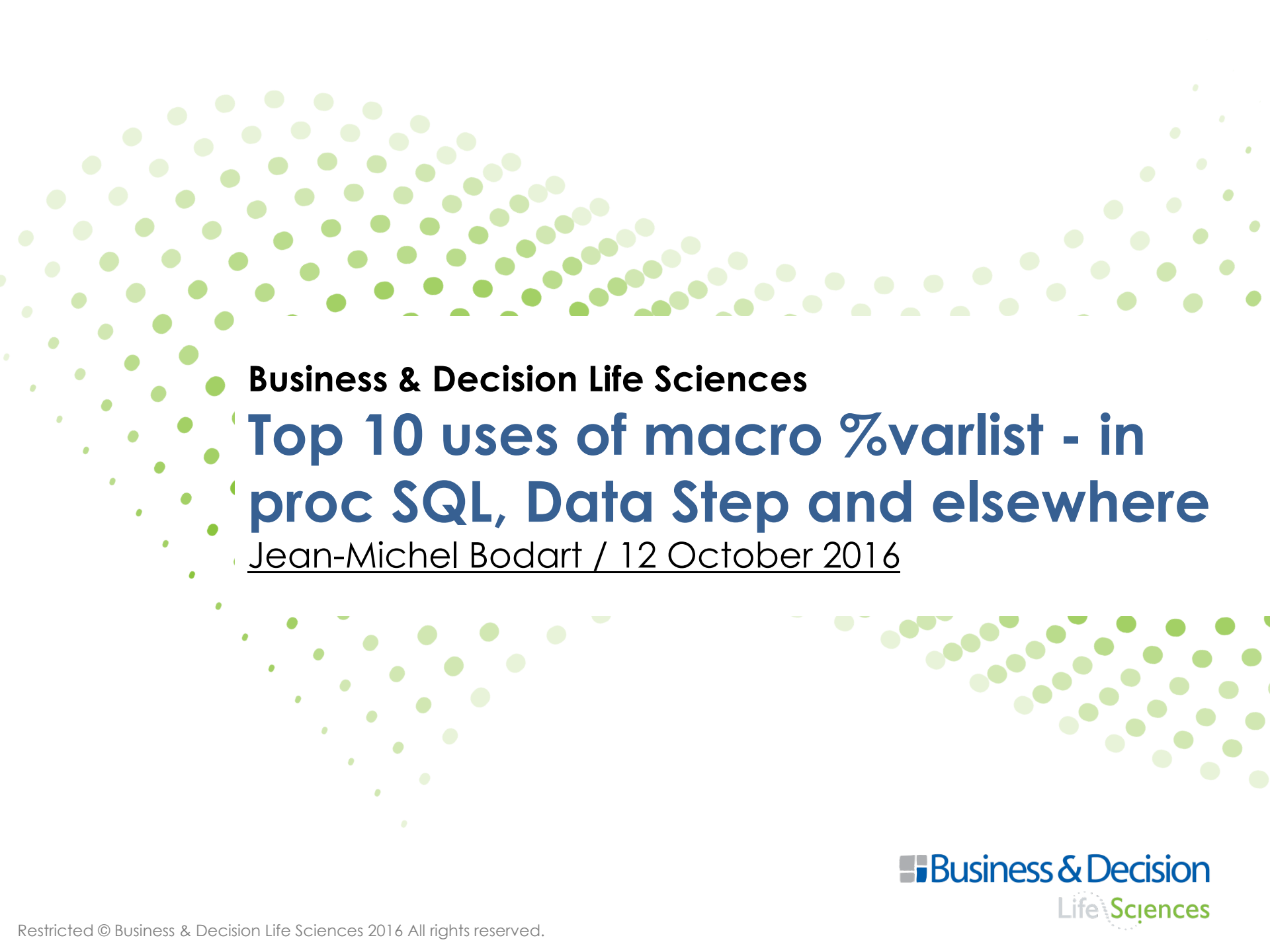


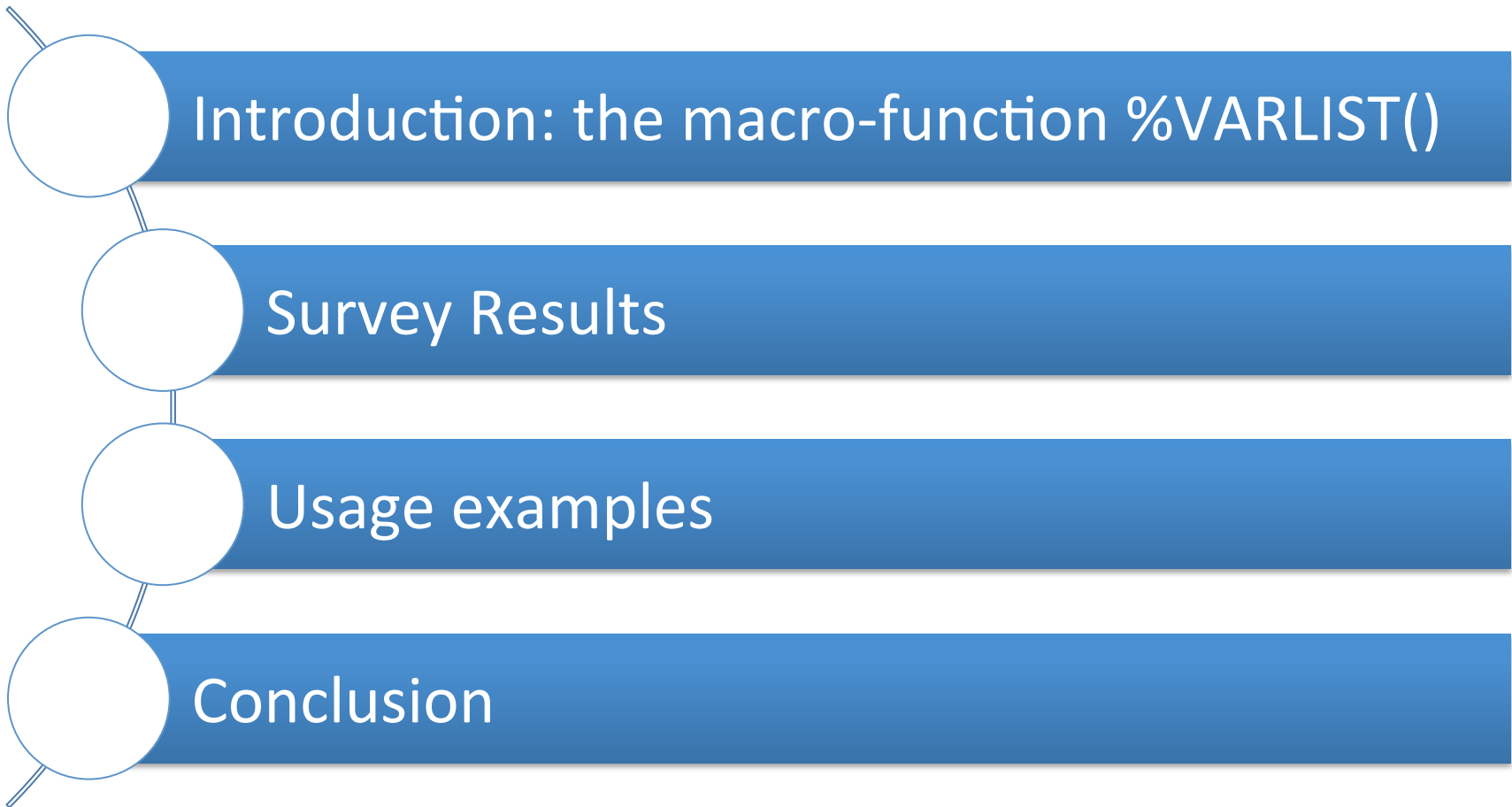


Business & Decision Life Sciences

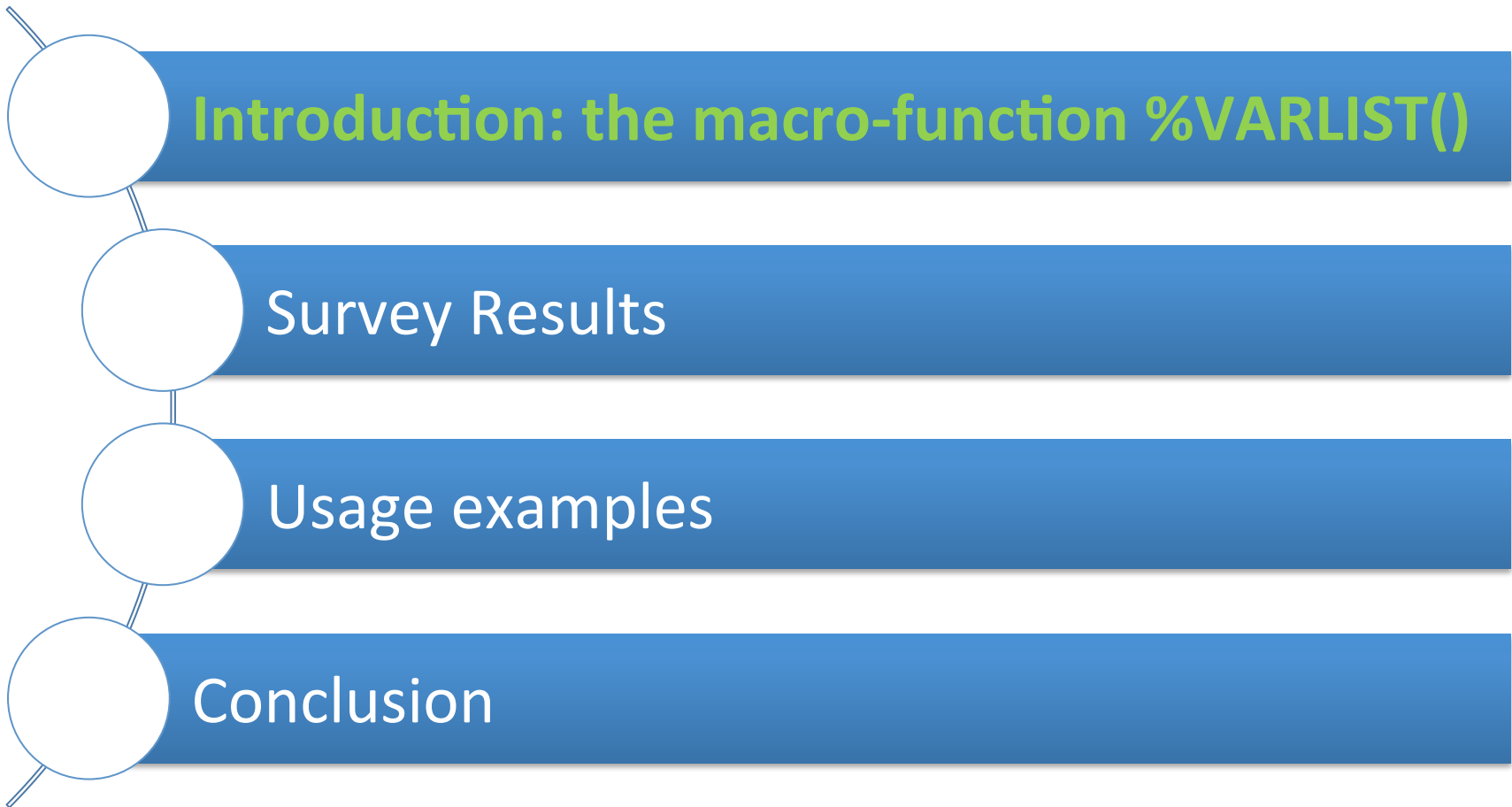
Top 10 uses of macro %varlist - in proc SQL, Data Step and elsewhere

Jean-Michel Bodart / 12 October 2016

Agenda



Agenda



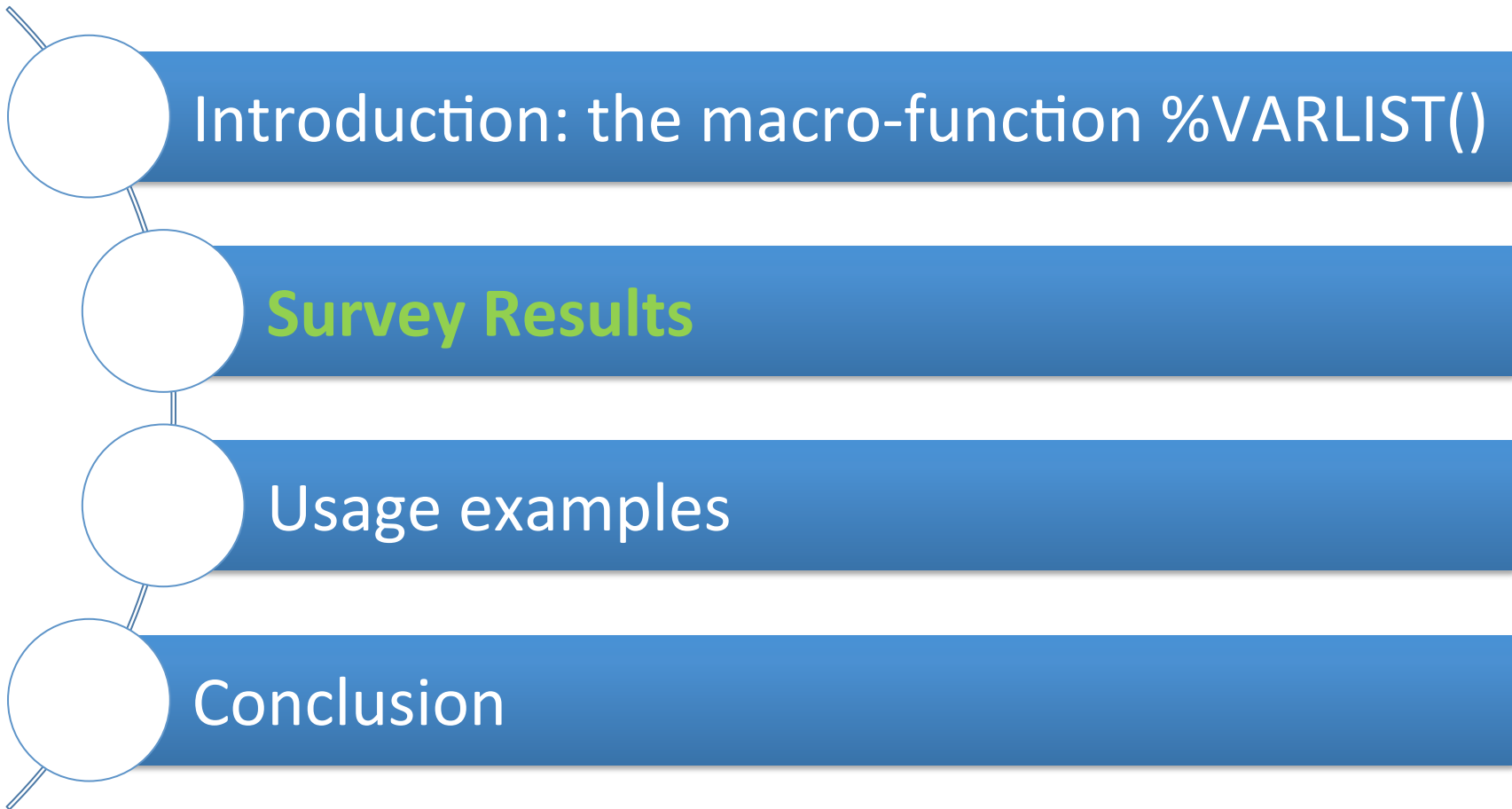
Introduction: the macro-function **%VARLIST()**

- **Macro-function** → think of: %UPCASE() or %SCAN()
- **User-written SAS Utility Tool**
 - Process lists of variables
 - Access variable attributes
- **“Input” parameters**
 - Data = [Dataset(s) [operator(s)]]
 - Var = [Variable Reference(s) [operator(s)]]
- **“Output” parameters (mainly)**
 - Sep = Keyword(s) | literal(s) [...] (default: #space#)
 - Pattern = Keyword(s) | literal(s) [...] (default: #var#)
- **Presented at PhUSE 2015 (CC05)**
 - Focus on (complex) SQL JOINS
- **Code available on PhUSE Wiki**
 - http://www.phusewiki.org/wiki/index.php?title=SAS_macro-function_%25VARLIST

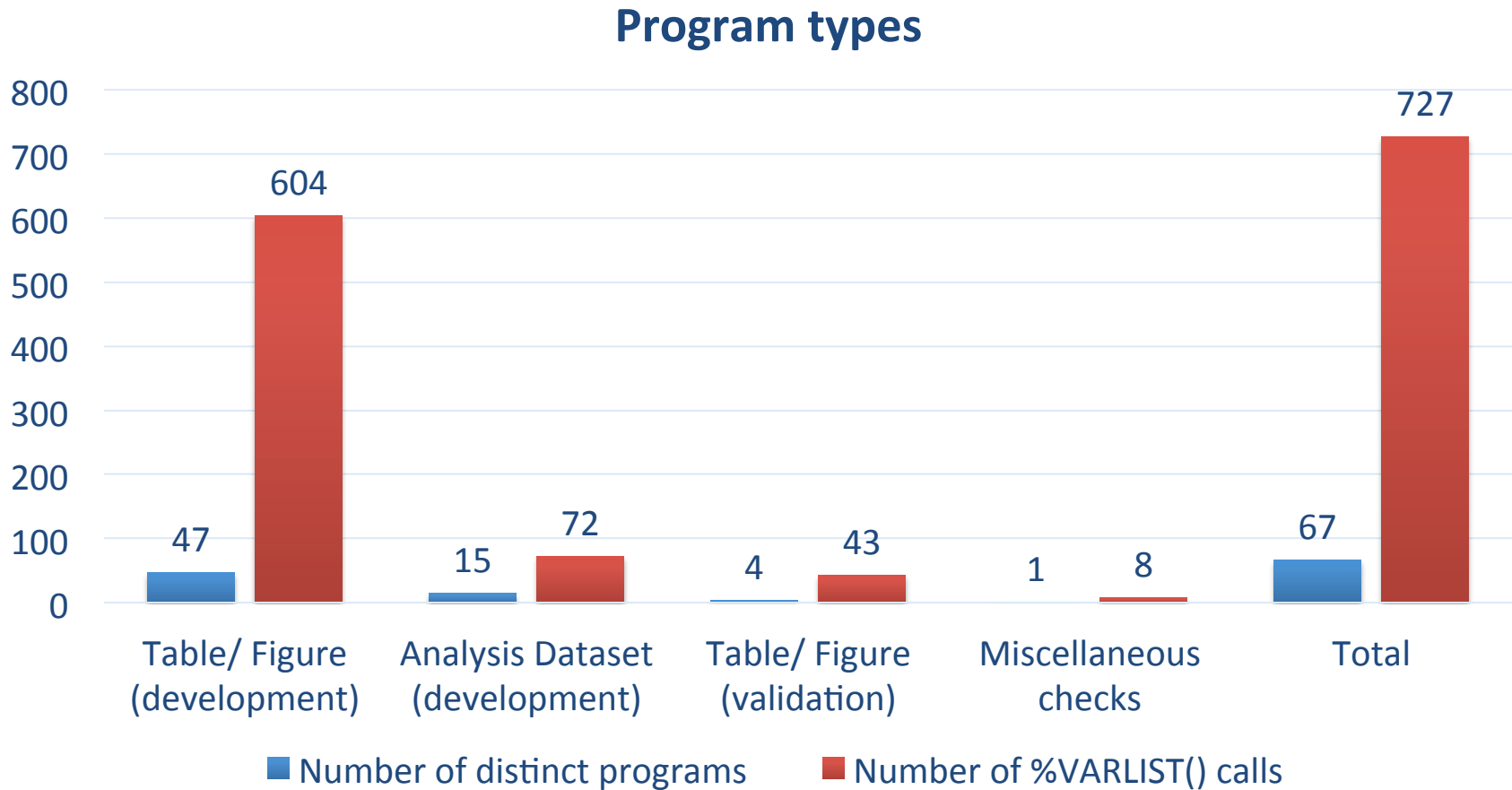
Survey of %VARLIST() Uses

- **%VARLIST() calls in SAS programs**
- **From daily work**
 - mainly Post-hoc and Exploratory analyses of clinical trials data
 - Analysis Datasets, Tables, Figures
- **Categorized by context and purpose**
 - both general and detailed levels
- **Performed in April 2016 (updated July 2016)**
 - 727 calls
 - 67 programs

Agenda



Survey of %VARLIST() Uses: Program Types

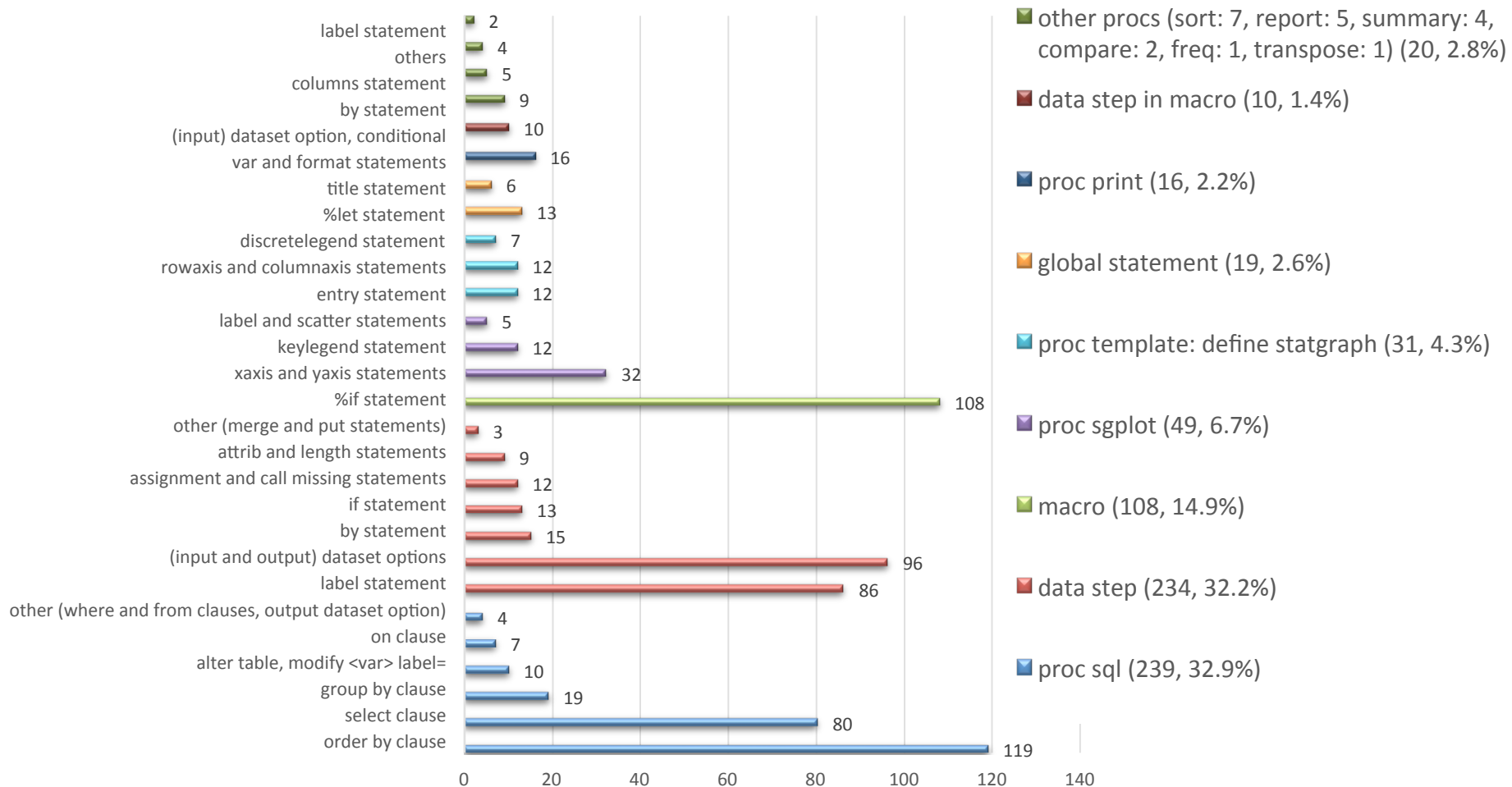


Survey of %VARLIST() Uses: Programming Context

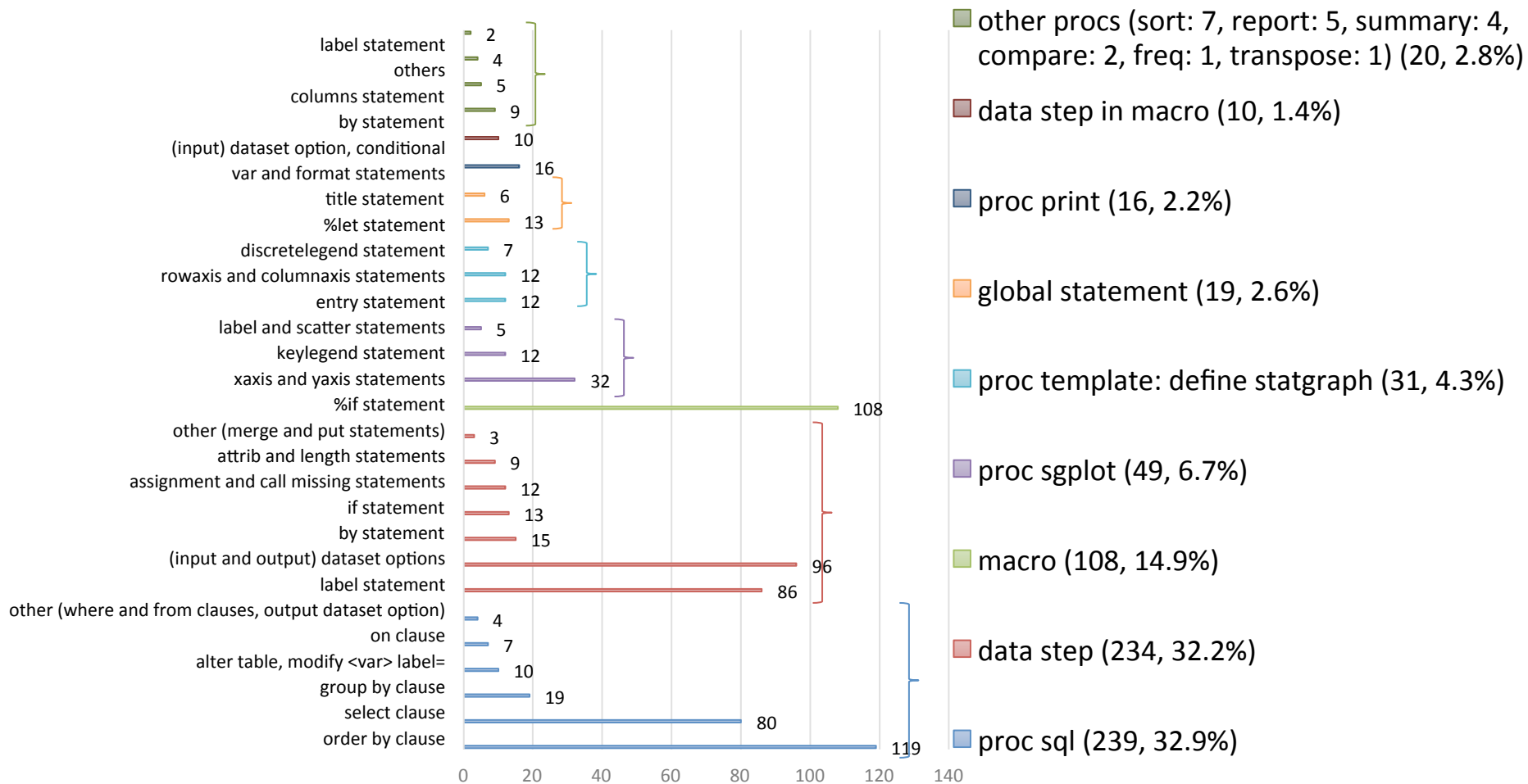
Context	Number and Percentage of calls	
proc sql	238	32.7%
data step	235	32.3%
macro	108	14.9%
proc sgplot	49	6.7%
proc template	31	4.3%
global statement	19	2.6%
proc print	16	2.2%
macro inside data step	10	1.4%

Context	Number and Percentage of calls	
proc sort	7	1.0%
proc report	5	0.7%
proc summary	4	0.6%
proc compare	2	0.3%
macro-definition	1	0.1%
proc freq	1	0.1%
proc transpose	1	0.1%

Survey of %VARLIST() Uses: Programming Context Details



Survey of %VARLIST() Uses: Programming Context Details



General Purposes of %VARLIST() Calls

Purpose	Number and % of Calls	
<u>Generate</u> list of unique variable names (from literal variable names and/or macro-variables or macro parameters containing 0 or more variable names) and optionally <u>insert</u> specific separators ; optionally <u>excluding</u> specific variable names	161	22.1%
<u>Check</u> whether (at least one of) a given (set of) variable(s) exist(s) in a given dataset	127	17.5%
<u>Transfer</u> variable labels (+ optional modification)	95	13.1%
<u>Retrieve</u> (and concatenate) label(s) of one or more variables and <u>assign</u> to title, footnote or graph element	92	12.7%
<u>Retrieve</u> those unique variable names from an input list (possibly with macro-variable references) that exist in one or more specific dataset(s) ; optionally <u>excluding</u> specific variable names	66	9.1%
<u>Insert</u> delimiters between space-separated list of literal variable names	39	5.4%
<u>Modify</u> existing label of a variable	36	5.0%

Top 10 uses of %VARLIST() by context and purpose (1/3)

1. **To check whether** (at least one of) **a given** (set of) **variable(s) exist(s)** in a given dataset, and **conditionally execute** some code accordingly, (e.g.) using **%IF statement** in macro definition (n=108).
2. **To generate a list of unique variable names** from literal variable names and/or macro-variables or macro parameters (containing 0 or more variable names) and optionally **insert specific separators**, with optional **exclusion of specific variable names**, (e.g.) in PROC **SQL SELECT** clause (n=87).
3. **To insert** additional **separators** (commas) between space-separated literal **variable names**, for **easier code maintenance**, so **the list** can be copied and pasted **unchanged between SQL** (e.g. ORDER BY clause) **and non-SQL code** (e.g. BY statement) (n=32).

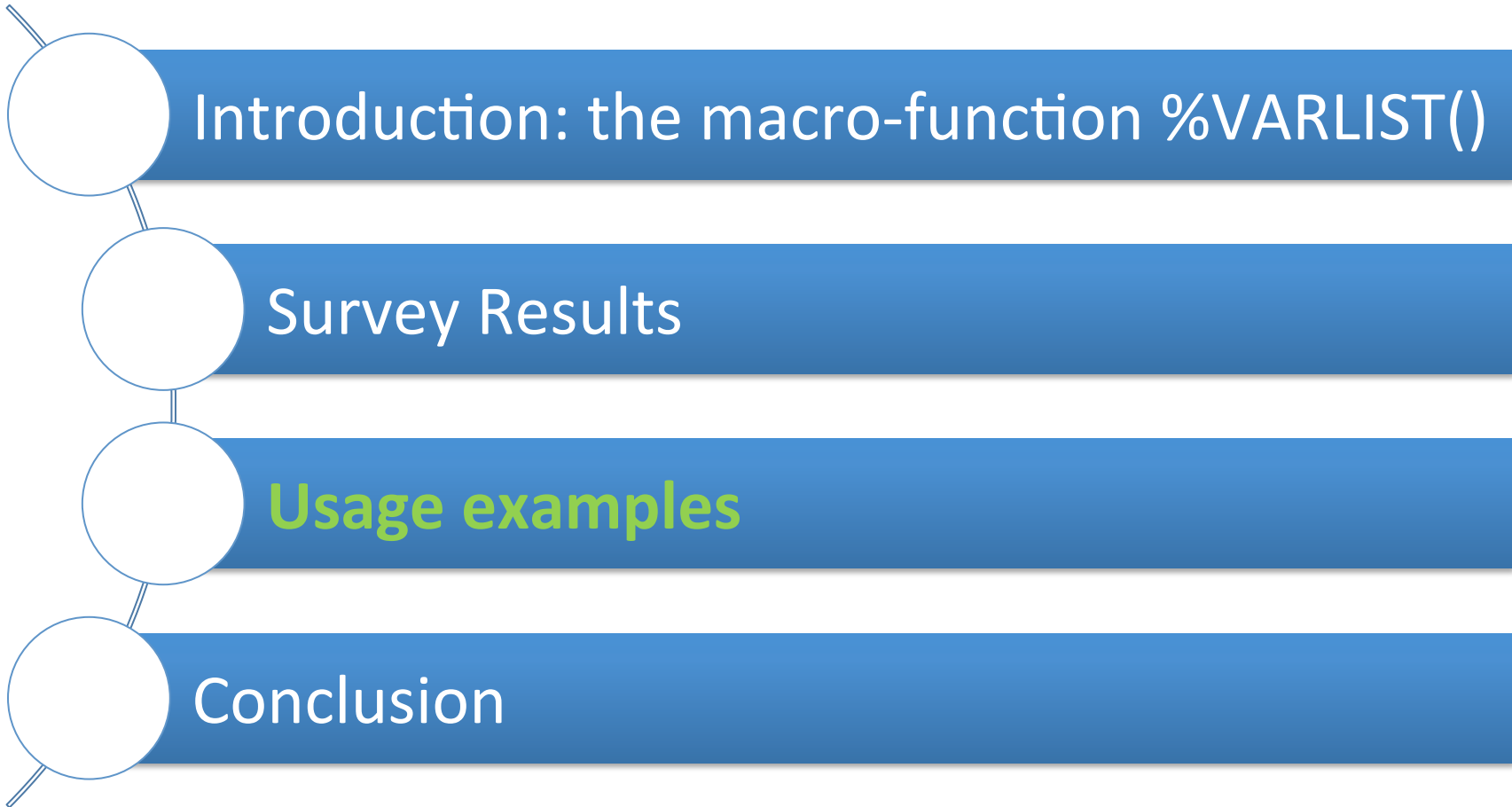
Top 10 uses of %VARLIST() by context and purpose (2/3)

4. **To retrieve the exact name** of a variable generated by SAS, that matches a **pre-defined pattern** (e.g. the bin variable from output dataset of the HISTOGRAM statement in SGPLOT procedure), and **rename it** to a specific name, using (input) **dataset option RENAME**, (e.g.) in DATA STEP **SET statement** (n=24).
5. **To transfer the label** from one variable to another variable, (e.g.) in DATA STEP **LABEL statement** (n=19).
6. **To retrieve** those **unique variables** that exist in one (or more) specific dataset(s), with optional **exclusion** of specific **variable names**, and by means of a specific pattern to generate a **comma-separated** list of variables with a **dataset name/alias qualifier** for use in **SQL JOIN** (n=18).
7. **To retrieve** (and concatenate) **the label(s)** of one (or more) variable(s) for **use in title, footnote or graph element label**, e.g. to assign an axis label in PROC SGPLOT **YAXIS statement** (n=18).

Top 10 uses of %VARLIST() by context and purpose (3/3)

8. **To retrieve** those **unique variables** from an **input list** (possibly with macro-variable references) **that exist** in one (or more) specific dataset(s), with optional **exclusion of specific variable names**, for use in **Input dataset option DROP**, (e.g.) in DATA STEP **SET statement** (n=14).
9. **To retrieve** those **unique variables** from an **input list** (possibly with macro-variable references) **that exist** in one (or more) specific dataset(s), with optional **exclusion of specific variable names**, for use in **Output dataset option DROP**, (e.g.) in DATA STEP **SET statement** (n=12).
10. **To generate** a list of **unique variable** names **from literal** variable names and/or **macro-variables** or macro parameters (containing 0 or more variable names) and optionally **insert specific separators**, with optional **exclusion of specific variable names**, (e.g.) in PROC **SQL ORDER BY and GROUP BY clauses** (n=12).

Agenda



Example 1: Summary Table Program

■ Descriptive Statistics

- change from baseline (efficacy parameters)
- by plasma concentration (study drug)
- over time / across time points
- by treatment sequence

■ Dataset: Cartesian Product (all combinations of categories)

- create records with n=0 for empty categories

■ Change request:

- Suppress concentration categories if empty for all treatment sequences

Week 96	Efficacy Parameter 1	Change from Baseline		
Drug A		Treatment sequence		
Concentration	Statistic	A-B	B-A	Overall
77.5 - <92.5	n	1		1
	Mean (SD)	-1.306 (NC)		-1.306 (NC)
	Median	-1.306		-1.306
	Min-Max	-1.31 - -1.31		-1.31 - -1.31
77.5 - <82.5	n			
	Mean (SD)			
	Median			
	Min-Max			
82.5 - <87.5	n			
	Mean (SD)			
	Median			
	Min-Max			
87.5 - <92.5	n			
	Mean (SD)			
	Median			
	Min-Max			
92.5 - <97.5	n	1		1
	Mean (SD)	-1.483 (NC)		-1.483 (NC)
	Median	-1.483		-1.483
	Min-Max	-1.48 - -1.48		-1.48 - -1.48

Example 1: Summary Table Program

Macro `%summary()`, parameters,
intermediate dataset `sum2a`

```
%macro summary(bygrp=, bygrp1=, xvar=, xvar1=, grp=, subgrp=, subgrp1= );
```

PARAMN	PARAMCD	AWEEKN	AWEEK	AVAL	AVALGRP	TRTSEQAN	TRTSEQA	N	MEAN	SD	MEDIAN	MIN	MAX
1	EFF1	96	Week 96	75	72.5 - <77.5	1	A-B	1	-1.306		-1.306	-1.31	-1.31
1	EFF1	96	Week 96	75	72.5 - <77.5	2	B-A	0					
1	EFF1	96	Week 96	75	72.5 - <77.5	99	Overall	1	-1.306		-1.306	-1.31	-1.31
1	EFF1	96	Week 96	80	77.5 - <82.5	1	A-B	0					
1	EFF1	96	Week 96	80	77.5 - <82.5	2	B-A	0					
1	EFF1	96	Week 96	80	77.5 - <82.5	99	Overall	0					
1	EFF1	96	Week 96	85	82.5 - <87.5	1	A-B	0					
1	EFF1	96	Week 96	85	82.5 - <87.5	2	B-A	0					
1	EFF1	96	Week 96	85	82.5 - <87.5	99	Overall	0					
1	EFF1	96	Week 96	90	87.5 - <92.5	1	A-B	0					
1	EFF1	96	Week 96	90	87.5 - <92.5	2	B-A	0					
1	EFF1	96	Week 96	90	87.5 - <92.5	99	Overall	0					
1	EFF1	96	Week 96	95	92.5 - <97.5	1	A-B	1	-1.483		-1.483	-1.48	-1.48
1	EFF1	96	Week 96	95	92.5 - <97.5	2	B-A	0					
1	EFF1	96	Week 96	95	92.5 - <97.5	99	Overall	1	-1.483		-1.483	-1.48	-1.48

```
%mend summary;
```

```
%summary(bygrp=AWEEKn, bygrp1=AWEEK, xvar=AVAL, xvar1=AVALGRP,
,grp=TRTSEQAN, grp=TRTSEQA );
```

```
%summary(bygrp= , bygrp1= , xvar=AVAL, xvar1=AVALGRP, grp=TRTSEQAN, grp=TRTSEQA );
```

Example 1: Summary Table Program

Suppressing empty categories from
intermediate dataset **sum2a**

```
proc sql noprint;
```

```
  create table sum2b as
```

```
  select * from sum2a
```

■ Group
by Categories

```
  group by
```

■ Remove Categories
with total n=0 across
treatment sequences

```
  having sum(n)>0
```

```
  order by
```

```
quit;
```

■ SQL GROUP BY

- Generate list of unique variables
- Separated by comma + space

```
%varlist (var= PARAMN PARAMCD PARAM  
&bygrp &bygrpl  
&xvarn &xvarl  
, sep= #cs# )
```

■ SQL ORDER BY

- Generate list of unique variables
- Separated by comma + space

```
%varlist (var= PARAMN PARAMCD PARAM  
&bygrp &bygrpl  
&xvarn &xvarl &grp.n &grp &grp.l  
&subgrp &subgrp &subgrpl  
statn  
, sep= #cs# , data=sum2a ) ;
```

Example 1: Summary Table Program

Suppressing empty categories from
intermediate dataset **sum2a**

```
MPRINT(SUMMARY):  proc sql noprint;
MPRINT(SUMMARY):  create table sum2b as select * from sum2a group by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEKKn, AWEK, AVAL, AVALGRP
MPRINT(SUMMARY):  having sum(n)>0 order by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEKKn, AWEK, AVAL, AVALGRP,
TRTSEQAn, TRTSEQA, TRTSEQA1, statn
MPRINT(SUMMARY):  ;
```

NOTE: The query requires remerging summary statistics back with the original data.

NOTE: Table WORK.SUM2B created, with 4290 rows and 33 columns.

```
MPRINT(SUMMARY):  proc sql noprint;
MPRINT(SUMMARY):  create table sum2b as select * from sum2a group by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEKKn, AWEK, AVAL, AVALGRP
MPRINT(SUMMARY):  having sum(n)>0 order by
MPRINT(VARLIST):   PARAMN, PARAMCD, PARAM, AWEKKn, AWEK, AVAL, AVALGRP,
TRTSEQAn, TRTSEQA, TRTSEQA1, statn
MPRINT(SUMMARY):  ;
```

NOTE: The query requires remerging summary statistics back with the original data.

NOTE: Table WORK.SUM2B created, with 4290 rows and 33 columns.

quit;

, sep= #CS#),

Example 1: Summary Table Program

Deriving page numbers with breaks every 3 by-groups

```
data sum2;  
  set sum2b(drop=
```

■ DATA step SET statement Dataset Option DROP=

Retrieve variables **page** and **vn** if found in dataset **sum2b** and **drop** them before creating RETAIN variables with same name

```
  %varlist(data=sum2b, var=page vn) );
```

```
by
```

■ DATA step BY statement

Generate list of **unique** names (first occurrence) from **var=** parameter, pass to **by** statement → define **first.<var>** and **last.<var>**

Same call (except no **sep=**) as in previous **SQL ORDER BY** → easy maintenance

```
%varlist(var=PARAMN PARAMCD PARAM &bygrp1 &xvar1  
&grp.n &grp &grp.1 &subgrp1 &subgrp &subgrp1 statn );
```

```
if first.%scan(
```

■ DATA step IF statement + %scan()

%scan() extracts the last name of list generated by **%varlist()** (from **var=** parameter) → refer to **first.<var>**

```
  %varlist(var=PARAMN PARAMCD PARAM &bygrp1  
, -1)
```

```
then do;
```

```
  vn=0; *- reset counter of &xvar1 by-groups -*;
```

```
end;
```

```
if first.&xvar1 then do;
```

```
  if mod(vn, 3)=0 then page+1; *- increment page every 3 by-groups -*;
```

```
  vn+1; *- increment vn at start of each &xvar1 by-group -*;
```

```
end;
```

```
run;
```

Example 1: Summary Table Program

Deriving page breaks every 3 by-groups

```

MPRINT (SUMMARY):      data sum2;
MPRINT (SUMMARY):      set sum2b(drop
MPRINT (SUMMARY):      = PAGE VN);
MPRINT (SUMMARY):      by
MPRINT (VARLIST):      PARAMN PARAMCD PARAM AWEEKn AWEEK AVAL AVALGRP TRTSEQAn
TRTSEQA TRTSEQAL statn
MPRINT (SUMMARY):      ;
MPRINT (SUMMARY):      if first.AWEEK then do;
MPRINT (SUMMARY):      vn=0;
MPRINT (SUMMARY):      *- reset counter of &xvarn by-groups -*;
MPRINT (SUMMARY):      end;
MPRINT (SUMMARY):      if first.AVAL then do;
MPRINT (SUMMARY):      if mod(vn, 3)=0 then page+1;
MPRINT (SUMMARY):      *- increment page every 3 by-groups -*;
MPRINT (SUMMARY):      vn+1;
MPRINT (SUMMARY):      *- increment vn at start of each &xvarn by-group -*;
MPRINT (SUMMARY):      end;
MPRINT (SUMMARY):      run;

```

Diagram illustrating the derivation of page breaks every 3 by-groups in the Summary Table Program. The program uses the `%varlist` macro to define variables and the `%scan` macro to process the data.

Key components and annotations:

- `%varlist(data=sum2b, var=page vn)`: Defines the data source and variables.
- `%varlist(var=PARAMN PARAMCD PARAM &bygrp &bygrpl &xvarn &xvarl &grp.n &grp &grp.l &subgrp &subgrpl statn)`: Defines the variables to be processed, including `PARAMN`, `PARAMCD`, `PARAM`, `&bygrp`, `&bygrpl`, `&xvarn`, `&xvarl`, `&grp.n`, `&grp`, `&grp.l`, `&subgrp`, `&subgrpl`, and `statn`.
- `%scan(%varlist(var=PARAMN PARAMCD PARAM &bygrp &bygrpl &xvarn &xvarl &grp.n &grp &grp.l &subgrp &subgrpl statn))`: Processes the variables, with `PARAM` and `&bygrp` highlighted.
- `AWEEKn AWEEK`: Variables used in the conditional logic.
- `AVAL AVALGRP`: Variables used in the conditional logic.
- `TRTSEQAn`: Variable used in the conditional logic.
- `TRTSEQA TRTSEQAL`: Variables used in the conditional logic.
- `statn`: Variable used in the conditional logic.

Example 2: Combine multiple datasets and check for duplicates

```
proc sql noprint;
  create table plasmalb as
    select subject, a.usubjid, a.SITEID, AVISITN, AVISIT, VISITNUM, VISIT
      , SAMPLEID, PARAMCD, METHOD, PCNAM, AVALC, AVAL, PARAM, PCDTCT
      , c.pATNF
      , d.RFsta
```

In SQL SELECT clause

- ☐ Retrieve unique variable names from dataset **adsl**
- ☐ Keep **all** except 'usubjid'
- ☐ Return list of variables separated by **comma** and **space** (#cs#),
- ☐ preceded by 'e.' to specify source dataset is **adsl**,
which was assigned alias name 'e' ☐ in SQL LEFT JOIN clause

```
      , %varlist(data=adsl ☐
        , var=#all# #not# usubjid ☐
        , sep=#cs# ☐ , pattern=e.#var# ☐ )
```

```
from plasma1 as a
  left join
    pATNF as c      on c.usubjid=a.usubjid
  left join
    RFsta as d      on d.usubjid=a.usubjid
  left join
    ADSL as e ☐ on e.usubjid=a.usubjid
```

[...]

Example 2: Combine multiple datasets and check for duplicates

[continued..]

order by

■ In SQL ORDER BY clause

Return list of variables unchanged but separated by **comma** and **space** (#cs#) (as required by SQL syntax)

This allows the (input) list of variables to be copied unchanged (e.g.) into DATA STEP BY STATEMENT (easy code maintenance).

```
%varlist(var= subject avisitn avisit visitnum visit
           sampleid paramcd pcdtc method pcnam
           ,sep= #cs# );

quit;

data plasmalc
  plasmalb_dups;
set plasmalb;
by subject avisitn avisit visitnum visit
   sampleid paramcd pcdtc method pcnam
   if (not last.pcnam) or (not first.pcnam)
   then output plasmalb_dups;
   if first.pcnam then output plasmalc;
run;
```



- Duplicates are identified
As multiple observations
in the same by-group

Example 3: Graph Program

Bland-Altman Plots and Histograms

1 – Generate Dataset

```
data BlandAltman;
  set measurements;
  MeanAB = (MethA + MethB) / 2;
  DiffAB = MethA - MethB;
  MeanAC = (MethA + MethC) / 2;
  DiffAC = MethA - MethC;
  *- append [variable name] to label -*;
  label
    ID = "Subject ID [ID]"
    X = "Age [X]"
    SEX = "Gender [Sex]"
    MethA = "Parameter P (Method A) [MethA]"
    MethB = "Parameter P (Method B) [MethB]"
    MethC = "Parameter P (Method C) [MethC]"
    MeanAB = "Average (Method A & Method B) in Parameter P [MeanAB]"
    DiffAB = "Difference (Method A - Method B) in Parameter P [DiffAB]"
    MeanAC = "Average (Method A & Method C) in Parameter P [MeanAC]"
    DiffAC = "Difference (Method A - Method C) in Parameter P [DiffAC]";
run;
proc print data=BlandAltman(obs=3) label noobs; run;
```

- Parameter P measured
- 3 methods: A (=Reference), B, C
- Bland-Altman plots:
 - Difference of 2 methods
 - Vs their Average

Example 3: Graph Program

Bland-Altman Plots and Histograms

1 – Generate Dataset

Subject ID [ID]	Gender [Sex] ↑	Age [X] ↑	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method B) in Parameter P [MeanAB]	Difference (Method A - Method B) in Parameter P [DiffAB]	Average (Method A & Method C) in Parameter P [MeanAC]	Difference (Method A - Method C) in Parameter P [DiffAC]
Alfred	M	14	119.0	112.5	101.5	115.75	6.5	110.25	17.5
Alice	F	13	106.5	84.0	129.0	95.25	22.5	117.75	-22.5
Barbara	F	13	115.3	98.0	115.0	106.65	17.3	115.15	0.3

label

ID = "Subject ID [ID]"

→ X = "Age [X]"

→ SEX = "Gender [Sex]"

MethA = "Parameter P (Method A) [MethA]"

MethB = "Parameter P (Method B) [MethB]"

MethC = "Parameter P (Method C) [MethC]"

MeanAB = "Average (Method A & Method B) in Parameter P [MeanAB]"

DiffAB = "Difference (Method A - Method B) in Parameter P [DiffAB]"

MeanAC = "Average (Method A & Method C) in Parameter P [MeanAC]"

DiffAC = "Difference (Method A - Method C) in Parameter P [DiffAC]";

run;

proc print data=BlandAltman(obs=3) label noobs; run;

Example 3: Graph Program

Bland-Altman Plots and Histograms

2 – Macro to Generate B-A Plots

```
%macro test(data=, x=, y=, group=, num=);
  data renamed&num.(drop= %varlist(data=&data, var= X Y)
    rename=(&x=X &y=Y));

  set &data;
  label &y="Variable Y [&y]";
  label &x=%sysfunc(tranwrd(%varlist(data=&data, var=&x, pattern=#vlabelq#)
    %str( in ), %str( +|+in )));

run;
```

```
2833 %test(data=BlandAltman, x=MeanAB, y=DiffAB, group=Sex, num=1);
MPRINT(TEST): data renamed1(drop
MPRINT(TEST): → rename=(MeanAB=X DiffAB=Y));
MPRINT(TEST): set BlandAltman;
MPRINT(TEST): label DiffAB = "Variable Y [DiffAB]";
MPRINT(TEST): label MeanAB=
MPRINT(TEST): "Average (Method A & Method B) +|+in Parameter P [MeanAB]";
MPRINT(TEST): run;
```

```
%mend test;
%test(data=BlandAltman, x=MeanAB, y=DiffAB, group=Sex, num=1); /* A vs B */
```

Example 3: Graph Program

Bland-Altman Plots and Histograms

2 – Macro to Generate B-A Plots

```
MPRINT (TEST): data renamed1(drop
MPRINT (TEST): = X rename=(MeanAB=X DiffAB=Y));
MPRINT (TEST): set BlandAltman;
MPRINT (TEST): label DiffAB = "Variable Y [DiffAB]";
MPRINT (TEST): label MeanAB=
MPRINT (TEST): "Average (Method A & Method B) +|+ in Parameter P [MeanAB]";
MPRINT (TEST): run;
```

```
proc print data=renamed&num(obs=3) label split="|" noobs;
  label Y="%sysfunc(translate(%varlist(data=&data, var=&y, pattern=#vlabel#)
  , ||, ( )) ***";
run;
```

```
MPRINT (TEST): proc print data=renamed1(obs=3) label split="|" noobs;
MPRINT (TEST): label Y=
MPRINT (TEST): "Difference || Method A - Method B || in Parameter P [DiffAB] ***";
MPRINT (TEST): run;
```

Subject ID [ID]	Gender [Sex]	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method B) + + in Parameter P [MeanAB]	Difference Method A - Method B in Parameter P [DiffAB] ***	Average (Method A & Method C) in Parameter P [MeanAC]	Difference (Method A - Method C) in Parameter P [DiffAC]
Alfred	M	119.0	112.5	101.5	115.75	6.5	110.25	17.5
Alice	F	106.5	84.0	129.0	95.25	22.5	117.75	-22.5
Barbara	F	115.3	98.0	115.0	106.65	17.3	115.15	0.3

Example 3: Graph Program

Bland-Altman Plots and Histograms

2 – Macro to Generate B-A Plots

```
%macro test(data=, x=, y=, group=, num=);
```

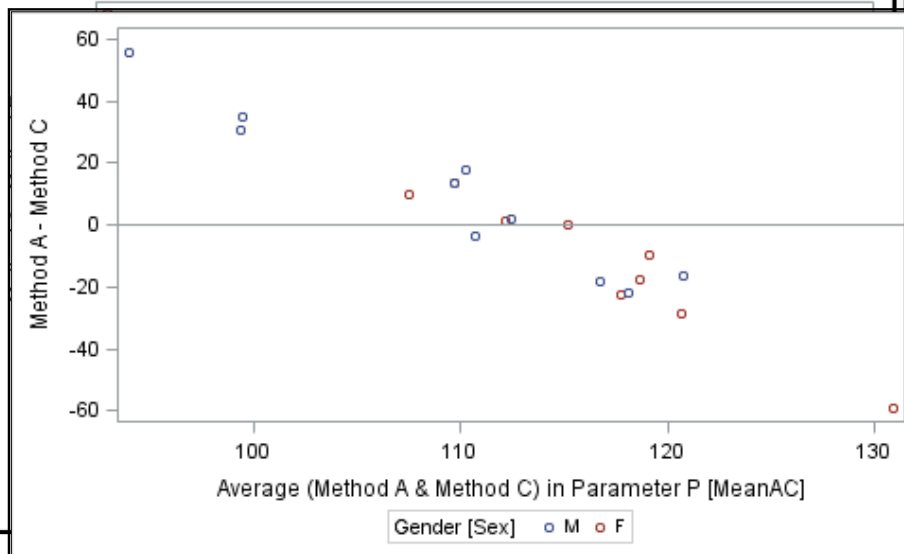
```
MPRINT(TEST):  proc sgplot data=renamed2;
MPRINT(TEST):  scatter x=X y=Y / group=Sex;
MPRINT(TEST):  refline 0 / axis=Y;
MPRINT(TEST):  xaxis label=
MPRINT(VARLIST):  "Average (Method A & Method C) in Parameter P [MeanAC]"
MPRINT(TEST):  ;
MPRINT(TEST):  yaxis label=
MPRINT(TEST):  "Method A - Method C"
MPRINT(TEST):  run;
```

```
scatter x=X y=Y / group=&group;
refline 0 / axis=Y;
xaxis label=%varlist(data=&data, v
yaxis label="%scan(%varlist(data=&
run;
```

```
%mend test;
```

```
%test(data=BlandAltman, x=MeanAB, y=DiffAB, group=Sex, num=1); /* A vs B */
```

```
%test(data=renamed1, x=MeanAC, y=DiffAC, group=Sex, num=2); /* A vs C */
```



Example 3: Graph Program

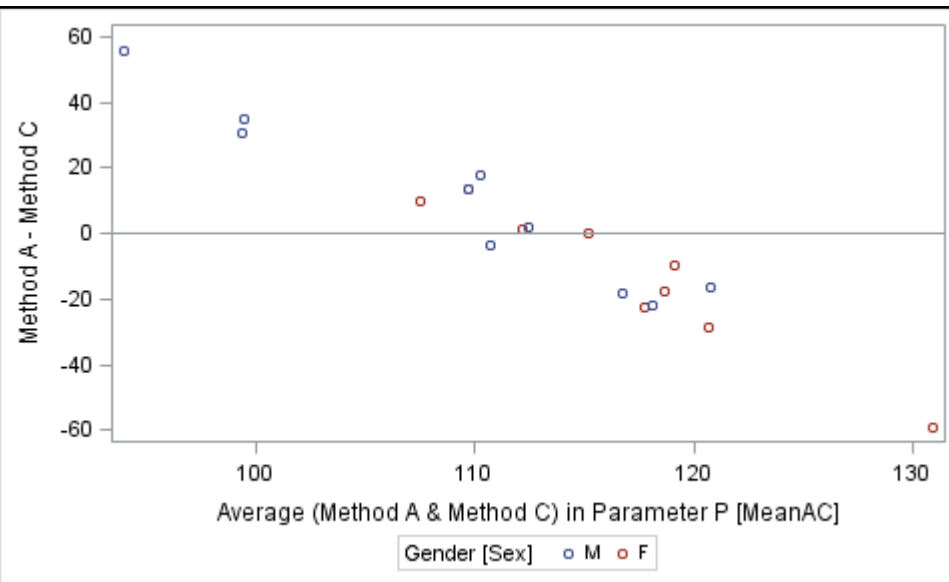
Bland-Altman Plots and Histograms

2 – Macro to Generate B-A Plots

```
%test(data=renamed1, x=MeanAC, y=
```

```
MPRINT(TEST): data renamed2(drop
MPRINT(TEST): = X Y rename=(MeanAC=X Diff
MPRINT(TEST): set renamed1;
MPRINT(TEST): label DiffAC = "Variable
MPRINT(TEST): label MeanAC=
MPRINT(TEST): "Average (Method A & Meth
MPRINT(TEST): run;
```

```
MPRINT(TEST): proc print data=renamed2(
MPRINT(TEST): label Y=
MPRINT(TEST): "Difference | Method A - M
MPRINT(TEST): run;
```



Subject ID [ID]	Gender [Sex]	Parameter P (Method A) [MethA]	Parameter P (Method B) [MethB]	Parameter P (Method C) [MethC]	Average (Method A & Method C) in Parameter P [MeanAC]	Difference Method A - Method C in Parameter P [DiffAC]
Alfred	M	119.0	112.5	101.5	110.25	17.5
Alice	F	106.5	84.0	129.0	117.75	-22.5
Barbara	F	115.3	98.0	115.0	115.15	0.3

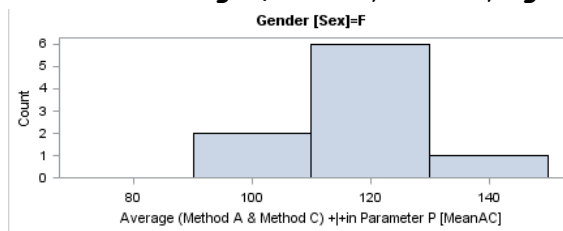
```
MPRINT(TEST): ;
MPRINT(TEST): yaxis label=
MPRINT(TEST): "Method A - Method C";
MPRINT(TEST): run;
```

Example 3: Graph Program

Bland-Altman Plots and Histograms

3 – Macro to Generate Histograms

```
%macro histogr(data=, var=, group=, binstart=, binwidth=, num=);
```



Obs	Sex	BIN X SCALE count	BINSTART 80 X	BIN X SCALE count	BINSTART 80 Y	X
1	F		80		0	117.75
2	F		100		2	115.15
3	F		120		6	112.15

```
proc sgplot data=&data;
```

```
by &group;
```

```
histogram &var / scale=count binstart=&binstart binwidth=&binwidth;
ods output sgplot=&var._&num.re=(Count>0));
run;
```

Obs	Sex	XBin	Count
2	F	100	2
3	F	120	6
4	F	140	1
11	M	100	4
12	M	120	6

```
data &var.Bins&num;
```

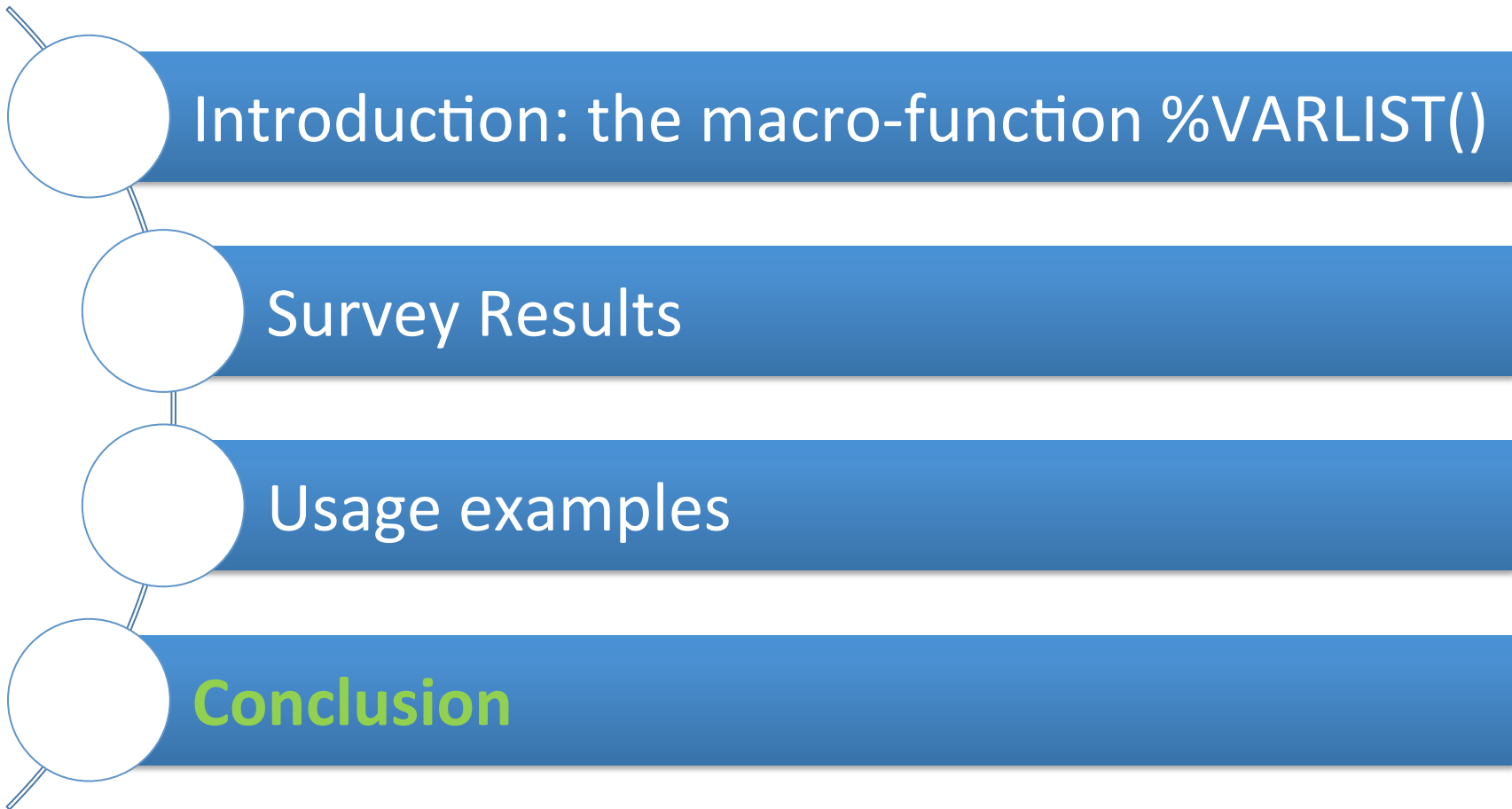
```
set &var._&num (rename= ( %varlist(data=&var._&num, var=Bin_&var._:Y) = Count
%varlist(data=&var._&num, var=Bin_&var._:X) = &var.Bin));
```

```
MPRINT(HISTOGR): data XBins2;
MPRINT(HISTOGR): set X_2 (rename
MPRINT(HISTOGR): =(BIN_X_SCALE_count_BINSTART_80__Y=Count BIN_X_SCALE_count_BINSTART_80__X=XBin));
MPRINT(HISTOGR): run;
```

```
%histogr(data=renamed1, var=X, group=Sex, binstart=80, binwidth=20, num=1);
```

```
%histogr(data=renamed1, var=Y, group=Sex, binstart=-25, binwidth=10, num=1);
```

Agenda



Conclusion: %VARLIST()

■ Widespread use

(much more than SQL: data step, macro code, proc steps, dataset options, titles/footnotes)

■ Rich features:

- **Generate lists** from input list (**var=**), including literals, macro-variables
- **Retrieve lists** from datasets (**data=**), including literals, macro-variables, matching patterns, #num#, #char#
- **Exclude** (list, #num#, #char#; variables found in other dataset(s))
- **Make unique**
- **Add separators** (**sep=**)
- **Retrieve variable attributes** (often label; also: type, length, (in)format)
- **Combine** together/ with literals as specified (**pattern=**)

Conclusion: %VARLIST()

■ Multiple purposes:

- **check** dataset variables
- **insert** commas for SQL
- **retrieve** (modify) **label** and assign in various ways
- **rename** variables matching pattern(s)
- create/modify new variables [with **prefix**, **suffix**, ...]
- ...

■ Top 10 uses ?

- Vary when we consider context, features and purpose, separately or in combinations

■ Availability

- Code, examples on PhUSE Wiki for everyone to use
- Contributions welcome: new features, examples



Thank you

Barcelona, Spain, 12 October 2016



Jean-Michel Bodart | Project Manager | Senior Statistical Programmer | jean-michel.bodart@businessdecision.com

Business & Decision Life Sciences

141 rue Saint-Lambert

B-1200 Brussels

T: +32 2 774 11 00

F: +32 2 774 11 99

lifesciences@businessdecision.com

<http://www.businessdecision-lifesciences.com/>