AdClin® TIPS

N°2

Table of contents

1. Introduction

1.1 TIP N°1 Follow-Up

It was not mentioned in TIP N°1, but we assumed that the TIP N°1 also applies to **%table2**.

1.2 N°2 Goal

In this TIP number, AdClin proposes TIPS to allow AdClin users to use %Table1 in non-standard conditions: Managing more than one by-variable and/or getting univariate statisitics labels within the columns. The TIP N°2 applies to %Table1 and does not concern %Table2.

1.3 Recall of the %Table1 Logic

%Table1 is meant to report into the same table univariate statistics and/or frequencies and percentages for various variables, coming from various datasets. The variables define blocks of reported figures, which are stacked up vertically. The different blocks line-up their results under a set of common columns, defined by a variable, as shown in the example below.

	Treatment A N=294	Treatment B N=296	All N=590
Sex			
Male	124 (42.2%)	129 (43.6%)	253 (42.9%)
Female	170 (57.8%)	167 (56.4%)	337 (57.1%)
Race			
Caucasian	285 (96.9%)	288 (97.3%)	573 (97.1%)
Black	2 (0.7%)	1 (0.3%)	3 (0.5%)
Oriental	2 (0.7%)	4 (1.4%)	6 (1.0%)
Other	5 (1.7%)	3 (1.0%)	8 (1.4%)
Age (years)			
N	294	296	590
Mean (Std)	44.1 (12.89)	43.9 (13.56)	44.0 (13.22)
Median	44.6	44.5	44.5
Min ; Max	18 ; 69	18 ; 71	18 ; 71

The columns are defined by the values of a column variable (here, Treatment) found in a reference dataset, the population dataset. The variables reported in the blocks may come from this population dataset, or from other datasets. In the latter case, the other datasets must have a one-to-one or many-to-one relation to the population dataset. In addition, by default, all column percentages will be computed versus frequencies computed once for the whole table, from the population dataset, and displayed as N= in the headings of the columns above.

1.4 Current Restrictions

Currently, %Table1 manages only **one level** of information as columns with **Colvar=** (e.g. Treatment) and also **one level** of by-variable with **By=**.

Furthermore the column levels are the categories of a frequency variable. There is a priori no possibility to represent 2 non-disjonctive sets in columns (e.g. "ITT" and "PP" or any variables).

Also, it is not possibe to display univariate statistics (Mean, Median, Std,…) as columns.

1.5 By= , Recall

The by-variable must be part of the "VarDataset" or part of the "PopDataset" if no other file is used.

We present below two examples generated from the "PopDataSet=Subjects" structure below:

SubjectId	Age	Compliance	Sex	Race	Trt	FAS
01-001	58.1	0.88	2	1	1	1
01-002	56.0	0.98	2	2	2	1
01-003	48.1	0.92	1	1	2	1
01-004	63.8	0.96	2	3	2	1
01-005	64.0	.	1	1	.	0
01-006	44.4	1	2	1	1	1
01-007	67.9	0.95	1	1	1	1
01-008	43.2	0.97	1	1	1	1
01-009	48.5	0.92	1	3	1	1
01-010	53.2	.	2	2	.	0
...	...	...	...	...	...	...

The syntax:

```
%Title(tip 01.1 Example 1: By= ByInBlocks=Yes)
%Table1( Popdataset=main2.subjects, PopId=subjectId,
    ColVar=Trt " ", By=Sex,
    Blocks=
        Race type=freq/
        Age  type=univ/
)
```

Gives:

Tip 01.1 Example 1: By= , ByInBlocks=Yes

	Drug A N=35	Drug B N=39	All N=74
Race			
Male			
Caucasian	11 (31.4%)	12 (30.8%)	23 (31.1%)
Black	1 (2.9%)	2 (5.1%)	3 (4.1%)
Asian	8 (22.9%)	5 (12.8%)	13 (17.6%)
Female			
Caucasian	7 (20.0%)	9 (23.1%)	16 (21.6%)
Black	0 (0.0%)	4 (10.3%)	4 (5.4%)
Asian	8 (22.9%)	7 (17.9%)	15 (20.3%)
Age at Baseline (years)			
Male			
N	20	19	39
Mean (std)	51.96 (8.83)	56.03 (9.40)	53.94 (9.23)
Median	53.82	53.86	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
Female			
N	15	20	35
Mean (std)	56.99 (8.72)	59.03 (9.55)	58.16 (9.13)
Median	58.13	59.71	59.52
Min / max	38.0 / 72.4	29.7 / 74.8	29.7 / 74.8

The syntax:

```
%Title(tip 01.1 Example 1: By= ByInBlocks=Yes)
%Table1( Popdataset=main2.subjects, PopId=subjectId,
    ColVar=Trt " ", By=Sex, ByInBlocks=No,
    Blocks=
        Race type=freq/
        Age  type=univ/
)
```

Gives:

Tip 01.2 Example 2: By= , ByInBlocks=No

	Drug A N=35	Drug B N=39	All N=74
Male			
Race			
Caucasian	11 (31.4%)	12 (30.8%)	23 (31.1%)
Black	1 (2.9%)	2 (5.1%)	3 (4.1%)
Asian	8 (22.9%)	5 (12.8%)	13 (17.6%)
Age at Baseline (years)			
N	20	19	39
Mean (std)	51.96 (8.83)	56.03 (9.40)	53.94 (9.23)
Median	53.82	53.86	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
Female			
Race			
Caucasian	7 (20.0%)	9 (23.1%)	16 (21.6%)
Black	0 (0.0%)	4 (10.3%)	4 (5.4%)
Asian	8 (22.9%)	7 (17.9%)	15 (20.3%)
Age at Baseline (years)			
N	15	20	35
Mean (std)	56.99 (8.72)	59.03 (9.55)	58.16 (9.13)
Median	58.13	59.71	59.52
Min / max	38.0 / 72.4	29.7 / 74.8	29.7 / 74.8

One can see that the ByInBlocks= Parameter controls how By-values are nested with Block-variable. By Default ByInBlocks=Yes.

Beyond the above mentioned restrictions, The By= parameter prevents the user to report another Block-variable, in the same table, like the Compliance, for which he does NOT want to apply the By-processing. Currently all Block-variables are By-processing dependent.

2. By processing, One Level, Without using By=

2.1 By processing using Macro Language: Reporting "Age" by "Sex" but "Race" globally in the same table.

Like this:

Tip 02.1 ML: Reporting "Age" by "Sex" but "Race" globally in the same table

	Drug A N=35	Drug B N=39	All N=74
Age			
Male			
N	20	19	39
Mean (std)	51.96 (8.83)	56.03 (9.40)	53.94 (9.23)
Median	53.82	53.86	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
Female			
N	15	20	35
Mean (std)	56.99 (8.72)	59.03 (9.55)	58.16 (9.13)
Median	58.13	59.71	59.52
Min / max	38.0 / 72.4	29.7 / 74.8	29.7 / 74.8
Race			
Caucasian	18 (51.4%)	21 (53.8%)	39 (52.7%)
Black	1 (2.9%)	6 (15.4%)	7 (9.5%)
Asian	16 (45.7%)	12 (30.8%)	28 (37.8%)

This can be done by using SAS Macro Language (In blue) as follow:

```
%Title(tip 02.1 ML: Reporting 'Age' by 'Sex' but 'Race' globally in the same table)
*--- De-Assign format on Sex;
data subjects; set main2.subjects; format Sex ; run;
*--- getting info;
Proc sql noprint;
    *--- Getting the code list for Sex;
    select distinct sex into: sexlist separated by " "
    from subjects where FAS=1;
    *--- Getting number of codes for Sex;
    select count (distinct sex) into: nsex
    from subjects where FAS=1;
Quit;

%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt.,
        Blocks=
            "Age"/
            %do sx = 1 %to &nsex;
                %*--- Getting the code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %*--- Getting the "decode";
                "%sysfunc(putn(&sex,sex.))" indent=1;
                Age@(Sex=&sex) " "  type=univ indent=2/
            %end;
            Race type=freq/
    )
%mend Temp;

%temp
```

Warning: The Sex code list assign to the macro variable "**&sexlist**" may not contain all the expected codes for Sex (e.g. If the Dataset contains only Males). This may be the case because we have searched for codes in the data. Depending on your situtation you may directly give the code list either from the data as shown above or by the simple syntaxe:

`%let sexlist = 1 2; %let nsex =  2;` if you know the codes and their number.

or from the Format Library if it exists to get all expected codes:

```
Proc format library=main2 cntlout=FmtOut;
Proc sql noprint;
    *--- Getting the code list for Sex;
    select distinct start into: sexlist separated by " "
    from FmtOut where fmtname='SEX';
    *--- Getting number of codes for Sex;
    select count (distinct start) into: nsex
    from FmtOut where fmtname='SEX';
Quit;
```

In this example we have done nothing more than repeating the example 1.1 for "Age" but reporting "Race" globally.

2.2 By processing using the macro %serlist: Reporting "Age" by "Sex" but "Race" globally in the same table.

%Serlist: Syntaxe and Basics

This macro generates a series from a template and a list.

The macro generates the template, repeated for each word in the list, and with substitutions performed.

Arguments

- Template — the template to repeat with substitutions
- List — the list of words for which the substitution should be done
- ListSep= — %str( ) the separator between words in the List argument
- Sep= — the separator between the repetitions
- Rep=? — the character(s) to replace by the words in the template

Restrictions

%SerList unquotes the template passed for each word in the list, which may result in unpredictable errors if the template contains some special characters like % or mismatched quotes or parentheses.

Examples

```
1) %SerList( c?=x? , a b c )                    will generate: ca=xa cb=xb cc=xc

2) %SerList( "?" , a b / c d , ListSep=/ )      will generate: "a b" "c d"

3) %SerList( "_mean?" ha , 1 2 3 A , Sep=!)

will generate: "_mean1" ha ! "_mean2" ha ! "_mean3" ha ! "_meanA" ha
```

```
4)
%SerList(
      data ?;
            set ? (keep=titi tata);
            length dataset $10;
            dataset="?";
      run;
, toto tutu )
```

Template

Liste

will generate:

```
data toto;
      set toto (keep=titi tata);
      length dataset $10;
      dataset="toto";
run;

data tutu;
      set tutu (keep=titi tata);
      length dataset $10;
      dataset="tutu";
run;

5)
%SerList(
      %SerList( a###? = b#? , x y z )
, 1 2 3, rep=# )

will generate:

a111x = b1x  a111y = b1y  a111z = b1z  a222x = b2x  a222y = b2y  a222z = b2z  a333x = b3x
a333y = b3y  a333z = b3z

6)
%SerList(
      %SerList( a##? = b##? , x y z )
, 1 2 3, rep=## )

will generate:

a1x = b1x  a1y = b1y  a1z = b1z  a2x = b2x  a2y = b2y  a2z = b2z  a3x = b3x  a3y = b3y  a3z = b3z
```

As shown, **two** %serlist can be nested. The list of the 2nd %serlist (x y z) being executed for each word of the list of the 1st %serlist (1 2 3) …

You may test %serlist before final use in your programs by running:

```
%put
      %SerList(
            %SerList( a##? = b##? , x y z )
      , 1 2 3, rep=## )

;
```

The %put will allow you to see the generated code of %serlist in a dummy way, in the LOG…

Using %serlist to report: "Age" by "Sex" but "Race" globally in the same table.

This can be done by using the Macro %serlist (In blue) as follow:

```
%Title(tip 02.2 Serlist: Reporting 'Age' by 'Sex' but 'Race' globally in the same table)
%Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
   ColVar=Trt " " trt.,
   Blocks=
       "Age"/
       %unquote(%serlist(
           "%nrstr(%%)sysfunc(putn(?,sex.))" indent=1;
           Age@(Sex=?) " "  type=univ indent=2/
       , &sexlist))
       Race type=freq/
    )
```

The result is the same as in 2.1:

Table 02.2 Serlist: Reporting "Age" by "Sex" but "Race" globally in the same table

	Drug A N=35	Drug B N=39	All N=74
Age			
Male			
N	20	19	39
Mean (std)	51.96 (8.83)	56.03 (9.40)	53.94 (9.23)
Median	53.82	53.86	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
Female			
N	15	20	35
Mean (std)	56.99 (8.72)	59.03 (9.55)	58.16 (9.13)
Median	58.13	59.71	59.52
Min / max	38.0 / 72.4	29.7 / 74.8	29.7 / 74.8
Race			
Caucasian	18 (51.4%)	21 (53.8%)	39 (52.7%)
Black	1 (2.9%)	6 (15.4%)	7 (9.5%)
Asian	16 (45.7%)	12 (30.8%)	28 (37.8%)

We remember you that the macro variable **&sexlist** contains the Sex codes: 1 2.

The %sysfunc macro allows retrieving SAS base information at Macro Language level. Here the Macro %sysfunc will resolve the "decode" of the Sex code (e.g. Male or Female). The Macro %nrstr (No Resolution of String) avoid the macro %sysfunc to resolve before %serlist, as normaly planned by SAS. '%' is replaced by %nrstr(%%). Finally the %serlist macro should not consider internal macro quoting. Thus %unquote prevents %serlist of doing this.

The genarated code is:

```
%Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
   ColVar=Trt " " trt.,
   Blocks=
       "Age"/
       "Male" indent=1;
       Age@(Sex=1) " "  type=univ indent=2/
       "Female" indent=1;
       Age@(Sex=2) " "  type=univ indent=2/
       Race type=freq/
    )
```

Even though the syntaxe appears quite esoteric, we have saved statements compared to the code typed in 2.1. More over there is no need to create the macro "%temp".

We do not recommand to use %serlist if you managed more than one By-variable level. But for the current example or similar ones, it worth it. For instance everytime you will have to use a Block syntaxe of the form: BlockVar@(Var=**x**) or BlockVar=**x** where '**x**' is a hardcoded category (e.g. =1, =2, ="A", etc…). In some cases, you may have to type a lot of hardcoded values e.g.: To get Confidence Intervalles of percentages for all categories for a frequency variable. You may type for Sex:

```
%Title(tip 02.3 Serlist: CI 95% On Sex percentages)
%Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
   ColVar=Trt " " trt., type=freq,
   Blocks=
       "Sex"/
       %unquote(%serlist(%str()
           Sex=? "%nrstr(%%)sysfunc(putn(?,sex.))" indent=1 ci=exact /
       , &sexlist))
    )
```

Generating:

```
%Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
   ColVar=Trt " " trt., type=freq,
   Blocks=
       "Sex"/
       Sex=1 "Male" indent=1 ci=exact /
       Sex=2 "Female" indent=1 ci=exact /
    )
```

resulting in:

Table 02.3 Serlist: CI 95% On Sex percentages

	Drug A N=35	Drug B N=39	All N=74
Sex			
Male	20 (57.1%)	19 (48.7%)	39 (52.7%)
95% confidence interval	[39.4% ; 73.7%]	[32.4% ; 65.2%]	[40.7% ; 64.4%]
Female	15 (42.9%)	20 (51.3%)	35 (47.3%)
95% confidence interval	[26.3% ; 60.6%]	[34.8% ; 67.6%]	[35.6% ; 59.3%]

Sex had two categories. One can imagine the savings obtained when reporting variables with dozen of categories.

3. By processing with more than one Level

3.1 With %table1: Reporting "Age" by "Sex" and "Race".

We want to display this:

Table 03.1-1 Reporting "Age" by "Sex" and "Race"

	Drug A N=35	Drug B N=39	All N=74
Age			
Male			
Caucasian			
N	11	12	23
Mean (std)	51.67 (9.57)	55.63 (10.56)	53.74 (10.07)
Median	54.03	53.73	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
Black			
N	1	2	3
Mean (std)	60.05 (-)	55.95 (13.09)	57.32 (9.55)
Median	60.05	55.95	60.05
Min / max	60.1 / 60.1	46.7 / 65.2	46.7 / 65.2
Asian			
N	8	5	13
Mean (std)	51.34 (8.42)	57.04 (6.85)	53.53 (8.08)
Median	51.03	55.12	53.60
Min / max	39.3 / 67.2	49.6 / 67.6	39.3 / 67.6
Female			
Caucasian			
N	7	9	16
Mean (std)	54.85 (6.23)	55.27 (10.83)	55.09 (8.84)
Median	54.64	57.98	56.31
Min / max	44.4 / 62.2	29.7 / 69.8	29.7 / 69.8
Black			
N	0	4	4
Mean (std)	- (-)	62.30 (5.28)	62.30 (5.28)
Median	-	63.05	63.05
Min / max	- / -	56.0 / 67.1	56.0 / 67.1
Asian			
N	8	7	15
Mean (std)	58.86 (10.50)	61.99 (8.95)	60.32 (9.60)
Median	61.03	63.76	62.11
Min / max	38.0 / 72.4	47.7 / 74.8	38.0 / 74.8

The related statements are:

```
*--- De-Assign format on Sex and Race;
data subjects; set main2.subjects; format Sex Race; run;
*--- getting info;
Proc sql noprint;
    *--- Getting the code list for Sex and Race;
    select distinct sex into: sexlist separated by " "
    from subjects where FAS=1;
    select distinct race into: racelist separated by " "
    from subjects where FAS=1;
    *--- Getting number of codes for Sex and Race;
    select count (distinct sex), count (distinct race)  into: nsex , : nrace
    from main2.subjects where FAS=1;
Quit;
```

Previous comments in 2.1, on all ways to get the code lists apply here as well.

```
%Title(tip 03.1-1 Reporting 'Age' by 'Sex' and 'Race')
%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt.,
        Blocks=
            "Age"/
            %do sx = 1 %to &nsex;
                %*--- Getting the Sex code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %*--- Getting the Sex "decode";
                "%sysfunc(putn(&sex,sex.))" indent=1/
                %do ra = 1 %to &nrace;
                    %*--- Getting the Race code;
                    %let race = %scan(&racelist,&ra,%str( ));
                    %*--- Getting the Race "decode";
                    "___%sysfunc(putn(&race,race.))" indent=2/
                    Age@(Sex=&sex and Race=&race ) " "  type=univ indent=3/
                %end;
            %end;
    )
%mend Temp;

%temp
```

It is recommanded here to use explicite Macro Language statements with indentation. The nested loops on Sex and Race can be better seen and understood. You can obviously permute the nesting as needed and create additional By-levels by piling the loops… The table layout outputed by %table1 will look like the above table: The By-levels labels will be indented from Left to Right until the reporting of the BlockVar (Here the Age). The parameter Mainlines=left or leftnoborder or span may give you few additional degrees of freedom for tuning the layout.

You can also obtained this presentation:

Table 03.1-2 Reporting "Age" by "Sex" and "Race" (Other Layout)

	Drug A N=35	Drug B N=39	All N=74
Age			
At Sex: Male and Race: Caucasian			
N	11	12	23
Mean (std)	51.67 (9.57)	55.63 (10.56)	53.74 (10.07)
Median	54.03	53.73	53.86
Min / max	33.0 / 67.9	36.8 / 71.9	33.0 / 71.9
At Sex: Male and Race: Black			
N	1	2	3
Mean (std)	60.05 (-)	55.95 (13.09)	57.32 (9.55)
Median	60.05	55.95	60.05
Min / max	60.1 / 60.1	46.7 / 65.2	46.7 / 65.2
At Sex: Male and Race: Asian			
N	8	5	13
Mean (std)	51.34 (8.42)	57.04 (6.85)	53.53 (8.08)
Median	51.03	55.12	53.60
Min / max	39.3 / 67.2	49.6 / 67.6	39.3 / 67.6
At Sex: Female and Race: Caucasian			
N	7	9	16
Mean (std)	54.85 (6.23)	55.27 (10.83)	55.09 (8.84)
Median	54.64	57.98	56.31
Min / max	44.4 / 62.2	29.7 / 69.8	29.7 / 69.8
At Sex: Female and Race: Black			
N	0	4	4
Mean (std)	- (-)	62.30 (5.28)	62.30 (5.28)
Median	-	63.05	63.05
Min / max	- / -	56.0 / 67.1	56.0 / 67.1
At Sex: Female and Race: Asian			
N	8	7	15
Mean (std)	58.86 (10.50)	61.99 (8.95)	60.32 (9.60)
Median	61.03	63.76	62.11
Min / max	38.0 / 72.4	47.7 / 74.8	38.0 / 74.8

By typing:

```
%Title(tip 03.1-2 Reporting 'Age' by 'Sex' and 'Race' (Other Layout))
%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt.,
        Blocks=
            "Age"/
            %do sx = 1 %to &nsex;
                %*--- Getting the Sex code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %do ra = 1 %to &nrace;
                    %*--- Getting the Race code;
                    %let race = %scan(&racelist,&ra,%str( ));
                    %*--- Getting the Sex and Race "decodes";
                    Age@(Sex=&sex and Race=&race ) "At Sex: %sysfunc(putn(&sex,sex.)) and Race:
                                                    %sysfunc(putn(&race,race.))"  type=univ indent=1/
                %end;
            %end;
    )
%mend Temp;

%temp
```

But beyond the above variations, if you want to change this layout principle. For instance to get one column per By-level with By-level Blocks spanning over the right one and/or the BlockVar, you may use %table1 and TDK as explained below.

3.2 With %table1, Outprefix= + TDK: Reporting "Age" by "Sex" and "Race".

Outprefix=

The following code:

```
%Title(tip 03.2-1 Outprefix= output)
%Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
    ColVar=Trt " " trt., outprefix=TEST_,
    Blocks=
        Age type=univ /
        Sex type=freq /
)
```

Will display:

Table 03.2 Outprefix= output

	Drug A N=35	Drug B N=39	All N=74
Age at Baseline (years)			
N	35	39	74
Mean (std)	54.12 (9.02)	57.57 (9.48)	55.94 (9.36)
Median	54.80	58.60	55.92
Min / max	33.0 / 72.4	29.7 / 74.8	29.7 / 74.8
Sex			
Male	20 (57.1%)	19 (48.7%)	39 (52.7%)
Female	15 (42.9%)	20 (51.3%)	35 (47.3%)

And will create the following datasets in the SAS Work (As explained in the related AdClin %table1 Documentation) :

- TEST_Block1
- TEST_Block2
- TEST_Block2Sub
- TEST_poplinen

The following proc print will help you to understand the generic structure of an univariate block, a freqency block or of the "line N" storage:

```
proc print data=TEST_Block1; title "TEST_Block1 Content"; run;
proc print data=TEST_Block2; title "TEST_Block2 Content"; run;
proc print data=TEST_Block2Sub; title "TEST_Block2Sub Content"; run;
proc print data=TEST_poplinen; title "TEST_poplinen Content"; run;
```

Proc print outputs: One can see that the tables results can be retrieved from the different outputed blocks.

TEST_Block1 Content

Obs	_N1	_N2	_NA	_Mean1	_Mean2	_MeanA	_Std1	_Std2	_StdA	_Median1
1	35	39	74	54.1151	57.5708	55.9363	9.01729	9.47771	9.36204	54.7981

Obs	_Median2	_MedianA	_Min1	_Min2	_MinA	_Max1	_Max2	_MaxA
1	58.6037	55.9233	32.9856	29.7495	29.7495	72.3559	74.7817	74.7817

Block 1 is an Univariate Block whose structure will always look this way. Each univariate statistics activated by the user (here the default) will be named ***_SASStatName*x** where **x** is the **rank order** of the colvar category or column. **x** = (**1,2,….,n,….,A**) where **A** represent the column All:

_MaxA contain the Max of Age for column All, _Std1 represent the Standard deviation of Age for Column number 1: e.g. Drug A.

It is important to understand that the columns are numbered from 1 to n then A, according to their rank based on the non-formatted categories of colvar.

This strategy allows to cope with either types of colvar (Numeric or Alphanumeric).

TEST_Block2 Content

Obs	_tObs1	_tNotM1	_tObs2	_tNotM2	_tObsA	_tNotMA
1	35	35	39	39	74	74

TEST_Block2Sub Content

Obs	Sex	_Freq1	_PctCol1	_Freq2	_PctCol2	FreqA	_PctColA
1	Male	20	0.57143	19	0.48718	39	0.52703
2	Female	15	0.42857	20	0.51282	35	0.47297

The Block Number 2 is a frequency Block. Each Frequency block will generate 2 datasets: the Block**n** and the Block**nSub.**

Block**n** contains the row totals of the frequency variable.

The same numbering principle applies to either Frequencies or Percentages:

_tobs1 is the total number of records for Sex for column 1: e.g. Drug A., _tNotM2 is the total of non-missing Sex values for column 2: e.g.Drug B.

Block**nSub** has as many records as categories found in the data.

_Freq2 at Sex=Male is the frequency of males in coumn 2: e.g. Drug B., _PctColA at sex=Female is the percentage of females in column A: All.

TEST_poplinen Content

Obs	_tPop1	_tPop2	_tPopA
1	35	39	74

Finally, The poplinen dataset contains the N= values of the table header. Also using the same numbering principle.

When statistical tests are commanded by the user, using the parameter Test=, the test results, depending on the variable type, will be stored either on additional columns of the univariate

*Outprefix*_block**n** or additional columns of the frequency *Outprefix*_block**n**. A Frequency *Outprefix*_block**nSub** never stores statistics.

A usefull macro as follows will be needed:

```
%macro OnetoNList(Ncols=, WithAll=Yes);
    %let OnetoNList = ;
    %do i = 1 %to &Ncols;
        %let OnetoNList = &OnetoNList &i;
    %end;
    %if &withAll=Yes %then %do
        %let OnetoNList = &OnetoNList A;
    %end;
%mend OnetoNList;
```

This macro creates a Macro variable **&OnetoNList** containing a list of values from 1 to Ncols separated by a "Blank" completed with a A if WithAll=Yes (1 2 3 4 A or 1 2 3 …).

Outprefix= + TDK: Reporting "Age" by "Sex" and "Race".

If we re-run the Table 03.1-2 with the outprefix= parameter:

```
%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt., outprefix=ForTDK_,
        Blocks=
            "Age"/
            %do sx = 1 %to &nsex;
                %*--- Getting the Sex code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %do ra = 1 %to &nrace;
                    %*--- Getting the Race code;
                    %let race = %scan(&racelist,&ra,%str( ));
                    %*--- Getting the Sex and Race "decodes";
                    Age@(Sex=&sex and Race=&race ) "At Sex: %sysfunc(putn(&sex,sex.)) and Race:
                                                    %sysfunc(putn(&race,race.))"  type=univ indent=1/
                %end;
            %end;
    )
%mend Temp;

%temp
```

As explained earlier, we will create 1 + 2 X 3 Univariate blocks named:

ForTDK_ Block1 to **ForTDK_**Block7 plus the **ForTDK_**poplinen. One difficulty will consist in not being mixed up by the blocks order.

- Block1 correspond to text Block for "Age" BUT will never be outputed. Numbering will start at 2. text Blocks are never outputed but they impact on the numberig of the outputed blocks.
- Block2 correspond to Block for Sex=1 and Race=1
- Block3 correspond to Block for Sex=1 and Race=2
- Etc…

It might become very complex to retrieve the right information…

We therefore propose to enrich the above Macro Language (in Orange) to keep a more documented track of the block number and their related meaning. Then we trigger a TDK step to display the "Age" by "Sex" and "Race" as expected at the end of § 03.1.

```
%Title(table 03.2-2 Outprefix= + TDK)
%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt., outprefix=ForTDK_,
        Blocks=
            "Age"/
            %let BlkNo = 1;
            %do sx = 1 %to &nsex,
                %*--- Getting the Sex code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %do ra = 1 %to &nrace;
                    %*--- Getting the Race code;
                    %let race = %scan(&racelist,&ra,%str( ));
                    %*--- Getting the Sex and Race "decodes";
                    %let BlkNo = %eval(&BlkNo+1)
                    %let Blk&sex.&race.age = &BlkNo;
                    Age@(Sex=&sex and Race=&race ) "At Sex: %sysfunc(putn(&sex,sex.)) and Race:
                                                    %sysfunc(putn(&race,race.))"  type=univ indent=1/
                %end;
            %end;
    )
```

Text Block "Age" is Block 1

Store Block N° in a Macro Variable whose name relate to context

The First Title is erased by a second one from the TDK step (the same):

```
    %Title(table 03.2-2 Outprefix= + TDK)
    *--- generating One to N list + All;
    %OnetoNList(Ncols=&ntrt)
    data _null_;
      %tstart
      %tr( "Sexe" r=2 ! "Race" r=2 ! "Age" r=2 ! %serlist( "%nrstr(%%)sysfunc(putn(?,trt.))" ha=c
                                                , &trtlist, sep=!) ! "All" )
      set ForTDK_poplinen;
      %tr( %serlist( "N=" _tpop? ha=c, &oneToNList, sep=!) )
      %trline
      %thend
      %*--- loops on Sex X race;
      %do sx = 1 %to &nsex;
          %*--- Getting the Sex code;
          %let sex = %scan(&sexlist,&sx,%str( ));
          %do ra = 1 %to &nrace;
              %*--- Getting the Race code;
              %let race = %scan(&racelist,&ra,%str( ));
              %*--- Getting the Sex and Race "decodes";
              %let BlkNo = &&Blk&sex.&race.age;
              set ForTDK_Block&blkNo;
              %if &race=1 %then %do;
                  %tr( "%sysfunc(putn(&sex,sex.))" r=%eval(&nrace*4) va=c  !
                       "%sysfunc(putn(&race,race.))" r=4 va=c !
                       "N" ! %serlist( _N? ha=c, &oneToNList, sep=! ))
              %end;
              %else %do;
                  %tr( "%sysfunc(putn(&race,race.))" r=4 va=c ! "N" ! %serlist( _N? ha=c, &oneToNList,
                       sep=! )  )
              %end;
              %tr( "Mean (Std)" ! %serlist( _mean? 12.1 "_(" _std? 12.2 ")" ha=c, &oneToNList, sep=! )  )
              %tr( "Median"     ! %serlist( _median? 12.1 ha=c, &oneToNList, sep=! )  )
              %tr( "Min ; Max"  ! %serlist( _min? 12.1 "_;_" _max? 12.1 ha=c, &oneToNList, sep=! )  )
              %trline
          %end;
      %end;
      %tstop
    run;
%mend Temp;

%temp
```

Create List: 1 2 A

List

Template

Retrieve Block N° and set the right block

Condition for Sex and Race Row span.

How By-balues span rows:
Sex: r = 3 races x 4 lines of statistics
Race: r= 4 lines of statistics

Condition for Race Row span

The expexted table of results is:

Table 03.2-2 Outprefix= + TDK

Sexe	Race	Age	Drug A N=35	Drug B N=39	All N=74
Male	Caucasian	N	11	12	23
		Mean (Std)	51.7 (9.57)	55.6 (10.56)	53.7 (10.07)
		Median	54.0	53.7	53.9
		Min ; Max	33.0 ; 67.9	36.8 ; 71.9	33.0 ; 71.9
	Black	N	1	2	3
		Mean (Std)	60.1 (-)	55.9 (13.09)	57.3 (9.55)
		Median	60.1	55.9	60.1
		Min ; Max	60.1 ; 60.1	46.7 ; 65.2	46.7 ; 65.2
	Asian	N	8	5	13
		Mean (Std)	51.3 (8.42)	57.0 (6.85)	53.5 (8.08)
		Median	51.0	55.1	53.6
		Min ; Max	39.3 ; 67.2	49.6 ; 67.6	39.3 ; 67.6
Female	Caucasian	N	7	9	16
		Mean (Std)	54.9 (6.23)	55.3 (10.83)	55.1 (8.84)
		Median	54.6	58.0	56.3
		Min ; Max	44.4 ; 62.2	29.7 ; 69.8	29.7 ; 69.8
	Black	N	0	4	4
		Mean (Std)	- (-)	62.3 (5.28)	62.3 (5.28)
		Median	-	63.0	63.0
		Min ; Max	- ; -	56.0 ; 67.1	56.0 ; 67.1
	Asian	N	8	7	15
		Mean (Std)	58.9 (10.50)	62.0 (8.95)	60.3 (9.60)
		Median	61.0	63.8	62.1
		Min ; Max	38.0 ; 72.4	47.7 ; 74.8	38.0 ; 74.8

In order to better understand the %serlist usage you may cut and paste some of the above typed %serlist within a **%put** …… **;** and run it, and look at the generated code.

You may have noticed that the amount of code will remain unchanged if treatment had 30 categories ! thanks to he %serlist macro.

Managing more than 2 By-variables can easily be extrapolated from our example.

We recommand to use the validated macro %table1 to compute almost everything and to store almost any kind of statistics. Then, **only if the layout does not fulfil the requirements**, you use the TDK just to reformat the already calculated results stored thanks to **OutPrefix=**.

4. By processing with more than one Level + Univariate Statistics in columns

4.1 Example 1

To illustrate this figure case we are going to report "Age" and "Compliance" (of Type=univ) by "Sex" and "Race". We want "N", "Mean (Std)" and "Min ; Max" to be reported in columns nested within the treatment code. The All column will be dropped. But a T-test will added in a Test column.

We expect the table bellow:

Table 04.1-1 By processing with more than 2 level + Statistics in columns (1)

Sexe	Race	Parameter	Drug A			Drug B			Drug A / Drug B Comparison T-Test
			N	Mean (Std)	Min ; Max	N	Mean (Std)	Min ; Max	
Male	Caucasian	Age	11	51.7 (9.57)	33.0 ; 67.9	12	55.6 (10.56)	36.8 ; 71.9	Pr>T= 0.359
		Compliance	11	1.0 (0.05)	0.8 ; 1.0	12	1.0 (0.03)	0.9 ; 1.0	Pr>T= 0.352
	Black	Age	1	60.1 (-)	60.1 ; 60.1	2	55.9 (13.09)	46.7 ; 65.2	Pr>T= 0.840
		Compliance	1	0.9 (-)	0.9 ; 0.9	2	0.8 (0.24)	0.7 ; 1.0	Pr>T= 0.791
	Asian	Age	8	51.3 (8.42)	39.3 ; 67.2	5	57.0 (6.85)	49.6 ; 67.6	Pr>T= 0.231
		Compliance	8	1.0 (0.03)	0.9 ; 1.0	5	1.0 (0.01)	1.0 ; 1.0	Pr>T= 0.078
Female	Caucasian	Age	7	54.9 (6.23)	44.4 ; 62.2	9	55.3 (10.83)	29.7 ; 69.8	Pr>T= 0.928
		Compliance	7	1.0 (0.04)	0.9 ; 1.0	9	0.9 (0.11)	0.7 ; 1.0	Pr>T= 0.449
	Black	Age	0	- (-)	- ; -	4	62.3 (5.28)	56.0 ; 67.1	Pr>T= .
		Compliance	0	- (-)	- ; -	4	1.0 (0.01)	1.0 ; 1.0	Pr>T= .
	Asian	Age	8	58.9 (10.50)	38.0 ; 72.4	7	62.0 (8.95)	47.7 ; 74.8	Pr>T= 0.548
		Compliance	8	1.0 (0.08)	0.8 ; 1.0	7	1.0 (0.02)	0.9 ; 1.0	Pr>T= 0.421

The statements below have been colored according to the same logic as the previous example. In addition, the above table colors have been matched to the related statements below:

```
%OnetoNList(Ncols=&ntrt,withall=No)
%Title(table 04.1-1 By processing with more than 2 level + Statistics in columns (1))
%macro Temp;
    %Table1( Popdataset=main2.subjects, PopId=subjectId, popfilter=FAS=1,
        ColVar=Trt " " trt., outprefix=ForTDK_,
        Blocks=
            %let BlkNo = 0;
            %do sx = 1 %to &nsex;
                %*--- Getting the Sex code;
                %let sex = %scan(&sexlist,&sx,%str( ));
                %do ra = 1 %to &nrace;
                    %*--- Getting the Race code;
                    %let race = %scan(&racelist,&ra,%str( ));
                    %let BlkNo = %eval(&BlkNo+1);
                    %let Blk&sex.&race.age = &BlkNo;
                    Age@(Sex=&sex and Race=&race ) "Age At Sex: &sex and Race: &race"  type=univ test=T/
                    %let BlkNo = %eval(&BlkNo+1);
                    %let Blk&sex.&race.compliance = &BlkNo;
                    Compliance@(Sex=&sex and Race=&race ) "Compliance At Sex: &sex and Race: &race"
                                type=univ test=T/
                %end;
            %end;
    )

    %Title(table 04.1-1 By processing with more than 2 level + Statistics in columns (1))

    data _null_;
        %tstart
        %tr( "Sexe"r=2 ! "Race" r=2 ! "Parameter" r=2 !
              %serlist( "%nrstr(%%)sysfunc(putn(?,trt.))" ha=c c=3 , &trtlist, sep=!) !
             "Drug A / Drug B§Comparison§T-Test" r=2)
```

```
        %trline
        %tr( %serlist( "N" ha=c ! "Mean (Std)" ha=c ! "Min ; Max" ha=c , 1 2 , sep=!) )
        %trline
        %thend
        %*--- loops on Sex X race;
        %do sx = 1 %to &nsex;
            %*--- Getting the Sex code;
            %let sex = %scan(&sexlist,&sx,%str( ));
            %do ra = 1 %to &nrace;
                %*--- Getting the Race code;
                %let race = %scan(&racelist,&ra,%str( ));
                %*--- Getting the Sex and Race "decodes";
                %if &race=1 %then %do;
                    %let BlkNo = &&Blk&sex.&race.age;
                    set ForTDK_Block&blkNo;
                    %tr("%sysfunc(putn(&sex,sex.))" r=%eval(&nrace*2) va=c  !
                        "%sysfunc(putn(&race,race.))" r=2 va=c ! "Age" !
                         %serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c !
                        _min? 12.1 "_;_" _max? 12.1 ha=c , &oneToNList, sep=! ) ! "Pr>T= " T_Prob Pvalue6.3)
                    %trline
                    %let BlkNo = &&Blk&sex.&race.compliance;
                    set ForTDK_Block&blkNo;
                    %tr(  "Compliance" ! %serlist( _N? ha=c !
                          _mean? 12.1 "_(" _std? 12.2 ")" ha=c ! _min? 2.1 "_;_" _max? 12.1 ha=c ,
                         &oneToNList , sep=! ) ! "Pr>T= " T_Prob pvalue6.3)
                %end;
                %else %do;
                    %let BlkNo = &&Blk&sex.&race.age;
                    set ForTDK_Block&blkNo;
                    %tr("%sysfunc(putn(&race,race.))" r=2 va=c ! "Age" !
                         %serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c !
                         _min? 12.1 "_;_" _max? 12.1 ha=c, &oneToNList, sep=! ) ! "Pr>T= " T_Prob pvalue6.3)
                    %trline
                    %let BlkNo = &&Blk&sex.&race.compliance;
                    set ForTDK_Block&blkNo;
                    %tr( "Compliance" ! %serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c !
                    _min? 12.1 _;_" _max? 12.1 ha=c, &oneToNList, sep=! ) ! "Pr>T= " T_Prob pvalue6.3)
                %end;
                %trline
            %end;
        %end;
        %tstop
    run;
%mend Temp;

%temp
```

This time, %table1 is used to compute statistics on "Age" and "Compliance" by "Sex" and "Race". The %table1 code has been lightened since it is not the final output. The block numbering is carefully managed as already explained. Then, the TDK step retrieves the stored Information from the **outprefix=ForTDK_** parameter and reformat the results.

The statement:

```
%serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c ! _min? 12.1 "_;_" _max? 12.1 ha=c , &oneToNList,
          sep=! )
```

allows to display statistics within the treatment category.

4.2 Example 2

This example is taken from the data structure below:

Dataset Subject:

subid	trt	itt
0101	2	1
0102	1	1
0103	2	1
0104	2	1
0105	2	1
0106	2	1
...	...	...

Dataset SubRLV:

subid	reader	lesno	reading	cs
0101	1	1	1	3
0101	1	1	2	-
0101	1	1	3	-
0101	1	2	1	1
0101	1	2	2	-
0101	1	2	3	-
0101	1	3	1	-
0101	1	3	2	1
0101	1	3	3	1
0101	2	2	1	1
0101	2	2	2	-
0101	2	2	3	-
0101	2	3	1	-
0101	2	3	2	1
0101	2	3	3	1
0102	1	1	1	2
0102	1	1	2	3
0102	1	1	3	2
0102	1	2	1	1
0102	1	2	2	3
...	...	...	...	...

The table to produced is this one:

Table 04.1-2 By processing with more than 2 level + Statistics in columns (2)

Contrast Score										
Reader	Reading	Drug A			Drug B			All		
		N	Mean (Std)	Min ; Max	N	Mean (Std)	Min ; Max	N	Mean (Std)	Min ; Max
1	1	35	1.7 (0.75)	0.0 ; 3.0	20	1.6 (0.82)	0.0 ; 3.0	55	1.7 (0.77)	0.0 ; 3.0
	2	33	2.2 (0.80)	0.0 ; 3.0	22	2.1 (0.92)	1.0 ; 3.0	55	2.1 (0.84)	0.0 ; 3.0
	3	41	2.2 (0.65)	1.0 ; 3.0	19	2.3 (0.82)	1.0 ; 3.0	60	2.3 (0.70)	1.0 ; 3.0
2	1	21	1.6 (0.58)	1.0 ; 3.0	16	1.4 (0.63)	1.0 ; 3.0	37	1.5 (0.60)	1.0 ; 3.0
	2	24	2.1 (0.58)	1.0 ; 3.0	18	1.9 (0.80)	1.0 ; 3.0	42	2.1 (0.68)	1.0 ; 3.0
	3	23	2.2 (0.59)	1.0 ; 3.0	18	2.0 (0.91)	1.0 ; 3.0	41	2.1 (0.74)	1.0 ; 3.0
3	1	32	1.4 (0.56)	0.0 ; 2.0	17	1.4 (0.61)	0.0 ; 2.0	49	1.4 (0.57)	0.0 ; 2.0
	2	34	2.0 (0.68)	1.0 ; 3.0	23	2.0 (0.80)	1.0 ; 3.0	57	2.0 (0.72)	1.0 ; 3.0
	3	32	1.7 (0.52)	1.0 ; 3.0	21	1.9 (0.73)	1.0 ; 3.0	53	1.8 (0.61)	1.0 ; 3.0

With this code:

```
*--- Info on By-levels and Treatment can be given by %let= ;
%let trtlist     = 1 2;
%let readerlist  = 1 2 3;
%let readinglist = 1 2 3;
%let ntrt        = 2;
%let nreader     = 3;
%let nreading    = 3;
%OnetoNList(Ncols=&ntrt)

%Title(table 04.1-2 By processing with more than 2 level + Statistics in columns (2))
%macro Temp;
    %Table1( Popdataset=main1.subject, PopId=subId, popfilter=ITT=1,
        ColVar=Trt " ", outprefix=ForTDK_,
        vardataset=main1.subrlv,
        Blocks=
            %let BlkNo = 0;
            %do rx = 1 %to &nreader;
                %*--- Getting the Reader;
                %let Reader = %scan(&readerlist,&rx,%str( ));
                %do ry = 1 %to &nreading;
                    %*--- Getting the Reading;
                    %let reading = %scan(&readinglist,&ry,%str( ));
                    %let BlkNo = %eval(&BlkNo+1);
                    %let Blk&reader.&reading.cs = &BlkNo;
                    cs@(Reader="&reader" and Reading=&reading ) "Contrast Score At Reader: &Reader and
                                Reading: &reading"  type=univ/
                %end;
            %end;
    )

   %Title(table 04.1-2 By processing with more than 2 level + Statistics in columns (2))
    data _null_;
        %tstart
        %tr( "Contrast Score" c=11 )
        %trline
        %tr( "Reader" r=2 ! "Reading" r=2 !  %serlist( "%nrstr(%%)sysfunc(putn(?,trt.))" ha=c c=3, &trtlist,
                                                    sep=!) ! "All" c=3)
        %trline
        %tr( %serlist( "N" ha=c ! "Mean (Std)" ha=c ! "Min ; Max" ha=c , &oneToNList , sep=!) )
        %trline
        %thend
        %*--- loops on Reader X Reading;
        %do rx = 1 %to &nreader;
            %*--- Getting the Reader code;
            %let reader = %scan(&readerlist,&rx,%str( ));
```

```
            %do ry = 1 %to &nreading;
                %*--- Getting the Reading code;
                %let reading = %scan(&readinglist,&ry,%str( ));
                %let BlkNo = &&Blk&reader.&reading.cs;
                set ForTDK_Block&blkNo;
                %if &reading=1 %then %do;
                    %tr( "&Reader" r=&nreading va=c  ! "&reading" va=c !
                          %serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c !
                                    _min? 12.1 "_;_" _max? 12.1 ha=c, &oneToNList, sep=! ) )
                %end;
                %else %do;
                    %tr( "&Reading" va=c !
                          %serlist( _N? ha=c ! _mean? 12.1 "_(" _std? 12.2 ")" ha=c !
                                    _min? 12.1 "_;_" _max? 12.1 ha=c, &oneToNList, sep=! ) )
                %end;
                %trline
            %end;
        %end;
        %tstop
   run;
%mend Temp;

%temp
```