***Categorical - Binary recodings***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

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

All rights reserved

*Recoding categorical variables into binary or vice versa.* Collection of macros for conversion categorical data into binary data or back; for example, creation of dummy variables or turning multiple response sets MRC (categorical multiple response) and MRD (dichotomous multiple response) – one type into another.

*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.

* To create binary dummy variables out of categorical variables with single (alternative) response use [!KO\_CATBIN](#_MACRO_!KO_CATBIN:_RECODING), [!KO\_ACATBIN](#_MACRO_!KO_ACATBIN:_AUTOMATIC) or [!KO\_ACATBIN2](#_MACRO_!KO_ACATBIN2:_AUTOMATIC). The first one is for “manual” recoding, and the other two are practically identical and are for “automatic” recoding.
* To create binary variables of a dichotomous multiple response set out of variables of a categorical multiple response set use [!KO\_MRCMRD](#_MACRO_!KO_MRCMRD:_RECODING), [!KO\_AMRCMRD](#_MACRO_!KO_AMRCMRD:_AUTOMATIC) or [!KO\_AMRCMRD2](#_MACRO_!KO_AMRCMRD2:_AUTOMATIC). The first one is for “manual” recoding, and the other two are practically identical and are for “automatic” recoding.
* To create variables of a categorical multiple response set out of variables of a dichotomous multiple response set, or to create a categorical variable out of dummy variables representing alternative response, – use [!KO\_MRDMRC](#_MACRO_!KO_MRDMRC:_RECODING). Input variables does not have to be in fact dichotomous.

# MACRO !KO\_CATBIN: RECODING VARIABLES INTO BINARY DUMMY VARIABLES

Version 4, Apr 2013 (Version 1, Apr 1999). Tested on SPSS Statistics 20, 22, 25.

!KO\_catbin vars= *v1 v2* /\*Variables, name-by-name, for which to create binary (dummy) variables

/values= *1 to 8* EXCEPT *6* /\*List (either name-by-name or range via “to”) of nonnegative values

/\*of interest in VARS; after the range may list, after kw EXCEPT, values to

/\*exclude from the range

/sqcond= /\*Optional: additional, referring to the sequence in VARS, limitation for

/\*recognizing a code a significant response; one of: GT LT GE LE EQ NE FIRST

/\*FEQ (see)

/separ= *'\_'* /\*Optionally: separator into the names of created binary variables

/ending= /\*Make ending of a dummy variable a value (VALUE, default) or an index (INDEX)

/label= YES /\*Label the variables created by the value labels: YES or NO (default)

/macro= /\*Unite dummies names in set as macro lists: don’t do (NONE, default),

/\*all dummies (ALL), or without first (BUTFIRST) or last (BUTLAST) dummy.

Minimal specification VARS, VALUES.

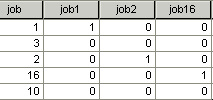
The macro creates binary dummy (indicator, one-hot) variables for user-indicated values of input variables. For each specified value, the macro will create the binary variable out of the input variable, where 1 = case is equal to this value; 0 = case is not equal to this value. A missing in the input variable will give system-missing in the dummy variables generated from it. If input variables are multiple, the suggested by you set of values is uniform for them all, and so each variable will yield the same number of binary variables.

Names of the dummy variables are concatenated of the input variables’ names and of the values specified (or of indices); amid these head and tail, a separator may be put. The dummy variables are created as new, so there should be no such names in the input dataset.

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_catbin vars= job /values= 1 2 16.



* Variable JOB is recoded into three binary ones corresponding to the 3 specified codes.
* Unit in a binary variable signifies that JOB has that (see variable name) code in the current case.

***Subcommands***

**VARS**

One or more numeric variables from which dummy variables need to create. Name-by-name list. If there are several variables, the macro treats them as independent. Typically VARS are categorical variables, but not necessarily. Fractional values are permitted in them. It is expected that at least partly values in the different variables are the same.

**VALUES**

List of significant codes (values in the input variables) that you need to “convert” into binary variables. That must be a list of nonnegative values. You can specify it either by (i) enumeration of all the needed codes (for example, *1 2 5 16*), or by (ii) a range min to max, for example, *2 to 9*. In the former case the codes need not be integers. When the codes are specified via a range, the macro forms their list as integers from min to max (both these numbers must be integer). If negative codes are important to you – see “specifying VALUES by variable names” below.

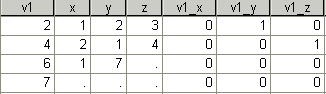
By insertion of keyword EXCEPT after the range you can specify a list of codes that you want to omit from the range. For example, if VALUES= *1 to 5* EXCEPT *2 4*, then the list of codes the macro will generate is *1 3 5*. If no codes follow the word EXCEPT, then the word is ignored.

*Specifying VALUES by variable names*. In case you are specifying VALUES by enumeration, you may specify as well (numeric) variable names in the list. This way it is possible to indicate different values to different respondents, for the recoding, and to set to them different number of the values. Moreover, values in the variables need not be nonnegative; though they need to have valid status in the variables: a user-missing value will be equated to an absent value.

The VALUES list must as a result consist of minimum two values.

EXAMPLE 2. VALUES in a form of variable list.

!KO\_catbin vars= v1 /values= x y z /separ= '\_'.



* Required is to recode categorical variable V1 into a series of binary variables.
* By the attributes X, Y, Z, which presence or absence the created variables V1\_X, V1\_Y, V1\_Z flag, there were meant in this instance different codes for different respondents. For example, attribute X is code 1 for the 1st respondent, but it is code 2 for the 2nd respondent.

EXAMPLE 3. A scale with negative values: -3 -2 -1 0 1 2 3.

do repeat val= -3 -2 -1 /var= m3 m2 m1.

compute var= val.

end repeat.

!KO\_catbin vars= v1 /values= m3 m2 m1 0 1 2 3 /separ= '\_'.

* This example shows that in order to process negative codes in V1, one has to create them as variables and then to indicate the variables in VALUES s/c.

**SEPAR**

In this optional subcommand, indicate a separator symbol (or a string of symbols) into the names of binary variables. Separator is visually comfortable when input variable names end with a digit. Recommended is to take the separator in quotes or apostrophes, especially when the separator ends with a period.

**ENDING**

By default and with ENDING=VALUE, a dummy variable name ends with the value (code) corresponding to it, of VALUES s/c. You may prefer an index, i.e., simple ordinal number 1, 2, 3, etc. as name ending. Then specify ENDING=INDEX.

**SQCOND**

This optional subcommand acts if input variables are multiple. It gives additional possibility to take into account the sequence in which variables are written in VARS. It can make sense in a situation when the variables *relate as “repeated measures”* of something one and the same, - for example, a respondent’s answers to the same question at different time points or under different conditions. If so, you might want to put a restriction to recognize a code "significant" (able to produce 1 in the corresponding binary variable), associated with the pattern of “dynamics” of the respondent’s answers in the sequence of variables. The following keywords set the condition to choose.

At recoding of a given input variable, a code in the current case can be recoded into 1 only under the extra condition that the code is:

GT - greater than the code in the preceding input variable

LT - lesser than the code in the preceding input variable

GE - greater or equal to the code in the preceding input variable

LE - lesser or equal to the code in the preceding input variable

EQ - equal to the code in the preceding input variable

NE - not equal to the code in the preceding input variable

FIRST - observed for the first time (in variables VARS sequence) in the case

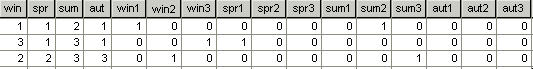
FEQ - observed for the first time or is equal to the code in the preceding input variable.

If the condition is not satisfied, the code is recoded into 0, not into 1.

*Notes*. (1) The condition is not applied to the first variable of VARS, because there is no preceding variable for it. (2) Inequality conditions GT, LT, GE, LE, NE mind any valid code “in the preceding variable”, i.e., the value “in the preceding variable” needs not belong to VALUES.

EXAMPLE 4. An investigator is going to study the choice of beverage type in seasonal perspective, and in order to deepen the analysis he wants to obtain binary variables out of categorical variables representing different seasons (with codes = beverage types selected). However, he is interested not simply in the fact a given beverage was chosen in a given season, but in the fact a given beverage was chosen in a given season *for the first time* in a year.

!KO\_catbin vars= win spr sum aut /values= 1 2 3 /sqcond= FIRST.



* Four input variables: WIN – SPR – SUM – AUT (winter – spring – summer – autumn). Values 1 2 3 are three beverages.
* Recoding of WIN result doesn’t depend on SQCOND because it is the first variable on VARS.
* Recoding for the rest variables: 1 is only where the given code was encountered for the first time in the sequence of VARS variables.

**LABEL**

By default and with LABEL=NO, the created dummy variables receive variable labels in the form of the codes (values) VALUES. If you specify LABEL=YES, the dummy variables receive variable labels in the form of value labels of those codes, borrowed by the macro from VARS variables attributes.

If there are variable names found in the VALUES list, then LABEL=NO will execute, even if LABEL=YES is specified.

**MACRO**

By this subcommand (MACRO=ALL), you can unite the names of dummy variables created from each given input variable, in a set in the form of a macro name. It is handy for substitution in syntax. The macro name is concatenated as !+VarName+Separ (if SEPAR ends with “.” or “\_”, that tail symbol is dropped). Order MACRO=BUTFIRST (or BUTLAST) if you want to exclude the name of the first (or the last) dummy variable from the macro (this removes multicollinearity of the set when VALUES exhaust all the present valid values of the input variable). By default, MACRO=NONE and no macros are produced.

EXAMPLE 5.

!KO\_catbin vars= job1 job2 /values= 1 to 13 /separ= '\_#' /macro= ALL.

frequencies !job1\_# !job2\_#.

* The macro created 13 dummies out of JOB1 and of JOB2, each, and combined their names in sets, as macros.
* These macro names persist through the SPSS Statistics session. In this example they are indicated on a later command FREQUENCIES, which thus becomes equivalent to: FREQUENCIES job1\_#1 to job1\_#13 job2\_#1 to job2\_#13.

***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 (“Horizontal tools”) also can create dummy variables.

# MACRO !KO\_ACATBIN: AUTOMATIC RECODING CATEGORICAL VARIABLES INTO BINARY DUMMY VARIABLES

Version 1, Dec 2013. Tested on SPSS Statistics 20, 22, 25.

!KO\_acatbin vars= *v1 v2* /\*Variables, name-by-name, for which to create binary (dummy) variables

/sort= /\*Sort the dummies order by variable values (YES, default) or don’t (NO)

/separ= *'\_#'* /\*Optionally: separator into the names of created binary variables

/ending= /\*Make ending of a dummy variable a value (VALUE, default) or an index (INDEX)

/label= /\*Label the variables created by the value labels: YES or NO (default)

/macro= /\*Unite dummies names in set as macro lists: don’t do (NONE, default),

/\*all dummies (ALL), or without first (BUTFIRST) or last (BUTLAST) dummy

/print= YES /\*Show frequencies: YES or NO (default).

Minimal specification VARS.

The macro creates binary dummy (indicator, one-hot) variables from input integer-valued categorical variables. It creates dummy variables “automatically” – for all observed values of an input variable. For each valid value present, the macro will create the binary variable out of the input variable, where 1 = case is equal to this value; 0 = case is not equal to this value. A missing in the input variable will give system-missing in the dummy variables generated from it. If input variables are multiple they do not have to have same values.

Macros !KO\_ACATBIN and [!KO\_ACATBIN2](#_MACRO_!KO_ACATBIN2:_AUTOMATIC) perform the same task and are identical in result. They differ in internal realization: !KO\_ACATBIN exploits matrix session, therefore its efficiency depends on RAM memory resource; !KO\_ACATBIN2 doesn’t use matrix session, is little dependent on RAM, is somewhat faster, and in situation of big data it can be preferable. !KO\_ACATBIN has subcommand SORT, which is absent in !KO\_ACATBIN2. !KO\_ACATBIN2 has subcommand DATASET, which is absent in !KO\_ACATBIN. Otherwise they are equivalent.

Names of the dummy variables are concatenated of the input variables’ names and of values (or of indices); amid these head and tail, a separator may be put. The dummy variables are created as new, so there should be no such names in the input dataset.

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_acatbin vars= job city.

* Variables JOB and CITY are recoded each in its own series of dummy variables. There are as many dummies in the series as there are valid values in the input variable.

***Subcommands***

**VARS**

One or more numeric variables from which dummy variables need to create. Name-by-name list. If there are several variables, the macro treats them as independent. The input variables must be categorical. Values (codes) in different VARS variables may be totally different. They should be integer (noninteger values the macro internally truncates to integers; you better avoid noninteger input). Negative values are allowed (but see s/c ENDING). There must be minimum two distinct valid values in a variable.

Valid value: *-999* is not allowed in the input variables, for it is reserved by the macro.

**SORT**

By default and with SORT=YES, the order of dummy variables at output will follow the sort order (ascending) of valid values of the input variable. With SORT=NO, the order of dummy variables at output will follow the order in which values were encountered (top to bottom) in the variable.

**SEPAR**

In this optional subcommand, indicate a separator symbol (or a string of symbols) into the names of binary variables. Separator is visually comfortable when input variable names end with a digit. Recommended is to take the separator in quotes or apostrophes, especially when the separator ends with a period.

**ENDING**

By default and with ENDING=VALUE, a dummy variable name ends with the value (code) corresponding to it. You may prefer an index, i.e., simple ordinal number 1, 2, 3, etc. as name ending. Then specify ENDING=INDEX.

If in the input variable there are negative valid values, then the subcommand will work as ENDING=INDEX, for that variable, and the macro will issue a warning if ENDING=VALUE was specified.

**LABEL**

By default and with LABEL=NO, the created dummy variables receive variable labels in the form of the codes (values). If you specify LABEL=YES, the dummy variables receive variable labels in the form of value labels of those codes, borrowed by the macro from VARS variables attributes.

**MACRO**

By this subcommand (MACRO=ALL), you can unite the names of dummy variables created from each given input variable, in a set in the form of a macro name. It is handy for substitution in syntax. The macro name is concatenated as !+VarName+Separ (if SEPAR ends with “.” or “\_”, that tail symbol is dropped). Order MACRO=BUTFIRST (or BUTLAST) if you want to exclude the name of the first (or the last) dummy variable from the macro (this removes multicollinearity of the set). By default, MACRO=NONE and no macros are produced.

EXAMPLE 2.

!KO\_acatbin vars= job1 job2 /separ= '\_#' /macro= BUTFIRST.

frequencies !job1\_# !job2\_#.

* The macro created dummy variables out of JOB1 and of JOB2, and combined their names in sets, as macros. The number of names in a set is one less than the number of dummies created from the variable, because MACRO=BUTFIRST.
* These macro names persist through the SPSS Statistics session. In this example they are indicated on a later command FREQUENCIES.

**PRINT**

Optional PRINT=YES displays in Output Viewer frequencies in each input variable. Value order in the display corresponds to s/c SORT.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. Besides, it turns off these regimes in the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* !KO\_CATCONT (“Categorical into Contrast”) creates not only dummy variables out of categorical ones, but also interaction variables between them.
* !KO\_HFREQ (“Horizontal tools”) also can create dummy variables.

# MACRO !KO\_ACATBIN2: AUTOMATIC RECODING CATEGORICAL VARIABLES INTO BINARY DUMMY VARIABLES

Version 1, Oct 2021. Tested on SPSS Statistics 20, 22, 25.

*The core idea of the macro is after Raynald Levesque*

!KO\_acatbin2 dataset= *data* /\*Working dataset’s name or RENAME

/vars= *v1*  /\*Variables, name-by-name, for which to create binary (dummy) variables

/separ= *'\_'* /\*Optionally: separator into the names of created binary variables

/ending= /\*Make ending of a dummy variable a value (VALUE, default) or an index (INDEX)

/label= /\*Label the variables created by the value labels: YES or NO (default)

/macro= ALL /\*Unite dummies names in set as macro lists: don’t do (NONE, default),

/\*all dummies (ALL), or without first (BUTFIRST) or last (BUTLAST) dummy

/print= YES /\*Show frequencies: YES or NO (default).

Minimal specification DATASET, VARS.

The macro creates binary dummy (indicator, one-hot) variables from input integer-valued categorical variables. It creates dummy variables “automatically” – for all observed values of an input variable. For each valid value present, the macro will create the binary variable out of the input variable, where 1 = case is equal to this value; 0 = case is not equal to this value. A missing in the input variable will give system-missing in the dummy variables generated from it. If input variables are multiple they do not have to have same values.

Macros [!KO\_ACATBIN](#_МАКРОС_!KO_ACATBIN:_АВТОМАТИЧЕСКАЯ) and !KO\_ACATBIN2 perform the same task and are identical in result. They differ in internal realization: !KO\_ACATBIN exploits matrix session, therefore its efficiency depends on RAM memory resource; !KO\_ACATBIN2 doesn’t use matrix session, is little dependent on RAM, is somewhat faster, and in situation of big data it can be preferable. !KO\_ACATBIN has subcommand SORT, which is absent in !KO\_ACATBIN2. !KO\_ACATBIN2 has subcommand DATASET, which is absent in !KO\_ACATBIN. !KO\_ACATBIN2 allows value: -999 in data. Otherwise they are equivalent.

Names of the dummy variables are concatenated of the input variables’ names and of values (or of indices); amid these head and tail, a separator may be put. The dummy variables are created as new, so there should be no such names in the input dataset.

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_acatbin2 dataset= DataSet0 /vars= job city.

* Variables JOB and CITY are recoded each in its own series of dummy variables. There are as many dummies in the series as there are valid values in the input variable.

***Subcommands***

**DATASET**

Specify the dataset name you work with. This macro needs the dataset be indicated to it. Or specify DATASET=RENAME, then the macro will rename the input dataset into *KO\_ACATBIN2#$.\_* and will work with it named thus. (The need for s/c DATASET is due to that the macro needs to know the name of the working dataset, but finding out that name is problematic for the macro.)

**VARS, SEPAR, ENDING, LABEL, MACRO, PRINT**

Subcommands identical to such ones of macro [!KO\_ACATBIN](#_MACRO_!KO_ACATBIN:_AUTOMATIC). As with !KO\_ACATBIN, input data should be integer. Value: -999 is allowed in data.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. Besides, it turns off these regimes in the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* !KO\_CATCONT (“Categorical into Contrast”) creates not only dummy variables out of categorical ones, but also interaction variables between them.
* !KO\_HFREQ (“Horizontal tools”) also can create dummy variables.

# MACRO !KO\_MRCMRD: RECODING CATEGORICAL MULTIPLE RESPONSE SET INTO DICHOTOMOUS MULTIPLE RESPONSE SET

Version 5, Aug 2013 (Version 1, Feb 1999). Tested on SPSS Statistics 20, 22, 25.

!KO\_mrcmrd vars= *v1 to v6* /\*Variables making an MRC, either name-by-name list or via “to”

/values= *1 to 8* EXCEPT *6* /\*List (either name-by-name or range via “to”) of nonnegative values

/\*of interest in VARS; after the range may list, after kw EXCEPT, values to

/\*exclude from the range

/\*If you specify "no answer" code here, omit NOANS

/dupl= /\*How to regard replicating of values in VARS: ignore duplicates (NO, default);

/\*return the result as count, not binary variables (COUNT);

/\*create own variable for each duplicate (CREATE integer)

/cap= *'bin.'* /\*Prefix into names of new being created binary variables (may quote)

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

/miss= SET /\*Make some cases missing in the created variables: (1) don’t do (NO, default);

/\*(2) if all values in the MRC set are missing in the case (SET);

/\*(3) if the case value is missing in the given variable (VAR varname)

/noans= *@* /\*Ending in the name of "no answer" variable; omit if the variable need not

/\*to create; after the ending may specify a label

/label= YES /\*Label the created MRD variables by value labels of the 1st VAR: YES or NO (default)

/regist= YES *'A Question'* /\*Register the created MRD set: YES or NO (default);

/\*after YES may specify a label for the set.

Minimal specification VARS, VALUES, CAP.

**Categorical multiple response set** (MRC) are several variables united by that they jointly are the common depository of data. Data (values) are discrete categories, response variants for a question of multiple (non-alternative) response; so that each respondent would have several different responses entered in their row (case). Each variable of an MRC set thus bears the meaning “one more answering by a respondent”. Cells that remain unfilled by response codes might be left empty or might be filled with a filler code which should be given missing value status in all variables of the MRC set. Usually, an MRC set is undertaken to follow the *regular build*: this is such fill by response codes whereby each next variable appears less and less filled with them; to put it differently, empty/nonvalid cell cannot be found more left than a valid code. The 1st variable is primary therefore. If a respondent did not give answers, “no answer” code is put in the 1st variable.

**Dichotomous or binary multiple response set** (MRD) is the alternative to MRC way to store the same kind of data. In MRD set, variables correspond to categories (response variants or attributes). Variables are as many as there are the variants, and values in the variables are binary: 1 (“yes”, “present”, “selected”) and 0 (“no”, “absent”, “not selected”). This resembles a set of dummy variables, however, unlike dummy variables, MRD set can contain more than one unit in a row, because MRD corresponds to a multiple response question, whereas dummy set corresponds to a single (alternative) response question. In parallel to how in categorical variables they usually provide the code for “no answer to the question”, in MRD they provide the variable “no answer to the question”, which is equal to 1 if all the rest of the set’s variables equal 0, and is equal to 0 otherwise.

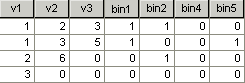
The macro recodes categorical variables with a common pool of values – a categorical multiple response set (MRC) – into a series of binary variables forming a dichotomous multiple response set (MRD). The user must offer the values of interest. For each specified value, the macro will create the binary variable out of the input set of variables, where 1 = case is equal to this value in some of the input variables; 0 = case is not equal to this value in all of the input variables. There exists an option to account for repeating values.

Names of the binary variables are concatenated of the prefix supplied by the user and of the values specified (or of indices). The variables are created as new, so there should be no such names in the input dataset. You can unite the binary variables in MRD (register as the multiple response set).

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_mrcmrd vars= v1 v2 v3 /values= 1 2 4 5 /cap= bin.



* Variables V1 V2 V3 are indicated as the MRC set (in this example, 0 is the filler code in the set, while other codes are responses to a “multiple response question”).
* Researcher indicated codes 1 2 4 5 for creating binary variables and prefix BIN for their names.
* The unit in a binary variable means that the code is present in the MRC set.

***Subcommands***

**VARS**

Input numeric variables constituting an MRC set (entire or its part) for you; the variables need not be registered in the data file as an MR set. Type either name-by-name or via “to” (referring to a range of consecutive variables in the dataset), but not in a mixed fashion. Typically, VARS are categorical variables, but not necessarily. Fractional values are permitted in them.

The MRC set does not have to have *regular build* (see above); the *filler* may be any code reserved for that (should be user-missing) or an absence of any code (system-missing).

If VARS is single variable, the output binary variables are dummy variables.

**VALUES**

List of significant codes (values in the input variables) that you need to “convert” into binary variables. That must be a list of nonnegative values. You can specify it either by (i) enumeration of all the needed codes (for example, *1 2 5 16*), or by (ii) a range min to max, for example, *2 to 9*. In the former case the codes need not be integers. When the codes are specified via a range, the macro forms their list as integers from min to max (both these numbers must be integer). If negative codes are important to you – see “specifying VALUES by variable names” below.

By insertion of keyword EXCEPT after the range you can specify a list of codes that you want to omit from the range. For example, if VALUES= *1 to 5* EXCEPT *2 4*, then the list of codes the macro will generate is *1 3 5*. If no codes follow the word EXCEPT, then the word is ignored.

Codes VALUES does not have to be valid values in VARS variables: !KO\_MRCMRD ignores user-missing statuses in VARS when it creates binary variables corresponding to VALUES.

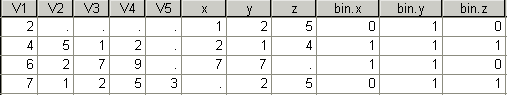
Code meaning “no answer to the question” can be put among VALUES; however, a better alternative to this may be a special subcommand NOANS (see).

*Specifying VALUES by variable names*. In case you are specifying VALUES by enumeration, you may specify as well (numeric) variable names in the list. This way it is possible to indicate different values to different respondents, for the recoding, and to set to them different number of the values. Moreover, values in the variables need not be nonnegative; though they need to have valid status in the variables: a user-missing value will be equated to an absent value.

The VALUES list must as a result consist of minimum two values.

EXAMPLE 2. VALUES as variables.

!KO\_mrcmrd vars= v1 to v4 /values= x y z /cap= 'bin.'.



* Categorical multiple response set: V1 – V4. X, Y, Z denotate three attributes, and in agreement with these the macro created three binary variables prefixed “BIN.”.
* Variables X, Y, Z show which codes mean “attribute X present”, “attribute Y present”, “attribute Z present”. For different respondents, these are different codes. For instance, attribute X for the 1st respondent is code 1, and for the 2nd it is code 2, and for the 3rd it is 7, while for the 4th respondent present attribute X is impossible (no code).
* For the 3rd respondent, code 7 designates presence either or both X and Y attributes.

EXAMPLE 3. Data with negative value.

compute var= -1.

!KO\_mrcmrd vars= mrc1 to mrc4 /values= var 0 1 2 3 /cap= mrd#.

* The MRC set contains codes interesting the user: -1, 0, 2, 3, 4. Negative code -1 cannot be directly specified in VALUES, but it is possible to create a variable out of it and specify its name in VALUES.

**DUPL**

By default/unspecifying and with DUPL=NO, the macro will create one binary variable per each code of VALUES. Unit in the binary variable flags that the code was encountered (at least once). MRC set sometimes is allowed to contain codes repeating within a case (for example, question “TV sets of which manufacturers do you have at home?” may permit a respondent to select several TV sets of the same company). If you are dealing with such MRC set and are not going to ignore the duplicates, you can do one of two ways:

COUNT - create the variables not binary, but count ones. Frequency of a code encounter will be in place of 1. Subcommand REGIST=YES will be impossible because the output variable set won’t be dichotomous, MRD.

CREATE *n* - create per each code not one, but *n* (positive integer >1) binary variables. Endings #1, #2, etc. will be added to the binary variables’ names, meaning “1st encounter”, “2nd encounter”, etc. If *n* is less than there are times the code encounter in the respondent’s data, only *n* its first encounters in VARS will be taken in stock.

Option DUPL=CREATE *n* is incompatible with the presence of variable names in VALUES.

**CAP**

This is the prefix the names of the being created variables will start with. You may take it in quotes or apostrophes (recommended if you end the prefix with a period: ‘bin2.’).

**ENDING**

By default and with ENDING=VALUE, an output variable name ends with the value (code) corresponding to it, of VALUES s/c. You may prefer an index, i.e., simple ordinal number 1, 2, 3, etc. as name ending. Then specify ENDING=INDEX.

**NOANS**

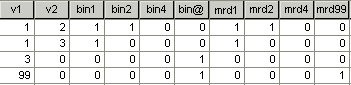
This subcommand creates, in last turn, the binary variable with the meaning “nothing of VALUES was found in VARS”; and it is = 1 in cases where the sum of the created variables, corresponding to VALUES, turned out to be 0. Specify a symbol (or a string of symbols) to become the ending of that variable’s name. After the symbol, you can suggest, in quotes or apostrophes, a label for the variable.

If VARS is an MRC set input in full, and VALUES are all possible (according to the questionnaire) response variants to the multiple response question, then s/c NOANS is the equivalent alternative to the direct mentioning of “no answer to the question” code in VALUES.

EXAMPLE 4. “No answer” variable.

!KO\_mrcmrd vars= v1 v2 /values= 1 2 4 /cap= bin /noans= @.

!KO\_mrcmrd vars= v1 v2 /values= 1 2 4 99 /cap= mrd.



* In the instance of NOANS s/c, the “no answer” variable BIN@ is created after the “BIN” variables as the “negative” of those created variables. It flags cases lacking VALUES codes.
* In the instance where code 99 (which in the MRC set meant “no answer to the question”, for the researcher) is introduced to VALUES, it is searched in the input variables as a usual code, giving its own variable, MRD99. Case 3 with sole code 3 (which is not listed among VALUES) has yielded 0 in all the “MRD” variables since there was no NOANS s/c.

**MISS**

Subcommand to account for / create missing values. By default/unspecifying and with MISS=NO, empty cells (system-missing) do not arise in the output variables. MISS allows to make system-missing some cases. For example, that could be respondents who were not asked the question. MISS makes a case system-missing in *all* created output variables. Indicate one of the two conditions:

SET - a case will be system-missing if it has missing values (user-missing or system-missing) in *all* input MRC variables (VARS).

VAR *varname* - a case will be system-missing if it has missing value (user-missing or system-missing) in this specific variable (it can be any variable of the dataset).

MISS makes cases system-missing *after* the output variables with their data are created.

**LABEL**

By default and with LABEL=NO, the created binary variables receive variable labels in the form of the codes (values) VALUES. If you specify LABEL=YES, the variables receive variable labels in the form of value labels of those codes, borrowed by the macro from the *first* VARS variable’s attributes.

If there are variable names found in the VALUES list, then LABEL=NO will execute, even if LABEL=YES is specified.

**REGIST**

Optional subcommand registering the MRD set in the .SAV file. For the system to know that the variables make up a dichotomous multiple response set. NO (default) – don’t register. YES – register; after YES you can specify, in quotes or apostrophes, a descriptive label for the set, for example: ‘Which of the listed TV programs do you watch?’. The set will be registered under the name indicated in CAP, from which ending symbol “\_” or “.” will be dropped if CAP ends with it.

REGIST=YES is impossible with DUPL=COUNT because the variables won’t be dichotomous.

***Special regimes***

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

***Related macros***

* If the variables of your MRC set represent not just answering attempts but ranks (1st variable = “1st place”, 2nd variable = “2nd place”…), then you can recode the set into the series of variables which values are not 1 and 0 but are those ranks, – by macro !KO\_RANKREV (“Series Response tools”).
* !KO\_HFREQ (“Horizontal tools”) also can create count variables, such as DUPL=COUNT does.

# MACRO !KO\_AMRCMRD: AUTOMATIC RECODING CATEGORICAL MULTIPLE RESPONSE SET INTO DICHOTOMOUS MULTIPLE RESPONSE SET

Version 1, Dec 2013. Tested on SPSS Statistics 20, 22, 25.

!KO\_amrcmrd vars= *v1 to v6* /\*Variables making an MRC, either name-by-name list or via “to”

/sort= /\*Sort the mrd variables order by the values (YES, default) or don’t (NO)

/dupl= /\*How to regard replicating of values in VARS: ignore duplicates (NO, default);

/\*or return the result as count, not binary variables (COUNT)

/cap= *'bin.'* /\*Prefix into names of new being created binary variables (may quote)

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

/miss= SET /\*Make some cases missing in the created variables: (1) don’t do (NO, default);

/\*(2) if all values in the MRC set are missing in the case (SET);

/\*(3) if the case value is missing in the given variable (VAR varname)

/noans= *@* /\*Ending in the name of "no answer" variable; omit if the variable need not

/\*to create; after the ending may specify a label

/label= YES /\* Label the created MRD variables by value labels of the 1st VAR:

/\*YES or NO (default)

/regist= YES *'A Question'* /\*Register the created MRD set: YES or NO (default);

/\*after YES may specify a label for the set

/print= YES /\*Show frequencies: YES or NO (default).

Minimal specification VARS, CAP.

See definitions what are MRC and MRD in the beginning of the description of [!KO\_MRCMRD](#_МАКРОС__!MRCMRD:).

The macro recodes integer-valued categorical variables with a common pool of values – a categorical multiple response set (MRC) – into a series of binary variables forming a dichotomous multiple response set (MRD). It creates binary variables “automatically” – for all observed values of the totality of input variables. For each valid value present, the macro will create the binary variable out of the input set of variables, where 1 = case is equal to this value in some of the input variables; 0 = case is not equal to this value in all of the input variables. There exists an option to account for repeating values (the variables are then created as count ones).

Macros !KO\_AMRCMRD and [!KO\_AMRCMRD2](#_MACRO_!KO_AMRCMRD2:_AUTOMATIC) perform the same task and are identical in result. They differ in internal realization: !KO\_AMRCMRD exploits matrix session, therefore its efficiency depends on RAM memory resource; !KO\_AMRCMRD2 doesn’t use matrix session, is little dependent on RAM. !KO\_AMRCMRD has subcommand SORT, which is absent in !KO\_AMRCMRD2. !KO\_AMRCMRD2 has subcommand DATASET, which is absent in !KO\_AMRCMRD. !KO\_AMRCMRD2 allows value: -999 in data. Otherwise they are equivalent.

Names of the binary variables are concatenated of the prefix supplied by the user and of the values specified (or of indices). The variables are created as new, so there should be no such names in the input dataset. You can unite the binary variables in MRD (register as the multiple response set).

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_amrcmrd vars= v1 v2 v3 /cap= bin /regist= YES 'What TV channels do you watch?'.

* Variables V1 V2 V3 are specified as the MRC set.
* The macro creates binary variables with prefix “BIN”; their number will equal the number of distinct integer values in V1-V3.
* REGIST=YES registers the binary variables as an MRD set.

***Subcommands***

**VARS**

Input numeric variables constituting an MRC set (entire or its part) for you; the variables need not be registered in the data file as an MR set. Type either name-by-name or via “to” (referring to a range of consecutive variables in the dataset), but not in a mixed fashion. The input variables must be categorical. Their values (category codes) must be integer (noninteger values the macro internally truncates to integers without a warning, but you should avoid entering noninteger data). Negative values are allowed (but see s/c ENDING). There must be minimum two distinct valid values in the data.

Valid value: *-999* is not allowed in the input variables, for it is reserved by the macro.

The macro does not demand user-missing definitions to be identical in the input variables. A category counts as valid the number of times it is observed in the variables’ data as valid.

The MRC set does not have to have *regular build* (see the beginning of the description of [!KO\_MRCMRD](#_МАКРОС__!MRCMRD:)); the *filler* may be any code reserved for that (should be user-missing) or an absence of any code (system-missing).

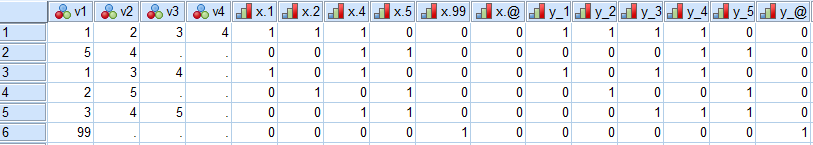
If VARS is single variable, the output binary variables are dummy variables.

EXAMPLE 2. Demonstartion of the different attitude of !KO\_AMRCMRD and !KO\_MRCMRD to values of input variables.

missing values v1 to v4 (3).

!KO\_amrcmrd vars= v1 to v4 /cap= 'x.' /noans= @.

!KO\_mrcmrd vars= v1 to v4 /values= 1 to 5 /cap= 'y\_' /noans= @.



* Categorical multiple response set: V1 – V4. These variables contain valid codes 1, 2, 3, 4, 5, and 99 (99 meant “no answer”). But MISSING VALUES command gave missing status to code 3 in V1-V4.
* !KO\_AMRCMRD recoded the input variables into binary X.1, X.2, X.4, X.5, X.99. Note that X.3 was not created, because 3 isn’t a valid value in all the V1-V4.
* !KO\_AMRCMRD also created in the end the requested “no answer” variable X.@, but it turned out empty (zeros) – because there already exists variable X.99 which was created for the valid code 99 (having meant “no answer”).
* !KO\_MRCMRD recoded the input variables into binary Y\_1, Y\_2, Y\_3, Y\_4, Y\_5 – in correspondence to the specified in VALUES codes 1, 2, 3, 4, 5. !KO\_MRCMRD ignores user-missing status of codes specified in VALUES, therefore variable Y\_3 was created and it contains units as if 3 were a valid value in V1-V4.
* !KO\_MRCMRD created in the end the requested “no answer” variable Y\_@. It equals 1 where all the Y\_1–Y\_5 equal 0. The “no answer” code 99 was not requested in VALUES, and thus Y\_@ came out to be corresponding to it.

**SORT**

By default and with SORT=YES, the order of binary variables at output will follow the sort order (ascending) of valid values of the input MRC set. With SORT=NO, the order of binary variables at output will follow the order in which values were encountered (top to bottom and starting from the 1st variable) in the set.

**DUPL**

By default/unspecifying and with DUPL=NO, the macro will create one binary variable per valid code of VARS. Unit in the binary variable flags that the code was encountered (at least once). MRC set sometimes is allowed to contain codes repeating within a case (for example, question “TV sets of which manufacturers do you have at home?” may permit a respondent to select several TV sets of the same company). If you are dealing with such MRC set and are not going to ignore the duplicates, you can specify DUPL=COUNT. Then the variables will be created not binary, but count ones. Frequency of a code encounter will be in place of 1. Subcommand REGIST=YES will be impossible because the output variable set won’t be dichotomous, MRD.

**CAP**

This is the prefix the names of the being created variables will start with. You may take it in quotes or apostrophes (recommended if you end the prefix with a period: ‘bin2.’).

**ENDING**

By default and with ENDING=VALUE, an output variable name ends with the value (code) corresponding to it. You may prefer an index, i.e., simple ordinal number 1, 2, 3, etc. as name ending. Then specify ENDING=INDEX.

If in the input variables there are negative valid values, then the subcommand will work as ENDING=INDEX, and the macro will issue a warning if ENDING=VALUE was specified.

**NOANS**

This subcommand creates, in last turn, the binary variable with the meaning “no valid codes were found in VARS”; and it is = 1 in cases where the sum of the created variables, corresponding to valid codes, turned out to be 0. Specify a symbol (or a string of symbols) to become the ending of that variable’s name. After the symbol, you can suggest, in quotes or apostrophes, a label for the variable. With ENDING=VALUE, try not to specify bare digits as the symbol, in avoidance of the conflict between variable names, what would cause error.

If in your VARS variables data code “no answer” is present and is valid, then commonly there is no sense in specifying NOANS subcommand. Also, with MISS=SET, the variable created by NOANS s/c will not contain units: all such cases will be replaced by system-missing.

**MISS**

Subcommand to account for / create missing values. By default/unspecifying and with MISS=NO, empty cells (system-missing) do not arise in the output variables. MISS allows to make system-missing some cases. For example, that could be respondents who were not asked the question. MISS makes a case system-missing in *all* created output variables. Indicate one of the two conditions:

SET - a case will be system-missing if it has missing values (user-missing or system-missing) in *all* input MRC variables (VARS).

VAR *varname* - a case will be system-missing if it has missing value (user-missing or system-missing) in this specific variable (it can be any variable of the dataset).

MISS makes cases system-missing *after* the output variables with their data are created.

**LABEL**

By default and with LABEL=NO, the created binary variables receive variable labels in the form of the codes (values). If you specify LABEL=YES, the variables receive variable labels in the form of value labels of those codes, borrowed by the macro from the *first* VARS variable’s attributes.

**REGIST**

Optional subcommand registering the MRD set in the .SAV file. For the system to know that the variables make up a dichotomous multiple response set. NO (default) – don’t register. YES – register; after YES you can specify, in quotes or apostrophes, a descriptive label for the set, for example: ‘Which of the listed TV programs do you watch?’. The set will be registered under the name indicated in CAP, from which ending symbol “\_” or “.” will be dropped if CAP ends with it.

REGIST=YES is impossible with DUPL=COUNT because the variables won’t be dichotomous.

**PRINT**

Optional PRINT=YES displays in Output Viewer frequencies in the input MRC set. Value order in the display corresponds to s/c SORT. If DUPL=COUNT, frequencies of responses are displayed; otherwise frequencies of respondents are displayed.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. Besides, it turns off these regimes in the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* !KO\_HFREQ (“Horizontal tools”) also can create count variables, such as DUPL=COUNT does.

# MACRO !KO\_AMRCMRD2: AUTOMATIC RECODING CATEGORICAL MULTIPLE RESPONSE SET INTO DICHOTOMOUS MULTIPLE RESPONSE SET

Version 1, Oct 2021. Tested on SPSS Statistics 20, 22, 25.

!KO\_amrcmrd2 dataset= *data* /\*Working dataset’s name or RENAME

/vars= *v1 to v6* /\*Variables making an MRC, either name-by-name list or via “to”

/dupl= /\*How to regard replicating of values in VARS: ignore duplicates (NO, default);

/\*or return the result as count, not binary variables (COUNT)

/cap= *'bin.'* /\*Prefix into names of new being created binary variables (may quote)

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

/miss= SET /\*Make some cases missing in the created variables: (1) don’t do (NO, default);

/\*(2) if all values in the MRC set are missing in the case (SET);

/\*(3) if the case value is missing in the given variable (VAR varname)

/noans= *@* /\*Ending in the name of "no answer" variable; omit if the variable need not

/\*to create; after the ending may specify a label

/label= YES /\* Label the created MRD variables by value labels of the 1st VAR:

/\*YES or NO (default)

/regist= YES *'A Question'* /\*Register the created MRD set: YES or NO (default);

/\*after YES may specify a label for the set

/print= YES /\*Show frequencies: YES or NO (default).

Minimal specification DATASET, VARS, CAP.

See definitions what are MRC and MRD in the beginning of the description of [!KO\_MRCMRD](#_МАКРОС__!MRCMRD:).

The macro recodes integer-valued categorical variables with a common pool of values – a categorical multiple response set (MRC) – into a series of binary variables forming a dichotomous multiple response set (MRD). It creates binary variables “automatically” – for all observed values of the totality of input variables. For each valid value present, the macro will create the binary variable out of the input set of variables, where 1 = case is equal to this value in some of the input variables; 0 = case is not equal to this value in all of the input variables. There exists an option to account for repeating values (the variables are then created as count ones).

Macros [!KO\_AMRCMRD](#_MACRO_!KO_AMRCMRD:_AUTOMATIC) and !KO\_AMRCMRD2 perform the same task and are identical in result. They differ in internal realization: !KO\_AMRCMRD exploits matrix session, therefore its efficiency depends on RAM memory resource; !KO\_AMRCMRD2 doesn’t use matrix session, is little dependent on RAM. !KO\_AMRCMRD has subcommand SORT, which is absent in !KO\_AMRCMRD2. !KO\_AMRCMRD2 has subcommand DATASET, which is absent in !KO\_AMRCMRD. !KO\_AMRCMRD2 allows value: -999 in data. Otherwise they are equivalent.

Names of the binary variables are concatenated of the prefix supplied by the user and of the values specified (or of indices). The variables are created as new, so there should be no such names in the input dataset. You can unite the binary variables in MRD (register as the multiple response set).

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_amrcmrd2 dataset= DataSet0 /vars= v1 v2 v3 /cap= bin /regist= YES 'What TV channels do you watch?'.

* Variables V1 V2 V3 are specified as the MRC set.
* The macro creates binary variables with prefix “BIN”; their number will equal the number of distinct integer values in V1-V3.
* REGIST=YES registers the binary variables as an MRD set.

***Subcommands***

**DATASET**

Specify the dataset name you work with. This macro needs the dataset be indicated to it. Or specify DATASET=RENAME, then the macro will rename the input dataset into *KO\_AMRCMRD2#$.\_* and will work with it named thus. (The need for s/c DATASET is due to that the macro needs to know the name of the working dataset, but finding out that name is problematic for the macro.)

**VARS, DUPL, CAP, ENDING, NOANS, MISS, LABEL, REGIST, PRINT**

Subcommands identical to such ones of macro [!KO\_AMRCMRD](#_MACRO_!KO_AMRCMRD:_AUTOMATIC). As with !KO\_AMRCMRD, input data should be integer. Value: -999 is allowed in data.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. Besides, it turns off these regimes in the dataset. It ignores temporary (under TEMPORARY) transformations.

***Related macros***

* !KO\_HFREQ (“Horizontal tools”) also can create count variables, such as DUPL=COUNT does.

# MACRO !KO\_MRDMRC: RECODING DICHOTOMOUS MULTIPLE RESPONSE SET INTO CATEGORICAL MULTIPLE RESPONSE SET

Version 4, Sep 2014 (Version 1, Oct 2000). Tested on SPSS Statistics 20, 22, 25.

*This macro needs SPSS Statistics 17 or higher*.

!KO\_mrdmrc vars= *v1 v2 v3 v4* /\*Name-by-name dichotomous variables

/selval= *1* /\*”Selected” value(s) there (default=1): nonnegative number(s) in list or in

/\*range via "THRU"; or COUNT n (see)

/codes= /\*Codes for the MRC being created, in correspondence to VARS: either list of

/\*nonnegative values in number of VARS, or EXTRACT (pick codes from VARS names),

/\*or ORDINAL (generate ordinal numbers, default)

/cap= *'cat.'* /\*Prefix into names of the new being created categorical variables (may quote)

/limit= /\*How many variables to create: necessary minimum (MIN, default), as many as

/\*there are input variables (MAX), or the specified number (number)

/filler= *0* /\*Optional: filler code for the set

/noans= *99 'NA'* /\*Optional: code for “no answer” (a label may follow then)

/miss= SET /\*Make some cases missing in the created variables: (1) don’t do (NO, default);

/\*(2) if all values in the MRD set are missing in the case (SET);

/\*(3) if the case value is missing in the given variable (VAR varname)

/label= YES /\*Label the created categories by the MRD variables' labels: YES or NO (default)

/regist= YES *'A Question'* /\*Register the created MRC set: YES or NO (default);

/\*after YES may specify a label for the set

/dropind= /\*Drop (YES) index 1 from the name if the output variable came out single;

/\*default = NO.

Minimal specification VARS, CAP.

See definitions what are MRC and MRD in the beginning of the description of [!KO\_MRCMRD](#_МАКРОС__!MRCMRD:).

The macro recodes a set of dichotomous variables or any other numeric variables communicating the status of a binary attribute (whether the attribute is present or not, or how many times is present), into categorical variables. If the input set of variables represented non-alternative attributes, i.e. it was a “dichotomous multiple response set” (MRD), a categorical multiple response set will come out of it. But if the input set represented alternative attributes, i.e. it was dummy variables, a single categorical variable will come out of it.

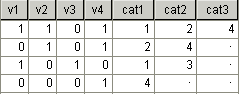
To each input variable, that is, to each attribute of response variant, there corresponds a respective numeric code in the output variables. You may offer these codes or may ask the macro to make them up automatically. The macro first creates categorical variables as empty and then fills them in with codes left to right in the order of attributes (i.e., the input variables as specified in VARS). The MRC set has *regular build* at output (see what it is in the beginning of [!KO\_MRCMRD](#_МАКРОС__!MRCMRD:) description).

Names of the categorical variables are concatenated of the prefix supplied by the user and of sequential indices 1, 2, ... The variables are created as new, so there should be no such names in the input dataset. You can unite the categorical variables in MRC (register as the multiple response set).

The macro creates temporary variables with names containing five consecutive symbols *$*, for example, *v$$$$$.\_2*. Therefore you should better avoid such names in your dataset.

EXAMPLE 1.

!KO\_mrdmrc vars= v1 v2 v3 v4 /cap= 'cat' /regist= YES 'An MRC set'.



* 4 binary variables V1 V2 V3 V4 correspond, by omitting of s/c CODES, to codes 1 2 3 4.
* Categorical variables prefixed “CAT” were created, and filled with codes from left to right. A code is inserted if in the given input variable the attribute is “present” (1).
* The categorical variables were registered as an MRC set.
* If variables V1 V2 V3 V4 were dummy variables, that is, they contained together not more than one unit is a row, the macro would create, by default, a single categorical variable CAT1.

***Subcommands***

**VARS**

Name-by-name list of input variables (two minimum): an MRD set of multiple response or any variables which the researcher takes for a set of attributes (each variable is one attribute that either is “present” or “absent”). The variables do not have to be actually binary (1 and 0) or even dichotomous or discrete, in general. Subcommand SELVAL allows to indicate one code or multiple values signifying “attribute present”. A missing in a variable equates to “attribute absent”, that is, to absence of “attribute present” code.

**SELVAL**

This subcommand informs the macro which value or values in variables VARS (in them all) should be understood as “attribute present” or “selected” status (while any other value or missing will be regarded “attribute absent” or “not selected”). By default, SELVAL=1, so the macro will account value 1 for “present” and any other value for “absent”. SELVAL also is used to inform the macro when VARS are count variables rather than status ones.

1) **When VARS are status variables** (showing an attribute either present or absent). List in SELVAL the values corresponding to the “attribute present” status. That must be nonnegative values (decimal fractional – permitted). For example: SELVAL= 6 8 9. Here any of the three values specified will be understood as “attribute present”. Repetition of a value, e.g., SELVAL= 6 6 8 9, won’t cause an error. If a specified value is not valid in some VARS variable, it will not be accounted for “attribute present” in it.

If in VARS a whole value range answers the “attribute present” status, you may indicate it with keyword THRU. For example: SELVAL= 1 THRU 5, where the second specified value must be not less than the first. Each valid value from 1 to 5, inclusive, will mean “attribute present”. You may specify SELVAR either by listing or by range, but not mixed.

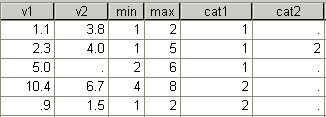
*Variable names in SELVAL*. In place of all or some numbers it is possible to type variable names, where there would contain values signifying “attribute present”. That means that you have an option to specify different such values to different respondents (dataset cases). For example: SELVAL=*SELCODE* refers to some variable *SELCODE* which values are the values meaning “attribute present” in VARS. Values is such variables need not be nonnegative. When specifying SELVAL by range, *X* THRU *Y*, both variables *X* and *Y* must be filled with valid values and the second value must be not less than the first.

2) **When VARS are count variables** (showing how many times an attribute is present). Values of VARS variables must be nonnegative integer (noninteger values the macro will internally truncate to integers, and negative values it treats as 0). The values are frequency – the number of times a given attribute (VARS variable) is observed. You must specify the subcommand this way: SELVAL= COUNT *n*, where *n* is a positive integer. Let the datum of VARS variable *X* in case *i* is *f*. If *f*>0 (and is valid in *X*), the macro will create min(*f*, *n*) copies of the code corresponding to *X*, in case *i* of the output variables.

Thus, SELVAL= COUNT *n* makes it possible to create, out of count input variables, an MRC set with repeating categories (responses).

EXAMPLE 2. Continuous input variables. And SELVAL as variables.

!KO\_mrdmrc vars= v1 v2 /selval= min THRU max /cap= cat.



* Variables V1 and V2 are continuous attributes. If a value lies within the bounds of MIN and MAX (and these are variable names in this instance), then it is taken for the attribute as “present”. These bounds are different for different respondents.
* Categorical multiple response set CAT1-CAT2 is obtained from those data. Say, the 1st respondent has attribute V1 (category code 1) because the value 1.1 fits in the range between 1 and 2, but attribute V2 (code 2) – this respondent doesn’t have it, since 3.8 – does not fit in the range.

EXAMPLE 3. SELVAL is a negative value.

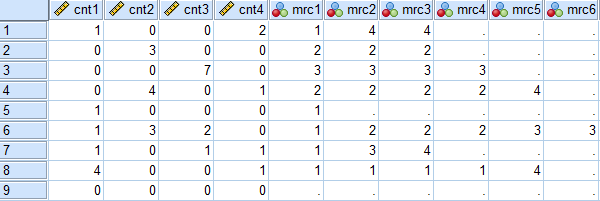
compute sel= -1.

!KO\_mrdmrc vars= v1 v2 /selval= sel /cap= cat.

* An attribute’s presence is coded in V1 and V2 by value: -1. You are not allowed to specify in SELVAL a negative value directly, but you may specify a variable name. The latter could have any values.

EXAMPLE 4. Creating an MRC with response duplicates with the help of SELVAL= COUNT n.

!KO\_mrdmrc vars= cnt1 cnt2 cnt3 cnt4 /cap= mrc /selval= COUNT 4.



* Variables CNT1 – CNT4 correspond to some 4 medical complaints. But the variables are not binary (presence vs absence of a complaint) but are count: how many days a week a patient experiences the complaint.
* These data are recoded into a categorical multiple response set (variables prefixed “MRC”) with codes 1 – 4 corresponding to the 4 complaints.
* For each respondent, a complaint is represented by as many code copies as the number of days per week the complaint was experienced; but not more than 4 copies (because SELVAL= COUNT 4).

**CODES**

Each variable of VARS stand for an attribute. The macro must give a numeric code in match with the attribute, by which the attribute (category) will be coded in the output variables. The subcommand sets the list of codes, the values of the future categorical variables. There are three ways to specify:

*code list* - enumerate the codes. These can be any nonnegative numbers, as many as there are input variables. They correspond to them orderly.

ORDINAL - (default/unspecifying) the macro builds the list automatically as integers 1, 2, …, number\_of\_input\_variables.

EXTRACT - the macro tries to extract codes from the input variables’ names. That must be just integer numbers. The input variable names must end with these numbers-codes. For example, variable named *VAR2* will yield code 2. Variable named *MRD3.2* will also give code 2. The macro picks an integer from the tail of a name.

EXAMPLE 5.

!KO\_mrdmrc vars= bin10.4 bin3.2 bin6 zxc\_11 /cap= 'mrc\_' /codes= 4 2 6 11.  
!KO\_mrdmrc vars= bin10.4 bin3.2 bin6 zxc\_11 /cap= 'mrc.' /codes= EXTRACT.

* Both commands in this example are equivalent.

**LIMIT**

How many variables to create. By default and with LIMIT=MIN, there are created the number of categorical variables minimally sufficient to accommodate all the data. LIMIT=MAX creates as many variables as there are input variables, i.e., with a reserve. (Under SELVAL= COUNT *n* this number is further multiplied by *n*.) LIMIT=*number* creates this number (>=1) of variables. Having specified an understated number, you will cut off some responses of respondents, whereas an overstated number will produce surplus empty variables.

**CAP**

This is the prefix the names of the being created variables will start with. You may take it in quotes or apostrophes (recommended if you end the prefix with a period: ‘cat2.’). The names will consist of the prefix and a sequential index.

**FILLER**

You can indicate a filler code for the MRC set. This can be any number (even negative) that is absent in CODES. The macro will appoint the filler the missing value status in the MRC set. In the special case when the filler code coincides with the “no answer” code you specify (NOANS), the filler code does not become missing value in the *first* variable of the set. This is generally correct since the MRC is of the *regular* build. However, it is not recommended to superpose the filler code and the “no answer” code.

**NOANS**

This optional subcommand requests to create the code (category) “no answer to the question” or, more precisely, “there is none of the VARS attributes” or “none selected”. Specify a nonnegative value. The code will appear only in the *first* MRC variable, in those cases that left unfilled with the CODES codes at all.

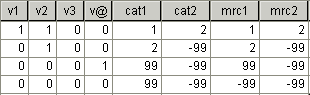
An alternative to NOANS can be specifying the “no answer” variable among the input VARS variables; however, it is equivalent only in the situation when the MRD set is input whole and the data in the “no answer” variable are accurate. Duplicating the “no answer” code within the code list (CODES) is allowed, whether it is correct or not depends on considerations. NOANS can be any nonnegative number. About combining NOANS code with filler code, see FILLER.

After the code you may leave a quoted label for it. It will be used if LABEL=YES.

EXAMPLE 6. Two ways to set the “no answer” code.

!KO\_mrdmrc vars= v1 v2 v3 /codes= 1 2 3 /cap= cat /filler= -99 /noans= 99.

!KO\_mrdmrc vars= v1 v2 v3 v@ /codes= 1 2 3 99 /cap= mrc /filler= -99.



* In the first instance there recoded are variables V1 V2 V3. It is requested to insert the code “no answer to the question” (99). The 3rd and the 4th respondents selected nothing in V1 V2 V3, and so they bear 99 in CAT1.
* In the second instance there was no request to insert the “no answer” code”; instead, an existing “no answer” variable V@ got used as one more input variable; and the code 99 was attached to it in CODES. Incidentally, data in V@ were not quite reliable: the 4th respondent ought to have 1 there. As the result, the filler code (-99) remained in MRC1 for that case.

**MISS**

Subcommand to account for / create missing values. By default/unspecifying and with MISS=NO, empty cells (system-missing) do not arise in the output variables, if you specified FILLER and NOANS. MISS allows to make system-missing some cases. For example, that could be respondents who were not asked the question. MISS makes a case system-missing in *all* created output variables. Indicate one of the two conditions:

SET - a case will be system-missing if it has missing values (user-missing or system-missing) in *all* input VARS variables.

VAR *varname* - a case will be system-missing if it has missing value (user-missing or system-missing) in this specific variable (it can be any variable of the dataset).

MISS makes cases system-missing *after* the output variables with their data are created.

**LABEL**

You can label by this subcommand (LABEL=YES) the created categorical values by the labels of input variables. The labeling will be done by the macro only in the *first* variable of the output categorical set, because that is enough for an MRC to function. By default and with LABEL=NO, no value labeling takes place. LABEL=YES requires Custom Tables module be installed because of the CTABLES command internally used.

**REGIST**

Optional subcommand registering the MRC set in the .SAV file, when the macro produces more than one categorical variable. For the system to know that the variables make up a categorical multiple response set. NO (default) – don’t register. YES – register; after YES you can specify, in quotes or apostrophes, a descriptive label for the set, for example: ‘Which of the listed TV programs do you watch?’. The set will be registered under the name indicated in CAP, from which ending symbol “\_” or “.” will be dropped if CAP ends with it. The first set’s variable will also get labeled if a label follows YES.

**DROPIND**

This optional subcommand acts only when LIMIT=MIN (which is the default). Then, if the variable created by the macro turned out to be just single (what is expected if VARS are dummy variables rather than a multiple response set), then DROPIND=YES will remove index 1 from its name. For example, with CAP= ‘CAT’, the variable at its exit from the macro will get name *CAT*, not *CAT1*. Also, if CAP ends with symbol “\_” or “.”, this will drop too. By default, DROPIND=NO.

***Special regimes***

The macro does not respond to weightedness, filteredness (FILTER, USE), splitness of the dataset. Besides, it turns off weights and split, if LABEL=YES. It ignores temporary (under TEMPORARY) transformations.