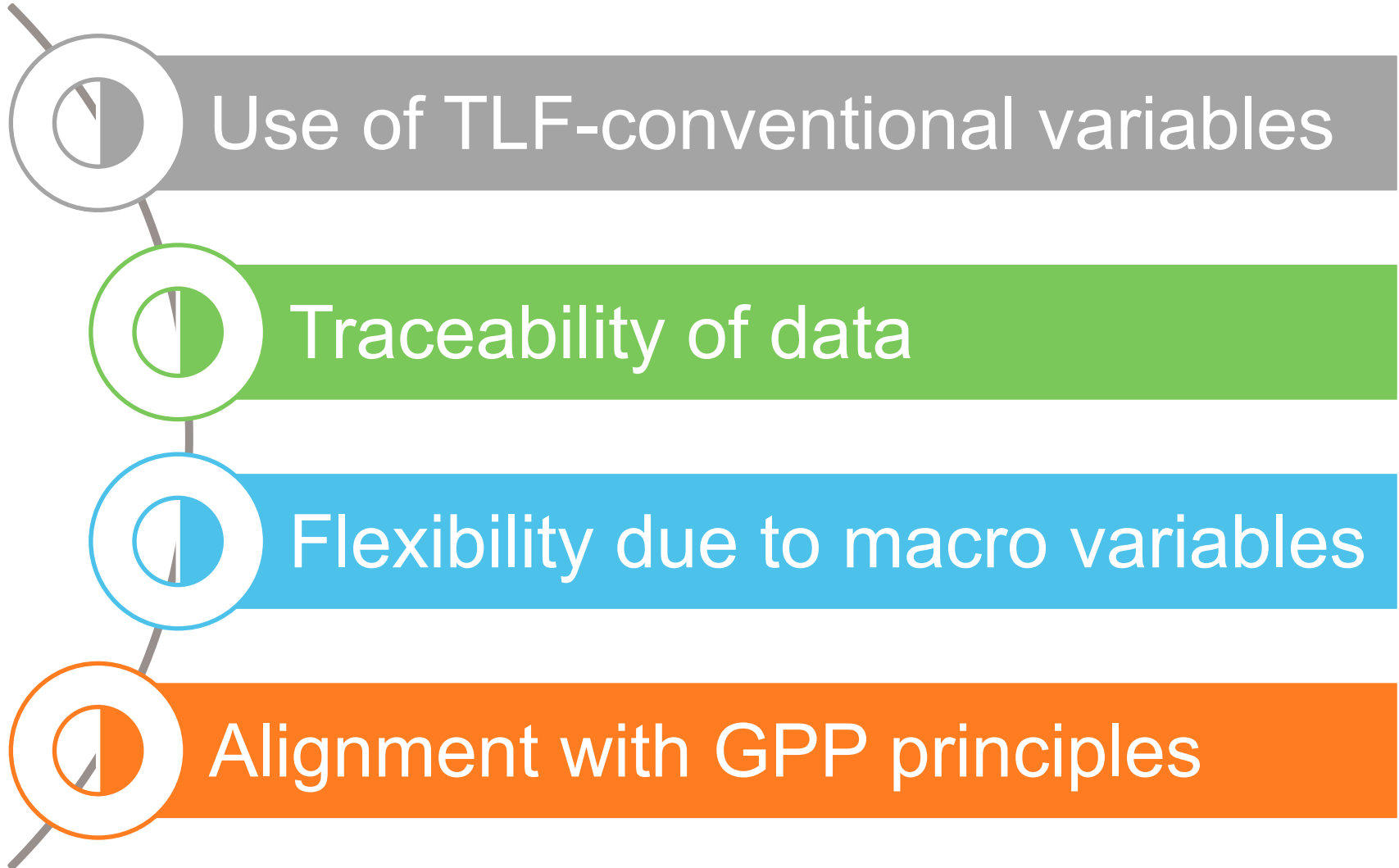


Subsequently, reducing the number of operations directly impacts:

- LOG file and debug;
- How much dissociated WORK datasets will be kept, reviewed and joined together;
- Adaptability to another task.

Basic elements helpful for TLF programming:

- **BY** statement allows to *repeat analysis by categories*, settled by *list of variables*;
- SDTM structure for Interventions and ADAM BDS standard variables perfectly match use of BY statement and provides traceability of result;
- We can reinforce BY with **OUTPUT** to create *categories* for TLF analysis;
- Reference to variables, *list of variables in BY statement* and other common settings (such as formatting) via **macro** variables to enable flexibility;
- Organize code following GPP principles in order to optimize work and result, thereof:
 - ✓ Do not derive anything in more than one place;
 - ✓ Perform only one task per module or macro.



TLF-conventional variables #1: groups, categories and analysis data



Variables in Dataset

- Subject-level **groups**:
 - TRT(N), GRP(N) – treatment/subject groups
 - *Example*: GRP = AGEGR1; GRPN = AGEGR1N;
- Data-level **categories**:
 - CAT1(N), CAT2(N) – grouping categories
 - Subject to be counted once per category
 - "Gender", "BMI (kg/m2)", "BMI group", AVISIT(N), PARAM(N), AEBODSYS
- Variables for **analysis** and **output**:
 - COL1(N), COL2(N) – columns to display
 - *Example 1*: "n", "Mean (SD)", "Any AE"
 - *Example 2*: RACE, AVALCAT1(N), CRITxx
 - AVALUE(N) – basic variables for analysis
 - PVALUE(N), LOGVALUE,...

Macro Variables

- &BYTRT, &BYGRP
- BYTRT = TRTAN TRTA;
- &BYCAT, &BYVIS, &BYPARM
- BYVIS= AVISITN AVISIT;
- BYPARM= PARCAT1 PARAMN
PARAMCD PARAM;
- BYCAT=PARCAT1N cat1;
- &BYMOCK
- BYMOCK = PARAMN PARAM
CAT1N CAT1 COL1N COL1;
- &BYVAL
- BYVAL= ASEVN ASEV;
- Names to be the same or similar

Not Recommended:

```
Data adsl;  
  set adsl;  
  output;  
  TRT01AN=0;  
  TRT01A = "Total";  
  output;  
Run;
```

- Treatment variable explicitly shown (+)
- Modification to other variable not flexible: many changes through code (-)
- WORK.ADSL not subject-level yet (-)
- Assigned “Total” for TRT01A(N) variable out of controlled terminology (-)

```
ANRIND = "Overall";  
AEBODSYS = propcase(AEBODSYS, ".");  
AEDECOD = " " || strip(aedecod);
```

Recommended:

```
Data subj_trt;  
  length TRTN 8 TRT $40;  
  set adsl;  
  trtn = trt01an; trt = trt01a;  
  output;  
  if not missing(trtn) then do;  
    trtn = 0; trt = "Total";  
    call missing(trt01an, trt01a);  
  output;  
end;
```

```
Run;  
%let bytrt= trtn trt;
```

- New TLF-conventional variable created;
- TRT01A(N) can be easily replaced; alternatively, global variable can be used;

```
coll = "Overall";  
cat1 = propcase(AEBODSYS, ".");  
cat2 = " " || strip(AEDECOD);  
%let bycat=AEBODSYS cat1 AEDECOD cat2;
```

TLF-conventional variables #2: control decimal alignment

Decimal Formats

- &Dec0 - &DecN – global variables to maintain *consistent* decimal alignment

```
%let dec0=3.;
%let dec1=5.1;
%let dec2=6.2;
%let dec3=7.3;
%let dec4=&dec3
%let dec5=&dec3
```

NDEC	COL1N	COL1	REZ
1	1 n		261
1	2 Mean		76.1
1	3 SD		19.98

```
length col1n 8 col1 $200 rez $20;
col1n = 1; col1 = "n";
rez = put(n,&dec0.);
output;
col1n = 2; col1 = "Mean";
rez = put(Mean,&dec1.);
output;
col1n = 3; col1 = "SD";
rez = put(SD,&dec2.);
output;
```

Macro Variables

- NDEC/&NDEC[=0,1,2,3...] – number of decimals for MIN, MAX univariates
 - Refer to *variable*, not eventual instances
 - Local formatting for macro calls

Utilize local dataset to track macro variables

```
%local decv decm decs;
%let byvar = avalcat1n avalcat1;
Data _localvars_;
  DecV=symget("dec"||put(&ndec.,1.));
  DecM=symget("dec"||put(&ndec.+1,1.));
  DecS=symget("dec"||put(&ndec.+2,1.));
  _byvar_frq=tranwrd("&byvar",' ','*');
  array lvars _ALL_;
  do over lvars;
    call symputx(vname(lvars),lvars);
  end;
```

Run;

DecV	DecM	DecS	_byvar_frq
5.1	6.2	7.3	avalcat1n*avalcat1

One-Proc Calculation with BY and OUTPUT: Adverse Events by Severity

- Each event representative have to be analyzed at 3 levels of categorization
- At each level one record per subject has to be selected
 - LVL (level of categorization) – supplementary variable for data-driven ordering based on frequency
 - CAT1 can be created after processing, but earlier initialization of non-missing variable is in place
 - ADAE severity variables can be replaced to relationship to study drug, etc.

OUTPUT

```

Data aecat;
length lvl 8 cat1 $200;
label cat1="SOC| Preferred Term";
set adae;
lvl=2;
cat1=" "||strip(aedecod);
output;
call missing(aedecod);
lvl=1;
cat1=propcase(aebodsys, '.');
output;
lvl=0;
cat1="Subjects with at least| one TEAE";
call missing(aedecod, aebodsys);
output;

run;
%let bycat = AEBODSYS AEDECOD lvl cat1;
%let byvar = ASEVN ASEV;
    
```

TABLE: Work.Aecat

LVL	CAT1	Unique Subject Identifier	Dictionary-Derived Term	Body System or Organ Class	Treatment Emergent Analysis Flag	Analysis Severity/Intensity	Analysis Severity/Inte (N)
2	Thermal burn	S049	Thermal burn	INJURY, POISONING AND PROCEDURAL COMPLICATIONS		Moderate	2
1	Injury, poisoning and procedural complications	S049		INJURY, POISONING AND PROCEDURAL COMPLICATIONS		Moderate	2
0	Subjects with at least one TEAE	S049				Moderate	2

ANY

- dataset
- variables
- # of levels

One-Proc Calculation with BY and OUTPUT: Adverse Events by Severity

BY

```
%let bycat= AEBODSYS AEDECOD lvl cat1; %let byvar= ASEVN ASEV; %let bytrt= trtN trt;
```

All set of treatment counts in one step

```
Proc Means data=subj&num;
  by &bytrt;
  var flag;
  output out=totals&num n=Nsub;
run; *Add column labels, macro vars...;
```

Traceability: counts and labels for treatment groups accessible from dataset

	TRTN	TRT	NSUB	TRTLABEL
1	1	Treatment 1	261	Treatment 1 N=261
2	2	Treatment 2	253	Treatment 2 N=253
3	3	Total	514	Total N=514

Merge subject groups with AE categories

```
Proc Sql noprint;
  create table data&num as select *
  from subj&num s, indata&num d
  where s.usubjid = d.usubjid;
quit;
```

Get AE with maximum severity at 3 levels

```
Data datasubj&num;
  set data&num;
  by &bytrt &bycat usubjid &byvar;
  if last.usubjid;
Run;
```

One-Proc Calculation

```
Proc Freq data=datasubj&num;
  by &bytrt &bycat &byvar;
  tables flag / out=count_subj&num
              (drop=percent);
Run;
```

Format table cells:

- Use TOTALSxx.Nsub for %;
- Format cells prior to any transpose;
- Setup columns other than default [treatments]

```
%let dec0 = 3.;
Data res_all&num;
  merge count_subj&num totals&num;
  by &bytrt;
  length rez $20 column $20 collbl $40;
  percent = 100*count/Nsub;
  length _perc $8;
  _perc = cats("(" ,put(percent,5.1),"%)" );
  rez = put(count,&dec0.)||"|"||right(_perc);
  *~Create columns to transpose~*;
  column= 10*trtn + asevn;
  collbl = ASEV;
Run;
```

One-Proc Calculation with BY and OUTPUT: Adverse Events by Severity

TABLE: Work.Res_all01

TRTN	TRT	LVL	CAT1	Analysis Severity (N)	Analysis Severity/	TRTLABEL	REZ	COLUMN	COLLBL
3	Total	0	Subjects with at least one TEAE	1	Mild	TotalN=514	88 (17.1%)	31	Mild
3	Total	1	Injury, poisoning and procedural complications	1	Mild	TotalN=514	25 (4.9%)	31	Mild
3	Total	2	Thermal burn	1	Mild	TotalN=514	1 (0.2%)	31	Mild

```
Proc Transpose data=res_all&num out=result&num prefix=trt;
```

```
by &bycat &byvar;
```

```
var rez;
```

```
id trtn;
```

```
idlabel trt;
```

```
run;
```

LVL	CAT1	Analysis Severity (N)	Analysis Severity/	Treatment 1N=261	Treatment 2N=253	TotalN=514
0	Subjects with at least one TEAE	1	Mild	40 (15.3%)	48 (19.0%)	88 (17.1%)
0	Subjects with at least one TEAE	2	Moderate	22 (8.4%)	22 (8.7%)	44 (8.6%)
0	Subjects with at least one TEAE	3	Severe	3 (1.1%)	4 (1.6%)	7 (1.4%)
1	Injury, poisoning and procedural complications	1	Mild	10 (3.8%)	15 (5.9%)	25 (4.9%)
1	Injury, poisoning and procedural complications	2	Moderate	10 (3.8%)	6 (2.4%)	16 (3.1%)
1	Injury, poisoning and procedural complications	3	Severe	1 (0.4%)	1 (0.4%)	2 (0.4%)
2	Thermal burn	1	Mild		1 (0.4%)	1 (0.2%)

Standard layout

& Customized (spanning)

```
Proc Transpose data=res_all&num out=result&num prefix=trt;
```

```
by &bycat;
```

```
var rez;
```

```
id column;
```

```
idlabel collbl;
```

```
Run;
```

LVL	CAT1	Mild	Moderate	Severe	Mild	Moderate	Severe	Mild	Moderate	Severe
0	Subjects with at least one TEAE	40 (15.3%)	22 (8.4%)	3 (1.1%)	48 (19.0%)	22 (8.7%)	4 (1.6%)	88 (17.1%)	44 (8.6%)	7 (1.4%)
1	Injury, poisoning and procedural complications	10 (3.8%)	10 (3.8%)	1 (0.4%)	15 (5.9%)	6 (2.4%)	1 (0.4%)	25 (4.9%)	16 (3.1%)	2 (0.4%)
2	Thermal burn				1 (0.4%)			1 (0.2%)		

Different Types of Analysis for Demographic and Baseline Characteristic

```
Data data_qual;
  length group $4 catln 8 cat1 $200
         colln 8 coll $200 pcat $200;
  set adsl;
  group = "QUAL";
  catln=1;  cat1="Gender";
  colln=ifn(sex="M",1,1,.);
  coll =put(sex,$genderf.);
  pcat = sex;
  output;
  catln=3;  cat1=vlabel(race);
  colln= aracen;
  coll = arace;
  pcat= ifc(race='WHITE',race,'OTHER','');
  output;
```

Run;

group	catln	cat1	colln	coll	pcat	USUBJID
QUAL	1	Gender	1	Male	M	DEM005
QUAL	3	Race	1	White	WHITE	DEM005
QUAL	1	Gender	2	Female	F	DEM006
QUAL	3	Race	2	Black or African...	NON-WHITE	DEM006

```
Proc Freq data=data_qual;
  by trtn trt catln cat1;
  tables colln*coll/out=freqs;
run;
```

```
Data data_quan;
  length group $4 catln 8
         cat1 $200 avalue ndec 8;
  set adsl;
  group = "QUAN";
  catln = 2;
  cat1 = "Age";
  avalue = age;
  ndec = 0;
  output;
  catln = 4;
  cat1="Duration at Study(weeks)";
  avalue = DURSTUDY;
  ndec = 1;
  output;
```

Run;

group	catln	cat1	avalue	ndec	USUBJID
QUAN	2	Age	58	0	DEM005
QUAN	4	Duration at Study (weeks)	69.7	1	DEM005
QUAN	2	Age	55	0	DEM006
QUAN	4	Duration at Study (weeks)	18.2	1	DEM006

```
Proc Means data=data_quan;
  by trtn trt catln cat1 ndec;
  var avalue;
  output out=means &means_out;
run;
```

Useful Tricks of PROC SQL to Generalize Study-Specific Programming

With VARIABLE LISTS as BY-parameters, any data-driven shell can be done

**this work well if full set of &BYVAL values appears at least once in dataset*

```
%let byparm=PARAMCD PARAM; %let byvis= AVISITN AVISIT; %let byval=AVALC;

Proc Sort data=data&oid nodupkey out=byparm&oid(keep=&byparm); by &byparm; run;
Proc Sort data=data&oid nodupkey out=byvis&oid(keep=&byvis); by &byvis; run;
Proc Sort data=data&oid nodupkey out=byval&oid(keep=&byval); by &byval; run;

Proc Sql ;
  create table shell&oid as select * from byparm&oid, byvis&oid, byval&oid;
quit;
```

Lists of parameters, data-driven formats etc. can be created and *printed*:

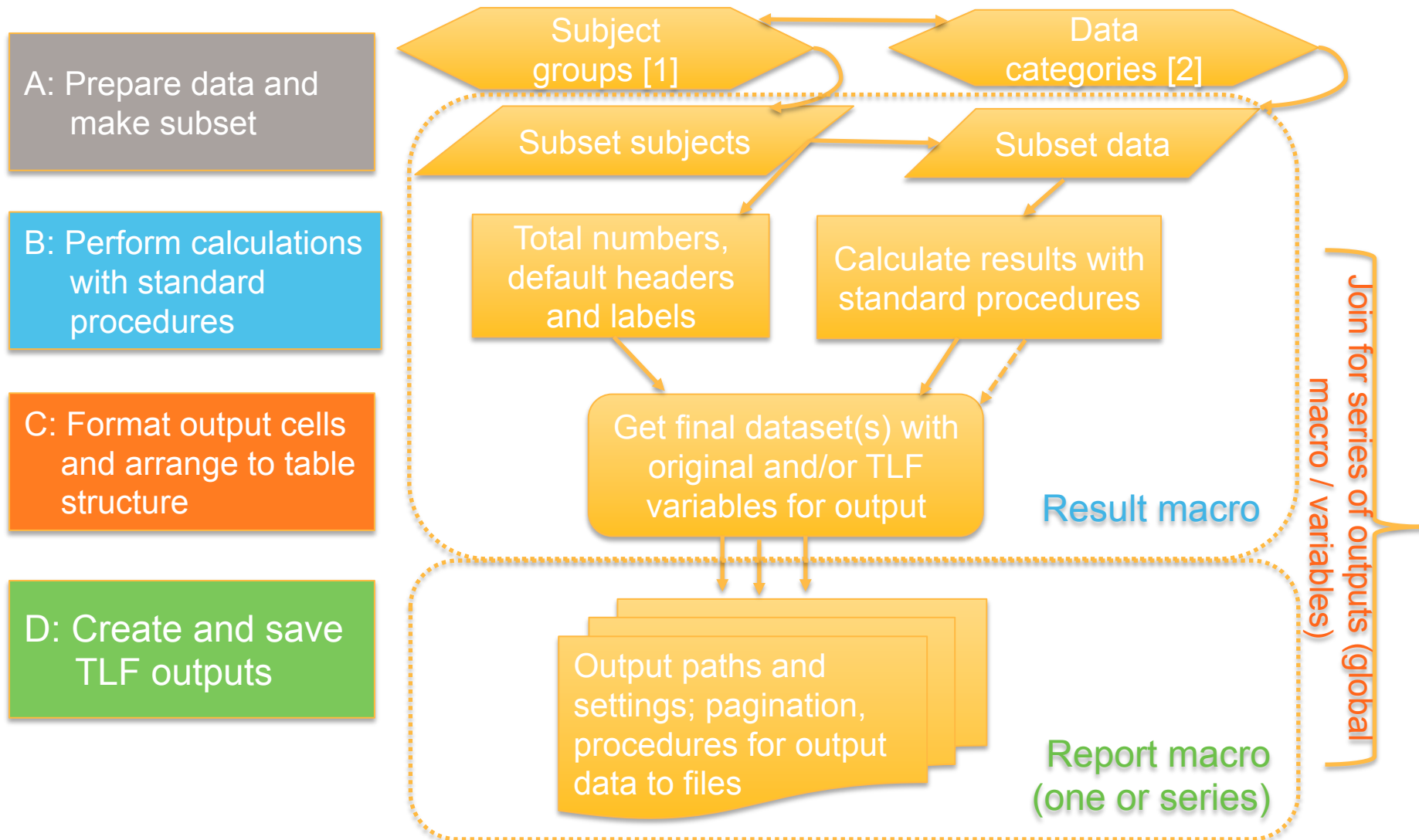
```
Proc Sql;
select distinct cats(avisitn,"=",avisit,"")
  into:_visfmt separated by ' ' from data&oid;
select distinct strip(paramcd) as ParamLst
  into:_paramlst separated by ' ' from data&oid;
quit;
```

```
Proc Format;
  value avisfmt &_visfmt;
run;
```

```
0='Baseline'
12='Week 12'
24='Week 24'
52='Week 52/Open-Label'
100='End of Study'
```

```
--ParamLst--
BMI
HEIGHT
PULSE
WEIGHT
```

From OPEN CODE to MACRO DESIGN



Appendix: Macro calls for Result and Report

```
*** Create Table for % of Responders***;  
%result_resp_yn(oid=01,  
  { insubj = adsl,  
    selsubj= %str(where fasfl='Y'),  
    bytrt  = trtsegan trtsega,  
    { indata = adefl,  
      seldata= %str(where anl01fl='Y'),  
      byval  = parcat1 avisitn avisit paramcd param,  
      avalue = avalc,  
      { percents = TOTAL);  
    }  
  }  
  { Result/Output ID  
    Subject-level  
    Data-level  
    Other settings  
}
```

* 4-column output by treatment sequence TRTSEQA *;
%**report_4trt**(oid=**01**,vispage=**2**);

< Macro call with the same parameters(or global settings), except for: oid= **02**, bytrt= trt01pn trt01p >

* 2-column output by planned treatments TRT01P *;
%**report_2trt**(oid=**02**,vispage=**3**);

- Number of data steps and procedure calls can be reduced to minimum: *one procedure for each type of analysis*
- GPP recommendations “do not derive anything in more than one place”, “perform only one task per module or macro” are reachable at SAS compiler level (not only due to repeated macro calls)
- Optimization of open-code enables us to develop powerful macro with high level of generalization

~~~~~THANK YOU!~~~~~

References

http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice

http://www.phusewiki.org/wiki/index.php?title=Good_Programming_Practice_Guidance

Acknowledges

The author would like to thank Roman Ganzha for his careful review and comments

Contact Information

Galyna Repetatska, PhD

Chiltern

51B Bohdana Khmelnytskogo str.

Kyiv / 01030, Ukraine

Email: Galyna.Repetatska@Chiltern.com

LinkedIn: <https://www.linkedin.com/in/halyna-repetatska>