

PHASE-III MACRO SYSTEM

A case of “fractal” system architecture using
programming languages and procedures from SAS
Institute Inc.

Wolf-Dieter Batz
PhUSE Conference Berlin, 2010

ACCOUNTANCY &
MA/CONSTRUCTI
CONTACT CENTR
ATIONS/EDUCATI
HNOLOGY/LEGAL
SAFETY/POLICY&
OURCES & MINING
NGINEERING/HU
LOGISTICS/FACILITIES MANAGEMENT/FINANCIAL
CIAL SERVICES/SOCIAL CARE/SALES & MARKETI
ING/ENERGY/OFFICE SUPPORT/RESPONSE MANA
HEALTHCARE/OIL & GAS/ARCHITECTURE/ASSESS
& DEVELOPMENT/PUBLIC SERVICES/ACCOUNTAN
NCY & FINANCE/EDUCATION/PHARMA/CONSTRU
NSTRUCTION & PROPERTY/RESOURCE MANAGEM
MENT/MANUFACTURING & OPERATIONS/RETAIL/I
INFORMATION TECHNOLOGY/SALES & MARKETING
RATEGY/BANKIN
MARKETING/ENE
INING/TELECOMS
HUMAN RESOURC
TRES/FINANCIAL
PHARMA/MANUF
HEALTHCARE/AR
PROCUREMENT/H

UCATION/PHARM
TY/CONTACT CEN
URING & OPERATI
ON TECHNOLOGY
NT/HEALTH & SAF
NKG/RESOURC
INSURANCE/ENG
RESOURCES/LOG
PUBLIC SERVICES
RESOURCES & MIN
ENGINEERING/H
CONTACT CENTRI
ES/SOCIAL CARE
NG/ENERGY/HEA
OFFICE SUPPORT
LEGAL/OIL & GAS

GENERAL (1)

PHASE-III MACRO SYSTEM

The Phase-III Macro System is a set of SAS macros. It is

- flexible,
- data independent and
- parameter controlled

GENERAL (2)

PHASE-III MACRO SYSTEM

Modules

- are small sized (three screen pages at maximum) for maintainability and
- avoid hard-coded references to any application related information like
 - data types,
 - labels and
 - Formats

GENERAL (3)

PHASE-III MACRO SYSTEM

At runtime, coding style makes broad use of

- automatic documentation and
- generation of
 - metadata and
 - lookup tables

PREFACE (1)

IMPLEMENTATION

The Phase-III Macro System is a highly interactive collection of macro modules providing transformation methods for study emergent datasets making use of all the information available in the description part of the dataset processed.

PREFACE (2)

IMPLEMENTATION

It provides subroutines that care for data types, formats, labels, headers, missing values, loops and more. Runtime generated information used to control processing is kept in standardized data structures using macro variable lists (mlists), SAS formats and datasets.

PREFACE (3)

IMPLEMENTATION

The user is provided with (an) output dataset(s) containing character columns with standard names and externally controlled attributes. Finally the Phase-III Macro System provides pre- and post processing functionality such as condense, struct and missline.

POSITIONING (1)

OBJECTIVE

The Phase-III Macro System is aimed at serving as a base for an extendable system that provides mechanisms for

- shaping input datasets,
- processing calculations and
- generating SAS datasets with ready made text content.

POSITIONING (2)

OBJECTIVE

The following requirements are met:

- Produce a wide variety of output with a minimum set of modules.
- Minimize maintenance efforts through self-documenting and limited program code.
- Be prepared to add new output structures without substantial delay.

THE QUEST

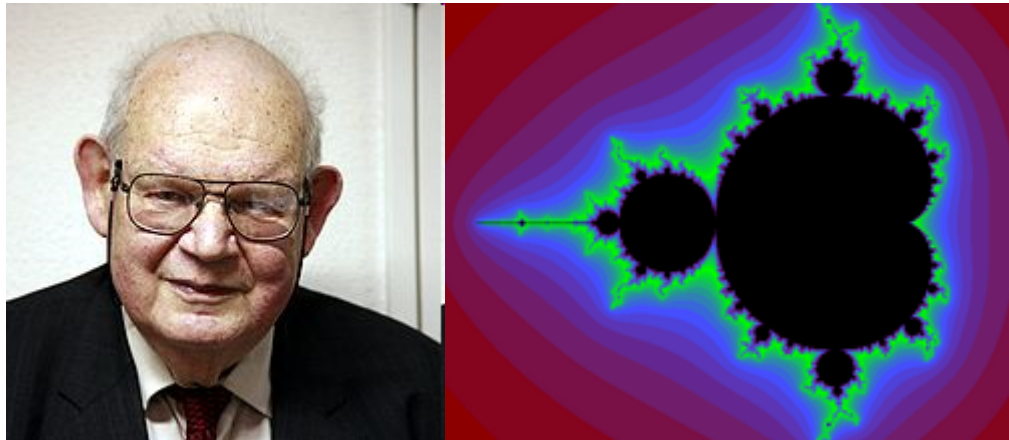
Total subjects treated	2821 (100)	2823 (100)
Start of study medication within 24 hrs after randomization		
Yes	2751 (97.5)	2775 (98.3)
No	66 (2.3)	47 (1.7)
Surgery delayed	59 (2.1)	41 (1.5)
Other	13 (0.5)	13 (0.5)
Missing	- (-)	- (-)
Missing	4 (0.1)	1 (0.0)
Study medication only from kit allocated		
Yes	2821 (100)	2823 (100)
Missing	- (-)	- (-)
Number of subjects who did not receive full loading dose	26 (0.9)	12 (0.4)
Reason:		
Adverse event	3 (0.1)	3 (0.1)
Technical failure / dosing error	17 (0.6)	8 (0.3)
Missing	6 (0.2)	1 (0.0)
At least 75% of loading dose administered		
Yes	2808 (99.5)	2819 (99.9)
No	8 (0.3)	4 (0.1)
Missing	5 (0.2)	- (-)
Number of subjects with maintenance dose interruptions longer than 1 hour	86 (3.0)	101 (3.6)
Number of maintenance dose interruptions		
1	79 (2.8)	98 (3.5)
2	4 (0.1)	4 (0.1)

APPROACH

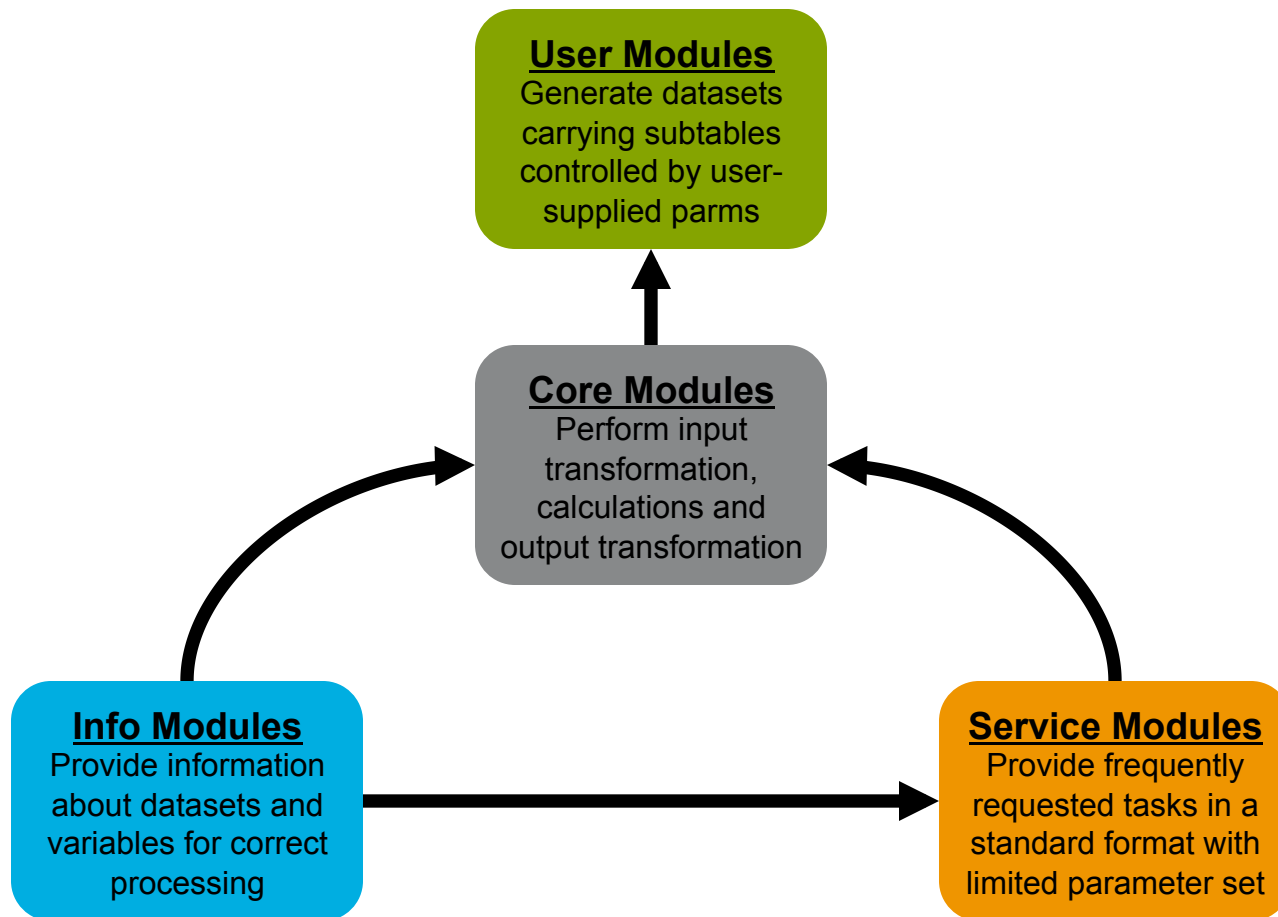
***“A problem well stated
is a problem half
solved”***

Charles Kettering

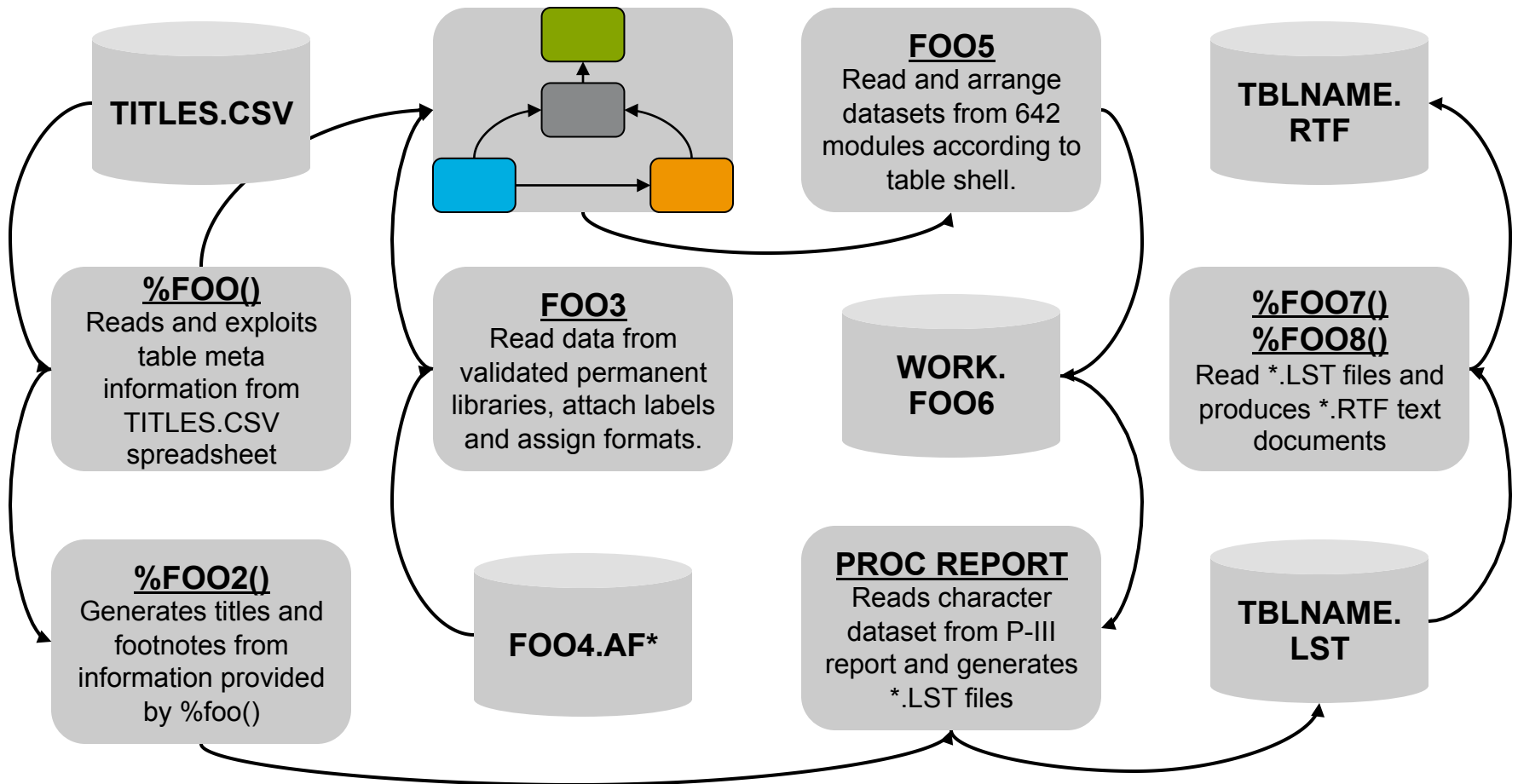
Benoit Mandelbrot



ARCHITECTURE



EMBEDDING



MODULES: INFO

%get_attr()

Return single attributes
like label, format, etc.

Reads dataset header and returns attributes as undeclared macro variables using the requested attributes names. Information becomes available when the particular variable is declared in the calling environment using a %global or %local statement.

%grp_desc()

Provide info about
a categorial variable

Investigates given categorial variable and provides results using undeclared macro variables: **&n_grp** - number of distinct values; **&v_grp** – structured list of distinct unformatted values; **&l_grp** – structured list of distinct formatted values.

%chk_list()

Provide info about
a list type macrovar

Reads supplied list of tokens and returns undeclared macro variables: **&n_lst** - number of list elements; **&v_lst** – structured list of supplied elements. Input list elements may be separated by blank and comma only.

MODULES: SERVICE (1)

%gen_intv()

Generate 2 formats
from list of intervals

Reads list of ranges and generates two formats; **&fmt_name.g** for correct sorting; **&fmt_name.f** for using the supplied ranges as a group label during tables generation.

%gen_type()

Generate list of `_type_`
for number of vars

Generates list of `_type_` values from supplied number of variables. Provided `_type_` values are calculated with respect to generation of hierarchical weights.

%gen_wgts()

Generate hierarchical
PCT-weights as Data

Generate matrix of weights to provide hierarchical percentages in table generation.

%gen_reco()

Generate character
format for recode

Read sequence of blank separated assignments to be used in Proc Format. A character format named according to the supplied `&fmt_name` is produced.

MODULES: SERVICE (2)

%exp_info()

Generate format from
two variables in dataset

Generates format from variable and primary key from dataset. Format generation is type sensitive.

%imp_info()

Generate new variable
In dataset using format

Generates new variable by mapping a primary key with a format and makes it available using an sql view.

%addgroup()

Ins. group information
using lookup dataset.

Transports information from dataset_A to dataset_B by combining the services from %exp_info and %imp_info.

%gen_mail()

Report execution status
to local e-mail recipient.

For batch execution immediately report which module executes in what report environment. Specific return codes supplied send appropriate messages.

MODULES: CORE (1)

%top_filt()

Generate standard data
structure for processing

Read from source dataset or view and perform filter and select operations

%row_filt()

Generate context dep.
std. data structure

Read from source dataset or view and perform filter and select operations with
respect to a given context variable's value.

MODULES: CORE (2)

%top_freq()

Calculate PCTs and
counts from std. data

Read output from *_filt module and perform calculations of frequencies and counts.

%row_freq()

Calculate PCTs/counts
fr. context dep. std. data

Read output from *_filt module and perform calculations of frequencies and counts.
allowing weighting with percentages from context variable.

%row_univ()

Calculate univ. stats fr.
context dep. std. data

Read output from *_filt module and perform calculations of univariate statistics
according to given list with respect to their sequence.

MODULES: CORE (3)

%top_outc()

Format categorial vals.
according to tab. shells

Read output from *_freq module and generate an output file with character formatted columns, totals, header and indentation.

%row_outc()

Format cat. vals. from
context dep. std. data

Read output from *_freq module and generate an output file with character formatted columns, indentation and weighted percentages. Add counts and percentages from context variable as headline.

%row_outv()

Format cont. vals. from
context dep. std. data

Read output from *_univ module and generate an output file with character formatted columns, stat labels and header.

MODULES: USER (1)

%row_bool()

Deliver PCT/count line
dataset using 1 decode.

Select single (,true') value for categorial processing from *_freq and *_catv modules.

%blk_bool()

Deliver PCT/count subt.
dataset from N bool.vars

Loop with %row_bool over array of categorial values using same name prefix and ,true' value and output results as one block.

%tab_bool()

Deliver PCT/count table
dataset from N bool.blks

Loop with %blk_bool over groups of variables using diferent name prefixes.

MODULES: USER (2)

%blk_catv()

Deliver PCT/count subt.
dataset from 1 cat. var.

Use categorial processing from *_freq and *_catv modules with restricted parameter set.

%tab_catv()

Deliver PCT/count table
dataset fr. N cat. vars

Loop with %blk_catv over array of names using same processing parameters.

MODULES: USER (3)

%blk_conv()

Deliver univ. subtable
dataset from 1 cont. var.

Use continuous variable processing from *_univ and *_conv modules with restricted parameter set.

%tab_conv()

Deliver univariate. table
dataset fr. N cont. vars

Loop with %blk_conv over array of names using same set of processing parameters.

MODULES: USER (4)

%two_catv()

Deliver PCT/count table
from 2 nested cat. vars

Perform nested processing of two categorial variables looping the context variable from the row_* modules over the categories of the ,outer' categories.

%two_boca()

Deliver PCT/count table
from bool./cat. vars

Perform nested processing with only one (,true') value select from the outer category.

%two_bobo()

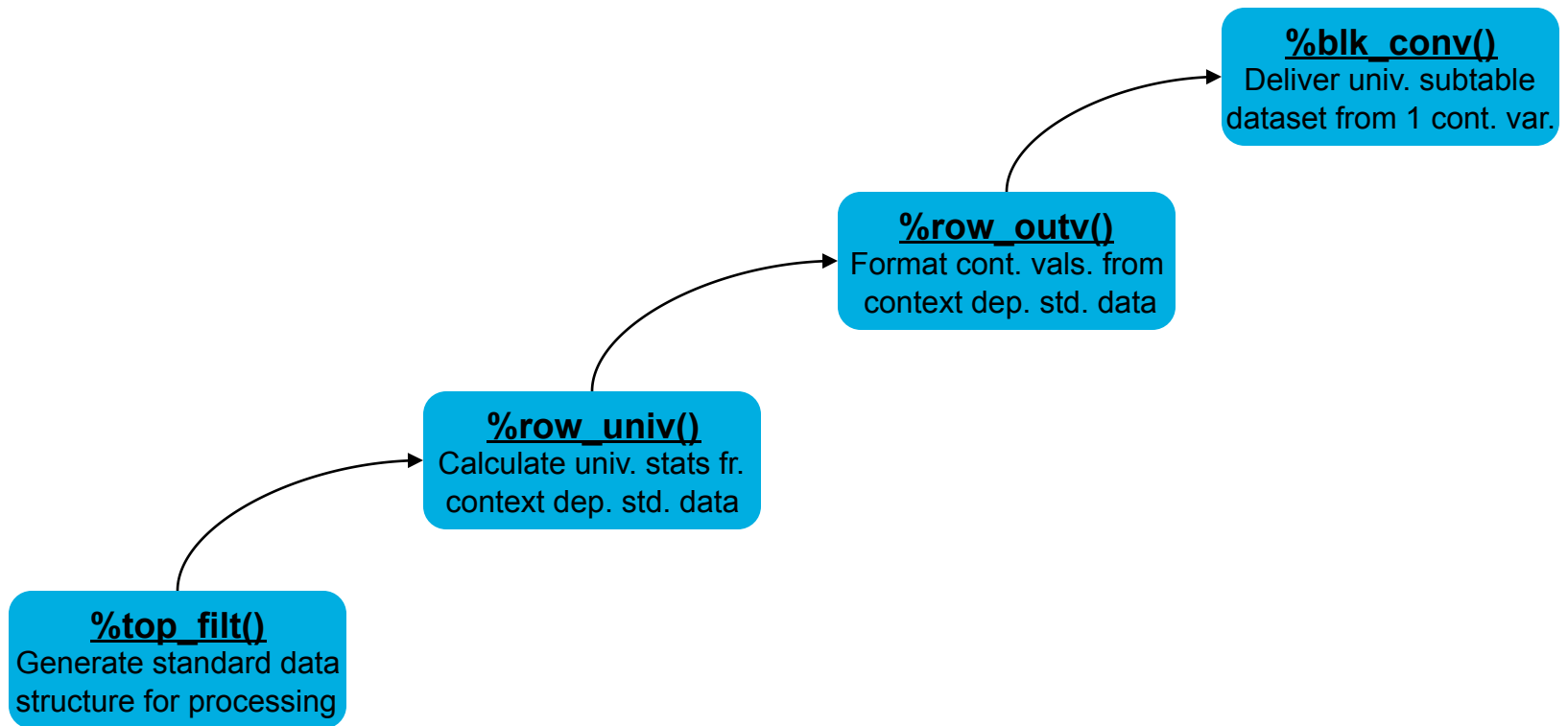
Deliver PCT/count table
from 2 nested bool vars

Perform nested processing with boolean (,true value') selection from the outer category and an array of boolean selections inside like in %blk_bool. True values may be chosen for each inside variable separately.

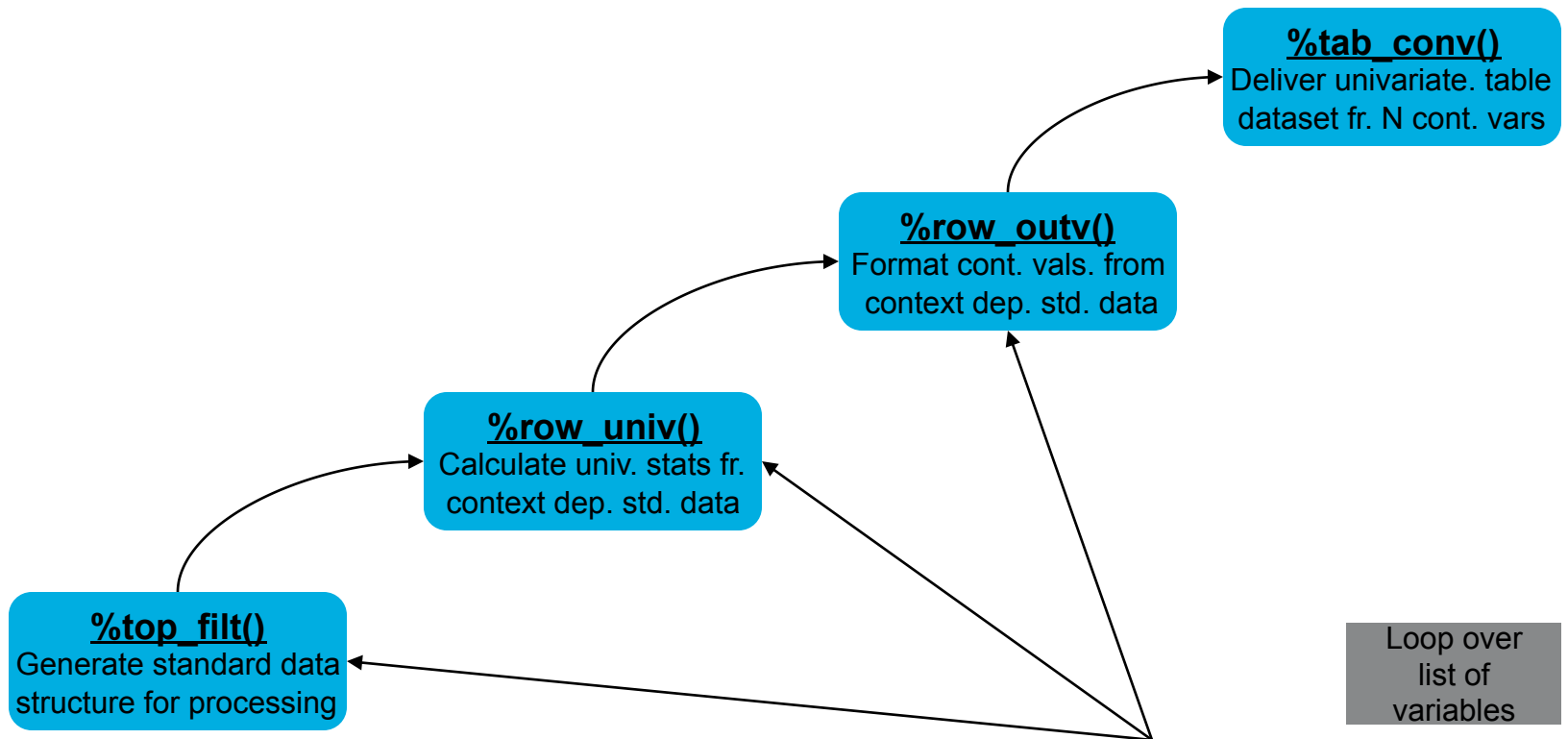
INSIDE USER MODULES CONTENTS

- Following is a sequence of structure pictograms giving an impression how core modules may be assembled to form user modules.
- To achieve this, several levels of processing may be combined through interchanging environment information in macrovariables and interim datasets.
- Currently the most complex modules provide two-level processing. Two specific service modules %gen_type() and %gen_wgts() extend these capabilities to n levels.

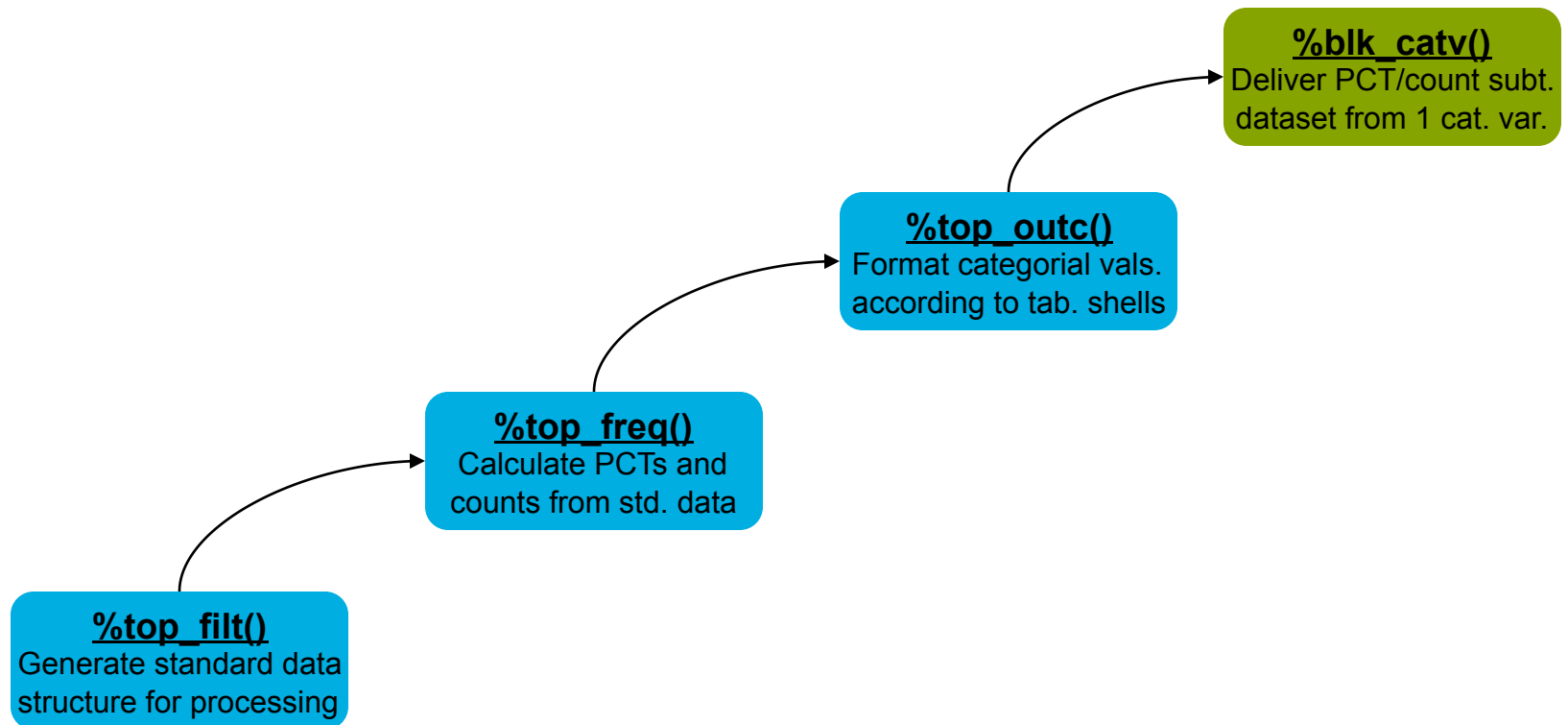
%blk_conv() – dataflow



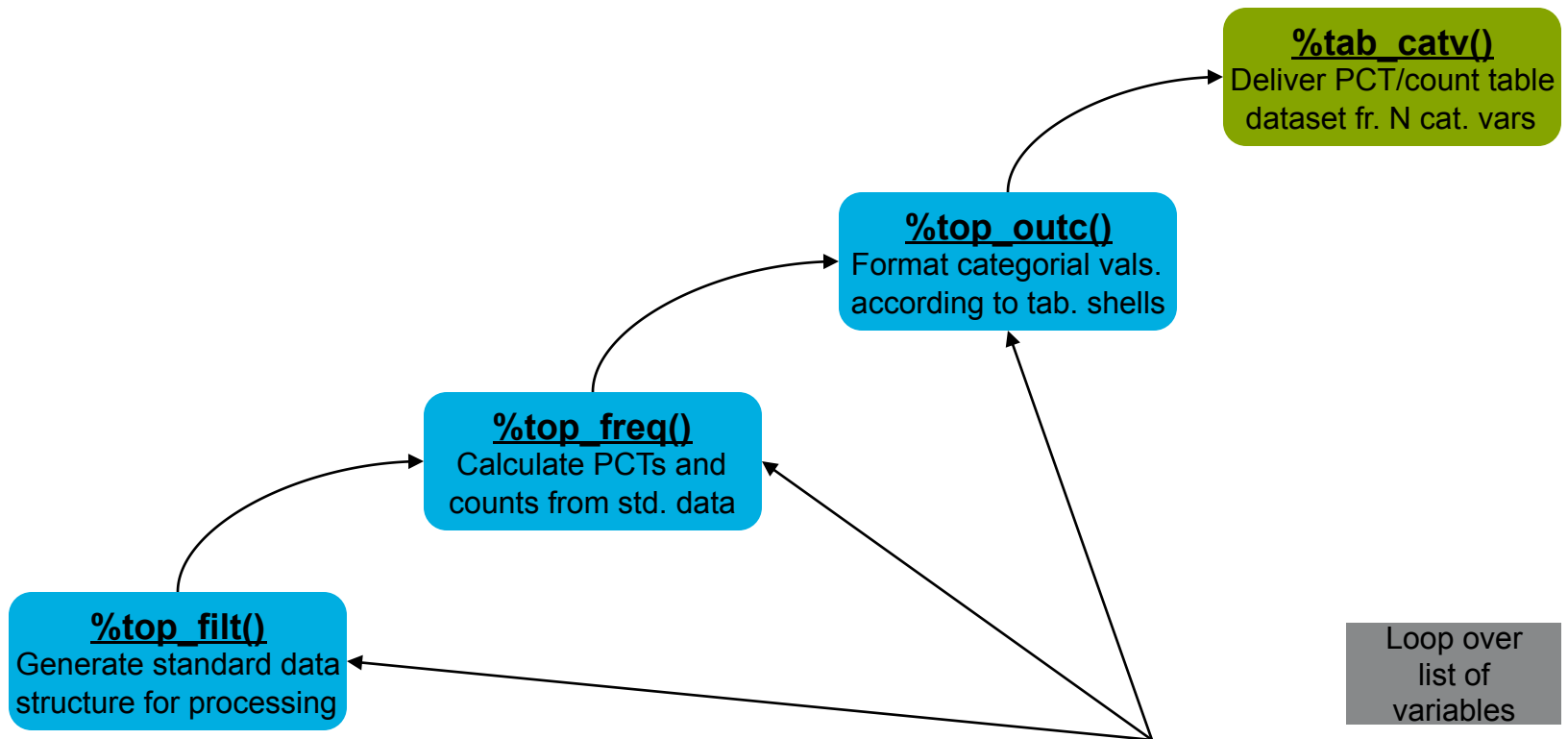
%tab_conv() – dataflow



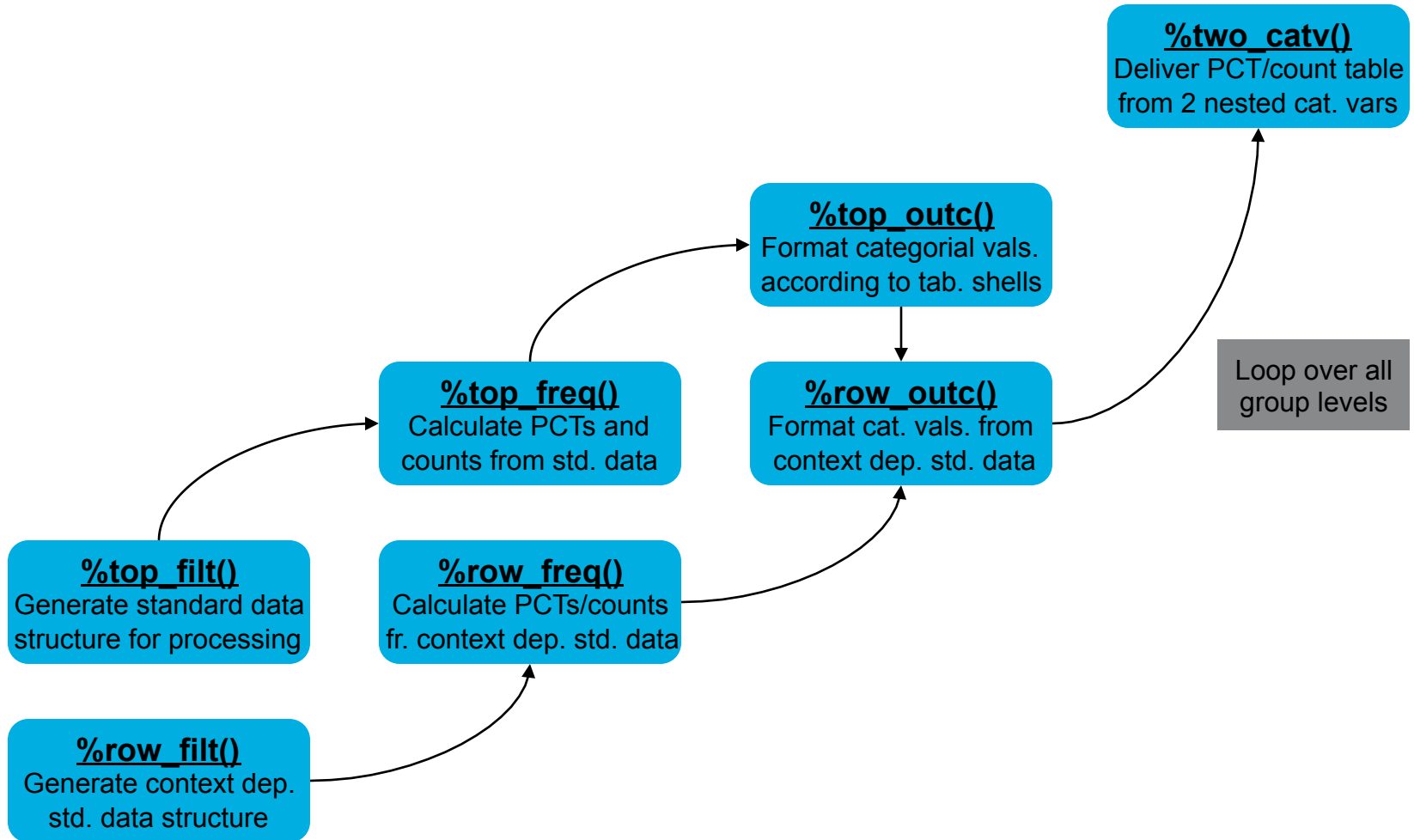
%blk_catv() – dataflow



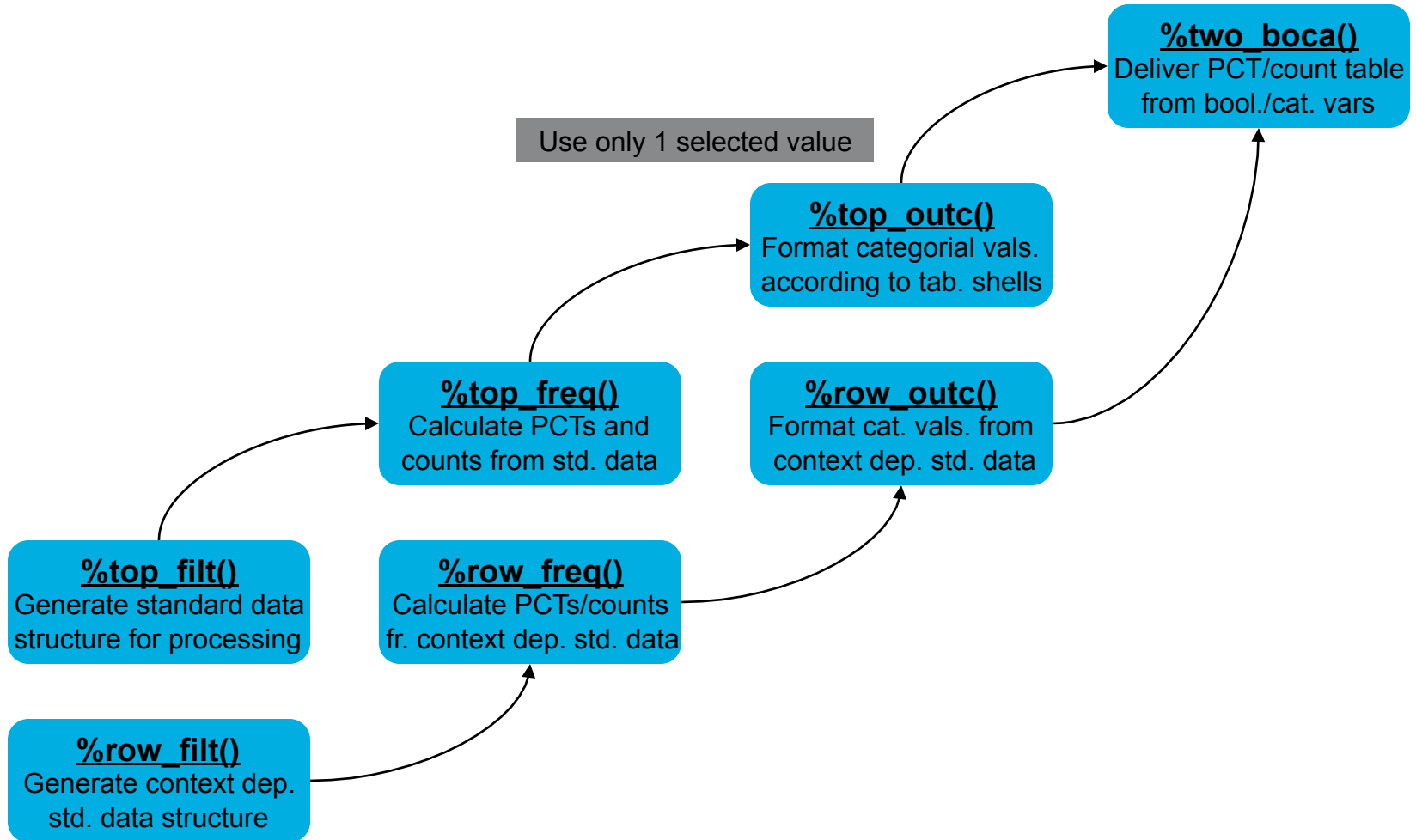
%tab_catv() – dataflow



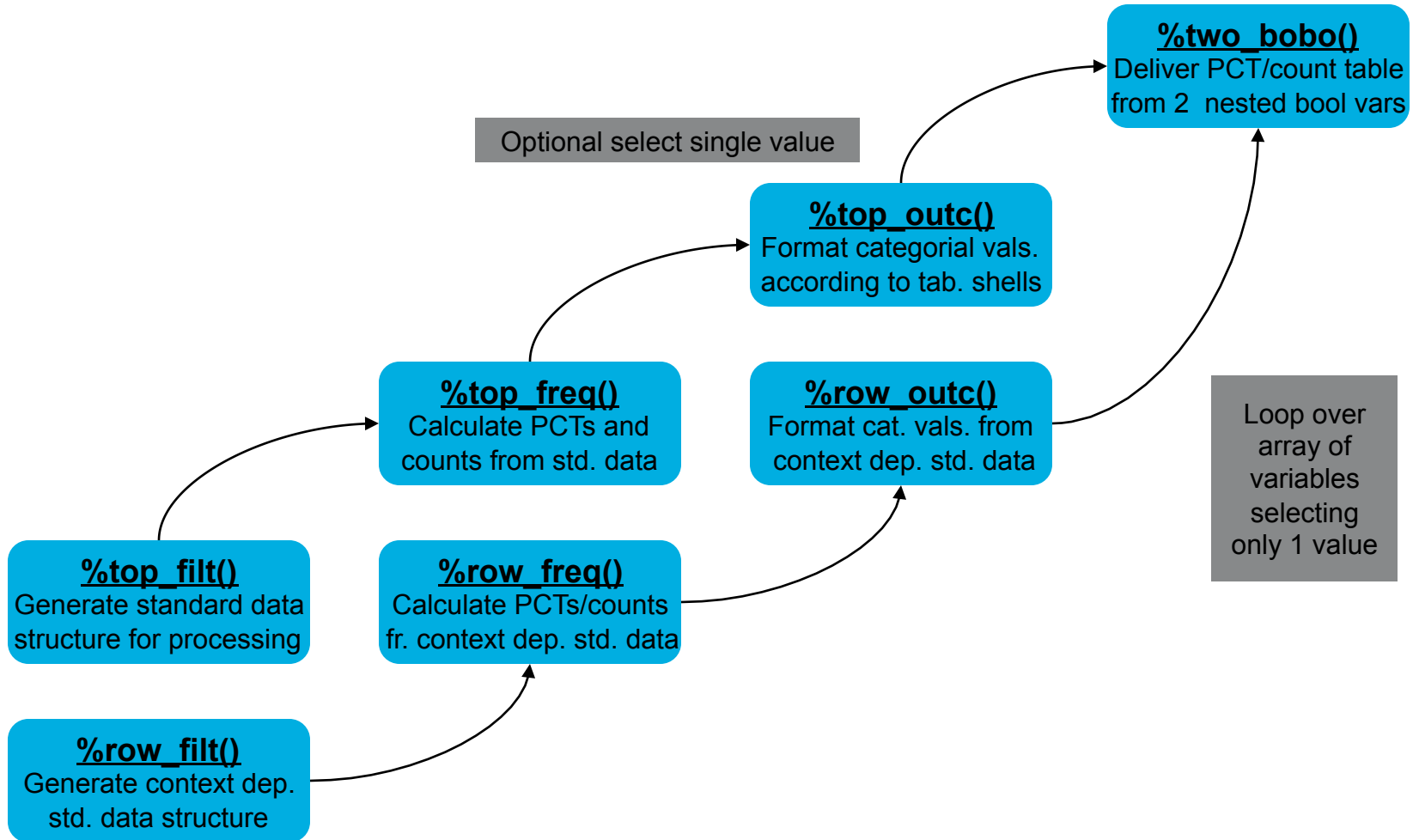
%two_catv() – dataflow



%two_boca() – dataflow



%two_bobo() – dataflow



EXAMPLES

1	Total subjects treated	2821 (100)	2823 (100)
2	Start of study medication within 24 hrs after randomization		
	Yes	2751 (97.5)	2775 (98.3)
	No	66 (2.3)	47 (1.7)
	Surgery delayed	59 (2.1)	41 (1.5)
	Other	13 (0.5)	13 (0.5)
	Missing	- (-)	- (-)
	Missing	4 (0.1)	1 (0.0)
3	Study medication only from kit allocated		
	Yes	2821 (100)	2823 (100)
	Missing	- (-)	- (-)
4	Number of subjects who did not receive full loading dose	26 (0.9)	12 (0.4)
	Reason:		
	Adverse event	3 (0.1)	3 (0.1)
	Technical failure / dosing error	17 (0.6)	8 (0.3)
	Missing	6 (0.2)	1 (0.0)
5	At least 75% of loading dose administered		
	Yes	2808 (99.5)	2819 (99.9)
	No	8 (0.3)	4 (0.1)
	Missing	5 (0.2)	- (-)
6	Number of subjects with maintenance dose interruptions longer than 1 hour	86 (3.0)	101 (3.6)
7	Number of maintenance dose interruptions		
	1	79 (2.8)	98 (3.5)
	2	4 (0.1)	4 (0.1)

CODE (1)

```
%BLK_CATV  
(dsn=_in_put_  
,row=sdw24h  
,rev=n  
,col=trnoat  
,space=2  
,total=o  
,head=n  
,indent=0  
,num=1  
,stat=n  
)
```

CODE (2)

```
%TWO_BOBO
(dsn=_in_put_
,use=
,use2=N
,row=sdw24h
,head=n
,row2=sdsudl#Y other#1 miss_5#1
,space=3
,col=trnoat
,indent=2
,indinc=2
,num=2
,total=n
,stat=n
,weight=y
,rev=y
)
```

CODE (3)

```
%BLK_CATV  
(dsn=_in_put_  
,row=sdon_der  
,rev=y  
,col=trnoat  
,space=2  
,total=n  
,head=y  
,indent=0  
,num=3  
,stat=n  
)
```

CODE (4)

```
%TWO_BOCA  
(dsn=_in_put_  
,use=0  
,row=loadfull  
,head=n  
,row2=nofusdrs  
,space=3  
,col=trnoat  
,indent=0  
,indinc=0  
,num=4  
,total=n  
,stat=n  
,weight=y  
)
```

CODE (5)

```
%BLK_CATV  
(dsn=_in_put_  
,row=vol75  
,rev=y  
,col=trnoat  
,space=2  
,total=n  
,head=y  
,indent=0  
,num=5  
,stat=n  
)
```

CODE (6)

```
%ROW_BOOL  
(dsn=_in_put_  
,row=infint1h  
,col=trnoat  
,space=0  
,total=n  
,indent=0  
,num=6  
,use=1  
,stat=n  
)
```

CODE (7)

```
%BLK_CATV  
(dsn=_in_put_  
,row=infmtno  
,rev=n  
,col=trnoat  
,space=  
,total=n  
,head=y  
,indent=0  
,num=7  
,stat=n  
)
```

INSIDE USER MODULES CONTENTS

Following pages show the sources of the currently available modules performing two-level processing. These examples intend to give an impression how to write additional user modules capable of generating complete subtables with quite complex structure and dependencies between calculations performed for specific lines.

source code (1) - %two_catv()

declares and upper level processing

```
%MACRO TWO_CATV(  
dsn=,exclude=,row=,row2=,col=,indent=0,num=,stat=N,weight=Y,space=2  
,condense=,struct=,struct2=,head2=N,misslin2=,indinc=2)  
  / store des="" ;  
  
%LOCAL n_grp v_grp n name;  
%LET name=TWO_CATV;  
  
%IF &STRUCT eq %THEN %LET struct =&DSN;  
%IF &STRUCT2 eq %THEN %LET struct2=&DSN;  
  
%GRP_DESC(dsn=&DSN,grp=&ROW,miss=n);  
  
%TOP_FILT(dsn=&DSN,grp=&ROW,by=&COL,grplvl=&NUM,var=,condense=&CONDENSE);  
%TOP_FREQ(dsn=top_filt,struct=&STRUCT,grp=&ROW,by=&COL);  
%TOP_OUTC(dsn=top_freq,head=n,total=n,stat=&STAT,indent=&INDENT,grp=&ROW  
,rev=n,use=,by=&COL,missline=);
```

source code (2) - %two_catv()

loop for lower level processing

```
%DO n=1 %TO &N_GRP;  
  
  %IF %SCAN(&V_GRP,&N) ne &EXCLUDE %THEN %DO;  
  
    %ROW_FILT(dsn=&DSN,context=&ROW,subgrp=&N,grp=&ROW2,by=&COL,var=,miss=n);  
    %ROW_FREQ(dsn=row_filt,sum=top_freq,struct=&STRUCT2,context=&ROW  
              ,grp=&ROW2,by=&COL,weight=&WEIGHT);  
    %ROW_OUTC(dsn=row_freq,sum=main_3rd,head=&HEAD2,stat=&STAT  
              ,indent=%EVAL(&INDENT+&INDINC),context=&ROW,grp=&ROW2  
              ,by=&COL,missline=&MISSLIN2);  
  
  %END;  
  
%END;
```

source code (3) - %two_catv()

care for naming and send completion mail

```
%IF &TAB_NAME ne %THEN %DO;

  data %SUBSTR(&TAB_NAME,1,3) &NUM%SUBSTR(&TAB_NAME,5,4);
    set
    %DO n=1 %TO &N_GRP;
      %IF &SPACE eq 1                                %THEN dummy ;
      %IF %SCAN(&V_GRP,&N) ne &EXCLUDE %THEN row&NUM._&N ;
      %IF &SPACE eq 2                                %THEN dummy ;
    %END;
      %IF &SPACE eq 3                                %THEN dummy ;
    ;
  run;

%END;

%GEN_MAIL(name=&NAME);
%MEND TWO_CATV;
```

source code (1) - %two_boca()

declares and upper level processing

```
%MACRO TWO_BOCA(  
  dsn=, use=, row=, row2=, col=, indent=0, num=, total=T, stat=Y, weight=Y, space=3  
  , condense=, struct=, struct2=, head=Y, head2=Y, misslin2=, indinc=2)  
  / store des="" ;  
  
  %LOCAL n_grp v_grp n name;  
  %LET name=TWO_BOCA;  
  
  %IF &STRUCT eq %THEN %LET struct =&DSN;  
  %IF &STRUCT2 eq %THEN %LET struct2=&DSN;  
  
  %TOP_FILT(dsn=&DSN, grp=&ROW, by=&COL, grplvl=&NUM, var=, condense=&CONDENSE) ;  
  %TOP_FREQ(dsn=top_filt, struct=&STRUCT, grp=&ROW, by=&COL) ;  
  %TOP_OUTC(dsn=top_freq, head=&HEAD, total=&TOTAL, stat=&STAT  
  , indent=&INDENT, grp=&ROW, rev=n, use=&USE, by=&COL) ;
```

source code (2) - %two_boca()

loop for lower level processing

```
%GRP_DESC (dsn=&DSN, grp=&ROW, miss=n) ;

%DO n=1 %TO &N_GRP;

  %IF %SCAN(&V_GRP, &N) eq &USE %THEN %DO;

    %ROW_FILT (dsn=&DSN, context=&ROW, subgrp=&N, grp=&ROW2, by=&COL, var=, miss=n) ;
    %ROW_FREQ (dsn=row_filt, sum=top_freq, struct=&STRUCT2
               , context=&ROW, grp=&ROW2, by=&COL, weight=&WEIGHT) ;
    %ROW_OUTC (dsn=row_freq, sum=, head=&HEAD2, stat=&STAT
               , indent=%EVAL(&INDENT+&INDINC), context=&ROW, grp=&ROW2
               , by=&COL, missline=&MISSLIN2) ;

  %END;

%END;
```

source code (3) - %two_boca()

care for naming and send completion mail

```
%IF &TAB_NAME ne %THEN %DO;

  data &TAB_DEF&NUM%SUBSTR(&TAB_NAME,5,4);
  set
    %IF &SPACE eq 1                %THEN dummy ;
      row&NUM._0
    %IF &SPACE eq 2                %THEN dummy ;
      row&NUM._&CURSUB
    %IF &SPACE eq 2                %THEN dummy ;
    %IF &SPACE eq 3                %THEN dummy ;
  ;
run;

%END;

%GEN_MAIL(name=&NAME);
%MEND TWO_BOCA;
```

source code (1) - %two_bobo()

declares and upper level processing

```
%MACRO TWO_BOBO(  
dsn=,use=,use2=,row=,row2=,col=,indent=0,indinc=2,num=,total=T,stat=Y  
,weight=Y,rev=N,space=2,condense=,struct=,struct2=,head=Y,head2=N)  
  / store des="Create hierarchy from two boolean vars";  
  
%LOCAL n_lst v_lst row_n use_n n_grp v_grp n i name;  
%LET name=TWO_BOBO;  
  
%IF &STRUCT eq %THEN %LET struct =&DSN;  
%IF &STRUCT2 eq %THEN %LET struct2=&DSN;  
  
%CHK_LIST(list=&ROW2);  
%GRP_DESC(dsn=&DSN,grp=&ROW,miss=n);  
  
%TOP_FILT(dsn=&DSN,grp=&ROW,by=&COL,grplvl=&NUM,var=,condense=&CONDENSE);  
%TOP_FREQ(dsn=top_filt,struct=&STRUCT,grp=&ROW,by=&COL);  
%TOP_OUTC(dsn=top_freq,head=&HEAD,total=&TOTAL,stat=&STAT,indent=&INDENT  
,grp=&ROW,rev=&REV,use=&USE,by=&COL);
```

source code (2) - %two_bobo()

loop for lower level processing

```
%DO n=1 %TO &N_LST;

%LET row_n = %SCAN(%SCAN(&V_LST,&N,|),1,#);
%LET use_n = %SCAN(%SCAN(&V_LST,&N,|),2,#);

%DO i=1 %TO &N_GRP;

  %IF %SCAN(&V_GRP,&I,|) eq &USE2 %THEN %DO;

    %ROW_FILT(dsn=&DSN,context=&ROW,subgrp=&I,grp=&ROW_N,by=&COL,var=,miss=n);
    %ROW_FREQ(dsn=row_filt,sum=top_freq,struct=&STRUCT2,context=&ROW
              ,grp=&ROW_N,by=&COL,weight=&WEIGHT);
    %LET cursub=&N;
    %ROW_OUTC(dsn=row_freq,sum=,head=&HEAD2,stat=&STAT
              ,indent=%EVAL(&INDENT+&INDINC),context=&ROW,grp=&ROW_N
              ,use=&USE_N,by=&COL);

  %END;

%END;

%END;
```

source code (3) - %two_bobo()

care for naming and send completion mail

```
%IF &TAB_NAME ne %THEN %DO;

  data %SUBSTR(&TAB_NAME,1,3) &NUM%SUBSTR(&TAB_NAME,5,4);
  set
    row&NUM._0(where=(class not contains "Missing"))
    %IF &SPACE eq 1 %THEN dummy;
  %DO n=1 %TO &N_LST;
    %IF &SPACE eq 2 %THEN dummy;
    row&NUM._&N
  %END;
    %IF &SPACE eq 2 %THEN dummy;
    row&NUM._0(where=(class contains "Missing"))
    %IF &SPACE eq 3 %THEN dummy;
  ;
run;

%END;

%GEN_MAIL(name=&NAME);
%MEND TWO_BOBO;
```

THANK YOU