Institute for Employment Research

The Research Institute of the Federal Employment Agency

IAB-Discussion Paper 16/2018

Articles on labour market issues

R Package hmi: A Convenient Tool for Hierarchical Multiple Imputation and Beyond

Matthias Speidel
Jörg Drechsler
Shahab Jolani

ISSN 2195-2663

R Package hmi: A Convenient Tool for Hierarchical Multiple Imputation and Beyond

Matthias Speidel (IAB)
Jörg Drechsler (IAB)
Shahab Jolani (Maastricht University)

Contents

Abstract

Applications of multiple imputation have long outgrown the traditional context of dealing with item nonresponse in cross-sectional data sets. Nowadays multiple imputation is also applied to impute missing values in hierarchical data sets, address confidentiality concerns, combine data from different sources, or correct measurement errors in surveys. However, software developments did not keep up with these recent extensions. Most imputation software can only deal with item nonresponse in cross-sectional settings and extensions for hierarchical data – if available at all – are typically limited in scope. Furthermore, to our knowledge no software is currently available for dealing with measurement error using multiple imputation approaches.

The `R` package `hmi` tries to close some of these gaps. It offers multiple imputation routines in hierarchical settings for many variable types (for example, nominal, ordinal, or continuous variables). It also provides imputation routines for interval data and handles a common measurement error problem in survey data: Biased inferences due to implicit rounding of the reported values. The user-friendly setup which only requires the data and optionally the specification of the analysis model of interest makes the package especially attractive for users less familiar with the peculiarities of multiple imputation. The compatibility with the popular `mice` package ensures that the rich set of analysis and diagnostic tools and post-imputation commands available in `mice` can be used easily once the data have been imputed.

Zusammenfassung

Anwendungen von Multipler Imputation sind längst über den klassischen Kontext der Behandlung von fehlenden Beobachtungen in Querschnittsstudien heraus gewachsen. Heutzutage wird Multiple Imputation auch verwendet um fehlenden Werten in hierarchischen Datensätzen zu imputieren, um Vertraulichkeits-Interessen zu begegnen, um Datensätze aus verschiedenen Quellen zu kombinieren oder um Messfehler aus Erhebungen zu korrigieren. Die meiste Imputationssoftware kann allerdings nur mit fehlenden Beobachtungen in Querschnittsdaten umgehen und Erweiterungen für hierarchische Daten - sofern überhaupt vorhanden - sind typischerweise in ihrem Umfang begrenzt. Unserem Kenntnisstand nach, ist aktuell keine Software für den Umgang mit Messfehlern, basierend auf Multiplen Imputationsmethoden, vorhanden. Das R-Packet `hmi` versucht einige dieser Lücken zu schließen. Es bietet Multiple Imputationsroutinen in hierarchischen Settings für viele Variablentypen (zum Beispiel nominal, ordinal oder stetige Variablen). Zudem stellt es Imputationsmethoden für Intervalldaten bereit und behandelt ein übliches Messfehlerproblem in Befragungsdaten: Verzerrungen aufgrund impliziten Rundens der berichteten Werte. Der nutzerfreundliche Aufbau, der nur die Daten und optional eine Spezifizierung des Analysemodels benötigt, macht das Paket besonders attraktiv für Nutzer die weniger vertraut mit den Besonderheiten von Multipler Imputation sind. Die Kompatibilität mit dem populären Paket `mice` stellt sicher, dass der reichhaltige Satz an Analyse- und Diagnosewerkzeugen, und Befehlen für das Imputationsergebnis aus `mice`, einfach angewandt werden kann, sobald die Daten imputiert wurden.

JEL classification: C38; C83

Keywords: hierarchical data, multiple imputation, multilevel models, measurement error, heaping, R

1 Introduction

Forty years after Donald Rubin's seminal paper (Rubin, 1978) which introduced the concept of multiple imputation, the approach has been shown to be useful in many contexts going far beyond the classical item nonresponse in cross sectional surveys for which it was originally proposed (Reiter/Raghunathan, 2007). Today, multiple imputation is used to deal with nonresponse in hierarchical data sets (Carpenter/Kenward, 2013: chap. 9), address confidentiality concerns by disseminating synthetic data instead of the original data (Drechsler, 2011), concatenate files from different data sources (Rubin, 1986; Rässler, 2003; Reiter, 2012), address measurement error in self-reported health information (Schenker/Raghunathan/Bondarenko, 2010), handle changes in the coding of variables in longitudinal studies (Clogg et al., 1991; Schenker, 2003), or impute plausible values for coarse data (Taylor/Schwartz/Detels, 1986; Heitjan/Rubin, 1990; Raghunathan et al., 2001). As discussed in Heitjan/Rubin (1991) coarse data are data for which the true values are not observed in a precise way. This includes missing data as a special case, but also rounding, grouping, censoring and interval data. Examples of applications of multiple imputation for coarse data include Gartner/Rässler (2005); Jenkins et al. (2011); Drechsler/Kiesl/Speidel (2015).

While classical imputation methodology as discussed for example in Rubin (1987) or van Buuren (2012) is sufficient for some of these applications, adjusted methodology is required for others. However, although all major statistical software such as `SPSS`, `Stata`, `SAS`, or `R` offer multiple imputation routines today, the available methodology is typically limited to the classical methodology for cross-sectional surveys. Some software also provides methods for dealing with hierarchical data structures, but as we will illustrate in Section 2.6, current implementations are limited in scope. With the exception of the recently implemented software package `synthpop` (Nowok/Raab/Dibben, 2016) which was specifically developed for generating synthetic data sets for disclosure protection, no software exists to our knowledge for applications such as the coarse data problem discussed above, which require modifications of the traditional multiple imputation framework.

The `R` package `hmi` closes some of the gaps of currently available software by offering four important contributions:

1. It offers imputation routines for hierarchical data using multilevel (mixed-effects) models for all variable types based on the sequential regression approach, which unlike the joint modeling approach can also handle item nonresponse if random slope models need to be estimated (see Section 2.4 for details)

2. It provides routines for dealing with rounding in reported values based on the methodology proposed in Heitjan/Rubin (1991).

3. It offers routines for imputing plausible values if it is only known (for some of the observations) that the exact value lies in certain intervals, for example if the data are censored. Currently, such imputation routines are only available in `Stata`.

4. It allows to deal with item nonresponse, interval information and rounding within the same variable simultaneously following the approach described in Drechsler/Kiesl/Speidel (2015).

The package also offers imputation tools for "classical" missing data problems by calling imputation routines available in the popular multiple imputation package `mice` (van Buuren/Groothuis-Oudshoorn, 2011). Since the objects generated using `hmi` are structured similar to objects generated using `mice` (both are `mids` objects), the rich set of analysis and diagnostic tools and post-imputation commands available in `mice` can be used easily once the data have been imputed. Furthermore, the package provides imputation routines for semi-continuous variables, that is, variables which have a spike at one value (typically zero), but can be considered continuous otherwise. These imputation routines are available in several software packages, but are not offered in `mice`.

To facilitate the usage of the package for less experienced users, the selection of suitable imputation models is highly automated, that is, the user only needs to provide the data. The package will identify the most appropriate imputation models for each variable with missing values using decision rules described in Section 5 of this paper. Additionally, the user can specify the substantive model he or she wants to run on the imputed data set. In this case `hmi` will use the same set of predictors and the same functional form as the substantive model for all imputation models in an effort to make the congeniality assumption more plausible. As discussed in Meng (1994), congeniality between the imputation model and the substantive model is important to avoid biased inferences based on the imputed data. We illustrate in Section 2.3 that specifying the substantive model is especially important if multilevel models will be fitted at the analysis stage since this will ensure that the hierarchical structure of the data will also be taken into account at the imputation stage. The package is available at `https://cran.r-project.org/package=hmi`.

The remainder of the paper discusses the main contributions of the package and provides detailed illustrations on how the package can be used. Specifically, Sections 2 to 4 address multiple imputation for hierarchical, interval and rounded data. Each section starts by illustrating the inferential problems caused by the various data deficiencies followed by a brief review of the required multiple imputation methodology for addressing the said problem. Limitations of currently available software and our contributions are also discussed. Section 5 describes the `hmi` package in detail: all mandatory and optional arguments, the internal checks, the handling of the model formula, the types of supported variables, and the implemented convergence checks will be presented. In Section 6 we provide real data applications to illustrate the implementation of the different features of the package. We end with a conclusion.

2 Multiple imputation for hierarchical data sets

Hierarchical data sets are data sets in which individual records are nested within groups. Typical examples include students in the same class or repeated measures of the same individual. In such settings, the assumption of independent observations, needed for the classical linear regression model, does not hold since records belonging to the same group tend to be more homogeneous than records belonging to different groups. To account for these cluster effects, multilevel models (also referred to as random effects or mixed effects models depending on the field of study) are often employed. In the following, we provide a brief summary of the methodology behind multilevel modeling starting with multilevel linear models for continuous variables. Then, we discuss extensions to multilevel generalized linear models for any variable type from the exponential family. A more detailed introduction can be found in any textbook on multilevel modeling, for example in Raudenbush/Bryk (2002). The brief overview will form the basis for our discussion of appropriate imputation strategies for hierarchical data and details about their implementation and available software in Sections 2.3 to 2.7.

2.1 Multilevel linear models

Paraphrasing from Speidel/Drechsler/Sakshaug (2017), multilevel linear models assume a linear relationship between the continuous target variable Y and some covariates X and Z. The effect of X on Y is governed by some global fixed effects β; the effect of Z on Y by some cluster specific random effects γ. Often Z is a subset of X, meaning that variables that are assumed to have a random effect are also included as fixed effect variables in the model.

The standard multilevel model has the form

$$\begin{aligned} y_{ij} &= x_{ij}\beta + z_{ij}\gamma_j + \varepsilon_{ij}, \\ \gamma_j &\sim N(0,\ \Sigma), \\ \varepsilon_{ij} &\sim N(0,\ \sigma^2), \end{aligned} \tag{1}$$

with $j = 1, \ldots, J$ being the index for the clusters, $i = 1, \ldots, n_j$ being the index for the units belonging to cluster j, and n_j being the number of observations in cluster j. The parameter β contains the global fixed effects, similar to the regression coefficients in classical linear regression models. The parameters γ_j are the cluster specific random effects, which are assumed to follow a normal distribution with zero mean vector and variance matrix Σ. These random effects and the normality assumption for them is a key difference to the classical linear regression model. The parameter ε_{ij} is the error term which is normally distributed with zero mean and variance σ^2, which is constant for all clusters.

Multilevel linear models can be generalized to more than two levels and residual variances being heteroscedastic across the clusters. Since `hmi` can only handle two levels of hierarchy and homoscedastic residuals at the moment, we do not cover these extensions here. The interested reader is referred to Raudenbush/Bryk (2002) or Snijders/Bosker (2011) for more details on these topics.

2.2 Multilevel generalized linear models

The step from multilevel linear models to multilevel generalized linear models (mglm) is analogous to the step from classical linear models to generalized linear models (glm). Both enable model estimation for variables from the exponential family using a linear predictor l and a link function f such that $E(Y) = \mu = f^{-1}(l)$. The major difference between mglm and glm is that the linear predictor in mglm also has random effect variables Z with regression coefficients $\gamma = \{\gamma_1, \ldots, \gamma_J\}$ leading to $l = X\beta + Z\gamma + \varepsilon$. These random effects and their covariance matrix Σ also have to be considered when estimating the model.

The link function is defined according to the type of variable that is modeled. For example for continuous variables the identity link is used and for count data the log-link. In general no closed form solution for the parameter estimates exist, so Markov Chain Monte Carlo (MCMC) methods or other iterative procedures are required for estimation (Gelman/Hill, 2006; Hadfield, 2010).

2.3 Dealing with missing values in hierarchical data

Hierarchical data are not spared from nonresponse and multiple imputation can be a convenient strategy to address this problem. Several researchers have shown that ignoring the hierarchical structure at the imputation stage will lead to biased inferences when analyzing the data (Reiter/Raghunathan/Kinney, 2006; van Buuren, 2011; Enders/Mistler/Keller, 2016; Zhou/Elliott/Raghunathan, 2016; Lüdtke/Robitzsch/Grund, 2017). Furthermore, accounting for the clustering by adding indicator variables for the clusters (fixed effects modeling) will still introduce bias if the analysis is based on a multilevel model (Taljaard/Donner/Klar, 2008; Andridge, 2011; Drechsler, 2015; Speidel/Drechsler/Sakshaug, 2017). To avoid this bias due to uncongeniality between the imputation and the analysis model, all manuscripts suggest using multilevel models also at the imputation stage.

2.4 Multiple imputation using multilevel models

With multiple imputation missing values are imputed multiply ($M \geq 2$ times) to be able to take the uncertainty from imputation into account. The imputed values are random draws from the distribution of the missing data given the observed data. Let $D = \{D_{obs},\ D_{mis}\}$ denote the data D separated into an observed part (D_{obs}) and a missing part (D_{mis}) and let θ contain the parameters which govern the distribution of D. To obtain approximate draws from $f(D_{mis} \mid D_{obs})$ multiple imputation repeatedly applies the following two steps:

1. Draw a new set of parameters $\theta^{\star}$ from their posterior distribution given the observed data: $f(\theta \mid D_{obs})$.
2. Draw replacements for the missing values from the predictive distribution of the missing data given the observed data and the drawn parameters from the previous step: $f(D_{mis} \mid \theta^{\star},\ D_{obs})$.

Valid inferences based on the imputed data can be obtained using the generic inferential procedures first described in Rubin (1978). For further details regarding the general properties of multiple imputation we refer to any textbook on multiple imputation, for example Rubin (1987); van Buuren (2012); Carpenter/Kenward (2013).

As pointed out above, if the model to be estimated on the imputed data is a multilevel model, a similar model specification should be used at the imputation stage to ensure unbiased results. Thus, for continuous variables the imputation model should follow the model specification given in Equation (1) and the two generic multiple imputation steps described above consist of the following two steps:

1. Draw a new set of parameters $\theta^\star = \{\beta^\star,\ \gamma^\star,\ \Sigma^\star,\ (\sigma^\star)^2\}$ from their posterior distribution.

2. Generate imputed values by drawing from

$$y_{ij}^{imp} = x_{ij}^{imp}\beta^\star + z_{ij}^{imp}\gamma_j^\star + \varepsilon_{ij}{}^\star \qquad \varepsilon_{ij}^\star \sim N(0,\ (\sigma^\star)^2), \tag{2}$$

where the superscript *imp* identifies all records for which Y is imputed. Unlike in the classical linear regression case, no closed form solutions exist for the posterior distribution of the parameters. Thus, Markov Chain Monte Carlo methods or other approximations (Jolani, 2018) are generally required to update the parameters. We refrain from providing the details of the iterative procedure here for brevity. The interested reader is referred to Goldstein (2011) for a detailed description of Gibbs sampling methods for hierarchical data and to Carpenter/Kenward (2013: chap. 9) and Drechsler (2015) for applications in the missing data context.

2.5 Joint modeling vs. sequential regression for multilevel multiple imputation

Two general strategies exist for imputing missing values if more than one variable is affected by nonresponse: joint modeling and sequential regression. The *joint modeling* approach specifies a joint distribution for all variables with missing data (potentially conditioning on fully observed variables) and draws imputed values based on this distribution. For example, if all variables to be imputed are continuous, a multivariate normal distribution is typically specified for those variables affected by nonresponse. A major drawback of the joint modeling approach in the multilevel context is that it cannot be used if missingness also occurs in the random slope variable(s) (Carpenter/Kenward, 2013; Enders/Mistler/Keller, 2016). Furthermore, the specification of a joint distribution can be difficult, if different variable types need to be modeled.

The *sequential regression* approach (also known as *chained equations* or *fully conditional specification*) does not require modeling the joint distribution directly. Instead, conditional distributions are specified for each variable to be imputed. The variables are imputed sequentially, conditioning on the other variables in the data set. However, some of the predictors in the imputation model might themselves contain imputed values. Thus, the model estimates will change if these imputed values are updated. To account for this, the procedure of sequentially imputing each variable has to be repeated several times, until the draws from the conditional distribution converge to draws from the implicitly specified joint distribution (see Raghunathan et al. (2001) for further details on the sequential regression approach).

A downside of the approach is that convergence is only guaranteed if this joint distribution exists. However, Liu et al. (2014) and Zhu/Raghunathan (2015) show that the joint distribution will exist under rather general conditions and even if this is not the case, inferences based on the imputed data will still be consistent as long as the conditional distributions are correctly specified.

2.6 Existing imputation routines for hierarchical data and their limitations

To our knowledge the only `R` (R Core Team, 2016) packages allowing hierarchical multiple imputation are `jomo` (Quartagno/Carpenter, 2018), `mice` (van Buuren/Groothuis-Oudshoorn, 2011), `micemd` (Audigier/Resche-Rigon, 2018) and `pan` (Schafer, 2016). Currently, `mice` is limited to continuous variables for hierarchical settings and cannot impute other variable types using a multilevel model. `micemd` also provides multilevel imputation functions for binary and integer variables, but not for categorical variables with more than two categories. A downside of `jomo` and `pan` is the fact that they rely on the joint modeling approach, with the drawbacks mentioned in the previous section.

Imputation routines based on multilevel models have also been developed for other statistical software packages: For `SAS` the external macro `MMI_IMPUTE` (Mistler, 2013) can be used. `Mplus` (Asparouhov/Muthén 2010) and the stand alone software `REALCOM-IMPUTE` also offer some multilevel multiple imputation routines. All of these imputation routines also use the joint modeling approach. To our knowledge, the only other software allowing multilevel imputation based on the more flexible sequential regression approach is the recently released standalone software `blimp` (Enders/Keller/Levy, 2017).

2.7 Our contribution for the imputation of hierarchical data

As mentioned in the introduction, `hmi` is designed to provide multilevel imputation routines for many relevant variable types, including semi-continuous variables based on the flexible sequential regression approach. Furthermore, it also offers single level models for all types of variables, for situation where a multilevel model is not applicable.

If an analysis model is specified, the package will automatically use the same set of predictors and the same functional form as the substantive model for all imputation models to avoid introducing bias in the analysis, because relationships which are important to the analyst are not reflected in the imputation models. If no analysis model is given, all variables are imputed using single level models by default. However, if desired, the user can manually specify which imputation models should be used for each variable.

For single level imputation, the package relies on the imputation routines implemented in `mice`. Own code is used for all multilevel imputation routines. The draws from the posterior distribution of the parameters of the multilevel models are obtained using MCMC methods implemented in the `MCMCglmm` package (Hadfield, 2010).

If multilevel imputations are employed, the package also stores the model parameters at each iteration of the MCMC chains, to enable the users to monitor the convergence of the chains. The users can either extract this information to run their own convergence diagnostics or they can rely on the checks implemented in the package. Per default the package runs Geweke's stationarity test (Geweke 1992) on each chain, plots those chains that failed the test and provides some summary information on the number of chains which failed the test (see Section 5.8 for details).

3 Multiple imputation for interval data

Interval data (sometimes called *bracketed response*) comprise all data for which an interval covering the true value is given instead of the exact value. According to this definition both, grouped and censored data can be treated as interval data. With *grouped data*, a set of precise observations is grouped into a single response group. For example in cancer research the number of positive lymph nodes might only be collected in categories 0, 1-3, 4-9 and 10+ (Royston, 2007) or age might only be reported in five year intervals for confidentiality reasons. Grouped data can also arise if surveys aim to maximize response rates for sensitive or difficult questions. For example, in the *Survey of Consumer Finances* (SCF) range cards are shown to respondents who refuse to provide information regarding their exact income, asking them to pick one of the ranges depicted (e.g. 0-5,000 $\$$) or to pick a category following a decision tree (Kennickell, 1991). A similar procedure is implemented in the *National Health Interview Survey* (NHIS), where initial nonresponders to the question regarding the yearly income are asked whether their income is above or below $20,000$ USD and in a next step a range card with 44 income categories is shown (Schenker et al., 2006). The German Panel Study *Labour Market and Social Security* (PASS) also asks initial nonresponders consecutive questions about intervals covering the true income (Trappmann et al., 2010). These approaches help to collect at least some information for respondents initially refusing to provide an answer (Drechsler/Kiesl/Speidel, 2015) or selecting “don’t know” for the exact income question (Kennickell, 1996).

Censoring refers to the situation in which values above (or below) a given threshold are not observed. The only information available is that the true value must be above (or below) the known threshold. Censoring from the left typically arises in situations in which technical equipment will not detect the measure of interest if its concentration is below a certain limit. For example, in the study presented in Pilcher et al. (2007), the number of HI viruses in human blood is only measurable once it is above a given threshold of detection. Censoring from the right often occurs in public use files, in which top coding is applied to reduce the risk of re-identification. This is for example done in the US-American Current Population Survey (CPS) (Larrimore et al., 2008). An example of right censoring in biology is the time to seed germination as the time it takes for a seed to germinate can be longer than the duration of the study (Scott/Jones, 1990).

3.1 Analyzing interval data

Obtaining valid inferences if only interval information is available for (parts of) some of the variables can be complex. The most common strategy is to adjust the likelihood accordingly. For example, in linear regression models, the well known tobit model (Tobin, 1958) can be used to account for censoring in the dependent variable. This approach can easily be extended to other forms of interval data but iterative procedures are typically required to find the maximum likelihood estimates in this case. Since most software packages do not offer routines for dealing with interval data beyond the tobit model, some applied researchers rely on naive approaches for analyzing the interval data: A common approach is to ignore the interval information completely, using only those observations for which exact information is observed. This approach is always inefficient, since available information is not used. It can also introduce bias, if those units that only provide interval information differ form those units which provide exact information. In fact, Heeringa/Little/Raghunathan (1997) showed that the tendency to only report intervals for income increases with income. Thus, results solely based on the exact reports are likely to be biased.

To simplify the analysis for applied researchers, imputation approaches can be used to generate plausible values given the interval information. This offers the advantage that the analysts no longer need to find appropriate ways for incorporating the interval information. They can rely on standard analysis models using the plausible values for inference. However, just like in the standard nonresponse context, care needs to be taken to ensure that valid inferences can be obtained from the imputed data.

For example, a naïve imputation approach which is sometimes applied in practice uses the midpoint or the upper bound of each reported interval as the imputed value (Law/Brookmeyer, 1992; Dorey/Little/Schenker, 1993). The data are then analyzed treating the imputed values as the true exact values. These approaches are valid only in very limited settings since they will generally underestimate the variance in the imputed data (Law/Brookmeyer, 1992; Kim/Xue, 2002).

To fully account for the uncertainty resulting from the fact that only intervals instead of exact values are observed initially, multiple imputation approaches are required which generate imputations by drawing from the conditional distribution of the exact values given the interval information (and additional information from other variables available in the data set).

Imputation approaches have been used for several data sets to facilitate the analysis for the user. For example, since 1995 the *Survey of Consumer Finances* generates imputed income values by drawing from truncated normal distributions using the bounds of the reported intervals as truncation points.

An application of the joint modeling approach for imputation of interval data is discussed in Heeringa (1993). The author imputed interval and missing data in the *Health and Retirement Survey* (HRS) using the *general location model*. One major disadvantage of the general location model is that the multivariate normal distribution needs to be estimated for each cell of the table spanned by crossclassifying all categorical variables. Thus, the approach can only be used if the number of categorical variables is very limited to ensure a sufficient number of observations for estimating the normal distribution within each cell. A second problem can be sparse cells in the interval variable, making the imputation model unreliable. The author noticed this problem especially for the largest income category which typically included only few, very wealthy individuals. The true income distribution in this category also might be very skewed, violating the normality assumption.

For settings with ordered income categories affected by item nonresponse, Bhat (1994) proposed an imputation method modeling the income distribution and the response probabilities jointly using a selection modeling approach.

Raghunathan et al. (2001) describes a general sequential regression approach for interval data. Plausible values are generated by drawing from truncated normal distributions. The parameters for the model are estimated using those observations for which an exact value is available. New parameters for the truncated normal model are drawn using sampling/importance resampling (SIR, Rubin 1988). This approach is also implemented in the multiple imputation software `IVEware` (Raghunathan et al., 2016). The software was also used to impute plausible values for interval answers in the *National Health Interview Survey* (NHIS), (Schenker et al., 2006).

Royston (2007) implemented an imputation model for interval data for `Stata`. He extended the approach of Raghunathan et al. (2001) by also using the information from the respondents that only provided an interval when estimating the parameters of the imputation model. To obtain parameter estimates the joint likelihood of the income of the exact reporters and the income of the interval reporters is maximized under the implicit assumption that the conditional distribution of the true income given the covariates in the model is the same for both groups. Instead of using SIR, draws from the posterior distribution of the parameters are only approximated by drawing from a multivariate normal distribution centered around the maximum likelihood estimates of the parameters. Compared to the approach of Raghunathan et al. (2001) this strategy offers the advantage that it uses all available information and that it can also be used if only interval information is available.

A similar approach was later used by Drechsler/Kiesl/Speidel (2015) for simultaneous imputation of interval, rounded, and missing data. For interval data without rounding, the approach simplifies to the method described by Royston (2007) and is separately implemented in `hmi`.

Several (multiple) imputation approaches have also been proposed for the special case of survival data (Taylor/Schwartz/Detels, 1986; Muñoz et al., 1989; Taylor et al., 1990; Dorey/Little/Schenker, 1993). In survival analysis censoring is a common problem since for those units that entered a certain state of interest (for example unemployment) previous to the start of the study or are still in that state at the time the study is terminated, the true time of entry or exit is unknown. Imputation routines for survival data differ systematically from the imputation routines for interval data in other data sets since survival models need to be used for imputation to ensure congeniality between the imputation and the analysis model. Multiple imputation routines for this special type of data are implemented in the `R` package `icenReg` (Anderson-Bergman, 2017). Imputations in `icenReg` can be based on proportional hazards, proportional odds or accelerated failure time models. Since `icenReg` already provides a convenient tool for dealing with survival data, we did not implement these routines in `hmi` and we limit the description of the imputation methodology in the next section to applications outside the survival analysis context. The interested reader is referred to Grover/Gupta (2015) or Anderson-Bergman (2017) for details regarding imputation routines for survival data.

3.2 Methodology of multiple imputation for interval data

Let $y = \{y_1, \ldots, y_n\}$ be the realizations of the variable of interest– possibly transformed to fulfill the normality assumption of linear regression models – for which only interval information is available for some or all of the n observations in the data. Let $x = \{x_1, \ldots, x_n\}$ be the realizations of any other variables X available in the data set which might help to predict the values of y. We assume that

$$Y|X \sim N(X\beta, \sigma^2) \tag{3}$$

If exact values would be observed for all records, the likelihood of the model parameters would be

$$L(\beta, \sigma^2|y, x) = \prod_{i=1}^{n} f\left(y_i, \mu_i = x_i'\beta, \sigma^2\right) \tag{4}$$

with f being the density of a normal distribution.

If only interval information is available for some of the respondents, we need to introduce some additional notation. Let I_i be an indicator function that equals zero if exact information is available and equals one if only interval information is available for individual i (the interval information includes missing data as a special case with interval bounds $-\infty$ and $+\infty$). Let $\underline{y_i}$ and $\overline{y_i}$ be the lower and upper bound of the interval for unit i. The extended likelihood that also takes the interval information into account is given by

$$L(\beta, \sigma^2|y, x) = \prod_{i=1}^{n} \left((1 - I_i)f(y_i, x_i'\beta, \sigma^2) + I_i \left[F(\overline{y_i}, x_i'\beta, \sigma^2) - F(\underline{y_i}, x_i'\beta, \sigma^2)\right]\right), \tag{5}$$

with F being the cumulative distribution function of the normal distribution. Maximizing this likelihood will provide estimates for the parameters $\theta = \{\beta, \sigma^2\}$. To approximate a draw from the posterior distribution of $f(\theta|y,\ x)$ under the assumption of flat priors for all parameters, we can draw from

$$\theta^{\star} \sim MVN(\hat{\theta},\ I(\hat{\theta})), \tag{6}$$

where $\hat{\theta}$ contains the maximum likelihood estimates of θ, and $I(\hat{\theta})$ is the negative inverse of the Hessian matrix of the log-likelihood with $\hat{\theta}$ plugged in.

Plausible values for interval respondents can be imputed by drawing from a truncated normal distribution $N_t(\mu, \sigma^2)$ with $\mu = x'\beta^{\star}$, $\sigma^2 = (\sigma^{\star})^2$, where $\beta^{\star}$ and $(\sigma^{\star})^2$ are the parameters drawn form the approximate posterior distribution as described above. The truncation points are given by the bounds of the reported interval. Imputations for those respondents that refused to provide any information are obtained by drawing from a normal distribution with parameters $\mu = x'\beta^{\star}$ and $\sigma^2 = (\sigma^{\star})^2$.

3.3 Our contribution for the imputation of interval data

To our knowledge, imputation routines for interval data following the procedures described above are currently only available in `Stata`. For the special case of survival data imputation routines following a completely different methodology are available in the `R` package `icenReg` by Anderson-Bergman (2017). The `hmi` package is the first `R` package to offer general imputation routines for interval data beyond the survival data context. The package also provides a new solution for storing information on lower and upper bounds of the interval information in *one* variable together with a set of functions for handling interval data.

The idea is to store the bounds in a character variable separated by a semicolon. Such an interval object can be generated using `generate_interval` or split into its lower and upper bounds by `split_interval`. See Section 5.5 for details and Section 6.2 for examples.

4 Multiple imputation for data affected by heaping

Another form of coarse data are data for which the reported values are implicitly rounded. The rounding can either be identical for all individuals (for example if individuals round off their age), or subject to different rounding degrees. Many individuals rounding to the same value lead to heaps in the empirical distribution of the data. Therefore, this form of rounding with unknown rounding degrees is often referred to as heaping in the literature. It typically occurs, if the respondent is unwilling or unable to provide an exact value and instead reports a value which is a multiple of some common rounding base to implicitly express his or her uncertainty regarding the estimate. In many cases, multiples of 10, 100, or 1,000 are used. In other situations, the respondent uses a higher level of aggregation (such as years instead of months or weeks instead of days) for the estimate. For example, Heitjan/Rubin (1990) studied reported ages for young children in Tanzania and noted several heaps at certain values, such as 6 or 12 months. Huttenlocher/Hedges/Bradburn (1990) found heaps at multiples of seven for questions which asked how many days ago an event took place. Wang/Heitjan (2008) identified several heaps at multiples of 20 in questions regarding cigarette consumption because the common pack of cigarettes contains 20 cigarettes.

Table 1 taken from Drechsler/Kiesl/Speidel (2015) illustrates the problem using reported monthly household income in the German panel study *Labour Market and Social Security* (PASS) (Trappmann et al., 2010) for the year 2008/2009. The table provides the percentage of the reported monthly income values that are divisible by a given round number. It seems that most respondents tend to round their income. More than 60 percent of the reported values are divisible by 100 and less than 16 percent of the values are not divisible by 5. Czajka/Denmead (2008) report similar problems for the *American Community Survey* and the *Current Population Survey*.

The major problem with heaping is that inferences will be biased if the reported values are treated as face value (Hanisch, 2005). For example, Drechsler/Kiesl (2016) illustrate that

Table 1: Percentage of reported monthly household income values that are divisible by a given round number in the PASS survey for the year 2008/2009.

Income divisible by	1,000	500	100	50	10	5
Relative frequency (%)	13.97	23.94	61.57	69.58	80.71	84.13

Source: PASS data, own calculations

important policy measures such as the poverty rate can be substantially biased if heaping in the reported income is not taken into account.

4.1 Analyzing rounded data

Starting with Sheppard (1898) several methods have been proposed to account for rounding at the analysis stage (see for example Hanisch 2005 or Schneeweiss/Komlos/Ahmad 2010 for a review). However, most of the rounding literature assumes symmetric rounding intervals that can be derived directly from the reported value. For example, if distance is reported in kilometers, it is assumed that the true distance must be in the interval *reported distance* $\pm$ *500 meters*. However, this does not generally hold for heaping. As illustrated below, the rounding interval can not be inferred directly with data affected by heaping.

Instead of accounting for the rounding at the analysis stage multiple imputation methodology can be used to account for the rounding at the data processing stage. A multiple imputation strategy to obtain plausible values for the true values based on the reported values accounting for the uncertainty from rounding was first proposed by Heitjan/Rubin (1990) for age data affected by heaping. Related approaches were later used for self-reported cigarette counts (Wang/Heitjan, 2008), rounded unemployment durations (van der Laan/Kuijvenhoven, 2011) and self-reported income (Drechsler/Kiesl/Speidel, 2015; Drechsler/Kiesl, 2016; Zinn/Würbach, 2016).

4.2 Methodology of multiple imputation for data affected by heaping

There is an important difference between interval observations treated in Section 3 and rounded observations: With interval observations the interval in which the true value must lie is known. This is not the case for rounded observations. For example, if the reported income is 1,800, we do not know whether this is the exact true value, or if the true value has been rounded to the closest 5, 10, 50, or 100. To account for this uncertainty, we also need to model the rounding process.

The methodology presented in this section is based on the ideas first discussed in Heitjan/Rubin (1990). We summarize the main ideas of the approach here borrowing heavily from Drechsler/Kiesl (2016). For further details we refer to Heitjan/Rubin (1990) or Drechsler/Kiesl (2016).

To be able to account for the heaping in a variable, two models need to be specified: one model for the variable of interest and one model for the rounding behavior. Let Y be the variable of interest. Similar to Section 3 we assume that the conditional distribution of Y

given some covariates X is given as

$$Y|X \sim N(X\beta, \sigma^2) \tag{7}$$

To model the rounding behavior, an ordered probit model can be specified, i.e., a normally distributed latent variable G is assumed which may (linearly) depend on Y and some covariates Z (where some or all components of Z might be in X and vice versa):

$$G|Y, Z \sim N(\gamma_0 + Y\gamma_1 + Z\gamma_2, \tau^2). \tag{8}$$

The thresholds of the ordered probit model separate the different degrees of rounding. For example, if the assumed possible degrees of rounding are 1, 10, 50, and 100, an ordered probit model with four categories would be estimated.

Based on these model assumptions, the joint distribution of Y and G can be specified. The set of parameters to be estimated is given by $\Psi = (\beta, \sigma^2, \gamma_1, \gamma_2, k_1, ..., k_{p-1})$, where $k_1, ..., k_{p-1}$ denote the thresholds of the probit model assuming p possible degrees of rounding (note that γ_0 is fixed at 0 and τ^2 at 1 to make the ordered probit model identifiable). For each individual i, $i = 1, \dots, n$, with n being the sample size, let s_i denote the rounded value which is observed instead of the true y_i, and $s = (s_1, \dots, s_n)$. The likelihood function for Ψ given s_i and covariates x_i, z_i (assuming independent observations) may then be written as

$$\begin{aligned} L(\Psi|s, x, z) &= \prod_i f(s_i|x_i, z_i, \Psi) \\ &\propto \prod_i \iint\limits_{A(s_i)} f(g, y|x_i, z_i, \Psi) dy dg, \end{aligned} \tag{9}$$

where $A(s_i)$ is the set of (g, y) that are consistent with an observed s_i. The parameter vector Ψ can be estimated by maximizing $L(\Psi|s, x, z)$ using numerical methods.

To generate imputations of Y, the first imputation step (drawing a new set of parameters from their joint posterior distribution) can again be approximated by drawing from

$$\Psi^\star \sim MVN(\hat{\Psi}, I(\hat{\Psi})), \tag{10}$$

where $\hat{\Psi}$ contains the maximum likelihood estimates of Ψ, and $I(\hat{\Psi})$ is the negative inverse of the Hessian matrix of the log-likelihood with $\hat{\Psi}$ plugged in.

For the second imputation step (generating imputed values for Y) a simple rejection sampling approach is implemented:

1. Draw candidate values for (y_i^{imp}, g_i) from a truncated bivariate normal distribution using parameters from $\Psi^\star$, where the truncation points are given by the maximal possible degree of rounding given the observed value s_i (for example, for an observed income value 850 with possible degrees of rounding 1, 10, 50, 100, and 1,000, y_i is bounded by 825 and 875 and g_i has to be in $]-\infty, k_3^\star[$).

2. Accept the drawn values for y_i as imputation value if they are consistent with the observed rounded value, i.e., when rounding the drawn value for y_i according to the drawn rounding indicator g_i gives the observed value s_i.

3. Otherwise draw again.

4.3 Our contribution for the imputation of data affected by heaping

The R package simPop (Templ et al., 2017) provides a function for generating plausible values if heaps only occur at multiples of 5 or 10. However, no other rounding degrees can be considered and no covariates can be incorporated into imputation model. hmi provides a more general imputation routine for variables affected by heaping following the methodology presented above. With hmi flexible degrees of rounding can be specified and covariates can be incorporated in both, the model for the rounding process and the imputation model. The package will declare variables to be affected by heaping if certain criteria are met, but it is also possible for the user to manually decide, which variables are affected. For details how to register variables accordingly see Section 5.1 and the *Rounded continuous variables* paragraph in Section 5.5.

It is also possible to use hmi for dealing with situations in which missing observations, interval observations and rounded observations occur simultaneously. This will typically be the case for surveys asking for income or other sensitive questions. Since nonresponse to the income question tends to be high, it is common practice to ask respondents whether their income lies in certain intervals if they are unwilling or unable to provide exact income values. In this situation three potential outcomes are possible: the respondent remains unwilling to provide any information at all and thus the income value is missing. Alternatively, the respondent might not provide an exact value but might be willing to indicate an interval in which his or her income lies. Finally, the respondent might report a supposedly exact value, which considering Table 1 will still be a rounded estimate of the true income in many cases. To deal with such a situation the likelihood function in Equation (9) needs to be extended to also account for the interval information:

$$
\begin{aligned}
L(\Psi|s,x,z) \propto \prod_{i=1}^{n} \Big\{ (1-I_i) \iint\limits_{A(s_i)} f(g,y|x_i,z_i,\Psi)dydg + \\
I_i\Big[F(\overline{y}_i, \mu_i = x_i'\beta, \sigma^2) - F(\underline{y}_i, \mu_i = x_i'\beta, \sigma^2)\Big]\Big\}
\end{aligned}
\tag{11}
$$

Imputed values for the interval data can be obtained by drawing from a truncated distribution as described in Section 3. See Drechsler/Kiesl/Speidel (2015) for an application and for further details regarding the imputation procedure. To our knowledge, hmi is the only imputation routine which is able to simultaneously impute rounded, missing and interval observations.

5 Software

The main function of the package hmi is the wrapper function called hmi. It performs all input checks, data preparations, calls of different imputation functions depending on the type of variable to be imputed and generates the output. In the simplest case the user just passes her or his data to hmi. In this case all variables with missing values are imputed based on a single level imputation model including all other variables in data as predictors. Under this scenario, the package works similar to other multiple imputation packages in R such as mice or mi (Su et al., 2011). The full flexibility of the package is unleashed, if the user additionally passes her or his (multilevel) analysis model to hmi and/or makes further specifications.

5.1 Input

These are the arguments which can be specified with hmi:

- data: The (partially observed/rounded) data set specified as a data.frame. Data in the matrix format are converted into a data.frame. For multilevel imputation the data have to be in the *long format*, meaning that observations belonging to the same cluster have to be stacked in rows and a cluster indicator needs to be available. Data in the *wide format* have to be converted to the *long format* using for example the packages reshape2 (Wickham, 2007) or tidyr (Wickham/Henry, 2018).
- model_formula: This argument requires a formula representing the desired analysis model which should be run once the data have been imputed. If model_formula is specified, hmi will try to set up imputation models which are in line with this model. In the multilevel case model_formula is used to identify fixed effects and random effects covariates and the cluster indicator. See Section 5.3 for details.
- family: A family object supported by glm (resp. glmer). This argument is not needed in the imputation process, it only facilitate the automated pooling (see Section 5.9) when the dependent variable in model_formula is not continuous. For example, for count data the appropriate call would be family = "poisson". Setting the family argument will ensure that the correct model is used when hmi calculates the appropriate multiple imputation inferences for the specified analysis model.
- additional_variables: With this argument the user can specify variables (separated by +, e.g. "x8 + x9") which should be included in the imputation models beyond those variables already included in the analysis model as specified in model_formula. Instead of using additional_variables the user might extend the model_formula and run a reduced analysis model with hmi_pool (or use the analysis tools provided by mice).
- list_of_types: If users are not satisfied with the automatic classification of the variable types by hmi (see Section 5.5), they can specify a list containing their own classifications. For example a user might want to treat a variable as continuous while

it was identified as count data (imputations would then be based on a linear regression model in this case instead of the Poisson model which is the default for count data). The explicit specifications in `list_of_types` are binding for `hmi` and overrule all other implicit specifications in any other attribute. For example, only missing values will be imputed in a variable specified to be continuous even if rounding degrees and/or a rounding formula are specified for this variable. To change this, the variable would need to be explicitly specified as rounded continuous in `list_of_types`. The list contains elements, named like the variables. Each element is a character of one keyword (e.g. `list_of_types = list(x1 = "cont", x2 = "categorical")`) to denote the imputation routine that shall be used for this variable. See Section 5.5 for all supported keywords and Section 5.6 for more explanations about the pre-definition of the variable types and Section 6.1 for an real data example.

- `M`: The number of imputed data sets that should be generated. The default value is 5.

- `maxit`: Similar to `mice`, `maxit` defines the number of cycles of the sequential regression imputation procedure that should be run before one imputed data set is stored (see also Section 2.5). The default value is 10, unless only one variable needs to be imputed. In this case the number of iterations is set to 1 as no updating of other variables is required.

- `nitt`: An integer defining the number of iterations that should be used for the Gibbs sampler whenever a variable is imputed using multilevel models based on the MCMC routines implemented in the package `MCMCglmm` (Section 2.4). Higher values imply a higher chance of convergence, but also increase the runtime of the imputation process. Convergence can be checked after imputation using the function `chaincheck` (see Section 5.8 for details). By default 22,000 iterations are run.

- `burnin`: An integer defining the number of MCMC draws of the `MCMCglmm` routines to be discarded as burn in. Higher values increase the chance of drawing values from a chain that has converged, but `burnin` has to be strictly lower than `nitt`. Furthermore a sufficient number of draws (say 1,000) should remain after discarding the burn in order to be able to effectively test convergence of the chain after the imputation run. The default value is 2,000.

- `pvalue`: By default `hmi` tries to include all variables as predictors in the imputation model. This can lead to unstable parameter estimates if the number of predictors is large. As a consequence imputations can vary erratically generating implausible imputed values way outside the observed range of values. A strategy to limit this problem is to exclude insignificant variables from the imputation model via a variable selection procedure (this strategy is also implemented in the multiple imputation software `IVEware`). If specified, the package `hmi` uses a backward selection procedure to identify the final imputation model: In the first step a (multilevel generalized) linear model is estimated using all variables as predictors. In the next step a new regression model is estimated such that the variable with the highest p value above `pvalue` is removed. This is repeated until each variable included in the model have a p value smaller or equal to `pvalue` or until only one variable remains in the model. Excluding insignificant variables stabilizes the imputation process in most situations, but will typically bias the (conditional) correlation between imputed and excluded variables towards zero in any analysis performed on the imputed data. Therefore we advise to use this option conservatively, that is, we recommend generating imputations using the default value (i.e., `pvalue = 1`, which means no variables are removed). Lower values – say, 0.5 or 0.2 – can be specified, if the imputations based on the default setting show unacceptably large variances. We also note that variables are automatically removed if their effect cannot be estimated, that is, if the estimated coefficient is `NA`.

- `mn`: Estimating cluster specific parameters based on very few observations can lead to unstable estimates. As an ad hoc approach the user can specify a minimum number (`mn`) of observations a cluster should contain. The smallest clusters with less then `mn` observations will then be collapsed with the second smallest cluster until all clusters have at least `mn` observations. As this approach violates the assumption of independent normally distributed cluster effects and the individual effects of the collapsed clusters will no longer be reflected in the imputed data, this approach should be used with caution. The default value is 1, leading to no collapsing.

- `k`: Categorical variables with many categories can lead to unstable estimates since a large number of dummy variables needs to be included in the imputation model and some categories might be sparsely populated. To avoid such problems, `k` gives the maximum number of categories a categorical variables is allowed to have when used as covariate in an imputation model. Variables with more than `k` categories will be excluded from all imputation models. By default the number is ∞, leading to no removal. A less restrictive solution to avoid unstable estimates is to prevent the inclusion of insignificant dummy variables in the imputation model by setting an appropriate values for `pvalue`. In some situation it could be acceptable to classify ordinal variables with many categories as continuous in `list_of_types`.

- `spike`: This argument accepts a single numeric value or a list for which the names of the list entries match the names of semi-continuous variables (variables which have a spike at one value of the distribution but can be considered continuous otherwise). By setting `spike` to be an integer, the user can specify at which value the spike(s) might be found in the variable(s). In many cases, a spike will be found at zero, for example if a household survey asks for the taxes payed or a business survey asks for the number of employees hired in the previous year. However, there could be situations in which a spike occurs at a different value. For example, responses regarding the monthly net income will typically have a spike at the social security transfer level. In cases of different spikes for different variables, the parameter `spike` should be a list. For example, if `x2` has a spike at 0 and `x7` has a spike at 416 (which is minimum amount of social security payments in Germany), the attribute would need to be specified as `spike = list(x2 = 0, x7 = 416)`. The function `list_of_spikes_-maker` can be used to generate such a list with suggested spikes (returning the mode for all variables for which more than 10 percent of the values are equal to the mode). This list can be adopted according to the needs of the user and then passed to `hmi` via the `spike` attribute. If `spike` contains a list, the names in the list implicitly define which variables should be treated as semi-continuous, that is, there is no need to additionally register the variables as `semicont` in `list_of_types`. However, if a different variable type is explicitly provided in `list_of_types` for this variable, the variable is treated according to this type since explicit specifications in `list_of_-types` dominate any implicit specifications through any of the other attributes. The *Semi-continuous variables* paragraph in Section 5.5 describes the heuristic used to decide whether a variable should be treated as semi-continuous if neither a numeric value nor a list is specified. It also provides details how semi-continuous variables are imputed.

- `rounding_degrees`: If the user wants to generate plausible values for variables affected by heaping following the methodology described in Section 4, she or he can specify the rounding degrees which should be included in the model. The argument can either be a single numeric vector or a list for which the names of the list entries match the names of the variables affected by heaping. In this case each element of list contains a numeric vector specifying the various rounding degrees. For example if the age of children is reported in months, heaps might occur at multiples of 1, 6, or 12 while the monthly income might be rounded to multiples of 1, 10, 100, or 1,000. If plausible values should be generated for both variables, the user would need to specify `rounding_degrees = list(age = c(1, 6, 12), income = c(1, 10, 100, 1000))`. The function `list_of_rounding_degrees_maker` generates such a list with individually suggested rounding degrees for each variable found to be affected by heaping. This list can be adapted by the user according to his or her needs. See the *Rounded continuous variables* paragraph in Section 5.5 for details regarding when a variable is considered to be heaped and what rounding degrees are used in which scenarios. In Section 6.3 a data example on imputing variables affected by heaping is given.

- `rounding_formula`: For heaped continuous variables the user can specify a formula for the rounding process, that is he or she can specify, which predictor variables should be included in Equation (8). The standard `formula` notation should be used but no dependent variable needs to be specified. To give an example, the formula specification could be `~y + x2 + x15`, where `y` represents the variable affected by rounding and `x2` and `x15` are two other variables from the data set. Again, the argument can either be a formula or a list with element names identical to the names of the heaped variables. In the letter case each list element must contain a formula for the rounding process. The function `list_of_rounding_formulas_maker` generates such a list. This list can be adapted by the user according to his or her needs. The default formula is `~.`, meaning that all variables are included as main effects in the model for rounding. We note that maximizing the likelihood in Equation (9) is tricky since the boundaries of the integrals also need to be estimated. If the rounding model is too complex or if too many rounding degrees are specified, the iterative procedure for maximizing the likelihood might not converge. The function `hmi` will issue a warning whenever the optimizer did not converge or when the Hessian matrix of the maximum likelihood procedure cannot be inverted (which is typically a strong indication of numerical problems of the estimation procedure). In such cases, we generally recommend to either drop predictors from the `rounding_formula` or discard some of the specified `rounding_degrees`.

- `pool_with_mice`: As long as `pool_with_mice` is set to be `TRUE`, which is the default, `hmi` internally uses the functions from `mice` to obtain the final results for the analysis model specified in `model_formula`. The results are returned as an additional attribute called `pooling` within the output object. Note that the output object generated by `hmi` differs from the output generated by `mice` in this case. This can be avoided if `pool_with_mice` is set to `FALSE`. Currently, the synergy of `hmi` and `mice` supports the automatic calculation of the final inferences for (generalized multilevel) linear analysis models. The default pooling of (ordered) categorical variables, is not supported, except for categorical variables in the single level case. A more flexible, but somewhat inconvenient function for pooling is `hmi_pool`, which is delivered with this `hmi` package (see Section 5.7 for details).

5.2 Checks and preparations

The package `hmi` runs several initial checks before starting with the actual imputation:

- All inputs are checked to ensure correct formating (e.g., `data` must be set up as a `data.frame`, many other attributes must either contain a list or a vector of numeric values, etc). See `?hmi` or the previous section for details on the attribute specifications.
- If any of the variables included in `data` has more than 90 percent missing values, the program asks the user whether he or she wants to keep this variable or to quit the program to adjust the `data` accordingly.
  - Variables which are completely missing will cause a warning; they do not contain any information and will not be imputed.
  - Observations with missing values for all variables will also cause a warning for the same reasons.
- Variables included in `model_formula` which are not in `data` will cause an error. Note that `hmi` currently only supports two levels of hierarchy in the multilevel imputation models. Thus, only one cluster ID can be specified in `model_formula`.
- If a multilevel model is specified in `model_formula` but less than three clusters are found, the user is asked to run a single level imputation or to process the data in a different manner.
- If a multilevel model is specified in `model_formula` and the cluster variable contains missing values, the user is asked whether those should be removed (recommended), categorically imputed (not recommended) or the imputation process should be canceled.
- If `nm` is specified, clusters with less than `nm` observations are collapsed (see Section 5.1 for details).

The following additional preparing steps are taken for each imputation model during the imputation process:

- If more than one constant variable is included in the imputation model, only one is kept to avoid multicollinearity. For the same reason one variable is dropped from multilevel imputation models of unordered categorical variables, whenever two predictor variables are highly correlated ($\rho > 0.99$).
- If a value for `k` is specified, categorical variables with more than `k` categories are removed from the current imputation model (see Section 5.1 for details).
- If a value for `pvalue` is specified, variables with p values larger than `pvalue` are removed from the current imputation model in an iterative procedure (see Section 5.1 for details).
- During the first imputation cycle, interval variables are treated as factors whenever they appear as covariates in one of the imputation models, until they have been imputed themselves: Once they have been imputed, the plausible values are used as predictors instead of the interval information. If there are many unique intervals in an interval variable, the user may consider setting a limit for the maximum number of allowed factors using the attributes `k`.

5.3 `model_formula`

In the single level case, the model specified in `model_formula` has to follow standard `formula` conventions for `lm` in `R` (see `?formula`). For multilevel models the notation used by `lmer` (`lme4` package by Bates et al. 2015) must be used. The notation for multilevel models as implemented in `lme4` closely follows the notation for single level models with the main difference that random effect variables are added in parentheses. The cluster identifier is also included within the parentheses separated from the random effect variables(s) by a vertical bar. To illustrate, a possible model specification might be `y ~ x1 + x2 + x3 * x4 + (1 + x2|ID)`. In this model an intercept, four main effects and one interaction are specified as fixed effects. The intercept and `x2` also have random effects. The variable `ID` contains the cluster identifier.

If interactions are specified in `model_formula`, they are also used as predictors in the imputation models of all other variables in an effort to achieve congeniality. Note that the package currently does not follow the sophisticated approach suggested by Carpenter/Goldstein/Kenward (2011) for dealing with interactions in the analysis model, instead it uses passive imputation meaning that after each iteration the interaction term is updated by multiplying the current imputed versions of the main effects (cf. e.g. Seaman/Bartlett/White 2012).

5.4 Imputation cycles

In the first cycle of the sequential regression imputation routine, the variables are sorted and imputed by increasing number of missing observations following the approach of

Raghunathan et al. (2001). In this cycle only those variables with no missing values or variables that have been previously imputed are used as predictors in the imputation model. If all variables have missing values, the variable with the lowest missing rate will be imputed by taking random samples from the observed cases of this variable. In all other imputation cycles, all variables are included as main effects in the imputation model, unless `pvalue` is specified. If `model_formula` is specified, the imputation model follows this model as closely as possible. This implies that the imputation and analysis model coincide when the dependent variable in the analysis model needs to be imputed. If, on the other hand, a covariate in the analysis model needs imputation, this variable takes the place of the dependent variable in the imputation model and the actual dependent variable in the analysis model becomes an independent variable in the imputation model. For example, if the analysis model is `y ~ 1 + x1 + x2 + (1 + x1|ID)` and the covariate `x1` needs imputation, the imputation model becomes `x1 ~ 1 + y + x2 + (1 + y|ID)`.

Depending on the situation the imputation model can either be a single or multilevel model. If `model_formula` contains a single level model, or when no analysis model is specified, the imputation model always will be a single level model. However, specifying a multilevel model in `model_formula` generally implies that a multilevel model will also be used for all imputation models. In the first cycle it can happen that the random effect covariate(s) have missing values. In such cases single level models are estimated until the random effect covariates(s) have been imputed. If the cluster ID has missing values, we recommend to remove the missing cases from the data set. In case the user opt against this, the missing cases are imputed using a single level model for categorical variables.

The number of cycles is defined by `maxit` unless only one variable contains missing values. In this situation, imputed values will be drawn from the correct distribution in the very first iteration (because all predictor variables are fully observed), and thus the number of iterations can be set to 1. The default number of imputation cycles, for situations with more than one missing variable, is 10. For a more cautious approach the user might set `maxit` to a larger value. After `maxit` cycles, the imputed values are stored, building a completed (imputed) data set. Then the process starts again, until `M` (default value: 5) imputed data sets have been generated.

5.5 The different supported types of variables

Different variable types (continuous, binary, etc.) require different imputation routines. For example, for binary variables it is not desirable in most cases to get imputed values different from `0` or `1`. And factor variables with levels `"A"`, `"B"` and `"C"` need an imputation routine different from the routines for binary and continuous variables.

The package `hmi` distinguishes nine different types of variables. The following section describes the internal strategies to assign a type to each variable and how the imputation model works for that type. Users not satisfied with these default choices can specify the types of variables in advance by setting up a `list_of_types`. Section 5.6 explains how this is done.

5.5.1 Binary variables (keyword `"binary"`)

Variables are considered to be binary if there are only two unique values in the observed data. This includes for example `0` and `1` or `"m"` and `"f"`. This default classification might fail for small data sets or if a third possible category is unobserved. For example, in a small health survey it could happen that non of the respondents reported to have had two (or more) Bypass surgeries. So here a count variable would falsely be classified as binary. (Multilevel) logistic regression models are used to impute `binary` variables.

5.5.2 Continuous variables (keyword `"cont"`)

Any numeric vector that is not one of the other types is considered to be continuous. Imputation models are based on (multilevel) linear regression models described in Section 2.4.

5.5.3 Semi-continuous variables (keyword `"semicont"`)

If a variable is not defined explicitly (via `list_of_types`) or implicitly (via an entry for this variable in `spike`), a variable is identified as semi-continuous by `hmi` if more than 10 percent of the observations share the same value (this value is then called *spike*), but the remainder of the observations can be considered continuous. To which spike the variable is tested, depends on the specifications in `spike`: if it is explicitly or implicitly defined, the value in `spike` is used (i.e. the numeric values of `spike` or the list element in `spike` for this variable - dependent on how `spike` is specified). If those elements are empty, `hmi` uses the mode (most frequent observation) of the variable, irrespectively of the 10 percent threshold. This threshold is only relevant if the variables are not explicitly or implicitly specified as semi-continuous. In theses cases, the mode, or `spike` if it is a numeric value, is used to check whether the 10 percent threshold is exceeded or not.

The approach for imputing semi-continuous variables implemented in `hmi` follows the ideas presented in Rubin (1987) and Raghunathan et al. (2001). The variable is imputed in two steps. In the first step a temporary indicator variable is generated that equals 0 if the observed value is equal to the spike and 1 otherwise. Missing values in this indicator variable are then imputed using (multilevel) logit models. In the second step, missing observations with an imputed value of 1 for the temporary indicator variable are imputed based on a (multilevel) linear regression imputation model, using only those observed cases of the semi-continuous variable that are not equal to the spike. The missing observations with an imputed value of 0 for the temporary indicator variable are replaced by the value of the spike.

5.5.4 Interval variables (keyword "interval")

Variables where some observations contain only interval information (e.g. $[2000; 3000]$) are called interval variables. The technical implementation requires a specification for interval data. To our knowledge there is no general technical standard for handling interval data in R. The packages survival (Therneau, 2018) and linLIR (Wiencierz, 2012) provide functionalities to handle interval data. Both packages generate auxiliary objects in which the information for the lower and upper bound are stored separately. We did not follow this approach for our package since it would require an inconvenient workflow to link both interval bounds (for all interval variables) appropriately. Instead we define a new class interval for interval variables. Technically each observation in such an interval variable is coded as "l;u" with l and u denoting the lower and upper bound of the interval. Both bounds can either be numerical values, NA, -Inf or Inf. Two examples would be "1234.56;3000" and "-1234.56;Inf".

We also implemented functions to run basic calculations on interval data (+, -, *, /, %%, exp, log, ^, sqrt, floor, ceiling, and round), to generate interval data based on one (as.interval) or two vectors (generate_interval), or to split interval data into their lower and upper bounds (split_interval). How to use these functions is illustrated in Section 6.2.

For interval variables, the imputation routine described in Section 3 is used. As mentioned in Section 5.2, interval variables are treated as factor variables during the first imputation cycle - until the variable itself has been imputed. Once plausible values have been generated for this variable, these imputed values will be used instead of the interval information in the following cycles whenever the (former) interval variable is used as a predictor in one of the other imputation models.

5.5.5 Rounded continuous variables (keyword "roundedcont")

Whether a variable is treated as “rounded continuous”, (i.e., when the variable is affected by heaping), depends on the information contained in the attributes list_of_types, rounding_degrees and rounding_formula.

- list_of_types is always binding. If there is an entry in list_of_types for the variable, it will be imputed using imputation routines appropriate for the specified type irrespective of the information provided in any of the other attributes. Thus, if the variable is registered as roundedcont in list_of_types, it will be treated as affected by heaping irrespective whether potential degrees of rounding are specified in rounding_degrees or not. Vice versa, if the variable is registered to be of any other type, its missing values will be imputed using imputation methods appropriate for this variable type, but the heaping in this variable will be ignored even if rounding degrees are specified for this variable.
- If no explicit method is specified for the variable in list_of_types, hmi checks whether

`rounding_degrees` or a `rounding_formula` are specified for it, implying that the variable should be treated as rounded continuous.

- If no explicit or implicit classification is found, `hmi` classifies the variable internally. The classification tests for rounding degrees $1,\ 10,\ 100,\ 1,000$ or, if given, the general vector in `rounding_degrees`. A variable is classified as "rounded continuous" if more than 50 percent of the values in this variable are divisible by the specified rounding degrees (ignoring rounding to the nearest integer).

Variables classified to be rounded continuous (including variables having heaps, missing values and intervals at the same time) are imputed following the methodology described in Section 4. Which rounding degrees are used for generating plausible values depends on the provided specifications:

- For variables explicitly or implicitly specified to be rounded continuous, the information provided in `rounding_degrees` is decisive. If `rounding_degrees` contains a vector, the values of this vector are used for all variables specified to be affected by heaping. If it contains a list and this list has an element for the variable under consideration, the rounding degrees specified in this list element are used. If the list element or `rounding_degrees` is `NULL`, the heuristic explained in Appendix A.1 is used for suggesting rounding degrees.
- For variables classified by `hmi` as rounded continuous, the rounding degrees 1, 10, 100, 1000 or, if given, the general vector in `rounding_degrees` is used.

5.5.6 Count variables (keyword `"count"`)

Except for variables which are identified to be semi-continuous all variables containing no more than 20 different integers are treated as count data per default. Variables with more than 20 integers are considered to be continuous to avoid treating continuous variables for which only integers are reported in the data (such as income data) as count data. The user can override these rules by simply specifying a variable with more than 20 different integers to be `count` or a variable with less than 20 integers to be `cont` in the `list_of_types`.

Imputations are generated based on a Poisson model for this variable type. `MCMCglmm` is used to obtain the required draws of the model parameters from their respective posterior distributions for both, single and multilevel models.

5.5.7 Categorical variables (keyword `"categorical"`)

Unordered factor variables (or variables with more than two categories - if they are not one of the previous types) are considered to be categorical variables.

To impute these variables in a single level setting `hmi` uses the `cart` approach implemented in `mice`. The approach constructs a classification tree based on the observed data and then samples imputed values from suitable leaves of this tree for individuals for which the variable is missing.

In the multilevel setting, we use the `categorical` specification in `MCMCglmm` to obtain draws of the model parameters from their posterior distribution based on a multilevel multinominal regression model. Imputations for the missing values are generated using own routines implemented in `hmi`.

5.5.8 Ordered categorical variables (keyword `"ordered_categorical"`)

If a factor variable is ordered, `hmi` treats it as `"ordered_categorical"`. Missing values in this variable are imputed based on an ordered logistic (for single level models) or ordered probit regression (for multilevel models). For single level models `mice` is used to generate the imputations. For multilevel models `MCMCglmm` is used to obtain the required draws of the model parameters from their posterior distribution and imputations are generated using own routines implemented in `hmi`.

5.5.9 Intercept variable (keyword `"intercept"`)

A variable for which all observed records share the same value is considered a constant variable and thus registered as an intercept variable. Missing values in this variable are replaced by the value observed for the other records.

If the user defines a `model_formula` containing an intercept variable (even if it is only implicit like in `y ~x1 + x2`) and there is no intercept variable in the data set, `hmi` temporarily includes such a variable for the imputation process. This can be suppressed by using `y ~0 + x1 + x2` or `y ~-1 + x1 + x2`. Vice versa, as mentioned in Section 5.2, if `model_formula` contains constant variables in addition to the intercept, these variables are automatically removed from the imputation model to keep the model identified.

5.6 Pre-definition of the variable types

The package `hmi` tries to make an educated guess, which imputation model is most suitable for which variable. Still, we encourage users to explicitly specify which imputation model should be used for each variable or at least to check whether the imputation models suggested by the package are reasonable. Imputation models for each variable can be specified using `list_of_types`. This attribute expects a `list` in which each element of it has the name of a variable in the data frame. The named element has to contain a single character string denoting the type of the variable (the keywords from the previous section). The user can pass her or his `data` to the function `list_of_types_maker` to see which imputation model would be suggested by `hmi` for which variable. Calling this function can also be useful to obtain an object which already contains a list with entries for all variables in the data set. This object can then be modified as required. Examples for generating and modifying this list are shown in Section 6.1.

We emphasize again that the specifications provided in list_of_types will dominate any other specifications. For example, if the attribute rounding_degrees contains specific degrees of rounding for variable x11, but this variable is specified as continuous in list_-of_types, the variable will be treated like any other continuous variables, meaning that only the missing values in this variable will be imputed based on a (multilevel) linear regression model. No adjustments will be performed to deal with the heaps in the data.

5.7 Output of hmi

The package is build to allow a seamless integration into mice. Most importantly, the output generated by hmi can be treated like a multiply imputed data set generated with mice, that is, all the tools available in mice for analyzing and modifying the imputed data sets can be applied directly. The technical details regarding the structure of the hmi output are described here, practical examples are shown in the *Monitoring convergence* and *Analyzing the imputed data* paragraphs of Section 6.1.

Similar to mice, hmi returns a so called mids-object (multiply imputed data sets). These objects contain the original data set, the imputed values, the chain means and variance of the imputed values, and several additional elements (see van Buuren/Groothuis-Oudshoorn 2011). The fact that hmi returns a mids-object enables users familiar with mice to use functions designed for mice-outputs without switching barriers. For example, running the generic plot()-function on a mids-object calls the function plot.mids showing the means and standard deviations of the imputed values for all variables over the different imputations and cycles, regardless whether the mids-object came from mice or hmi. Another example is the complete-function delivered by mice which returns the imputed data set.

The function hmi returns two additional elements within the mids-object which are not available from mice: gibbs and pooling. The former allows checking the convergence of the gibbs-sampler chains generated by MCMCglmm (a convenient tool for checking convergence is available through the function chaincheck, see Section 5.8 for details). The later gives the pooled results (that is the final inferences based on the combining rules for multiply imputed data) from passing the model_formula to the pooling functions from mice (see Section 5.9 for details).

5.8 Convergence checks

For every imputed variable, the function plot.mids (delivered by mice) shows the mean and standard deviation of the imputed values across the maxit iterations and M imputation cycles. See Figure 1 in Section 6.1 as an example. This tool helps to evaluate whether draws based on the sequential regression approach converged to draws from the underlying joint distribution of the missing data given the observed data (see van Buuren/Groothuis-Oudshoorn 2011 for more details on this convergence measure).

If multilevel models are used for imputation (or if a Poisson model is used in general) additional convergence tests are necessary since the posterior draws of the model parameters are obtained using a Gibbs sampler in these cases. Thus, we need to ensure that the Gibbs sampler actually converged before the parameters were drawn. Detailed information about all the MCMC chains from all models is available through the element `gibbs`. This is a multidimensional list. The first dimension distinguishes the different imputation runs. The elements in this layer are therefore called `"imputation1"`, `"imputation2"`, `...`, `"imputation[M]"`. The second layer is for the cycles with names `"cycle1"`, `...`, `"cylce[maxit]"`. The next layer is for the variable that has been imputed. For example, an element named `"x1"` stands for the imputation of `"x1"`. The last layer distinguishes between `"Sol"` and `"VCV"`. The names are adopted from `MCMCglmm` where the elements `"Sol"` and `"VCV"` in the output represent the point estimates (of the fixed effects and cluster specific effects) and the variance parameter estimates (the elements of the random effects covariance matrix and the residual variance), receptively. `hmi` only exports the fixed effects point estimates from `"Sol"` due to workspace considerations: `MCMCglmm` estimates `nitt` cluster specific effects for every random effects variable in every cluster. This would imply that if the user wants to run `nitt = 5000` iterations for a random intercepts and slopes model with only one fixed effects variable on a data set with 60 clusters, the dimension of the resulting matrix would already be $5000 \times (2+2\cdot 60)$. If such a matrix would be saved for two variables and the imputation procedure is based on `maxit = 10` iterations and `M = 20` imputations, the final output would already contain $20\times 10\times 2\times 5000\times(2+2\cdot 60)\approx 2$ million elements. Thus, to keep the size of the generated output manageable even if several variables are imputed based on multilevel models and/or the number of clusters is large, convergence can only be monitored for the fixed effects and the variance components.

To facilitate the convergence evaluations, the user can apply the function `chaincheck` to the output provided by `hmi`. The function implements the stationarity test proposed by Geweke (1992) and plots the results. The null hypothesis of the stationary test is that the expected values behind the means $\bar{x}_A$ and $\bar{x}_B$ of the first 10 percent and last 50 percent of the chain (after discarding the burn in) are equal. The test statistic for this test is $T = (\bar{x}_A - \bar{x}_B)/\sqrt{\hat{\sigma}(\bar{x}_A)^2 + \hat{\sigma}(\bar{x}_B)^2}$, where $\hat{\sigma}(\bar{x}_A)^2$ and $\hat{\sigma}(\bar{x}_B)^2$ are the estimated variances of the arithmetic means of the first 10 percent and last 50 percent of the chain after discarding the burn in. T asymptotically follows a standard normal distribution. So if $|T|$ exceeds the $1-\alpha/2$ quantile of the standard normal distribution, the null hypothesis can be rejected. The test is implemented in the function `geweke.diag` from the `R` package `coda` (Plummer et al., 2006) and `chaincheck` calls this function. Beyond the `mids`-object generated by `hmi` the user can also pass the desired significance level `alpha` for the test statistic and the desired `burnin` (expressed as a percentage of the total length of the chain) to the `chaincheck` function. By default (`plot = TRUE`), `chaincheck` will plot all chains for which the null hypothesis was rejected. Each plot contains the information which parameter and which variable, in which cycle and imputation is depicted. Furthermore, the test statistic T is shown. Note that no adjustments are made for the multiple testing problem and thus a certain number of tests will show significant results (“chain did not converge”) by chance (Type I error). For example in a setting with `maxit = 5`, `M = 5`, two variables to impute and an imputation model with 2 fixed effects and two random effects variables and a significance level of `alpha = 0.01`, the number of expected false positives is $5\cdot 5\cdot 2\cdot(2+4+1)\cdot\alpha = 3.5$. The function `chaincheck` will print the actual and expected number of failed test. Note that the test is only meant to highlight potentially convergence problems. The provided plots can then be used to decide, whether the identified chains really indicate problems of the Gibbs sampler.

For large numbers of chains and thus larger numbers of expected false positives, it might be more convenient not to plot the chains failing the convergence test. This can be done by setting the function parameter `plot = FALSE`. We note that users are free to use their own convergence diagnostics since results from all the chains are available in the `gibbs` attribute of the `mids`-object generated by `hmi`.

High autocorrelation can increase the number of false positives. The parameter `thinning` allows to increase the thinning of the Chains to reduce auto correlation (the default value is `1`). As a rule of thumb, the number of values in the chain should not fall below 1000. By setting `thinning = NULL`, the number of remaining values is set to be approximately 1000. Note that setting a value for `thinning` will not affect the imputation procedures. The parameters will only affect which chain values are used when computing Geweke's test.

If the Gibbs sampler apparently did not converge, (currently) a new call of `hmi` has to be initiated with an increased number of iterations for the Gibbs-sampler (parameter `nitt`).

5.9 Pooling

The functions `with` and `pool` from `mice` are flexible tools for analyzing and pooling multiply imputed data sets. `hmi` uses these functions to obtain the final results for the analysis model specified by `model_formula` and `family`. The results can be accessed in the `mids`-object through its element `pooling`. Currently, `mice` only pools global fixed effects of multilevel regression model. In some situations, other parameters such as the variance components from the different levels of the hierarchical model might be relevant for the user. Therefore `hmi` delivers the function `hmi_pool` as a flexible alternative to the functionality available in `mice`. The function needs two inputs:

1. the multiply imputed data set (the `mids` object created with `hmi` or `mice`) and
2. a predefined analysis function which takes a completed data set as input, and returns a vector with the desired complete data statistics (e.g. the regression coefficients or random effects variance estimates).

`hmi_pool` calculates the parameters defined in the analysis function on each of the completed data sets in the `mids`-object and averages them, that is `hmi_pool` will only provide point estimates but not their associated estimated variances. The pooling is only valid when averaging is reasonable. For example it would be invalid to pool factor loadings from factor analysis where the signs of loadings have no meaning (comparable to whether `"m"` or `"f"` is the reference category in a regression model). Examples how to use `hmi_pool` are given in the *Analyzing the imputed data* paragraph in Section 6.1 and on the help page `?hmi_pool`.

6 Application examples

To illustrate the generation of plausible values for multilevel data, interval data and variables affected by heaping three step-by-step examples from three real data sets are given.

6.1 Multilevel data

To illustrate the main functionality of the package `hmi`, we use the data set Gcsemv containing information on the General Certificate of Secondary Education (GSCE) in the UK. The data set, which was collected in 1989 and contains 1905 students in 73 schools, is one of the data sets used in Goldstein (2011). It is freely available on the website of the *Centre for Multilevel Modelling* (CMM) at the University of Bristol under the following URL `http://www.bristol.ac.uk/cmm/media/team/hg/msm-3rd-ed/gcsemv.xls`. It is also included in the package `hmi` to allow users to replicate the examples given in this section. We thank Harvey Goldstein and the CMM for allowing us to incorporate the data into the `hmi` package. The variables contained in the data set are described in Table 2. A more detailed description of the data can be found in Creswell (1991).

Table 2: Variables included in the Gcsemv data.

variable	description
`school`	School ID
`student`	Student ID within this school[1]
`gender`	Gender (0 = boy, 1 = girl),
`written`	(Numeric) score in a written questionnaire
`coursework`	(Numeric) score for a coursework

Source:
`http://www.bristol.ac.uk/cmm/learning/mmsoftware/data-rev.html#gcsemv`

6.1.1 Before starting imputation

If the package has not been installed previously, the very first step is to install the `hmi` package via `install.packages("hmi")`. Once the package has been installed it can be attached to the current session, and the `Gcsemv` data can be loaded. The code for these two steps is:

```
library("hmi")
data(Gcsemv)
```

A short summary of the data shows (among other information) that the data set has 202 missing values in the written exam covariate and 180 missing values in the coursework covariate. Thus, the missing rate in those variables is 10.6 percent and 9.4 percent respectively. There are no rows with missing values in both variables, so the number of incomplete observations in total is 382 or 20.0 percent.

[1] the student ID is not unique since students in different schools can have the same ID

```
summary(Gcsemv)
     school          student       gender       written          coursework
 68137   : 104    77       :  14   0: 777   Min.   : 0.625   Min.   :  9.259
 68411   :  84    83       :  14   1:1128   1st Qu.:37.500   1st Qu.: 62.963
 68107   :  79    53       :  13            Median :46.875   Median : 75.926
 68809   :  73    66       :  13            Mean   :46.798   Mean   : 73.435
 22520   :  65    27       :  12            3rd Qu.:55.625   3rd Qu.: 86.111
 60457   :  54    110      :  12            Max.   :90.000   Max.   :100.000
 (Other):1446    (Other):1827              NA's   :202      NA's   :180
```

A list containing the suggested variable types for each variable in the data set can be obtained by:

```
list_of_types_maker(Gcsemv)
$school
[1] "categorical"

$student
[1] "categorical"

$gender
[1] "binary"

$written
[1] "cont"

$coursework
[1] "cont"
```

If the user is not satisfied with the suggested types, he or she might save the list, modify it, and pass the modified list to `hmi`. For example, if `coursework` contained the average grade of every student and the user prefers to treat that variable as ordered categorical, he or she can type:

```
modified_list <- list_of_types_maker(Gcsemv)
modified_list$coursework <- "ordered_categorical"
```

The modified list would then be passed to `hmi` by setting the attribute `list_of_types = modified_list`.

6.1.2 Running the imputation

The next (optional) step is to set up the `model_formula`, i.e. the final model of interest which should be estimated based on the multiply imputed data (see Section 5.3). In the example given below, interest lies in the influence of gender and performance in previous coursework on the written exam. The intercept and the effect of gender are allowed to vary across the schools. They are added as random effects in the `model_formula`.

```
model_formula <- written ~ 1 + gender + coursework + (1 + gender|school)
```

Now the data and `model_formula` can be passed to the wrapper function `hmi`. The results are saved in an object called `dat_imputed`. Note that for full reproducibility a seed for the pseudo-random number generator is specified. Since no value is specified for the number of imputations, the default number of `M=5` imputed data sets will be generated. `hmi` will provide a progress bar during the imputation process.

```
set.seed(123)
dat_imputed <- hmi(data = Gcsemv, model_formula = model_formula)
Imputation progress:
0%   20%  40%  60%  80%  100%
|----|----|----|----|
```

6.1.3 Monitoring convergence

Before running any analysis models on the newly generated `mids`-object, it is always a good idea to check the convergence of all imputation routines. Some examples of how to do this based on the output generated by `hmi` are presented in this section.

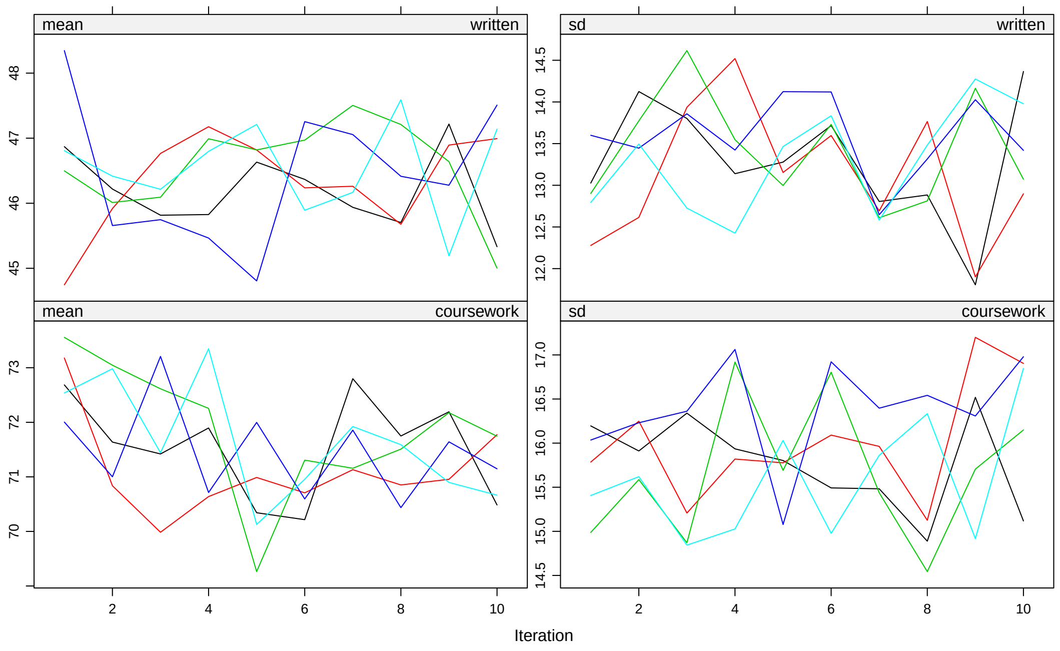
Diagnostic plots regarding the convergence of the sequential regression procedure can be obtained for example by `plot(dat_imputed)`. The command will plot the arithmetic mean and standard deviation of the imputed values for each imputed variable across the `maxit` cycles separately for each of the `M` imputations. In the given example calling the `plot` command will produce graphs for the variables `"written"` and `"coursework"` since these are the only two variables which have been imputed previously. Each graph contains five different lines for each of the `M = 5` imputations. Each line consists of ten points for each of the `maxit = 10` iterations.

```
plot(dat_imputed, layout = c(2, 2))
```

Convergence (potentially after some burnin iterations) can be assumed if the following two points are fulfilled:

1. There is no inherent trend in any of the lines.
2. The lines from the different imputations mix well, i.e. there is sufficient overlap between the different lines.

Figure 1: Mean (left) and standard deviation (right) for the imputed variables in the Gcsemv data across 10 iterations for 5 imputations.



Source: Gcsemv data, own calculations

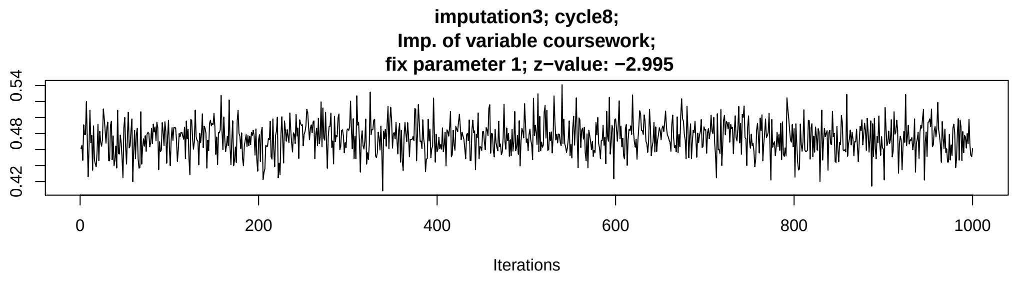
Examining the plots in Figure 1, both requirements seem to be met.

Given that the model specified in `model_formula` is a hierarchical model, multilevel models have also been used as imputation models. Since these models can only be estimated using MCMC methods, formal checks regarding the convergence of these models are also required. The function `chaincheck` runs convergence tests using the Geweke statistic for each chain of the MCMC method and plots traceplots for all those parameters for which the test indicates a failure of convergence (see Section 5.8 for details on the test). The function also provides the information how often the null hypothesis is rejected and compares this number to the expected number of false rejections due to type I error.

```
chaincheck(dat_imputed, thin = NULL)
12 out of 695 chains (1.73%) did not pass the convergence test.
For alpha = 0.01 the expected number is 6.95.
```

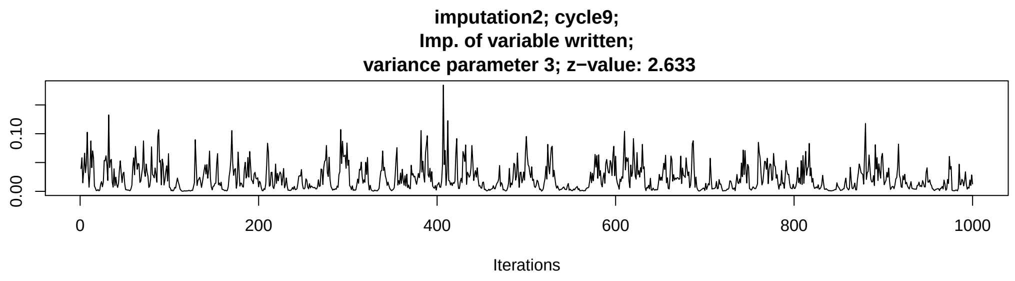
For the given example the traceplots for the fixed effects in the models which did not pass the stationarity test show no problematic pattern (one traceplot is shown in Figure 2 the others are omitted for brevity). But the plots for the variance parameters show signs of autocorrelation (one chain is shown in Figure 3). For highly autocorrelated chains it is more likely that the mean of the first 10 percent of the chain differs from the mean of the last 50 percent of the chain and thus the null hypothesis of the Geweke test (which basically assumes equivalence of the two means) is rejected. Note however, that autocorrelation would only be a problem, if multiple draws from the same chain would be used. Since only one value from a chain is used for each imputation in `hmi`, autocorrelation within a chain is generally irrelevant for `hmi`. Therefore it can be concluded, that for the package's purposes, all parameters in all imputation models show good convergence properties.

Figure 2: Traceplot of one fixed effects parameter which formally did not pass the stationarity test.



Source: Gcsemv data, own calculations

Figure 3: Traceplot of a variance parameter showing signs of high autocorrelation.



Source: Gcsemv data, own calculations

6.1.4 Analyzing the imputed data

In this section different possibilities for obtaining valid inferences based on the imputed data are shown. In general, valid inferences can be obtained by analyzing each completed data set separately and combining the results according to Rubin's combining rules (Rubin, 1987).

The package `mice` offers the functions `with` and `pool` to obtain final inferences based on the imputed data sets for a broad class of analyses. These functions can also be used with objects generated by `hmi` since they only require a `mids`-object as input. We refer to van Buuren/Groothuis-Oudshoorn (2011) for more details how to use these functions. Note that `hmi` also calls these functions internally if a model is specified in `model_formula` and `pool_with_mice = TRUE` (which is the default). The regression results are directly available through the element `pooling` from the `mids` object. This element is not available in `mids` objects generated by `mice`; it is a special feature of `hmi`. It will not be included if `pool_with_mice = FALSE`.

```
summary(dat_imputed$pooling)
                   est         se          t        df     Pr(>|t|)
(Intercept) 21.4285513 1.54661329 13.855145 228.08527 0.000000e+00
```

```
gender1     -5.4004356 0.59328192 -9.102647 153.58281 4.440892e-16
coursework   0.4042744 0.01919767 21.058509  64.10292 0.000000e+00
                 lo 95     hi 95 nmis       fmi    lambda
(Intercept) 18.3810747 24.476028   NA 0.1306011 0.1230109
gender1     -6.5724822 -4.228389   NA 0.1642805 0.1534680
coursework   0.3659238  0.442625  180 0.2666766 0.2441485
```

However, `pool` can only be used with estimation commands that return a list of coefficients and their variance matrix. Thus, for example, no information is returned regarding the variance components on the different levels if `pool` is used to provide the results of a multilevel analysis. However, the estimated variances on the different levels can be of interest in some applications. For this reason `hmi` offers the option to pass an analysis function setup by the user to the function `hmi_pool` which will run the specified analyses on each imputed data set and return the final point estimates but not their variances. Thus, this function can be used in situations in which the variance of the point estimates cannot be estimated (or is not of interest to the analyst), but averaging the point estimates from the different data sets is still a valid approach.

In the following example, the user is interested in the global fixed effects and the elements of the random effects covariance matrix of the multilevel model from the running example. To obtain the final results, she or he would first need to specify the analysis function:

```
#The input of the function is a complete data set
#(which will be provided by hmi_pool later).

analysis_function <- function(complete_data){

  # Generate an empty list for storing the results of interest
  parameters_of_interest <- list()

  # Specify the analysis model of interest
  my_model <- lmer(written ~ 1 + gender + coursework + (1 + gender | school),
              data = complete_data)

  # Specify, which parameters from the model should be returned.
  # The fixed effects:
  parameters_of_interest[[1]] <- fixef(my_model)
  # The covariance matrix of the random effects:
  parameters_of_interest[[2]] <- VarCorr(my_model)[[1]][ , ]

  # Turn the list into a vector to simplify labeling:
  ret <- unlist(parameters_of_interest)

  # Optionally: label the output:
  names(ret) <- c("intercept", "gender", "coursework",
                   "sigma0", "sigma01", "sigma10", "sigma1")

  #Return the results.
  return(ret)
```

This function can then be passed to hmi_pool to obtain the final point estimates for the specified parameters. As the analysis_function in this example calls lmer from the lme4 package, this packages has to be loaded in advance.

```
library("lme4")
hmi_pool(mids = dat_imputed, analysis_function = analysis_function)

 intercept     gender coursework     sigma0    sigma01    sigma10     sigma1
21.4285513 -5.4004356  0.4042744 42.3474332 -2.7057949 -2.7057949  3.1604561
```

The final results for the global fixed effects are identical to the results obtained with mice, but the output now also contains the final point estimates of the covariance matrix of the random effects.

6.2 Interval data

To illustrate the usage of the provided functions for interval-objects and the imputation of interval data, hmi includes three versions of a subset of the 2015-2016 Income File of the National Health and Nutrition Examination Survey (NHANES) (Centers for Disease Control and Prevention (CDC) and National Center for Health Statistics (NCHS), 2015-2016). The data set nhanes_sub (accessible by typing data(nhanes_sub) once the package is loaded) contains the data in their original format (compared to the version available on the NCHS website the data have been slightly modified, for example by coding some variables as factors or collapsing several nonresponse categories into a single category). In the data set nhanes_mod some variables have been changed to the internal interval variable format, which is required if plausible values should be imputed for these variables. Finally, nhanes_imp contains a multiply imputed data set in which missing and interval information has been replaced with plausible values following the methodology outlined in Sections 2 and 3. These data sets are included for illustrative purposes so that users of the package can compare different versions of the data sets to get a better understanding of how this imputation function works. Table 3 lists the variables present in the nhanes data sets.

As an illustrative example, the required steps to prepare the variable ind310 for generating plausible values, that is, the transformation of the categorical variable from nhanes_sub to the interval variable in nhanes_mod, are presented here (the interval variable for ind235 was generated in a similar fashion). Separate lower and upper bounds are defined for each observation (based on the description of https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/INQ_I.htm); subsequently they are merged to an interval object by the function generate_interval

Table 3: Variables included in the nhanes data sets.

variable	description
inq020	Income from wages/salaries (1 = Yes, 2 = No)
inq012	Income from self employment (1 = Yes, 2 = No)
inq030	Income from Social Security or Railroad Retirement (1 = Yes, 2 = No)
inq060	Income from other disability pension (1 = Yes, 2 = No)
inq080	Income from retirement/survivor pension (1 = Yes, 2 = No)
inq090	Income from Supplemental Security Income (1 = Yes, 2 = No)
inq132	Income from state/county cash assistance (1 = Yes, 2 = No)
inq140	Income from interest/dividends or rental (1 = Yes, 2 = No)
inq150	Income from other sources (1 = Yes, 2 = No)
ind235	Monthly family income (13 categories/an interval object)
ind310	Total savings/cash assets for the family (8 categories/an interval object)
inq320	How do you get to the grocery store? (10 categories)

Source: https://wwwn.cdc.gov/Nchs/Nhanes/2015-2016/INQ_I.htm

```
#generate an empty vector of length n
data(nhanes_sub)
low <- array(dim = nrow(nhanes_sub))
up <- array(dim = nrow(nhanes_sub))

#fill in the lower bounds depending on the reported savings category
low[nhanes_sub$ind310 == 1] <- 0
low[nhanes_sub$ind310 == 2] <- 3001
low[nhanes_sub$ind310 == 3] <- 5001
low[nhanes_sub$ind310 == 4] <- 10001
low[nhanes_sub$ind310 == 5] <- 15001
low[nhanes_sub$ind310 == 6] <- 0
low[nhanes_sub$ind310 == 7] <- 20001
low[nhanes_sub$ind310 == 8] <- 0

#fill in the upper bounds depending on the reported savings category
up[nhanes_sub$ind310 == 1] <- 3000
up[nhanes_sub$ind310 == 2] <- 5000
up[nhanes_sub$ind310 == 3] <- 10000
up[nhanes_sub$ind310 == 4] <- 15000
up[nhanes_sub$ind310 == 5] <- 20000
up[nhanes_sub$ind310 == 6] <- 20000
up[nhanes_sub$ind310 == 7] <- Inf
up[nhanes_sub$ind310 == 8] <- Inf

#generate the interval variable
ind310interval <- generate_interval(low, up)

#inspect the first few entries in the generated object
head(ind310interval)
"20001;Inf" "3001;5000" "0;3000" "3001;5000"  "0;3000" "3001;5000"
```

Once the variables are registered as `interval` variables, the data set can be passed to the `hmi` wrapper function. `hmi` will automatically generate plausible values for all variables registered as `interval` variables. For the imputation of the missing and interval data in `nhanes_mod`, we increased the number of iterations to 50, as diagnostic plots showed that the sequential regression procedure did not converge after the default number of 10 iterations.

```
set.seed(123)
nhanes_imp <- hmi(nhanes_mod, maxit = 50)
```

6.2.1 Some useful functions for interval data

The package `hmi` also includes some useful functions to analyze and manipulate interval data. This section provides a short summary of some of the functions available.

`table.interval`: Variables stored in interval format are interpreted as a vector of characters or a factor by most `R` functions including the `table` command. Without `table.interval`, `table` would order the intervals alphabetically, which can be arbitrary. The function `table.interval` offers improved sorting options. By default, it orders the intervals first by the value of their lower bound and if they are equal, by the value of the upper bound. If the attribute `sort` is set to `"mostprecise_increasing"`, the intervals are first ordered by their length (from small to large) and if the lengths are equal, by the value of the lower bound (from small to large). Using the `table` command on an interval variable will automatically invoke `table.interval` if `hmi` is loaded.

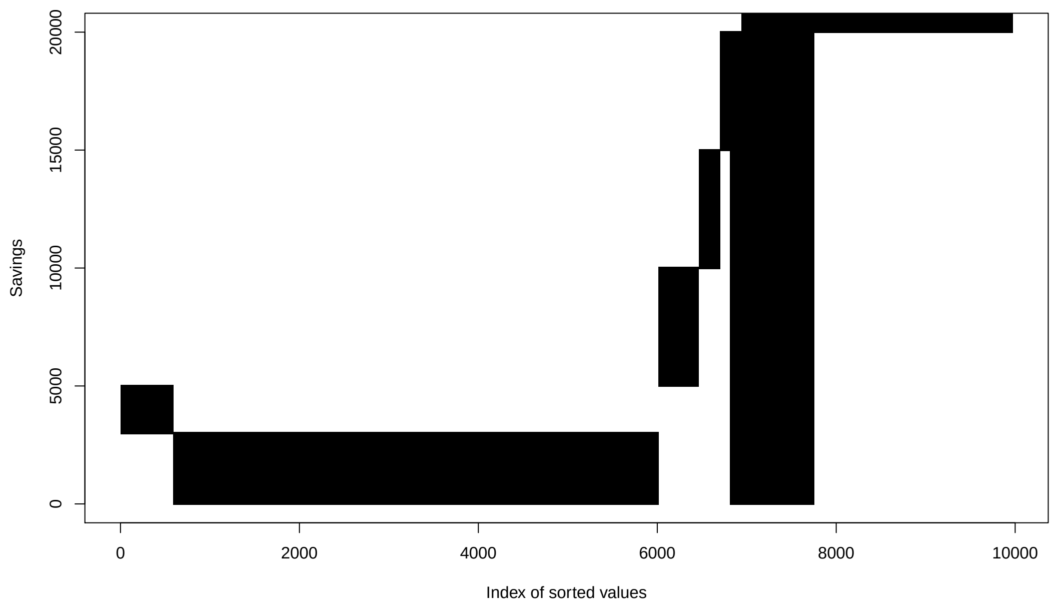
```
table(nhanes_mod$ind310)

     0;3000      0;20000        0;Inf    3001;5000   5001;10000  10001;15000
       5426          128          814          588          450          237
15001;20000    20001;Inf
        110         2218
```

`plot.interval`: To inspect interval variables graphically, the generic plotting function `plot` can be used, which will call `plot.interval`. For example, Figure 4 containing the results for the savings variable from `nhanes_mod` is generated using the following code:

```
plot(nhanes_mod$ind310, ylab = "Savings", sort = "mostprecise_increasing")
```

The figure shows the interval values for `ind310` sorted first by the interval lengths and then by the lower bound. A second option is `sort = "lowerbound_increasing"` sorting the intervals first by the lower bound and then by the upper bound. If no argument is specified for `sort`, the intervals are sorted by their appearance in the data. For each observation the

Figure 4: An interval-data scatter plot.



Source: nhanes data, own depiction

plot draws a line from its lower to its upper bound (plus a small margin to make very small intervals and point precise observations visible). As the lines for observations sharing the same interval are grouped together, they form an area. Thus, the width of the area is an indicator for the relative frequency of this interval. Note that in the example the upper bound for the highest savings category and for the nonrespondents is ∞ which cannot be plotted. Therefore the upper limit of the y-axis by default is the highest finite bound observed (plus a small margin). The axis bounds can be manually altered by the parameters `xlim` and `ylim`.

`center.interval`: This function simply returns a numeric vector containing the midpoint of the reported interval for each observation (for example 1,500 if the interval is `"0;3000"`). Intervals including `Inf` or `-Inf` will return `Inf` or `-Inf`, unless the interval is `"-Inf;Inf"` or the parameter `inf2NA` was set to be `TRUE`. In those cases `NA` will be returned for these intervals. This function can potentially be useful for some descriptive statistics, but we caution the user that treating the midpoint of the reported interval as if it were the originally reported value is rarely a good idea.

```
midpoints <- center.interval(nhanes_mod$ind310)
table(midpoints)
x
   1500  4000.5  7500.5   10000 12500.5 17500.5     Inf
   5426     588     450     128    2371     110    3032
```

`idf2interval` and `interval2idf`: Interval variables are also accepted in some other `R` packages. For example, the package `linLIR` by Wiencierz (2012) provides methods for regression models with interval variables. However when using this package, the data containing the interval information need to be coded as `idf` (*imprecise data frame*). To ensure that users can switch easily between `idf` and `interval` objects, we implemented `idf2interval` and `interval2idf` which convey an object from one format to the other. Technically, `idf` objects can contain multiple interval variables, so when transforming an `idf` object to fit to the `interval` setting, the (multiple) interval variables from `idf` are stored as variables in a `data.frame`.

```
idf <- interval2idf(nhanes_mod$ind310)
intervaldf <- idf2interval(idf)
```

`split_interval`: This function is basically the inverse function of `generate_interval`. It returns a two column matrix containing the lower bound for each reported interval in the first column and the upper bound in the second column:

```
bounds <- split_interval(nhanes_mod$ind310)
head(bounds)
```

```
      [,1] [,2]
[1,] 20001  Inf
[2,] 3001  5000
[3,]    0  3000
[4,] 3001  5000
[5,]    0  3000
[6,] 3001  5000
```

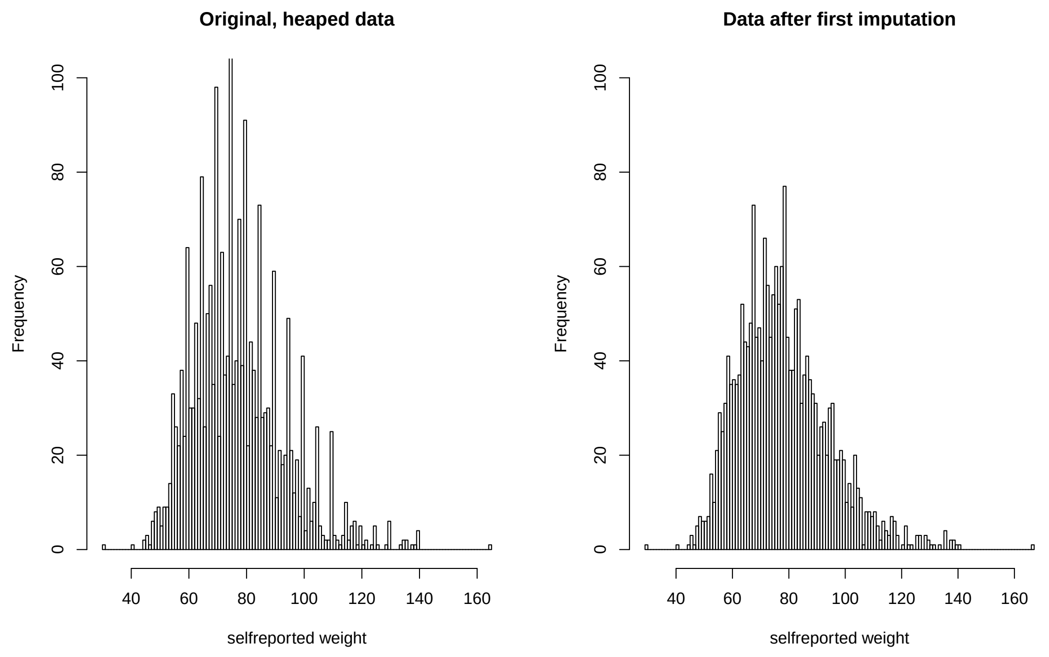
Finally, we note that basic arithmetics (`+`, `-`, `*`, `/`, `%%`) and transformations (`log`, `exp`, `^`, `sqrt`, `round`, `floor`, `ceiling`) can be applied to interval data (for example to change the currency for the reported values):

```
log_savings_in_euro <- log(nhanes_mod$ind310 * 0.8)
```

6.3 Variables affected by heaping

To briefly illustrate how to generate plausible values for a variable affected by heaping, we use the `selfreport` data from the `mice` package. The data set contains 2060 records and 15 variables, merged from multiple Dutch data sets. The left panel of Figure 5 shows a histogram of the self reported weight (variable `wr` in the data set). Heaps at multiples of 5 and 10 are clearly visible and thus, it seems plausible to assume that many respondents round their true weight to the closest 5 or 10 kilograms. Counting the number of records that are divisible by 5 and 10 reveals that almost 40 percent of the records are divisible by 5 and approximately 20 percent of the reported values are divisible by 10:

Figure 5: Selfreported weight from the selfreport data as originally observed (left) and after generating plausible values accounting for potential rounding of the reported values.



Source: selfreport data, own depictions and calculations

```
library("mice")
data(selfreport)
sum(selfreport$wr %%  5 == 0)/nrow(selfreport)
0.3800971

sum(selfreport$wr %% 10 == 0)/nrow(selfreport)
0.1941748
```

Note that these fractions are slightly below the thresholds setup in the heuristic for suggesting rounding degrees as implemented in `list_of_rounding_degrees_maker`. The heuristic would identify 5 as a rounding degree if 40 percent of the data would be divisible by this value and register 10 as a rounding degree if 20 percent of the data are divisible by this value (see Appendix A.1 for details). For this reason, explicit rounding degrees must be provided in this example when calling `hmi`. For the purpose of a short runtime, only two variables are used for imputation in this illustration: the self reported weight (`wr`) and the self reported height (`hr`):

```
set.seed(123)
selfreport_imputed <- hmi(selfreport[, c("hr", "wr")],
                          rounding_degrees = list(wr = c(1, 5, 10)))
```

By default, every variable in the data set is included in the model for the rounding behavior, that is, into the model specified in Equation (8). The model can be adjusted using `rounding_formula`. For example, if only the weight variable (and the intercept) should be used in the rounding behavior model, this could be achieved by setting `rounding_formula = ~wr`. The right panel of Figure 5 shows the histogram after imputation. The heaps in the data have disappeared.

7 Conclusion

With `hmi` we provide comprehensive, but easy to handle tools for multiple imputation for hierarchical data sets. The package supports imputation methods for all common types of variables. Furthermore, imputation tools for interval and heaped variables are provided. Several internal features of the package ensure that sensible default settings are selected automatically. Thus, even inexperienced users will find the package convenient to use since all they need to provide is their data and potentially the analysis model they want to run on the imputed data. The final results (according to the given analysis model) will also be returned by default. Still, the package offers great flexibility since almost all settings can be defined manually if desired. Multiple imputation point estimates for analyses not supported in `mice` can also be obtained using an additional function provided with the package.

Currently, `hmi` still has some limitations which we hope to address in future releases of the package: Most importantly, the package does not provide any tools for imputing variables from the second level of the hierarchical model, that is, variables which are constant within clusters. A convenient tool for imputing such variables is available in `mice`. Furthermore, the multilevel imputation models are currently limited to two levels of hierarchy and homoscedastic error terms. Finally, ensuring that all Gibbs samplers of the multilevel imputation models have converged is currently left to the user. In future versions of the package, we hope to implement some routines that will automatically ensure that all chains run long enough to ensure convergence.

References

Anderson-Bergman, Clifford (2017): icenReg: Regression Models for Interval Censored Data in R. In: Journal of Statistical Software, Vol. 81, No. 12, p. 1–23.

Andridge, Rebecca R. (2011): Quantifying the Impact of Fixed Effects Modeling of Clusters in Multiple Imputation for Cluster Randomized Trials. In: Biometrical Journal, Vol. 53, No. 1, p. 53–74.

Asparouhov, Tihomir; Muthén, Bengt (2010): Multiple Imputation with Mplus. In: MPlus Web Notes, URL https://www.statmodel.com/download/Imputations7.pdf.

Audigier, Vincent; Resche-Rigon, Matthieu (2018): micemd: Multiple Imputation by Chained Equations with Multilevel Data. URL https://CRAN.R-project.org/package=micemd, r package version 1.2.0.

Bates, Douglas; Mächler, Martin; Bolker, Ben; Walker, Steve (2015): Fitting Linear Mixed-Effects Models Using lme4. In: Journal of Statistical Software, Vol. 67, No. 1, p. 1–48.

Bhat, Chandra R. (1994): Imputing a Continuous Income Variable From Grouped and Missing Income Observations. In: Economics Letters, Vol. 46, No. 4, p. 311–319.

Carpenter, James R.; Goldstein, Harvey; Kenward, Michael G. (2011): REALCOM-IMPUTE Software for Multilevel Multiple Imputation with Mixed Response Types. In: Journal of Statistical Software, Vol. 45, No. 5, p. 1–14.

Carpenter, James R.; Kenward, Michael G. (2013): Multiple Imputation and its Application. John Wiley & Sons.

Centers for Disease Control and Prevention (CDC) and National Center for Health Statistics (NCHS) (2015-2016): National Health and Nutrition Examination Survey Data. URL https://wwwn.cdc.gov/nchs/nhanes/.

Clogg, Clifford C.; Rubin, Donald B.; Schenker, Nathaniel; Schultz, Bradley; Weidman, Lynn (1991): Multiple Imputation of Industry and Occupation Codes in Census Public-Use Samples Using Bayesian Logistic Regression. In: Journal of the American Statistical Association, Vol. 86, No. 413, p. 68–78.

Creswell, Michael (1991): A Multilevel Bivariate Model. In: Prosser, Robert; Rasbash, Jon; Goldstein, Harvey (Eds.) Data Analysis with ML3, London: Institute of Education.

Czajka, John L.; Denmead, Gabrielle (2008): Income Data for Policy Analysis: A Comparative Assessment of Eight Surveys. Final report to the U.S. Department of Health and Human Services submitted by Mathematica Policy Research, Inc., U.S. Department of Health and Human Services, URL https://aspe.hhs.gov/system/files/pdf/75721/report.pdf.

Dorey, Frederick J.; Little, Roderick J. A.; Schenker, Nathaniel (1993): Multiple Imputation for Threshold-Crossing Data With Interval Censoring. In: Statistics in Medicine, Vol. 12, No. 17, p. 1589–1603.

Drechsler, Jörg (2015): Multiple Imputation of Multilevel Missing Data – Rigor Versus Simplicity. In: Journal of Educational and Behavioral Statistics, Vol. 40, No. 1, p. 69–95.

Drechsler, Jörg (2011): Synthetic Datasets for Statistical Disclosure Control: Theory and Implementation, Vol. 201. Springer Science & Business Media.

Drechsler, Jörg; Kiesl, Hans (2016): Beat the Heap: An Imputation Strategy for Valid Inferences from Rounded Income Data. In: Journal of Survey Statistics and Methodology, Vol. 4, No. 1, p. 22–42.

Drechsler, Jörg; Kiesl, Hans; Speidel, Matthias (2015): MI Double Feature: Multiple Imputation to Address Nonresponse and Rounding Errors in Income Questions. In: Austrian Journal of Statistics, Vol. 44, p. 59–71.

Enders, Craig Kyle; Keller, Brian T.; Levy, Roy (2017): A Fully Conditional Specification Approach to Multilevel Imputation of Categorical and Continuous Variables. In: Psychological Methods.

Enders, Craig Kyle; Mistler, Stephen Andrew; Keller, Brian T. (2016): Multilevel Multiple Imputation: A Review and Evaluation of Joint Modeling and Chained Equations Imputation. In: Psychological Methods, Vol. 21, No. 2, p. 222–240.

Gartner, Hermann; Rässler, Susanne (2005): Analyzing the Changing Gender Wage Gap based on Multiply Inputed Right Censored Wages. Tech. Rep., IAB-Discussion Paper 05/2005, URL `http://doku.iab.de/discussionpapers/2005/dp0505.pdf`.

Gelman, Andrew; Hill, Jennifer (2006): Data analysis Using Regression and Multilevel/Hierarchical Models. Cambridge university press.

Geweke, John (1992): Evaluating the Accuracy of Sampling Based Approaches to Calculating Posterior Moments. In: Bernando, J. B.; Berger, J. O.; Dawid, A. P.; Smith, Adrian F. M. (Eds.) Bayesian Statistics 4, Oxford, UK: Clarendon Press, p. 169–193.

Goldstein, Harvey (2011): Multilevel Statistical Models. Chichester (UK): Wiley, 4 ed..

Grover, Gurprit; Gupta, Vinay K. (2015): Multiple Imputation of Censored Survival Data in the Presence of Missing Covariates Using Restricted Mean Survival Time. In: Journal of Applied Statistics, Vol. 42, No. 4, p. 817–827.

Hadfield, Jarrod D. (2010): MCMC Methods for Multi-Response Generalized Linear Mixed Models: The `MCMCglmm` `R` Package. In: Journal of Statistical Software, Vol. 33, No. 2, p. 1–22.

Hanisch, Jens U. (2005): Rounded Responses to Income Questions. In: Allgemeines Statistisches Archiv, Vol. 89, No. 1, p. 39–48.

Heeringa, Steven G. (1993): Imputation of Item Missing Data in the Health and Retirement Survey. URL `http://ww2.amstat.org/sections/srms/Proceedings/papers/1993_013.pdf`.

Heeringa, Steven G.; Little, Roderick J. A.; Raghunathan, Trivellore E. (1997): Imputation of Multivariate Data on Household Net Worth. URL http://ww2.amstat.org/sections/SRMS/Proceedings/papers/1997_020.pdf.

Heitjan, Daniel F.; Rubin, Donald B. (1991): Ignorability and Coarse Data. In: The Annals of Statistics, Vol. 19, No. 4, p. 2244–2253.

Heitjan, Daniel F.; Rubin, Donald B. (1990): Inference from Coarse Data via Multiple Imputation with Application to Age Heaping. In: Journal of the American Statistical Association, Vol. 85, No. 410, p. 304–314, URL http://www.jstor.org/stable/2289765.

Huttenlocher, Janellen; Hedges, Larry V.; Bradburn, Norman M. (1990): Reports of Elapsed Time: Bounding and Rounding Processes in Estimation. In: Journal of Experimental Psychology: Learning, Memory, and Cognition, Vol. 16, No. 2, p. 196–213.

Jenkins, Stephen P.; Burkhauser, Richard V.; Feng, Shuaizhang; Larrimore, Jeff (2011): Measuring Inequality Using Censored Data: A Multiple-Imputation Approach to Estimation and Inference. In: Journal of the Royal Statistical Society A, Vol. 174, No. 1, p. 63–81.

Jolani, Shahab (2018): Hierarchical Imputation of Systematically and Sporadically Missing Data: An Approximate Bayesian Approach Using Chained Equations. In: Biometrical Journal, Vol. 60, No. 2, p. 333–351.

Kennickell, Arthur B. (1996): Using Range Techniques with CAPI in the 1995 Survey of Consumer Finances. URL http://ww2.amstat.org/sections/srms/Proceedings/papers/1996_073.pdf.

Kennickell, Arthur B. (1991): Imputation of the 1989 Survey of Consumer Finances: Stochastic Relaxation and Multiple Imputation. URL http://ww2.amstat.org/sections/srms/Proceedings/papers/1991_001.pdf.

Kim, Mimi Y.; Xue, Xiaonan (2002): The Analysis of Multivariate Interval-Censored Survival Data. In: Statistics in Medicine, Vol. 21, No. 23, p. 3715–3726.

Larrimore, Jeff; Burkhauser, Richard V.; Feng, Shuaizhang; Zayatz, Laura (2008): Consistent Cell Means for Topcoded Incomes in the Public Use March CPS (1976-2007). In: Journal of Economic and Social Measurement, Vol. 33, No. 2,3, p. 89–128.

Law, C. Gordon; Brookmeyer, Ron (1992): Effects of Mid-Point Imputation on the Analysis of Doubly Censored Data. In: Statistics in Medicine, Vol. 11, No. 12, p. 1569–1578.

Liu, Jingchen; Gelman, Andrew; Hill, Jennifer; Su, Yu-Sung; Kropko, Jonathan (2014): On the stationary distribution of iterative imputations. In: Biometrika, Vol. 101, No. 1, p. 155–173.

Lüdtke, Oliver; Robitzsch, Alexander; Grund, Simon (2017): Multiple Imputation of Missing Data in Multilevel Designs: A Comparison of Different Strategies. In: Psychological Methods, Vol. 22, No. 1, p. 141–165.

Meng, Xiao-Li (1994): Multiple-Imputation Inferences with Uncongenial Sources of Input. In: Statistical Science, Vol. 9, No. 4, p. 538–573.

Mistler, Stephen A. (2013): A SAS Macro for Applying Multiple Imputation to Multilevel Data. In: Proceedings of the SAS Global Forum, URL https://support.sas.com/resources/papers/proceedings13/438-2013.pdf.

Muñoz, Alvaro; Wang, Mei-Cheng; Bass, Sue; Taylor, Jeremy M. G.; Kingsley, Lawrence A.; Chmiel, Joan S.; Polk, B. Frank; The Multicenter AIDS Cohort Study Group (1989): Acquired Immunodeficiency Syndrome (AIDS)-free Time After Human Immunodeficiency Virus Type1 (HIV-1) Seroconversion in Homosexual Men. In: American Journal of Epidemiology, Vol. 130, No. 3, p. 530–539.

Nowok, Beata; Raab, Gillian M.; Dibben, Chris (2016): synthpop: Bespoke Creation of Synthetic Data in R. In: Journal of Statistical Software, Vol. 74, No. 11, p. 1–26.

Pilcher, Christopher D.; Joaki, George; Hoffman, Irving F.; Martinson, Francis E.A.; Mapanje, Clement; Stewart, Paul W.; Powers, Kimberly A.; Galvin, Shannon; Chilongozi, David; Gama, Syze; Price, Matthew A.; Fiscus, Susan A.; Cohen, Myron S. (2007): Amplified Transmission of HIV-1: Comparison of HIV-1 Concentrations in Semen and Blood During Acute and Chronic Infection. In: AIDS, Vol. 21, No. 13, p. 1723–1730.

Plummer, Martyn; Best, Nicky; Cowles, Kate; Vines, Karen (2006): CODA: Convergence Diagnosis and Output Analysis for MCMC. In: R News, Vol. 6, No. 1, p. 7–11, URL https://www.r-project.org/doc/Rnews/Rnews_2006-1.pdf.

Quartagno, Matteo; Carpenter, James (2018): jomo: A package for Multilevel Joint Modelling Multiple Imputation. URL http://CRAN.R-project.org/package=jomo, r package version 2.6-1.

R Core Team (2016): R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, URL http://www.R-project.org/.

Raghunathan, Trivellore E.; Lepkowski, James M.; van Hoewyk, John; Solenberger, Peter (2001): A Multivariate Technique for Multiply Imputing Missing Values Using a Series of Regression Models. In: Survey Methodology, Vol. 27, p. 85–96.

Raghunathan, Trivellore E.; Solenberger, Peter W.; Berglund, Patricia A.; van Hoewyk, John (2016): IVEware: Imputation and Variance Estimation Software. URL http://www.src.isr.umich.edu/wp-content/uploads/IVEware-Version-0.3-User-Guide-linked.pdf.

Rässler, Susanne (2003): A Non-Iterative Bayesian Approach to Statistical Matching. In: Statistica Neerlandica, Vol. 57, No. 1, p. 58–74.

Raudenbush, Stephen W.; Bryk, Anthony S. (2002): Hierarchical Linear Models: Applications and Data Analysis Methods. Thousand Oaks: Sage Publications, Inc, 2 ed..

Reiter, Jerome P. (2012): Bayesian Finite Population Imputation for Data Fusion. In: Statistica Sinica, Vol. 22, No. 2, p. 795–811, URL http://www.jstor.org/stable/24310035.

Reiter, Jerome P.; Raghunathan, Trivellore E. (2007): The Multiple Adaptations of Multiple Imputation. In: Journal of the American Statistical Association, Vol. 102, No. 480, p. 1462–1471.

Reiter, Jerome P.; Raghunathan, Trivellore E.; Kinney, Satkartar K. (2006): The Importance of Modeling the Sampling Design in Multiple Imputation for Missing Data. In: Survey Methodology, Vol. 32, No. 2, p. 143–150.

Royston, Patrick (2007): Multiple Imputation of Missing Values: Further Update of `ice`, with an Emphasis on Interval Censoring. In: Stata Journal, Vol. 7, No. 4, p. 445–464, URL `http://www.stata-journal.com/article.html?article=st0067_3`.

Rubin, Donald B. (1988): Using the SIR Algorithm to Simulate Posterior Distributions. In: José Miguel Bernardo, Dennis Victor Lindley, Morris Herman DeGroot; Smith, Adrian Frederick Melhuish (Eds.) Bayesian Statistics, Vol. 3, Oxford University Press, p. 395–402.

Rubin, Donald B. (1987): Multiple Imputation for Nonresponse in Surveys. Hoboken (NJ): John Wiley & Sons.

Rubin, Donald B. (1986): Statistical Matching Using File Concatenation with Adjusted Weights and Multiple Imputations. In: Journal of Business & Economic Statistics, Vol. 4, No. 1, p. 87–94, URL `http://www.jstor.org/stable/1391390`.

Rubin, Donald B. (1978): Multiple Imputations in Sample Surveys - a Phenomenological Bayesian Approach to Nonresponse. URL `http://ww2.amstat.org/sections/SRMS/Proceedings/papers/1978_004.pdf`.

Schafer, Joseph L. (2016): `pan`: Multiple Imputation for Multivariate Panel or Clustered Data. URL `https://CRAN.R-project.org/package=pan`, r package version 1.4.

Schenker, Nathaniel (2003): Assessing Variability Due To Race Bridging: Application to Census Counts and Vital Rates for the Year 2000. In: Journal of the American Statistical Association, Vol. 98, No. 464, p. 818–828.

Schenker, Nathaniel; Raghunathan, Trivellore E.; Bondarenko, Irina (2010): Improving on Analyses of Self-Reported Data in a Large-Scale Health Survey by Using Information from an Examination-Based Survey. In: Statistics in Medicine, Vol. 29, No. 5, p. 533–545.

Schenker, Nathaniel; Raghunathan, Trivellore E.; Chiu, Pei-Lu; Makuc, Diane M.; Zhang, Guangyu; Cohen, Alan J. (2006): Multiple Imputation of Missing Income Data in the National Health Interview Survey. In: Journal of the American Statistical Association, Vol. 101, No. 475, p. 924–933.

Schneeweiss, Hans; Komlos, John; Ahmad, Amar S. (2010): Symmetric and Asymmetric Rounding: A Review and Some New Results. In: Advances in Statistical Analysis, Vol. 94, No. 3, p. 247–271.

Scott, S. Jeffrey; Jones, Richard A. (1990): Generation Means Analysis of Right-Censored Response-Time Traits: Low Temperature Seed Germination in Tomato. In: Euphytica, Vol. 48, No. 3, p. 239–244.

Seaman, Shaun Robert; Bartlett, Jonathan W.; White, Ian R. (2012): Multiple Imputation of Missing Covariates with Non-Linear Effects and Interactions: An Evaluation of Statistical Methods. In: BMC Medical Research Methodology, Vol. 12, No. 46, p. 1–13.

Sheppard, William Fleetwood (1898): On the Calculation of the Most Probable Values of Frequency-Constants, for Data arranged according to Equidistant Division of a Scale. In: Proceedings of the London Mathematical Society, Vol. s1-29, No. 1, p. 353–380.

Snijders, Tom Augustinus Benedictus; Bosker, Roel (2011): Multilevel Analysis - An Introduction to Basic and Advanced Multilevel Modeling. London: SAGE, 2nd revised edition. ed..

Speidel, Matthias; Drechsler, Jörg; Sakshaug, Joseph W. (2017): Biases in Multilevel Analyses Caused by Cluster-Specific Fixed-Effects Imputation. In: Behavior Research Methods.

Su, Yu-Sung; Gelman, Andrew; Hill, Jennifer; Yajima, Masanao (2011): Multiple Imputation with Diagnostics (mi) in R: Opening Windows into the Black Box. In: Journal of Statistical Software, Articles, Vol. 45, No. 2, p. 1–31.

Taljaard, Monica; Donner, Allan; Klar, Neil (2008): Imputation Strategies for Missing Continuous Outcomes in Cluster Randomized Trials. In: Biometrical Journal, Vol. 50, No. 3, p. 329–345.

Taylor, Jeremy M. G.; Muñoz, Alvaro; Bass, Sue M.; Saah, Alfred J.; Chmiel, Joan S.; Kingsley, Lawrence A. (1990): Estimating the Distribution of Times from HIV Seroconversion to AIDS Using Multiple Imputation. In: Statistics in Medicine, Vol. 9, No. 5, p. 505–514.

Taylor, Jeremy M. G.; Schwartz, Kendra; Detels, Roger (1986): The Time from Infection with Human Immunodeficiency Virus (HIV) to the Onset of AIDS. In: The Journal of Infectious Diseases, Vol. 154, No. 4, p. 694–697, URL http://www.jstor.org/stable/30104955.

Templ, Matthias; Meindl, Bernhard; Kowarik, Alexander; Dupriez, Olivier (2017): Simulation of Synthetic Complex Data: The R Package simPop. In: Journal of Statistical Software, Vol. 79, No. 10, p. 1–38.

Therneau, Terry M. (2018): A Package for Survival Analysis in S. URL https://CRAN.R-project.org/package=survival, version 2.42-3.

Tobin, James (1958): Estimation of Relationships for Limited Dependent Variables. In: Econometrica, Vol. 26, No. 1, p. 24–36.

Trappmann, Mark; Gundert, Stefanie; Wenzig, Claudia; Gebhardt, Daniel (2010): PASS: a Household Panel Survey for Research on Unemployment and Poverty. In: Schmollers Jahrbuch. Zeitschrift für Wirtschafts- und Sozialwissenschaften, Vol. 130, No. 4, p. 609–622.

van Buuren, Stef (2012): Flexible Imputation of Missing Data. United States: Taylor & Francis Group.

van Buuren, Stef (2011): Multiple Imputation of Multilevel Data. In: Hox, Joop J.; Roberts, J. Kyle (Eds.) The Handbook of Advanced Multilevel Analysis, chap. 10, Milton Park, UK: Routledge Academic, p. 173–196.

van Buuren, Stef; Groothuis-Oudshoorn, Karin (2011): mice: Multivariate Imputation by Chained Equations in R. In: Jornal of Statistical Software, Vol. 45, No. 3, p. 1–67.

van der Laan, Jan; Kuijvenhoven, Léander (2011): Imputation of Rounded Data. Statistics Netherlands Discussion Paper no. 201108, Statistics Netherlands, URL `https://www.cbs.nl/-/media/imported/documents/2011/08/2011-x10-08.pdf`.

Wang, Hao; Heitjan, Daniel F. (2008): Modeling Heaping in Self-Reported Cigarette Counts. In: Statistics in Medicine, Vol. 27, No. 19, p. 3789–3804.

Wickham, Hadley (2007): Reshaping Data with the `reshape` Package. In: Journal of Statistical Software, Vol. 21, No. 12, p. 1–20, URL `http://www.jstatsoft.org/v21/i12/`.

Wickham, Hadley; Henry, Lionel (2018): `tidyr`: Easily Tidy Data with '`spread()`' and '`gather()`' Functions. URL `https://CRAN.R-project.org/package=tidyr`, r package version 0.8.0.

Wiencierz, Andrea (2012): `linLIR`: linear Likelihood-based Imprecise Regression. URL `https://CRAN.R-project.org/package=linLIR`, r package version 1.1.

Zhou, Hanzhi; Elliott, Michael R.; Raghunathan, Trivellore E. (2016): Synthetic Multiple-Imputation Procedure for Multistage Complex Samples. In: Journal of Official Statistics, Vol. 32, No. 1, p. 231–256.

Zhu, Jian; Raghunathan, Trivellore E. (2015): Convergence Properties of a Sequential Regression Multiple Imputation Algorithm. In: Journal of the American Statistical Association, Vol. 110, No. 511, p. 1112–1124.

Zinn, Sabine; Würbach, Ariane (2016): A statistical Approach to Address the Problem of Heaping in Self-Reported Income Data. In: Journal of Applied Statistics, Vol. 43, No. 4, p. 682–703.

A Appendix

A.1 Suggestion for rounding degrees

If the user registers a variable as potentially being affected by heaping (by setting the variable type to `roundedcont`) but does not provide `rounding_degrees` for this variable, `hmi` tries to make an educated guess, regarding the possible degrees of rounding which should be used when modeling the heaping. The following heuristic is used to suggest the rounding degrees:

1. For a given continuous variable all possible rounding degrees (*factors* or *divisors* in mathematical terms), are derived for each observation. To give an example, the factors of 10 are $1, 2, 5, 10$. We will call $1, 2, 5$ *subfactors* of 10.

2. For each possible factor identified in step 1, the number of observations divisible by this factor is tabulated.

3. A rough estimate (based on the assumption of a discrete uniform distribution between 0 and ∞) for the expected number of observations being divisible by a factor s is n/s,

where n is the number of records in the data set. For example, the expected number of observations being divisible by $s = 5$ for a data set containing 10,000 records is $n/s = 2000$. If the observed number of individuals being divisible by factor s is at least twice the expected number, s is a “candidate rounding degree”.

4. Starting with the highest candidate rounding degree, each candidate has to fulfill two conditions to be stored as an actual rounding degree.

   - At least 20 percent of the data have to be divisible by this candidate; observations which are also divisible by larger rounding degrees which has been previously identified to be an actual rounding degree are not considered. The removal of these records ensures that the currently considered candidate actually contributes to the heaping. For example when 40 percent of the data are divisible by 100, at least 40 percent of the data have to be divisible by 50. By requesting that at least 60 percent of the data are divisible by 50 (if 100 has been identified previously as an actual rounding degree) it is ensured that the fact that a large proportion of the data is divisible by 50 is not only a spurious effect because many observations are rounded to the closest 100.
   - The considered candidate must be a subfactor of at least two other factors found in the data. This prevents that a rounding degree only “explains itself”. For example $4,000$ would not be considered to be an actual rounding degree if 27 percent of the individuals reported a value of $4,000$, but no one reported $8,000$ or $12,000$ etc. This condition ensures that lower (and thus more general) rounding degrees such as 1000 are favored.

Imprint

IAB-Discussion Paper 16/2018
5 June 2018

Editorial address
Institute for Employment Research
of the Federal Employment Agency
Regensburger Straße 104
D-90478 Nuremberg

Editorial staff
Ricardo Martinez Moya, Jutta Palm-Nowak

Technical completion
Renate Martin

Website
http://www.iab.de

Download of this Discussion Paper
http://doku.iab.de/discussionpapers/2018/dp1618.pdf

ISSN 2195-2663

For further inquiries contact the authors:

Matthias Speidel
Phone +49.911.179.7956
E-mail Matthias.Speidel@iab.de

Jörg Drechsler
Phone +49.911.179.4021
E-mail Joerg.Drechsler@iab.de