***Impute missing data***

SPSS macros by Kirill Orlov

kior@akado.ru, ttnphns@gmail.com

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

All rights reserved.

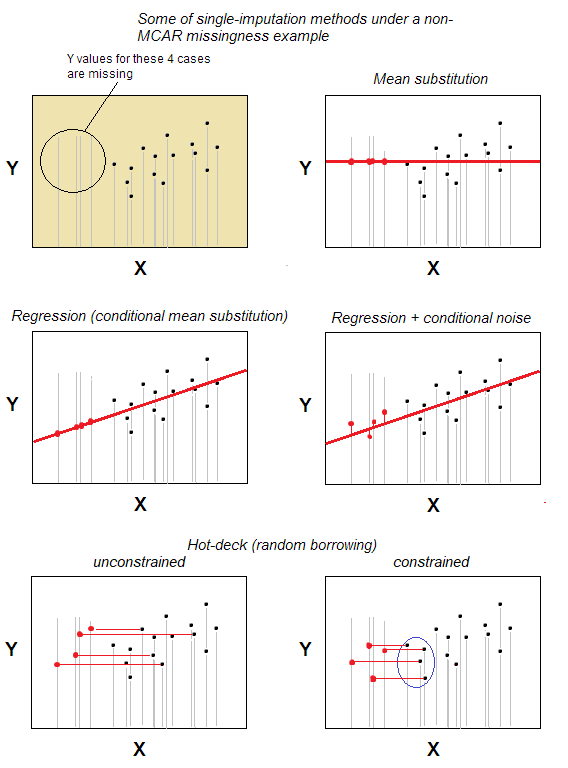
*Imputation of missing data*. The macros perform hot-deck imputation of missing values, borrowing valid values from cases which are similar to cases with missing data by some background characteristics. A separate macro performs an arbitrary, user-defined borrowing of values from some cases by other cases.

*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.

* [!KO\_HDIMPUTE](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK) and [!KO\_HOTDECK](#_MACRO_!HOTDECK:_HOT-DECK_1) perform the same task – hot-deck imputation, but these macros differ in options and algorithms, so result of imputation, even under the same random number seed, will usually be not identical with them. The method of hot-deck imputation is that a case with missing value receives a valid value from some case randomly chosen from those cases that are sufficiently similar to the missing one, based on *background* variables specified by the user (these variables are also called “deck” variables). Hot-deck imputation is popular because it is both simple in idea and, at the same time, suitable for situations where such simple methods of processing missing values as listwise deletion or mean/median substitution will not do for the reason the missings are allocated in the data not chaotically – not according to MCAR pattern (Missing Completely At Random). The mentioned background variables must be categorical. Note generally, that any imputation is recommended to do only if there is no more than about 20% of cases are missing data in a variable.
* [!KO\_BORROW](#_MACRO_!BORROW:_BORROWING_2) is intended for manual, user-defined throwing of values (any, not necessarily valid, in general case) over from case to case within a variable (“vertically”). This macro could be used, in particular, in order to extend a made imputation solution from one to multiple variables.

**Fig. 1. Hot-deck imputation along with some other methods of single imputation**. Hot-deck imputation does not model or estimate values for the missings, it selects them randomly from the abundance of present valid cases, and if needed – under restriction conditions.



# MACRO !KO\_HDIMPUTE: HOT-DECK IMPUTATION (THOROUGH ALGORITHM)

Version 3, Jun 2020 (Version 1, Sep 2011). Tested on SPSS Statistics 20, 22, 25.

*This macro needs SPSS Statistics 17 or higher.*

!KO\_hdimpute vars= *income nchildren* /\*Numeric variables with missing values, name-by-name

/nmiss= 40 36 /\*How many missing values to attempt to impute in each:

/\*integers by the number of VARS, *integer* ALL

/bgvars= *marriage position agegr* /\*Numeric categorical background variables

/\*on which to match

/match= /\*Matching type: search for full match (ALL, default),

/\*partial match (ANY *number*), hierarchical partial match (HIE *number*)

/varsbg= *income* /\*Take account of correlations these variables (name-by-name list of

/\*variables from VARS)

/discret= /\*If VARSBG specified: scale variables in this list

/mult= ATONE /\*For multiple VARS: try to impute all missings in a recipient "at one"

/\*by the same donor (ATONE, default), or impute them independently (INDEP)

/samp= NOREPLACE /\*Use donor: strictly once (NOREPLACE, default),

/\*or no more than this num of times (number)

/\*or allow repeated usage without without limitation (REPLACE)

/print= YES /\*Dynamic printout of the imputation process: YES или NO (default).

Minimal specification VARS, NMISS, BGVARS.

The macro performs operation of hot-deck imputation on variables with missings (imputation variables). Each case with missing value is called a recipient. Cases similar to the recipient but having valid values in the same variable appear to be potential donors; among those, one case gets selected randomly and becomes the donor of the valid value for the recipient. If there are no potential donors, the recipient is left with missing value (failed imputation).

In this macro, you must specify how many missings in each imputation variable it should try to substitute. You can choose a regime via which cases similar to the recipient on the background variables should be searched. Optionally you can take account of associations between the imputation variables. Finally, you can control repeated use of a case as a donor.

For each variable *NAME* of VARS, the macro creates in the working dataset: (1) variable IMPUTE\_*NAME* with imputed values in addition to the originally valid ones; this variable can be used in subsequent analyses; (2) variable DONOR\_*NAME*, that shows, for each case-with-missing in the dataset, its final state: either the case number of the donor case which imputed here or the note that the imputation did not occur due to no donor found by the algorithm.

The macro creates temporary variables with names ending by *#$#$*. Therefore it is recommended that the input dataset lacks variables with names having that ending.

The result depends on random numbers and changes from a run to a run. You can manage random number seed (in order to repeat a result, for example) in SPSS menu Transform – Random Number Generators.

*Speed performance*. The macro is not fast runner, because it searches a donor for each case with missings or even for each missing value one at a time. Because of this, for large datasets, many variables VARS, or plenty of missings NMISS, a faster [!KO\_HOTDECK](#_MACRO_!HOTDECK:_HOT-DECK_1) may be recommended, although it is less flexible and thorough. But see also EXAMPLE 3. Comparison of the two macros:

|  |  |
| --- | --- |
| !KO\_HDIMPUTE | !KO\_HOTDECK |
| Slower | Fast (recommended for big dataset, many variables or many missings) |
| Imputes a specified number of missings | Imputes all missings |
| If there is a donor, imputation is guaranteed | If there is a donor, imputation is very likely |
| Allows partial match on the set of background variables | No |
| Donor is being selected from suitable candidates randomly; in partial match regime the probability of selection depends on the strength (potentiality) of a candidate | Donor is being selected from suitable candidates randomly (they all are equipotential) |
| Can take account of associations between imputation variables (constrained hot-deck) | No |
| There are modes “at one” and “independently” | Mixed mode |
| Can prohibit repeated use of a donor. Can set a limit to the number of times to use | Can attenuate probability of repeated use of a donor |
| Order in VARS may be of significance | Order in VARS plays almost no role |
| Supports dataset case filter | No |

**Algorithm**

Principle: to consider each individual missing value and seek a donor to it. !KO\_HDIMPUTE processes variables VARS one at a time, and within each of them cases with missings (*the recipients*) are processed in turn.

1) Set of recipients. At processing of a current variable, missing cases are separated from valid cases and are sorted in random order, and NMISS first of them are appointed to undergo imputation (be the recipients). Also, the missing cases that had already been processed (were recipient) at the preceding variables, are forced into the head of that random queue of NMISS recipients (and in random order too) – to be imputed prior of the rest: thus, the continuity among VARS in respect to the set of recipients is kept.

2) Selecting the donor. The recipient being processed (in the queue of NMISS ones) is being searched candidates to: *potential* donors. These are cases valid on the current variable, and which similarity on the background variables BGVARS with the recipient is not below the threshold set by condition MATCH. The magnitude of the similarity surpassing the threshold can be individual in potential donors and is called their strength, or *potentiality P*. A potential donor with maximal value *P2U* (where *U* is a random number from uniform distribution, generated at the spot and independently for each potential donor) realizes as the *donor* and imputes its value into the recipient. In other words, the donor is selected randomly with the probability of selection proportional to the squared potentiality. See *Note*.

3) Imputation. Under MULT=ATONE the donor imputes “at one” also into those subsequent VARS of the recipient where the latter is missing, and where the donor can offer a valid value (i.e., is valid in those VARS). Under MULT=INDEP the donor limits itself imputing only in the current variable, so that imputation of different VARS becomes independent (will likely be done by different donors for the same recipient case).

The imputed by a valid value recipient cannot become a donor of that imputed value (snowballing imputation is prohibited). If the recipient was originally valid in other than the current variable, it is able to become a donor when that variable is processed.

*Note*: When s/c VARSBG (see) is specified, variables VARS themselves are considered in their initial state as one more set of background variables; and always with the threshold condition equivalent to “MATCH= ANY 1”. So then, only those valid cases will be considered potential donors, which satisfy *both* threshold conditions: this one and the one that was specified in MATCH s/c. As for the magnitude of potentiality *P* of potential donors, it is the *sum* of the two similarities: according to this and according to that conditions. Let for example, VARS= V1 V2 V3, BGVARS= G1 G2 G3, MATCH= ANY 2, VARSBG= V1 V2 V3. Then, to hit among potential donors, a valid value must not only coincide with the recipient on some two of G1, G2, G3, but also be similar to the recipient on at least one of V1, V2 or V3 (which logically requires the recipient to be valid in at least one of these VARSBG variables). Let the recipient be missing in V1 and be valid in V2, V3, and some potential donor coinciding with it only on V3 and also on all the G1, G2, G3. Then the strength *P* of this potential donor = 1+3=4.

***Subcommands***

**VARS**

Name-by-name list of numeric variables with missings that you want to subject to imputation. Missings can be system-missing as well as user-missing, replaced will be both. Specifying a variable having no missings in the list leads to no error and is permissible.

Order of the variables in the list may have significance, especially in conditions /SAMP=NOREPLACE and /MULT=INDEP under limited pool of potential donors and fairly long VARS list. Since the missings get all imputed first in one variable, then in another, the variables standing in the tail of VARS may, in principle, experience shortage of donors generally or shortage of stronger (more potential) donors.

**NMISS**

Specify how many missings you want to replace with valid values in each VARS variable. Specify individual number for each variable (then NMISS will contain as much numbers as there are VARS variables), or specify single number with the keyword ALL after it (the number will propagate over all the VARS). The numbers should be nonnegative integers.

Maximal permitted number is 1000. It coincides with the limit for macro-cycles in SPSS Statistics. But you will be able to specify a bigger number if you raise the said limit by command SET MITERATE (see *Command Syntax Reference*, SET command). In any case, if you specify in NMISS a number greater than the set limit for the number of macro-cycles, SPSS Statistics will not run the macro and will issue an error.

If the specified number *n* for a variable is less than there are really missings in the variable, then *n* missings will randomly be selected for imputation procedure, while the rest missings in the variable will be ignored. If you specify *n* greater than there are missings, it will not spoil the result but the macro will be working longer because of idle wheeling.

There exists the continuity among VARS variables: cases imputed in past variables get imputed first of all when the next variable is processed.

You may use filter to select cases for the imputation (see “Special regimes”).

**BGVARS**

Specify name-by-name list of background variables on which the recipient will be compared with other cases in search of potential donors. The variables must be numeric. They must be categorical; and the number of categories affects the search success of potential donors: the fewer are categories in a background variable, the easier potential donors are to find. A continuous variable - discretize it first into a categorical. If you don’t have any background variables (which means that any valid case in the dataset can be a potential donor for you) just create a variable with valid value constant for all cases, and specify the variable.

Only valid values in BGVARS variables can be the base of affinity of a recipient and a donor.

In the list, there should be no variables of VARS list (see special subcommand VARSBG for them).

**MATCH**

This subcommand plays role when background variables are multiple, and it sets the regime of matching of cases on the background variables. Matches only by valid values are considered. Select:

ALL - (default) In order a valid case to enroll for potential donors, it must agree with the recipient on all of background variables: it is the regime of complete match.

ANY *n* - In order a valid case to enroll for potential donors, it must agree with the recipient on any in BGVARS list *n* variables. The more there are any variables in BGVARS list on which there is agreement, the more potential becomes this case as a potential donor.

HIE *n* - In order a valid case to enroll for potential donors, it must agree with the recipient on first in BGVARS list *n* variables. The more there are first variables in BGVARS list on which there is agreement, the more potential becomes this case as a potential donor. This option is sensitive to the order of variables in BGVARS list; write more important, for matching, variables earlier in the list.

Number *n* must be integer from 1 to number\_of\_background\_variables-1.

Selection on the background attributes requested in MATCH yields a collection of potential donor cases for the current recipient. If the list of potential donors happens empty, no imputation for the recipient will take place. If there are more than one potential donor, the choice of the donor from them is done randomly. In the regimes of partial match – MATCH= ANY *n* or HIE *n* – the picked potential donors may differ by their potentiality *P* (the degree of match with the recipient), and the random choice among them is weighted by the potentiality squared: the donor comes as the case with maximal quantity *P2 \* random number*, or, to put equivalently, *P* \* rv[~U(0*, P*)], U being uniform distribution.

EXAMPLE 1.

!KO\_hdimpute vars= v1 /nmiss= 120 /bgvars= region sex agegr marriage /match= HIE 3.

* Imputation of missings in *V1*, controlling for the region of residence of respondents, their sex, age group, marital status. For a valid case to have the chance to become a donor, it must match a recipient case on at least three first of these four variables. If it matches on all the four – its chance to become a donor increases.
* 120 randomly selected missings will undergo imputation. Or that much as there are in *V1*, if they are less than 120.

**VARSBG, DISCRET**

These subcommands act if there are multiple variables in VARS. Subcommand VARSBG allows to do “constrained” hot-deck imputation, that is, taking into account (pre-imputation) associations/correlations between variables VARS themselves (see the picture in the beginning). In VARSBG, specify name-by-name those variables of VARS, associations between which and each being imputed variable of VARS you want to take into account when imputing the latter. You may indicate one, several or all of VARS list variables in the VARSBG list.

The problem of taking account of associations between imputation variables is solved technically by that VARSBG list forms one more set of background variables, besides BGVARS. This toughens requirements for potential donors: a donor should match the recipient not only on BGVARS (by MATCH condition) but also match it on (at least one of) VARSBG. Matching on VARSBG is the protection of associations/correlations among VARS from its “erosion” by imputation process. When a recipient case is missing in variable X of VARS but is not missing in some variables Y of VARSBG, it is selected a donor which coincides with it on those variables Y. The donor must match it at least on one of Y, but the potentiality of the donor is the higher the higher is the number of Y it matches the recipient on (see above “Algorithm, *Note*”). If a recipient is missing in all VARSBG, it will remain unimputed because it is unclear what donor to look after for it (you may complete imputation in such recipients by a second run of the macro, now without specifying VARSBG).

Subcommand DISCRET is auxiliary to VARSBG subcommand. If there are non-categorical variables present in VARSBG – i.e. continuous, scale – you must list them in DISCRET, for the macro to discretize them (internally, without altering the data in the dataset) into categorical. This is necessary because background variables, which VARSBG are, must be categorical. The macro uses SPSS command RANK /NTILES(*k*) for that binning. It partitions the range of a quantitative variable with > *k* unique values into about *k* approximately equal, by the number of cases, groups. By default, *k*=5, but you can change this parameter, adding subcommand /KBINS= *number* to the macro call.

EXAMPLE 2.

!KO\_hdimpute vars= v1 v2 v3 /nmiss= 45 76 61 /bgvars= region sex /varsbg= v1 v2 v3 /discret= v1 v2.

* Imputation with the control of region of residence and sex of a respondent, and also with the intention to account of, where possible, associations between imputation variables VARS themselves: they are listed in s/c VARSBG. Moreover, *V1* and *V2* of them are not categorical, so they demand to be binned (s/c DISCRET).

**MULT**

The subcommand acts if there are multiple variables in VARS. It sets the manner of imputation into a case that is missing in more than one VARS variable.

ATONE - (default) imputation “at one”. The donor case imputes in the recipient’s missing value of the current (being processed) variable and also of all the subsequent variables of VARS list where the recipient is missing while the donor is valid.

INDEP - imputation “independent”. The donor case imputes in the recipient’s missing value of the current variable only.

MULT=ATONE means that the recipient with multiple missings will likely be replaced by some same donor (the act can be understood as creation of a duplicate case out of the donor), whereas under MULT=INDEP such recipient is likely to be imputed by different donors (the act can be understood as creation of a hybrid case out of the donors). While duplicating of cases in a dataset usually preserves the correlational configuration, hybridization of cases in a dataset with pronounced correlations usually tends to attenuate them.

Support of correlations among VARS depends also on special s/c VARSBG (see). While MULT=ATONE intends to create a donor’s duplicate in place of a recipient with multiple missings, VARSBG makes use of non-missings of a recipient, if there are any, to select a correlation-saving donor for it.

EXAMPLE 3. Speeding up the job with the help of [!KO\_BORROW](#_MACRO_!BORROW:_BORROWING_2) macro in the situation when missing cases are “all through”.

!KO\_hdimpute vars= v1 v2 v3 v4 v5 v6 v7 v8 /nmiss= 100 ALL /bgvars= region sex /mult= ATONE.

* Missings in variables *V1 – V8* mostly belong to the same cases, i.e., they are predominantly “all through” missing cases. The macro will impute in the MULT=ATONE regime (also default), that is, it will try to replace, if possible, all missing values of a recipient case by one donor case.
* Eight VARS variables is many: the macro will prepare and execute not quickly. An alternative and fast approach in this situation, giving fairly similar result, is to impute only one variable and then propagate its results over other variables, by macro !KO\_BORROW.

!KO\_hdimpute vars= v1 /nmiss= 100 /bgvars= region sex.

!KO\_borrow vars= v2 v3 v4 v5 v6 v7 v8 /donor= donor\_v1 /cap= 'impute\_'.

* Only one variable, *V1*, is processed in !KO\_HDIMPUTE, which is quicker. At output there will be *IMPUTE\_V1* variable and *DONOR\_V1* variable; the latter containing case numbers of the donors having “hit”.
* !KO\_BORROW, by using this variable in its DONOR subcommand, forces the above donors to give their valid values into variables *V2 – V8*. Since missings in these variables are for the most part the same cases as were missing in *V1*, the result of such action is close to what had been in the first !KO\_HDIMPUTE with all 8 variables.

**SAMP**

This subcommand is in charge of repeated using of cases as donors. By default/unspecification and with SAMP=NOREPLACE, a donor case can impute only in one recipient case; in other words, donor selection is a “selection without replacement” of it w.r.t. the pool of potential donors. With SAMP=REPLACE, a case having become a donor for a recipient case, keeps a chance to impute in other recipient cases; in other words, donor selection is a “selection with replacement”. You may also specify SAMP=*number* (set positive integer), then a case won’t be able to impute more than in this number of recipient cases. SAMP=1 is equivalent to SAMP=NOREPLACE.

**PRINT**

This subcommand lets output to Viewer reporting about the flow of imputation, at run time. This is convenient to follow its current and order, when there are many missings requested to impute. Specify PRINT=YES. By default, PRINT=NO and no reporting.

***Special regimes***

The macro does not obey case weighting. It is not suited for split state of the dataset (SPLIT FILE) and for selection of cases by USE command. Do not run before the macro temporary operations (under TEMPORARY command). The macro obeys filtering (FILTER), with which you can manage the selection of cases for participating in imputation. The filtered-out valid case will not act a donor, while a filtered-out case-with-missing will receive status “did not undergo imputation” in the output.

***Some questions***

*What’s the difference between BGVARS and VARSBG subcommands?* Both subcommands set lists of background variables (to search a donor fitted for the recipient). Their difference is not theoretical but rather convenience. BGVARS is a required subcommand where one has to indicate variables outside of the VARS list. It is assumed that there are no or few missings in BGVARS variables, so that they themselves don’t need imputation. VARSBG is an optional subcommand with which you can draw the imputation variables VARS to play background variables too. Match condition on BGVARS variables is specified by you in MATCH s/c (by default it is “ALL”). Match condition on VARSBG variables is preset by the macro and is always equivalent to “ANY 1” (such a liberal condition is explained by that in input VARS variables there can be many missings). BGVARS variables must be already categorical at input. VARSBG variables, if they are not categorical, need be indicated in DISCRET, the macro first to bin them internally.

*What’s the difference between SAMP and MULT subcommands?* SAMP is about whether in one or in multiple *cases with missing* valuesone donor case is allowed to impute in. MULT is about whether one or multiple *missing values* in a recipient case are allowed to impute in by a donor case. MULT plays a role if a recipient may have multiple missings – i.e., if VARS is more than one variable.

# MACRO !KO\_HOTDECK: HOT-DECK IMPUTATION (QUICK ALGORITHM)

Version 3, Jun 2020 (Version 1, Sep 2011). Tested on SPSS Statistics 20, 22, 25.

*This macro was written by Teresa Myers (teresaannmyers@gmail.com) and later modified/enhanced by me.*

!KO\_hotdeck vars= *income nchildren* /\*Numeric variables with missing values, name-by-name

/bgvars= *marriage position agegr* /\*Numeric categorical background variables

/\*on which to match

/samp= REPLACE /\*Seek to use donor only once (RESTRREPLACE, default),

/\*or allow repeated usage (REPLACE).

Minimal specification VARS, BGVARS.

The macro performs, like macro [!KO\_HDIMPUTE](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK), operation of hot-deck imputation on variables with missings. Another algorithm (technical solution) is used than in !KO\_HDIMPUTE, so results, being as valid, come out different here.

For each variable *NAME* of VARS, the macro creates in the working dataset: (1) variable IMPUTE\_*NAME* with imputed values in addition to the originally valid ones; this variable can be used in subsequent analyses; (2) variable DONOR\_*NAME*, that shows, for each case-with-missing in the dataset, its final state: either the case number of the donor case which imputed here or the note that the imputation did not occur due to no donor found by the algorithm.

The macro creates temporary variables with names ending by *#$#$*. Therefore it is recommended that the input dataset lacks variables with names having that ending.

The result depends on random numbers and changes from a run to a run. You can manage random number seed (in order to repeat a result, for example) in SPSS menu Transform – Random Number Generators.

!KO\_HOTDECK is much quicker than !KO\_HDIMPUTE and easily copes with large datasets containing many cases with missing values. !KO\_HOTDECK has no limitation for the number of such cases to impute: it always attempts to impute to all cases-with-missings whatever their count. On the other hand, !KO\_HOTDECK supports only full-agreement regime of matching on background variables (this corresponds to specification /MATCH=ALL of macro !KO\_HDIMPUTE), which makes it less flexible than !KO\_HDIMPUTE. The algorithm in !KO\_HOTDECK is not absolutely thorough, therefore on rare occasions it can leave a missing value unimputed despite that there exists a suitable donor. See the comparison table of the two macros in the description of !KO\_HDIMPUTE.

**Algorithm**

Principle of !KO\_HOTDECK: collect together similar cases, valid and missing, and undertake transfers of values among them. (1) Dataset cases are sorted by the BGVARS variables. Consequently, cases identical on all BGVARS form groups of neighbouring cases; and those go in random order there. Within a group, *recipients* (cases with missings in VARS) are mixed with valid cases (*potential* donors). (2) On running across cases in each such group, the macro does the following in each VARS variable one by one: (3) if the case is missing (a recipient) in the variable, the macro examines its neighbour cases (up to 3 positions below and above it) and takes the first encountered valid of them as the donor, and it imputes the value into the recipient. Since the sequence of this examining of neighbours is always the same, the recipient that is missing in several of the VARS will receive imputations in those VARS by the same or by different donor cases, depending on how saturated by valid values are the recipient’s neighbours. If a case has become a donor to a recipient in the first variable and the case is valid in other VARS too, then, as a rule (albeit not always), the case will become the donor to it in those other VARS either.

Recipient case imputed with a valid value cannot become a donor of this imputed value (snowballing imputation is prohibited).

EXAMPLE 1.

!KO\_hotdeck vars= v1 v2 /bgvars= region sex agegr marriage.

* Impute in missings of *V1* and *V2*, controlling for the region of residence of respondents, their sex, age group, marital status. For a valid case to have the chance to become a donor, it must match a recipient case on all the four variables.

***Subcommands***

**VARS**

Name-by-name list of numeric variables with missings that you want to subject to imputation. Missings can be system-missing as well as user-missing, replaced will be both.

**BGVARS**

Specify name-by-name list of background variables on which the recipient will be compared with other cases in search of potential donors. The variables must be numeric. They must be categorical; and the number of categories affects the search success of for potential donors: the fewer are categories in a background variable the easier potential donors are to find. A continuous variable - discretize it first into a categorical. If you don’t have background variables (which means that any valid case of the dataset can be a potential donor, for you), just create a numeric constant variable with a valid value in the dataset and indicate that variable.

Variables from VARS cannot enter the list. If in some of BGVARS there are missing values, those cases won’t be able to become donors, because !KO\_HOTDECK accepts only complete identity on all BGVARS between a recipient and a donor, and a missing value cannot be a base of identity.

**SAMP**

This subcommand is in charge of repeated using of cases as donors. By default/unspecification and with SAMP=RESTRREPLACE, repeated use of cases as donors is restricted: it is rarely that a donor case will be able to impute into more than one recipient case (and when it happens, it is not in the same VARS variable). With SAMP=REPLACE, a case having become a donor for a recipient case retains considerable chance to impute into other recipient cases.

***Special regimes***

The macro does not obey case weighting. It is not suited for split state of the dataset (SPLIT FILE) and for selection of cases by USE command. Do not run before the macro temporary operations (under TEMPORARY command). The macro does not obey filtering (FILTER), unlike macro !KO\_HDIMPUTE.

# MACRO !KO\_BORROW: BORROWING VALUES FROM SPECIFIED OTHER CASES

Version 1, Dec 2012. Tested on SPSS Statistics 20, 22, 25.

*This macro needs SPSS Statistics 17 or higher.*

!KO\_borrow vars= *v1 v2 v3* /\*Variables in which to carry over values, name-by-name

/donor= *donor* /\*Variable with donor case numbers, what to borrow values from

/ifrecval= ANY /\*Replace value if it is: any (ANY), valid (VALID),

/\*missing (MISSING, default), system missing (SYSMIS), empty string (BLANK)

/ifdonval= VALID /\*Replace by value if it is: any (ANY), valid (VALID, default),

/\*missing (MISSING), system missing (SYSMIS), empty string (BLANK)

/cap= '*new*\_' /\*Optional: prefix to create new variables.

Minimal specification VARS, DONOR.

This macro accomplishes borrowing of values among cases: a carrying over of values from cases to cases within the same variable. Whence (from which case) must the current case copy its value should be written down in a special variable which name you will specify in DONOR subcommand.

If for the current case (the recipient) the DONOR variable is positive integer *i* (and not greater than the number of cases in the dataset) then the recipient will borrow the value from the *i*-th case (the donor), if in regard to the two values ­– the original recipient’s and the donor’s – both conditions, IFRECVAL и IFDONVAL, are being held, respectively. The macro carries over values “vertically”, within the same processed variable. You may process one or several such variables in a single run.

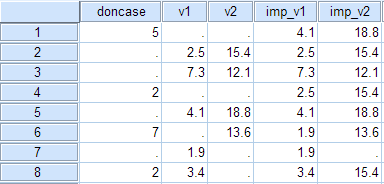
In particular, the macro is suited to propagate the results of hot-dock imputation of missings from one variable onto multiple variables – from the same donor cases. [!KO\_HDIMPUTE](#_МАКРОС_!HDIMPUT:_КОЛОДНАЯ_(HOT-DECK) and [!KO\_HOTDECK](#_MACRO_!HOTDECK:_HOT-DECK_1) save variable DONOR\_*NAME* to the dataset, that shows what cases acted as donors. You indicate that variable in s/c DONOR of macro !KO\_BORROW. See EXAMPLE 3 in the description of !KO\_HDIMPUTE.

In !KO\_BORROW, if a recipient case is itself a donor to some other cases, it can give them only those its values which it had initially. So, recipients in the dataset always borrow just initial values of their donors.

The macro creates temporary variables with names ending by *#$#$.* Therefore it is recommended that the input dataset lacks variables with names having that ending.

EXAMPLE 1.

!KO\_borrow vars= v1 v2 /donor= doncase /cap= imp\_ .



* Variables within which cases must borrow values from other cases are *V1* and *V2*. Variable DONCASE contains case numbers of donor cases, for the current cases as recipients. Requested is to output new variables with prefix *IMP\_*.
* By default of both subcommands IFRECVAL and IFDONVAL, the condition of substitution is this: a recipient value is to be replaced by the donor’s one if the former is missing while the latter is valid.
* Result: missings were substituted by those values, according to that condition.

***Subcommands***

**VARS**

Specify name by name variables within which to do borrowing of values across cases. The variables may be either numeric or string, all of the same type.

**DONOR**

Specify name of numeric variable which values are the case numbers of the donor cases (for the current cases, recipients). Any values in this variable except valid positive integers are ignored by the macro (and so the current case won’t be a recipient).

**IFRECVAL, IFDONVAL**

Two subcommands putting the condition (term) under which the borrowing of value from a donor will come true. IFRECVAL is the condition “if the value of the recipient case (i.e., the current case) in the given variable of VARS is…”. IFDONVAL is the condition “if the value of the donor case (i.e., of the mentioned in DONOR variable) in that given variable of VARS is…”. One of the following keywords:

ANY any value

VALID valid value

MISSING missing (system- or user-)

SYSMIS system-missing (usable with numeric VARS)

BLANK blank (absent) text, valid or missing (usable with string VARS)

For example, /IFRECVAL=ANY /IFDONVAL=ANY allows replacement of any value of a recipient by any value of its donor. By default/unspecification, IFRECVAL is set as MISSING; and by default/unspecification, IFDONVAL is set as VALID; thus, if you omit both subcommands, there will take place imputation of missings by valid data.

Attention, in SPSS Statistics below version 21 don’t use conditions VALID and MISSING for string variables wider than 8 bytes.

**CAP**

Bу default, the macro modifies variables VARS themselves. If you don’t want to affect them and instead want the macro to create new variables out of them, specify here the prefix into the names of the new variables. You may take the prefix in quotes or apostrophes (they might be recommended if you end the prefix by period as separator: ‘x.’). If the input dataset already contains variables of these names, those will be overwritten.

***Special regimes***

The macro ignores dataset splitting (SPLIT FILE) and temporary (under TEMPORARY) transformations. The macro ignores case filtering (commands FILTER or USE) and weighting. If you need to forbid some cases to be recipients or donors – insert the corresponding edit right in the DONOR variable. Any command standing before the macro call and removing some cases (for example SELECT IF) radically affects results since it shifts the ordinal case numbers of the remaining cases.