***Horizontal tools***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

<https://www.spsstools.net/en/KO-spssmacros>

All rights reserved

*Some horizontal operations.* Collection of macros performing some wanted things (such as standardization, sorting, ranking, binning or counting up frequncies or unique values) within cases, horizontally.

*Read “*[*About SPSS macros*](https://www.spsstools.net/en/KO-aboutmacros)*” what are they and how to run them.*

*The “Protected directory” error.* Some of the macros described in the current document write temporary files to hard disc. If you don't have full Administrator rights of your computer, it may cause error saying, among things: *“SPSS Statistics cannot access a file... specifies a protected directory...”*, meaning that the default directory the macro wants to use is protected on your PC. To solve the problem, in Syntax window issue command: CD 'myfolder'., where 'myfolder' is the path/name of some folder where you are allowed to save files to.

* For horizontal rescaling (standardizing etc.) use [!KO\_HRESC](#_MACRO_!KO_HRESC:_HORIZONTAL).
* For horizontal ranking use [!KO\_HRANK](#_MACRO_!KO_HRANK:_HORIZONTAL).
* For horizontal sorting or randomization use [!KO\_HSORT](#_MACRO_!KO_HSORT:_HORIZONTAL), and also [!KO\_HQSORT](#_MACRO_!KO_HQSORT:_QUICK).
* For horizontal binning of quantitative data into equal-frequency categories – [!KO\_HBINEF](#_MACRO_!KO_HBINEF:_HORIZONTAL).
* For horizontal binning of quantitative data into equal-interval categories – [!KO\_HBINEI](#_MACRO_!KO_HBINEI:_HORIZONTAL).
* To count horizontally frequencies of values – [!KO\_HFREQ](#_MACRO_!KO_HFREQ:_FREQUENCIES,).
* To count within cases the diversity and repetition of values – [!KO\_HCOUNT](#_MACRO_!KO_HCOUNT:_HORIZONTAL).
* To measure gaps (differences) among values in their distribution – [!KO\_HGAPS](#_МАКРОС_!KO_HGAPS:_STATS).

These macros, except !KO\_HFREQ, don’t use restructuring or transposing of the dataset. They are based on transformation commands.

# MACRO !KO\_HRESC: HORIZONTAL RESCALING

Version 2, Dec 2020 (Version 1, May 2009). Tested on SPSS Statistics 20, 22, 25.

!KO\_hresc vars= *v1 v3 to v10* /\*Numeric variables (name-by-name and/or via “to”) which values to

/\*rescale horizontally (at least two)

/cap= *'tr\_'* /\*Prefix (may quote) for output variables names; if not specified,

/\*VARS themselves will be transformed

/trans= Z /\*Transformation: Z, C, SS, SUM, RANGE, RESCALE, EXP, CLR, ZABS,

/\*DMED, SMED1, SMED2 (see)

/discret= /\*Optional: discretize output values: RND d or TRUNC d

/missing= /\*Take user-missing values in processing (INCLUDE, possible only with CAP

/\*not specified) or don’t take (EXCLUDE, default)

/nvalid= *7* /\*Optional: empty a case with the number of valid values less than the threshold

/\*(specify positive integer)

/flag= YES /\*If CAP specified: count valid values and flag constant cases:

/\*YES or NO (default).

Minimal specification VARS, TRANS.

Rescales values of numeric variables horizontally, that is, within cases. Under the umbrella word “rescaling”, various types of centering, standardizing, or normalizing are meant here. You may request to output the transformed data as new variables or to replace the input variables themselves.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hresc vars= v1 to v10 /cap= z\_ /trans= Z.

* Variables’ V1 – V10 data are being standardized horizontally, yielding variables Z\_1 – Z\_10 with Z-scores.

***Subcommands***

**VARS**

Specify numeric variables (two minimum) which values to transform horizontally. Name-by-name list and/or using “to” convention. If the variables are too many, using “to” is recommended: this can spare the run time. The data are perceived as quantitative (not nominal).

**CAP**

Prefix into the names of output variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). Output variables will be named by the prefix and an index from 1 to the number of input variables. There must be no variables with these names in the input dataset.

If not specify or omit CAP subcommand, input variables VARS themselves will be the output ones: all values in them will be replaced, and any user-missing definitions abolished. Value labels will remain former, but will become irrelevant since the variables’ values will come new.

**TRANS**

Specify the needed data transform, that will be done individually within each vector (“vector” is a case of the dataset, variables VARS).

Z - **z-standardization**. From value, the vector mean is subtracted, and the result is divided by the vector standard deviation (computed on “df=n-1”). As the result, in the vector the mean will be 0 and the standard deviation 1. If the vector was a constant, the macro will make it zero vector.

C - **centration** (deviation from mean). From value, the vector mean is subtracted. As the result, in the vector the mean will be 0.

SS - **normalizing to sum of squares 1**. Value is divided by the square root of the sum of squares in the vector. As the result, in the vector the sum-of-squares will become 1. If the vector was zero constant, the macro will leave it as is.

SUM - **normalizing to sum 1**. This option requires nonnegative data. Value is divided by the sum in the vector. As the result, the sum in the vector will become 1. If the vector was zero constant, the macro will leave it as is.

RANGE - **taking to range of 1**. Value is divided by the range in the vector. As the result, the range in the vector will become 1. If the vector was a constant, the macro will leave it as is.

RESCALE - **rescaling to range [0,1]**. From value, the vector minimum is subtracted, and the result is divided by the range in the vector. As the result, the vector values will have range [0,1]. If the vector was a constant, the macro will make it constant 0.5.

ZABS - **standardization with average absolute deviation**. Value is translated to deviation from mean, and is divided by the mean of absolute magnitudes of these deviations. As the result, in the vector the mean will be 0 and the average absolute deviation around mean will be 1. If the vector was a constant, the macro will make it zero vector.

DMED - **deviation from median**. From value, the vector median is subtracted. As the result, in the vector the median will be 0.

SMED1 - **median standardization with average absolute deviation around median**. Value is translated to deviation from median, and is divided by the mean of absolute magnitudes of these deviations. As the result, in the vector the median will be 0 and the average absolute deviation around median will be 1. If the vector was a constant, the macro will make it zero vector.

SMED2 - **median standardization with median absolute deviation around median**. Value is translated to deviation from median, and is divided by the median of absolute magnitudes of these deviations. As the result, in the vector the median will be 0 and the median absolute deviation around median will be 1. If the vector was a constant, the macro will make it zero vector.

EXP - nonlinear transform **exponential normalization**, or **softmax**. It is normalizing to sum of 1 done after exponentiation of values (*e*val). As the result, the vector values will be nonnegative and their sum will be 1.

CLR - nonlinear transform **centered logratio transform**. This option requires positive data. First, values of the vector are normalized to sum of 1. In the obtained vector, values are divided by the geometric mean in it, and logarithm is taken. As the result, the sum in the vector will be 0.

Keep in mind that some of these transformations require nonnegative or positive input data. The macro does not check at input whether data suit. Input cases that are constants – you can trace them by subcommand FLAG (see).

**DISCRET**

Optional subcommand to discretize (bin) values after the transform. Specify keyword RND or TRUNC, meaning the method – rounding or truncation, and after it – the figure 0, 1, 2, or 3, meaning the number of decimal digints to retain in a value.

**MISSING**

How to treat user-missing values in the input variables: don’t take them in transformation, consider them as system-missing (MISSING=EXCLUDE, also default/unspecifying), or take as valid values (MISSING=INCLUDE, allowed only with CAP s/c not specified).

**NVALID**

You may specify the bottom limit for the number of valid values in a case, positive integer. If there happens less valid values than that threshold, the case does not enter processing, but it will be empty (sysmis) in the variables output from the macro. If MISSING=INCLUDE, user missings count as valid values.

By default, NVALID=1 for all TRANS except Z. For TRANS=Z, default is NVALID=2 (and NVALID=1 is considered as NVALID=2).

**FLAG**

With CAP specified, FLAG=YES creates or updates in the dataset variable *NVALID#$* showing the number of valid values in each case, at input. Besides, FLAG=YES creates or updates in the dataset variable *CONSTA#$* marking constant cases, at input. Constants equal to zero get flagged by code 0, and other constants – by code 1.

FLAG=YES is not permitted with CAP not specified. FLAG does not react to s/c NVALID.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

# MACRO !KO\_HRANK: HORIZONTAL RANKING

Version 2, Nov 2020 (Version 1, Jul 2004). Tested on SPSS Statistics 20, 22, 25.

!KO\_hrank vars= *v1 to v10* /\*Numeric variables (adjacent, via “to”) which values to rank

/\*horizontally

/cap= *rnk* /\*Prefix (may quote) for output variables names; if not specified,

/\*VARS themselves will be ranked

/trans= RANK /\*Transformation: ranks (RANK), normal scores (NORMAL),

/\*percentile groups (NTILES k)

/order= D /\*Order of ranking: ascending (A, default), descending (D)

/ties= /\*Treatment of equal values: average ranks (MEAN, тж п/у),

/\*give lower rank (LOW), give upper rank (HIGH),

/\*condense ranks (CONDENSE), give consecutive ranks (GRADE)

/fraction= /\*With TRANS=NORMAL, way to estimate proportion: BLOM (default), TUKEY,

/\*RANKIT, VW (see)

/missing= /\*Take user-missing values in processing (INCLUDE) or don’t take (EXCLUDE, default)

/nvalid= *8* /\* Optional: empty a case with the number of valid values less than the threshold

/\*(specify positive integer)

/flag= YES /\*If CAP specified: count valid values: YES or NO (default).

Minimal specification VARS, TRANS.

Converts values of numeric variables into ranks, horizontally, that is, ranks data within cases. You may request to output the ranks as new variables or to replace the input variables themselves. You can treat equal values (ties) in various ways. The macro can also recalculate data onto normal distribution (normal scores) or do binning of the data in k equally-sized groups (percentile groups). This macro does the same as command RANK does, only does it “within cases” rather than “within variables”.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hrank vars= v1 to v10 /cap= rnk /trans= RANK.

* Variables’ V1 – V10 data are being ranked horizontally, yielding variables RNK1 – RNK10 with the ranks.

***Subcommands***

**VARS**

Specify via “to” numeric variables which values to rank horizontally. Those must be consecutive variables in the dataset. The data are perceived as quantitative (not nominal).

**CAP**

Prefix into the names of output variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). Output variables will be named by the prefix and an index from 1 to the number of input variables. There must be no variables’ with these names in the input dataset.

If not specify or omit CAP subcommand, input variables VARS themselves will be the output ones: all values in them will be replaced, and any user-missing definitions abolished. Value labels will remain former, but will become irrelevant since the variables values will come new.

**TRANS**

Specify the needed data transform (that will be done individually within each case of the dataset):

RANK - rank values – convert them to ranks.

NORMAL - having done ranking, translate then ranks into normal scores, that is, to values which would have been in the input data if they had been from standard normal distribution.

NTILES *k* - having done ranking, group then ranks by their magnitude into *k* approximately equally-sized (by frequency) groups. It is binning (quantizing) of an observed distribution of quantitative data by the method of percentile groups. Specify positive integer not less than 2. For example, NTILES 4 will assign code 1 to values up to the 25th percentile, code 2 to values between the 25th and the 50th percentile, code 3 to values between the 50th and the 75th percentile, code 4 to values beyond the 75th percentile. It is expected that *k* is not great – less than the number of unique values in the distribution, i.e. in the case of the dataset.

All three transforms are done identically (same formulas) to how command RANK does them (see SPSS Statistics Algorithms).

For technical reasons, the macro does not do transforms NORMAL and NTILES in case of TIES=GRADE.

Another macro which can fulfil the same task that the option NTILES does – is [!KO\_HBINEF](#_MACRO__!KO_HBINEF:). The latter uses somewhat different approach, direct calculation of percentiles by the “Aempirical” method (instead of ranking done in !KO\_HRANK). Categorization results by !KO\_HBINEF and !KO\_HRANK coincide quite often. !KO\_HBINEF is somewhat faster.

**TIES**

How to process equal (tied) values during ranking. Because ranking is performed with any TRANS, setting TIES affects all those types of transform.

MEAN - (also default/unspecifying) assign them their mean rank.

LOW - assign them their lower rank.

HIGH - assign them their higher rank.

CONDENSE - assign them their lower rank and squeeze ranks: the next rank will differ by 1 from this one (in Rank Cases menu, this method is designated “Sequential ranks for unique values”).

GRADE - ties receive different ranks, consecutive integers.

Illustration of these variants (ranking is ascending order):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Value* | *3* | *6* | *6* | *7* | *9* |
| GRADE | 1 | 2 | 3 | 4 | 5 |
| MEAN | 1 | 2.5 | 2.5 | 4 | 5 |
| HIGH | 1 | 3 | 3 | 4 | 5 |
| LOW | 1 | 2 | 2 | 4 | 5 |
| CONDENSE | 1 | 2 | 2 | 3 | 4 |

The maximal rank for all the variants except CONDENSE always equals the number of valid values in the case.

**ORDER**

Direction of ranking: ascending, i.e. the greater rank to the greater value (ORDER=A, also default/unspecifying) or descending: the greater rank to the smaller value (ORDER=D).

**FRACTION**

This subcommand serves TRANS=NORMAL. It sets the method of estimation, in a continuous distribution, of the cumulative proportion corresponding to a specific rank.

BLOM - (also default/unspecifying) Blom’s method.

TUKEY - Tukey’s method.

RANKIT - the rankit method.

VW - Van der Waerden method.

**MISSING**

How to treat user-missing values in the input variables: don’t take them in ranking, consider them as system-missing (MISSING=EXCLUDE, also default/unspecifying), or take as valid values (MISSING=INCLUDE).

**NVALID**

You may specify the bottom limit for the number of valid values in a case, positive integer. If there happens less valid values than that threshold, the case does not enter processing, but it will be empty (sysmis) in the variables output from the macro. If MISSING=INCLUDE, user missings count as valid values. By default, NVALID=1.

**FLAG**

With CAP specified, FLAG=YES creates or updates in the dataset variable *NVALID#$* showing the number of valid values in each case, at input. If MISSING=INCLUDE, user missings count as valid values.

FLAG=YES is not permitted with CAP not specified. FLAG does not react to s/c NVALID.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

# MACRO !KO\_HSORT: HORIZONTAL SORTING OR RANDOMIZATION

Version 2, Dec 2020 (Version 1, Mar 2005). Tested on SPSS Statistics 20, 22, 25.

!KO\_hsort vars= *v1 to v10* /\*Numeric variables (adjacent, via “to”) which values to sort or

/\*randomize within cases

/cap= *s\_*  /\*Prefix (may quote it) into the names of output variables; if omit, VARS

/\*themselves will be sorted (randomized)

/cases= FIRST /\*Sort (randomize) within all cases independently (ALL, default); or

/\*sort (randomize) just within the 1st case, the other being caught up (FIRST)

/selvar= *sel=1* /\*Optionally: variable to select cases to be processed by the macro:

/\*name and condition

/order= /\*Sort or randomize: sort ascending (A, default), descending (D), randomize (R)

/missing= /\*Take user-missing values into processing (INCLUDE) or not to take

/\*(EXCLUDE, default)

/misspos= ORIG /\*With sorting: missing values to sort out to the end (TAIL, default) or keep

/\*in their place (ORIG)

/redupl= /\*With sorting: keep reduplicating values at the sort (LEAVE, default) or

/\*remove the duplicates (CROP).

Minimal specification VARS.

Sorts or randomizes values of numeric variables horizontally, that is, within cases. You may request to output new variables or to replace the input variables themselves. A subset of cases can be excluded from that value permutation procedure; and of the included cases it is possible, by wish, to sort (randomize) only one case, the other cases being caught up along with it. The macro does not do nested sorting (i.e., sorting of a case within identical values of another case).

Sorting algorithm used by !KO\_HSORT is moderately fast. To sort horizontally a great number of long vectors (i.e., many cases and large number of variables VARS) use faster macro [!KO\_HQSORT](#_MACRO_!HQSORT:_QUICK).

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hsort vars= v18.1 to v18.9 /order= A /cap= 's18.'.

!KO\_hsort vars= v18.1 to v18.9 /order= A /cap= 's18\_' /cases= FIRST.

* In the first run, values of variables V18.1 – V18.9 are sorted left to right ascendingly. That is done for each case of the dataset. Output variables: S18.1 – S18.9.
* In the second run on the same variables, values of only the first case in the dataset are sorted. The rest of cases permute their data following the first case. Output variables: S18\_1 – S18\_9.

***Subcommands***

**VARS**

Specify via “to” numeric variables which values to sort (or randomize) horizontally. Those must be consecutive variables in the dataset.

**CAP**

Prefix into the names of output variables with permuted (reordered) values. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). Output variables will be named by the prefix and an index from 1 to the number of input variables. There must be no variables with these names in the input dataset.

If not specify or omit CAP subcommand, input variables VARS themselves will be the output ones: values in them will be permuted, and any user-missing definitions abolished (but: see s/c SELVAR). Value labels will remain former, but may become irrelevant since the variables values will be horizontally permuted.

**SELVAR**

By default, all cases of the dataset are selected to be affected by the macro. With this subcommand, you may indicate one variable, existing in the dataset but not from VARS list, selecting cases to process by the macro. After the variable name, specify one simple condition built of a relational operator (=, <>, >, <, >=, <=, EQ, NE, GT, LT, GE, LE) and a valid value (or a name of some other variable in place of a value). Examples of permitted specifications: SELVAR= *REGION=2*; or: SELVAR= *HEIGHT>26.8*; or: SELVAR= *X<>Y*. You may use parentheses if you want, e.g.: SELVAR= *(REGION=2)*.

Only those cases will be processed by the macro that satisfy the condition. Besides that, if there are missing values in the variable indicated in SELVAR s/c, those cases will come out empty (system missing) in the output variables.

If s/c SELVAR is specified but CAP is not, the macro will not abolish user-missing definitions, if any in VARS variables, at the output, and will notify of it.

EXAMPLE 2.

!KO\_hsort vars= q4 to q21 /selvar= region<>5.

!KO\_hsort vars= q4 to q21 /selvar= region=5 /missing= INCLUDE /redupl= CROP.

* The first command sorts values of cases which satisfy “REGION is not 5”.
* The second command sorts values of cases which satisfy “REGION is 5”. Also, user-missing values are taken as valid data, while duplicates among values are eliminated.

**CASES**

This subcommand indicates which of the cases affected (processed) by the macro are to be *value-permuted* – i.e., their values will be sorted (or randomized) left to right, – and which are to be *caught up* into the permutation. One of two keywords:

ALL - (also default/unspecification of the s/c) permuted, independently of each other, will be all dataset cases affected: in all of them values will be sorted (or randomized) left to right. No caught up cases will be.

FIRST - permuted case is single – the first in the dataset, while the rest of the affected cases will be caught up by the permutation. In other words, this is the permutation of *variables* (data columns only, without permuting variable names) following the reordering of values in the first case of the dataset.

Under sorting (ORDER= A or D) with CASES=FIRST, if the first case has missing values, only system-missing values will come out under them in the output data. In other words, the missing sorted value cannot catch up values in other cases.

**ORDER**

Type of permuting: sort left to right by ascending of values (ORDER=A, default/unspecification), sort left to right by descending of values (ORDER=D), or randomize value order (ORDER=R)[[1]](#footnote-1).

For sorting (ORDER= A or D), the macro uses a stable Selection algorithm of sorting. A “stable” sorting algorithm means that the order among identical (repeating) values will be preserved. The stability becomes apparent in CASES=FIRST. For instance, if in the first case some variable *X*=2 and some variable *Y*=2, and *X* is found more left in the dataset than *Y*, a stable sort algorithm will put the contents of column *X* guaranteed more left than the *Y*’s column contents.

**MISSING**

How to treat user-missing values in the input variables in the *value-permuted* cases: consider them missing, like system-missing (MISSING=EXCLUDE, also default/unspecifying), or take them as valid values (MISSING=INCLUDE).

**MISSPOS**

This subcommand is inactive under ORDER=R. It allows to request missing values in the *value-permuted* cases to be kept on their original places (MISSPOS=ORIG), rather than to be sorted to the tail – after valid values (MISSPOS=TAIL, also default/unspecifying). Speaking differently, with MISSPOS=ORIG, a cell with originally missing value won’t be able to contain a valid value at output, such a cell will be system-missing.

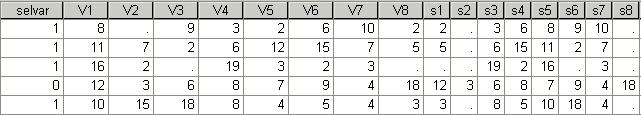
**REDUPL**

This additional option (inactive under ORDER=R) is the sifting off of repeating values, at the sorting. With REDUPL=CROP, if in a *value-permuted* case some value encounters more than once, this value will encounter only once in that case at output. By default/unspecifying and with REDUPL=LEAVE, no elimination of reduplicates of values will take place.

In sorting under CASES=FIRST and REDUPL=CROP, the column of caught-up values that is located in the output data under a value that was cropped (bereaved of duplicates), corresponds to the *last* (rightmost) encounter of the value in the input data.

EXAMPLE 3.

!KO\_hsort vars= v1 to v8 /order= A /cap= s /cases= FIRST /selvar= selvar=1 /misspos= ORIG /redupl= CROP.



* Variables V1 – V8 values to be sorted left to right ascendingly. Only the first case will be sorted, while the other – be caught up by the sorting (CASES=FIRST). Output data are new variables S1 – S8.
* According to variable SELVAR, the 4th case is excluded from the process at all: in variables S1 – S8 its data therefore are not moved comparing to the input variables.
* In the case being sorted (the first one) there is missing value in the second variable. Since MISSPOS=ORIG, in the sorted data of this case that missing remained on its place rather than went to the tail (the right side).
* In the tail (S8), there a missing occurred in the first case for a different reason: it was requested to crop duplicating values (REDUPL=CROP). Value 2 encountered twice in the sorted case: and now it is only one.
* In variable S1 the cropped value, 2, caught up data which was under the last (second) value 2 in the input data (V8, and not V5).
* Missing values in the first case are not accompanied by any caught-up values: columns S2 and S8 are empty.

***Special regimes***

The macro does not respond to weightedness, splitness of the dataset. It ignores temporary (under TEMPORARY) operations. The macro does not obey commands FILTER or USE: to select cases for the procedure, use subcommand SELVAR of the macro. Be wary to run, before the macro, commands removing cases from dataset (such as SELECT IF or SAMPLE), when you are setting CASES=FIRST, for there might happen that the first case in the dataset will be not the one you are thinking of.

# MACRO !KO\_HQSORT: QUICK HORIZONTAL SORTING

Version 2, Dec 2020 (Version 1, Dec 2007). Tested on SPSS Statistics 20, 22, 25.

!KO\_hqsort vars= *v1 to v1000* /\*Numeric variables (adjacent, via “to”) which values to

/\*sort within cases

/cap= *s\_* /\*Prefix (may quote it) into the names of output variables; if omit, VARS

/\*themselves will be sorted

/selvar= /\*Optionally: variable to select cases to be processed by the macro:

/\*name and condition

/order= /\*Sort ascending (A, default) or descending (D)

/redupl= /\*Keep reduplicating values at the sort (LEAVE, default) or

/\*remove the duplicates (CROP).

Minimal specification VARS.

Sorts values of numeric variables horizontally, that is, within cases. You may request to output new variables or to replace the input variables themselves. A subset of cases can be excluded from sorting procedure. The macro does not do nested sorting (i.e., sorting of a case within identical values of another case).

The macro uses Quicksort sorting algorithm (with pivot value in the middle of the subvector), which is fast and is suitable for a great number of long vectors (i.e., many cases and large number of variables VARS). It will yield the same result as macro [!KO\_HSORT](#_MACRO_!HSORT:_HORIZONTAL). However, !KO\_HQSORT lacks options which !KO\_HSORT has, specifically, variants ORDER=R, CASES=FIRST, MISSPOS=ORIG. !KO\_HQSORT sorts all (selected for the procedure) cases of the dataset independently of each other. Missings are sorted to the tail. !KO\_HQSORT has no MISSING subcommand: if you want to include user-missing values in sorting, remove user-missing definitions off the input variables yourself.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hqsort vars= f1 to f1042 /order= A /selvar= sex=1.

* Values of variables F1 – F1042 are being sorted left to right ascendingly. This is done for each case in the dataset, for which SEX=1. The output variables are F1 – F1042 themselves.

***Subcommands***

**VARS**, **CAP**, **SELVAR**

These subcommands are identical to those in [!KO\_HSORT](#_MACRO_!HSORT:_HORIZONTAL) (see).

**ORDER**

Direction of sorting: sort left to right by ascending of values (ORDER=A, default/unspecification) or by descending of values (ORDER=D).

**REDUPL**

This additional option is the sifting off of repeating values, at the sorting. With REDUPL=CROP, if in a case some value encounters more than once, this value will encounter only once in that case at output. By default/unspecifying and with REDUPL=LEAVE, no elimination of reduplicates of values will take place.

***Special regimes***

The macro does not respond to weightedness, splitness of the dataset. It ignores temporary (under TEMPORARY) operations. The macro does not obey commands FILTER or USE: to select cases for the procedure, use subcommand SELVAR of the macro.

# MACRO !KO\_HBINEF: HORIZONTAL BINNING (EQUAL FREQUENCY GROUPS)

Version 1, Nov 2021. Tested on SPSS Statistics 20, 22, 26.

!KO\_hbinef vars= *v1 to* *v20* /\*Numeric variables (adjacent, via “to”) which values to bin

/\*horizontally

/cap= *bin* /\*Prefix (may quote) for output variables names; if not specified,

/\*VARS themselves will be binned

/k= *4* /\*Number of groups (bins, categories)

/order= A /\*Order of sorting: ascending (A), descending (D); if sorting

/\*in cases is already done, then keyword SORTED may follow

/incp= SEN /\*Include cut-point: in the junior (JUN) or in the senior (SEN) category

/trunc= NO /\*Truncate percentile level: YES [d] (default) or NO

/missing= /\*Take user-missing values (INCLUDE) or don’t take (EXCLUDE, default)

/nvalid= /\*Optional: empty a case with the number of valid values less than the threshold

/\*(specify positive integer)

/cutp= *cp#* /\*Optional: create variables with cut-points (specify prefix)

/freq= *fr#* /\*Optional: create variables with frequencies in categories (specify prefix)

/flag= YES /\*If CAP specified: count valid values: YES or NO (default).

Minimal specification VARS, K, ORDER, INCP.

Bins (categorizes, quantizes) values of quantitative variables horizontally, that is, within cases. You may request to output new variables or to replace the input variables themselves. Ordered categories being created are equal-frequency percentile groups.

Another macro that also can create equal-frequency percentile groups – [!KO\_HRANK](#_MACRO_!KO_HRANK:_HORIZONTAL). See comparison below.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hbinef vars= v1 to v50 /cap= bin /k= 4 /order= A /incp= JUN.

* Data in variables V1 – V50 are being binned horizontally, forming variables with categories, BIN1 – BIN50.
* The binning is into 4 equally sized percentile groups, i.e. quartile groups (K=4). Binning goes from low values to high values (order=A). Therefore, the macro will assign code 1 to values up to the 25th percentile (in the within-case distribution of values), code 2 to values between the 25th and the 50th percentile, code 3 to values between the 50th and the 75th percentile, code 4 to values beyond the 75th percentile. That is done separately and independently in each case of the dataset.
* If a value exactly equals the percentile (the cut-point), it will be assigned the smaller code of the two competing ones (INCP=JUN). For example, value concurring with the 25th percentile will receive code 1 (not 2).

**Algorithm**. The following is done independently in each case.

1. Sorting. *n* valid data in the case are internally sorted ascendingly, after what they are given order indices *i* = 1, 2, ..., *n*.
2. Cut-points. *k*-1 cut-points of approximately equal-frequency groups are the values of percentiles corresponding to the percentile levels (percentile ranks) 1/*k*\*100%, 2/*k*\*100%, ..., (*k*-1)/*k*\*100%. For example, with *k*=4, the cut-points will become percentile values: 25%-th, 50%-th, 75%-th. Percentile values are determined by the “Empirical method with averaging” (method AEMPIRICAL in SPSS command EXAMINE – see “Command Syntax Reference” and “SPSS Statistics Algorithms”).
3. Condensing. Cut-point duplicates, if any, are removed. Therefore, in some cases there will be left less cut-points than *k*-1. Condensing will not affect the grouping (binning) of data; it is performed for the sake of group codes being contiguous 1, 2, ..., no lacunae.
4. Assigning codes to data. Let the number of cut-points after (3) be *p* and their values be *Q1*, *Q2*, ..., *Qp*. Then, for *c* = 1, 2, ..., *p*: if the value of the datum < (with INCP=SEN) or ≤ (with INCP=JUN) than *Qc*, then give the datum code *c*, and break to turn to next datum.

So, a case data receive codes (categories) from 1 to *p* (with the ideal *p*=*k*), lower values receiving lesser code. *p* categories are equally filled with data, to the extent possible. If ORDER=D (rather than A), then in (1) data are sorted descending, and in (4), instead of “<” or “≤”, there used are “>” or “≥”; in the result, higher values receive lesser code.

On rare occasions, when the number of unique values in a case is little for the specified *k*, the codes can come out not consecutive despite of the (3); for instance, coding 1 and 3 is possible (instead of 1 and 2).

* Binning performed by !KO\_HBINEF coincides in result with binning accomplished by the menu-based Visual Binning procedure, option “Equal Percentiles” (as checked on vers. 26). Visual Binning categorizes data within variables, while !KO\_HBINEF does so within cases.
* SPSS command OPTIMAL BINNING with the specification METHOD=EQUALFREQ yields results, as a rule, close to Visual Binning / !KO\_HBINEF (as checked on vers. 26). OPTIMAL BINNING uses the similar algorithm, but it watches the succession (condensation) of the final codes more strictly.
* Macro [!KO\_HRANK](#_MACRO_!KO_HRANK:_HORIZONTAL) with the specification /TRANS= NTILES *k* also performs, like !KO\_HBINEF, binning into equally sized percentile groups, but uses somewhat different algorithm based on ranks (rather than on computation of percentile values), and where the results of binning depend on the way ties are processed. Categorization results by !KO\_HBINEF and !KO\_HRANK coincide quite often. !KO\_HBINEF is somewhat faster.

***Subcommands***

**VARS**

Specify via “to” numeric variables which values to bin horizontally. Those must be consecutive variables in the dataset. The data are perceived as quantitative (not nominal).

**CAP**

Prefix into the names of output variables with category codes. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). Output variables will be named by the prefix and an index from 1 to the number of input variables. There must be no variables with these names in the input dataset.

If not specify or omit CAP subcommand, input variables VARS themselves will be the output ones: all values in them will be replaced, and any user-missing definitions abolished. Value labels will remain former, but will become irrelevant since the variables’ values will come new.

**K**

Make this number of equal-frequency groups (categories). Specify integer not less than 2. It is expected that *k* is not great – less than the number of unique values in the distribution, i.e. in the case of the dataset.

**ORDER**

Direction of data sorting that the macro will do internally in each case. Sorting of values is done prior of all, before computing the cut-points. It will define the order of final codes. If sorting is ascending (A), a group of a lesser code will contain lower data values. If sorting is descending (D), a group of a lesser code will contain higher data values. ORDER=D has the same effect as when one switches sign of the input data and then uses ORDER=A.

In the situation where you have data in each case already sorted (in order A or D), you may add keyword SORTED after the keyword A or D. It will save time: the macro will not sort input data. Important: in the sorted data, all missings (system-, user-) must be in the tail (right side) of a case.

Let us remark that ORDER=D is not the same thing and brings not always the same effect as reversing the order of codes at output, i.e., turning over the scale of assigned codes (1 2 3... into ...3 2 1), what is done, for example, in Visual Binning. If a user wants to reverse the final scale, they could do it themselves.

**INCP**

In which one of the two adjacent groups to include a value that is equal to the cut-point between them. INCP=JUN puts that value in the “junior” group, i.e., the one with the lesser *code*. INCP=SEN puts that value in the “senior” group, i.e., the one with the greater *code*. In short, JUN is *cut-point]*, and SEN is *]cut-point*.

**TRUNC**

This subcommand determines the precision of calculation percentile levels (percentile ranks). It potentially affects percentile values, therefore, fine nuances among cut-points. Specify TRUNC=YES *d* (where *d*, digit from 1 to 6, signifies the number of decimal positions to retain in a fractional value) or TRUNC=NO.

Default/unspecifying of the s/c, likewise TRUNC=YES without parameter *d*, corresponds to specification TRUNC= YES 4. Every percentile level will be truncated up to 4 decimal digits. Say, with K=3, the percentile levels will be: 33.3333% and 66.6666%. If set *d* equal to 2, the levels will be 33.33% and 66.66%. If TRUNC=NO, then no truncation is applied and fractional percentile levels appear with the machine precision (15-16 decimal digits).

**MISSING**

How to treat user-missing values in the input variables: don’t take them in categorization, consider them as system-missing (MISSING=EXCLUDE, also default/unspecifying), or take as valid values (MISSING=INCLUDE).

**NVALID**

You may specify the bottom limit for the number of valid values in a case, positive integer. If there happen less valid values than that threshold, the case does not enter processing, but it will be empty (sysmis) in the variables output from the macro. If MISSING=INCLUDE, user missings count as valid values. By default, NVALID=1.

**CUTP**

You can request to create new variables showing cut-points (after condensing – see “Algorithm” above). Suggest a prefix for the names of these variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”).

**FREQ**

You can request to create new variables showing frequencies in the groups (categories) obtained. Suggest a prefix for the names of these variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”).

**FLAG**

With CAP specified, FLAG=YES creates or updates in the dataset variable *NVALID#$* showing the number of valid values in each case, at input. If MISSING=INCLUDE, user missings count as valid values.

FLAG=YES is not permitted with CAP not specified. FLAG does not react to s/c NVALID.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

# MACRO !KO\_HBINEI: HORIZONTAL BINNING (EQUAL INTERVAL GROUPS)

Version 1, Nov 2021. Tested on SPSS Statistics 20, 22, 26.

!KO\_hbinei vars= *v1 to v20* /\*Numeric variables (name-by-name and/or via “to”) which values

/\*to bin horizontally

/cap= *bin* /\*Prefix (may quote) for output variables names; if not specified,

/\*VARS themselves will be binned

/k= *4* /\*Number of groups (bins, categories)

/order= A /\*Order: ascending (A, default), descending (D)

/sw= /\*Start value and interval width; or AUTO (default): determine

/\*automatically

/incp= JUN /\*Include cut-point: in the junior (JUN) or in the senior (SEN) category

/outcodes= *-999 999* /\*If SW specified: two codes for values outside of the working range

/midinter= /\*Codes in the form of midinterval values (YES) or ordered integers (NO, default)

/missing= /\*Take user-missing values (INCLUDE) or don’t take (EXCLUDE, default)

/nvalid= /\*Optional: empty a case with the number of valid values less than the threshold

/\*(specify positive integer)

/cutp= *c#* /\*Optional: create variables with cut-points (specify prefix)

/freq= *f#* /\*Optional: create variables with frequencies in categories (specify prefix)

/flag= YES /\*If CAP specified: count valid values: YES or NO (default).

Minimal specification VARS, K, INCP.

Bins (categorizes, quantizes) values of quantitative variables horizontally, that is, within cases. You may request to output new variables or to replace the input variables themselves. Ordered categories being created are equal-interval groups.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hbinei vars= v1 to v50 /cap= bin /k= 4 /order= A /incp= JUN.

* Data in variables V1 – V50 are being binned horizontally, forming variables with categories, BIN1 – BIN50.
* The binning is into 4 equal interval groups. Binning goes from low values to high values (order=A). By default of SW s/c, the beginning of the 1st interval and the interval width are determined automatically: the start is the minimal observed value, and the observed range in the data is divided in K=4 intervals. That is done separately and independently in each case of the dataset.
* If a value exactly equals the border between intervals (the cut-point), it will be assigned the smaller code of the two competing ones (INCP=JUN). For example, value concurring with the border between the 1st and the 2nd intervals will receive code 1 (not 2).

***Subcommands***

**VARS**

Specify numeric variables (two minimum) which values to bin horizontally. Name-by-name list and/or using “to” convention. If the variables are too many, using “to” is recommended: this can spare the run time. The data are perceived as quantitative (not nominal).

**CAP**

Prefix into the names of output variables with category codes. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). Output variables will be named by the prefix and an index from 1 to the number of input variables. There must be no variables with these names in the input dataset.

If not specify or omit CAP subcommand, input variables VARS themselves will be the output ones: all values in them will be replaced, and any user-missing definitions abolished. Value labels will remain former, but will become irrelevant since the variables’ values will come new.

**K**

Make this number of equal-interval groups (categories). Specify integer not less than 2. It is expected that *k* is not great – less than the number of unique values in the distribution, i.e. in the case of the dataset.

**ORDER**

Direction of data intervalization (i.e., cutting of the scale of values into equal intervals) the macro will do in each case. This will define the order of final codes. If intervalization is ascending (A), a group of a lesser code will contain lower data values. If intervalization is descending (D), a group of a lesser code will contain higher data values.

**SW**

Subcommand which sets the parameters of intervalization of the scale of values – the start of cutting it into *k* equal intervals and the width of the interval.

AUTO - (also default/unspecifying) determines the parameters for each dataset case automatically. The start is the minimal (*min*, with ORDER=A) or maximal (*max*, with ORDER=D) valid value in a case. The range being notched coincides with the observed range of valid data in the case, *max-min*. The interval width = *(max-min)/k*. With ORDER=A the notching goes from *min* to *max*, and with ORDER=D – backward. Under SW=AUTO all valid values in a case will fall into this or that of the *k* categories.

*Start* *Width* - manual set of the start and the interval width. Specify two numbers; the second (width) must be positive. The first (start), as expected, would be from the area of low values with ORDER=A, because the notching will go from it up. And it is expected that the start would be from the area of high values with ORDER=D, because the notching will go from it down. With the manual specification of SW, it won’t be necessary so that all valid values in a case fall in some of the *k* categories; values having fallen in none of the categories will be recoded into OUTCODES codes (see).

Any one or both of *Start* and *Width* parameters can be specified by a variable name, not just a number. Those must be dataset variables containing the needed numeric values. This way you can set different parameter values to different cases. The width must always be a positive value.

If the variable with *Start* values contains missings, these cases will retain, in the end of the job, their input valid values: they won’t be binned in any way (however, user-missing values in them will turn into *sysmis*, if MISSING=EXCLUDE). You can use this opportunity, if you need the macro not to touch some cases in the dataset.

EXAMPLE 2.

!KO\_hbinei vars= v1 to v50 /cap= cat /k= 5 /order= D /sw= ptile90 2.5 /incp= JUN /outcodes= 999 99.

* Data in variables V1 – V50 are being binned horizontally, forming variables with categories, CAT1 – CAT50.
* The binning is into 5 equal interval groups. It goes (groups 1, 2, 3, 4, 5) from high values to low values (order=D), starting from the value contained in the variable PTILE90 and with the interval width 2.5.
* Values exceeding PTILE90 will be recoded into 999, and values that are too low to get in the last (5th) interval will be recoded into 99 (s/c OUTCODES).
* If a value exactly equals the border between adjacent intervals, it enters the group with the smaller code (INCP=JUN).

**INCP**

In which one of the two adjacent groups to include a value that is equal to the cut-point between them. INCP=JUN puts that value in the “junior” group, i.e., the one with the lesser *code*. INCP=SEN puts that value in the “senior” group, i.e., the one with the greater *code*. In short, JUN is *cut-point]*, and SEN is *]cut-point*.

When you specify SW parameters manually, INCP instruction applies to the notched range’s boundaries as well, i.e., to the points *Start* and *cut-point\_k*. Therefore, under INCP=JUN a value equal to *Start* will not enter group 1, it will appear beyond the categories and will be coded by the first code indicated in OUTCODES. Under INCP=SEN a value equal to *cut-point\_k* will not enter group k, it will appear beyond the categories and will be coded by the second code indicated in OUTCODES. While under SW=AUTO, INCP instruction concerns only cut-points lying inside of the boundaries *Start* and *cut-point\_k*. So that a value equal to *Start* will always enter group 1, and a value equal to *cut-point\_k* will always enter group k.

**OUTCODES**

The subcommand is ignored if SW=AUTO. Specify two values (codes) into which every valid value not hitting within the notched range, must be recoded. The first code will be assigned to values lying “before” *Start*. The second code will be assigned to values lying “after” *cut-point\_k*. With ORDER=A “before” are low values and “after” are high values; with ORDER=D it is vice versa. The codes may be identical.

You may specify one or both values as a variable name, not just a number. These should be dataset variables containing the needed numeric codes. This way you can set different codes to different cases. You can also use keyword $SYSMIS (system-missing) in place of a value.

**MIDINTER**

By default and with MIDINTER=NO, category codes being assigned to the groups are integers 1, 2, ..., *k*. With MIDINTER=YES, category codes are the values equal to the middles of the intervals. For example, group 2 will be coded not as 2 but as *(cut-point\_2 + cut-point\_1) / 2*.

**MISSING**

How to treat user-missing values in the input variables: don’t take them in categorization, consider them as system-missing (MISSING=EXCLUDE, also default/unspecifying), or take as valid values (MISSING=INCLUDE).

**NVALID**

You may specify the bottom limit for the number of valid values in a case, positive integer. If there happen less valid values than that threshold, the case does not enter processing, but it will be empty (sysmis) in the variables output from the macro. If MISSING=INCLUDE, user missings count as valid values. By default, NVALID=1.

**CUTP**

You can request to create new variables showing the cut-points into equal intervals. Suggest a prefix for the names of these variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). The start point of intervalization is shown as the cut-point with index 0.

**FREQ**

You can request to create new variables showing frequencies in the groups (categories) obtained. Suggest a prefix for the names of these variables. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”).

**FLAG**

With CAP specified, FLAG=YES creates or updates in the dataset variable *NVALID#$* showing the number of valid values in each case, at input. If MISSING=INCLUDE, user missings count as valid values.

FLAG=YES is not permitted with CAP not specified. FLAG does not react to s/c NVALID.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations. If you need the macro not to touch some cases in the dataset, specify parameter *Start* as a variable with missings (read in SW subcommand).

# MACRO !KO\_HFREQ: FREQUENCIES, HORIZONTALLY

Version 1, Dec 2018. Tested on SPSS Statistics 20, 22, 26.

!KO\_hfreq vars= *v1 v5 to v50* /\*Numeric variables (name-by-name and/or via “to”) which value

/\*frequencies to count horizontally

/values= *-1 THRU 10* /\*Optional: among which specifically values to count

/discret= RND *1* /\*How to discretize input values: RND d or TRUNC d (default RND 0)

/cap= f# /\*Prefix (may quote) into names of output frequency variables

/ending= /\*Make ending of these variables a value (VALUE, default) or an index (INDEX).

Minimal specification VARS, CAP.

Counts frequencies of values horizontally, that is, within cases. The result is a new unnamed dataset with cases that are cases of the input dataset and variables corresponding to values of the input data; values in these variables are the counts.

Variable *NTOTAL#$* in the output dataset contains sum of frequencies. This variable will not be created if in input data there were no missings and all valid values existing in the input data were processed by the macro (s/c VALUES allows not to take all existing values into processing).

The macro treats user-missing values as missings and does not count frequencies for them. The macro does not modify data in the input dataset in any way.

EXAMPLE 1.

!KO\_hfreq vars= v1 to v50 /cap= f#.

* Frequencies of distinct values existing in columns V1 – V50 are being counted, separately in each dataset case. Values that are not integer are internally rounded to integers (see s/c DISCRET).
* New dataset will be created. Its cases are the cases of the input one, and variables (columns) in it are the values existing in the input data (the value itself concatenated in the variable name). The data in the dataset are frequencies.

***Subcommands***

**VARS**

Specify numeric variables (one at least) which values’ frequencies to count horizontally. Name-by-name list and/or using “to” convention. If the variables are too many, using “to” is recommended: this can spare the run time.

**VALUES**

By default, the macro counts frequency for every encountered valid value. Here you can indicate a range or list of values of interest for you. The counting of frequencies will then be only among valid values hitting into this range/list. To specify range, use the following keywords (accepted in SPSS commands COUNT and RECODE): THRU, LO (or LOWEST), HI (or HIGHEST). For example, 0 6 THRU HI will mean: range from 6 and up and additionally value 0.

EXAMPLE 2.

!KO\_hfreq vars= v1 to v50 /cap= f# /values= -3 THRU 3.

* Data of variables V1 – V50 are some values from -10 to 20. But the macro will be interested only in frequencies of values found in the range -3 to 3, so frequencies for every unique value only from this range will be returned.

**DISCRET**

This subcommand plays a role if the input data have continuous or fractional valid values not sifted out by s/c VALUES. You should indicate whether the macro will round or truncate them, and with what precision, prior to counting their frequencies. Specify, respectively, DISCRET= RND *d* or DISCRET= TRUNC *d*, where *d* is integer 0, 1, 2, or 3 – that is the number of decimal digits which should be retained in a value.

By default/unspecifying of the subcommand, DISCRET= RND 0, i.e., the macro rounds every value to integer at input. DISCRET acts *after* sifting of values by VALUES subcommand.

EXAMPLE 3.

!KO\_hfreq vars= v1 to v50 /cap= f# /values= 2 3 6 /discret= TRUNC 1.

* The macro will count frequencies of values 2, 3, 6 only (other data values will be conflated to missings). Because VALUES s/c acts before DISCRET s/c, and values 2, 3, 6 are integer – no decimal digits, DISCRET= TRUNC 1 amounts to the same as TRUNC 0 or the default RND 0.

**CAP**

Prefix into the names of output dataset variables with frequencies. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”).

**ENDING**

By default and with ENDING=VALUE, names of variables of the output dataset end with the values themselves (if a value is negative, “-” is replaced with “\_”). For example, if CAP= *F#*, then variables corresponding to values: *-2.5, -2, 0, 1, 3.2* will be named: *F#\_2.5, F#\_2, F#0, F#1, F#3.2*. (You can prefer other symbol instead of “\_” to substitute the negative sign; for this, add subcommand /NEGSIGN= '*symb*', where *symb* is the wanted symbol that is allowed to use in variable names.)

With ENDING=INDEX, ordinal numbers will be used in ascending order of values. The names will then be *F#1, F#2, F#3, F#4, F#5*.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* !KO\_HFREQ is quite universal. You can obtain dummy variables with it, which macros !KO\_CATBIN, !KO\_!ACATBIN/!KO\_!ACATBIN2 (“Categorical - Binary recodings” collection) produce. If you input single variable in !KO\_HFREQ, the output will be variables with frequencies either 1 or 0, i.e. binary dummy variables. Input of multiple variables will imitate macros !KO\_MRCMRD, !KO\_!AMRCMRD/!KO\_!AMRCMRD (“Categorical - Binary recodings” collection).

# MACRO !KO\_HCOUNT: HORIZONTAL COUNT OF UNIQUE VALUES AND DUPLICATES

Version 2, Jan 2021 (Version 1, Feb 2006). Tested on SPSS Statistics 20, 22, 25.

!KO\_hcount vars= *v1 to v10* /\*Numeric variables (name-by-name and/or via “to”) in which to count

/\*unique values and duplicates within cases

/values= /\*Optional: among which values to count

/missing= INCLUDE /\*User-missing not to count (EXCLUDE, default) or count as valid (INCLUDE)

/rtimes= /\*Optional: number of unique values which repeat specifically this num of times, show it

/cap= *'cnt\_'* /\*Optional: supply prefix in the names of being computed variables.

Minimal specification VARS.

Counts, within cases, how many there encounters distinct values (“unique values”), how many values of these ones do not repeat (i.e., no duplicates) and how many do repeat, and what are the maximal and the minimal number of repetitions (copies) among the values. The macro creates (or updates, if exist already in the dataset) 5 variables – *UNIQ#*, *UNIQ\_S#*, *UNIQ\_R#*, *R\_MAX#*, *R\_MIN#*, showing these quantities.

Let a respondent’s data be: 8 2 3 2 7 3 5 2.

Here we have:

* Number of distinct encountered values (*UNIQ#*): **5** (namely, 2, 3, 5, 7, 8)
* Of these, this number of values encounter once (*UNIQ\_S#*): **3** (namely, 5, 7, 8)
* And this number of values exist in duplicates (*UNIQ\_R#*): **2** (namely, 2, 3)
* The maximal observed number of duplicates (*R\_MAX#*): **3** (value 2 encounters 3 times)
* The minimal observed number of duplicates (*R\_MIN#*): **2** (value 3 encounters 2 times)

So, the macro returns the highlighted five counts.

The macro counts only valid values. You may order to treat user-missing values as valid. You may also specify some other conditions.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

EXAMPLE 1.

!KO\_hcount vars= v1 to v10.

* Horizontal counting how many there are different valid values, repeating and not repeating, there are in V1 – V10.

***Subcommands***

**VARS**

Numeric variables in which to do the counting horizontally. Name-by-name list and/or using “to” convention. If the variables are too many, using “to” is recommended: this can spare the run time.

**VALUES**

By default, the macro counts among all encountered valid values. Here you may specify the range or list of values you are exclusively interested in. The counting will be among only these values encountered. To specify a range, use the following keywords (accepted in SPSS commands COUNT and RECODE): THRU, LO (or LOWEST), HI (or HIGHEST). For example, 0 6 THRU HI will mean: range from 6 and up and additionally value 0.

Note that if in the domain specified in VALUES there fall any user-missing values, while MISSING=EXCLUDE, those values won’t be counted in the variables where they are user-missing.

EXAMPLE 2.

!KO\_hcount vars= v1 v4 to v41 /values= -5 THRU 12 13 THRU 19.

!KO\_hcount vars= v1 v4 to v41 /values= 10 12 16 99 /missing= INCLUDE.

* In the first command, “horizontal” counts are dedicated only to (valid) values in the range from -5 to 12 and 13 to 19.
* In the second command, “horizontal” counts are dedicated only to values from the list 10, 12, 16, 99, and the user-missing values in VARS variables are taken by the macro as valid.

**MISSING**

You may count user-missing values in VARS as valid: MISSING=INCLUDE. By default/unspecification, MISSING=EXCLUDE and user-missing values are ignored.

**RTIMES**

This subcommand affects only variable *UNIQ\_R#*. By default, *UNIQ\_R#* shows how many distinct values encounter more than once, no matter how many times specifically. With RTIMES, you may ask to show how many distinct values encounter specifically so many times. Indicate operator (=, >, <, >=, <=, <>) and the number of times. For example, RTIMES= =3 will show how many different values encounter 3 times each. RTIMES= >3 will show how many different values encounter more than 3 times each. RTIMES= <>3 will show how many different values encounter not 3 times each (i.e., encounter 2 or 4 or 5 etc. times).

You may specify, in place of a number, a name of a variable containing the needed positive integers (this way, for each case a separate number of times can be specified).

RTIMES= =1 is an error, because variable *UNIQ\_R#* is dedicated only to repeating values; for values found only one time each, *UNIQ\_S#* is dedicated. RTIMES= >1 is equivalent to unspecifying RTIMES s/c.

**CAP**

By default, the macro creates or updates in the dataset the variables named *UNIQ#*, *UNIQ\_S#*, *UNIQ\_R#*, *R\_MAX#*, *R\_MIN#* (if these variables have user-missing definitions, the macro immediately takes off those). By CAP subcommand, you can request to create new variables with the above names with a prefix attached. Specify the prefix. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). If you are specifying a prefix, the input dataset must have no variables with the such formed names.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* Counting of specific duplicating values, in particular while imposing a ban to count some data cells, is possible to do with macro !KO\_VALVARS (“Series Response tools”).

# МАКРОС !KO\_HGAPS: STATS OF GAPS BETWEEN VALUES IN VARIATIONAL SERIES, HORIZONTAL

Version 2, Jan 2021 (Version 1, Nov 2007). Tested on SPSS Statistics 20, 22, 25.

!KO\_hgaps vars= v1 to v10 /\*Numeric variables (name-by-name and/or via “to”) among which values

/\*to measure gaps within cases

/sorted= /\*Values within cases are sorted (YES) not (NO, default)

/missing= INCLUDE /\*User-missing are missing (EXCLUDE, default) or treat as valid (INCLUDE)

/cap= /\*Optional: supply prefix in the names of being computed variables.

Minimal specification VARS.

A gap is an absolute difference, the distance between two nonequal consecutive values in a data distribution (i.e., in a series, called variational series, of data values sorted ascendingly (or descendingly). For each case of the dataset the macro will compute statistics of gaps: the number of existing gaps in it, the minimal and the maximal gap, averaged gap, standard deviation among gaps, creating, respectively, five variables: *NGAP#*, *MINGAP#*, *MAXGAP#*, *MEANGAP#*, *SDGAP#*. Statistics of gaps are useful, in particular, in exploring outliers and the degree of data nonhomogeneity.

Let a respondent’s data be: 9 0 3 2 8 3 5 11.

Then, as it’s easy to make sure if you order the values by their magnitude: 0 2 3 3 5 8 9 11,

* We have *NGAP#* = 6 gaps in all: (0-2) (2-3) (3-5) (5-8) (8-9) (9-11)
* The minimal gap, *MINGAP#*, between (nonequal) values equals **1**: specifically, 3-2=1, also 9-8=1
* And the maximal gap, *MAXGAP#*, equals **3**: specifically, 8-5=3
* Mean magnitude of all gaps, *MEANGAP#*, equals [(2-0)+(3-2)+(5-3 )+(8-5)+(9-8)+(11-9)] / 6 = **1.83**
* And the standard deviation of them, *SDGAP#*, equals **0.75**.

The macro will return the five highlighted values. Number of gaps *NGAP#* is always equal to the number of distinct values minus 1. If all values of a case are identical, all the five variables will be 0. If all values of a case are missing, *NGAP#* will be - 1, and the other variables will be 0.

The macro processes only valid values. You may order to treat user-missing values as valid.

There must be no variables *NVARS###* and *NVARS##$* in the working dataset.

***Subcommands***

**VARS**

Numeric variables among which to analyze gaps horizontally. Name-by-name list and/or using “to” convention. If the variables are too many, using “to” is recommended: this can spare the run time. Values need not be sorted left to right. The data are perceived as quantitative.

**SORTED**

By default and with SORTED=NO, the macro uses algorithm which does not require data be sorted; however, it is slow on big data (i.e., many cases and large number of variables VARS). If in your VARS variables (valid) values are sorted within cases, you may specify SORTED=YES; then the macro uses more quick algorithm. It doesn’t matter if the existing sortedness is ascending or descending, and if valid values are interlaid or not with missings. You may sort values within cases beforehand by macro !KO\_HSORT or (faster) by macro !KO\_HQSORT.

**MISSING**

You may count user-missing values in VARS as valid: MISSING=INCLUDE. By default/unspecification, MISSING=EXCLUDE and user-missing values are ignored.

**CAP**

By default, the macro creates or updates in the dataset the variables named *NGAP#*, *MINGAP#*, *MAXGAP#*, *MEANGAP#*, *SDGAP#* (if these variables have user-missing definitions, the macro immediately takes off those). By CAP subcommand, you can request to create new variables with the above names with a prefix attached. Specify the prefix. You may take it in quotes/apostrophes (recommended if you end the prefix with period: “var.”). If you are specifying a prefix, the input dataset must have no variables with the such formed names.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. It ignores temporary (under TEMPORARY) transformations.

1. Managing of random number seed in SPSS Statistics: menu Transform – Random Number Generator. [↑](#footnote-ref-1)