Utrecht University

LISS Income Imputations

Adam Maghout, Quinty Boer, Ward Eiling & Zhipei Wang

Survey Data Analysis 201300001

Utrecht University

Faculty of Social and Behavioural Sciences

Methodology and Statistics for the Behavioural, Biomedical and Social Sciences

Contents

1 Introduction

1.1 LISS panel survey

The LISS (Longitudinal Internet Studies for the Social Sciences) panel is a longitudinal panel survey conducted online by CentERdata from Tilburg University in which participants are required to fill in a set of questions every month concerning work, education, descent, income, political views, values and personality (Scherpenzeel, 2011). The sample frame of the LISS is the nation-wide address frame of Statistics Netherlands. A simple random sample of 10,150 addresses was drawn from this frame at the inception of the panel in 2007, with only households in which at least one individual understood Dutch being included in the survey (Scherpenzeel, 2011). Households that did not have access to internet or a suitable device were provided with one by CentERdata in order to account for coverage bias. This was a particular concern in 2007 as internet coverage was only 88% (Knoef & Vos, 2008) compared to over 95 % in 2021 in the Netherlands (de Clercq et al., 2023).

The eight core modules of the panel are the following:

- Family and Household
- Economic Situation and Housing
- Work and Schooling
- Social Integration and Leisure
- Health
- Personality
- Religion and Ethnicity
- Politics and Value

These are supplemented with an open access data collection, the themes of which are put forward by external academic researchers (Scherpenzeel, 2011).

Only four data sets from the LISS panel are reviewed here, containing the core variables of the LISS panel from July 2018, July 2019, July 2020 and July 2021. The variable of interest was nettoink, the net household monthly income in euros. Associated Missing values were imputed by the original researchers and were stored under the name nettoink_f.

1.2 Income-related questions and missingness

There exists a wealth of research supporting the claim that information about net income is harder to acquire than other types of information. Indeed, Yan et al. find that 20-40% item non-response is to be expected on income-related questions (Yan et al., 2010). Non-repsonse rates for the current data are presented in the table below and are uncharacteristically lower, perhaps due to the sampling frame or the high unit non-response, evidenced by large fluctuations in sample sizes between waves.

There is an ongoing discussion surrounding the usefulness and methods for imputing missing income values.

	2018	2019	2020	2021	Overall
	(N=10702)	(N=9649)	(N=11040)	(N=10110)	(N=41501)
nettoink					
Mean (SD)	1350 (3270)	1390 (2450)	1470 (2750)	1520 (2810)	1430 (2840)
Median [Min, Max]	1250 [0, 231000]	1250 [0, 145000]	1390 [0, 147000]	1450 [0, 147000]	1300 [0, 231000]
Missing	714 (6.7%)	692 (7.2%)	779 (7.1%)	693 (6.9%)	2878 (6.9%)

Unfortunately, in the case of the LISS panel, no information is provided in the datasets or documentation concerning the likely missing data mechanism or the reason why the data is missing (such as not knowing or refusing). Chi-squared test comparing net income being missing and other variables can however be conducted to evaluate whether MCAR (Missing Completely At Random) can be assumed. P-values from these tests

across all years for various variables are presented in the following table. Comparisons for specific levels of categorical variables are not presented for conciseness.

```
##  Missing data analysis: nettoink      p
##                         geslacht  0.001
##                          positie <0.001
##                         leeftijd <0.001
##                         aantalhh <0.001
##                         aantalki <0.001
##                          partner  0.386
##                         burgstat <0.001
##                         woonvorm <0.001
##                           woning <0.001
##                         belbezig <0.001
##                           oplcat <0.001
```

Since most of the obtained p-values are small, it is unlikely that the missing data is MCAR. Upon further investigation, several intuitive patterns between net household income being missing and other variables can be drawn. Participants that are not the household head (included in positie) are less likely to have provided their net household income, perhaps because that information was not available to them. This effect carries over to the number of people living in the household (aantalhh), since the higher the number, the less likely it is that the respondent will be the household head. This theory is further supported by a chi-square test between both variables (p-value $<$ 2.2e-16), suggesting a link between them. Similarly, participants who were unable to work or were not working (belbezig) are more likely to have provided their net household income, perhaps because this did not involve disclosing their individual income. Respondents whose highest level of education was primary school (oplcat) are also more likely to have responded. This may be due to other respondents having a higher monthly income and thus not wanting to declare it. This poses a problem for imputation as it will be hard to predict higher values where they were not given. The aforementioned patterns provide some insight into whether the missing values were obtained through lack of knowledge or refusal.

1.3 LISS imputation procedure

According to the documentation, imputation of missing net income values was initially performed for participants whose gross income was available. This also holds true for the other way around, insofar as gross income was imputed from available net income values. The formula used to do so is given by:

$$nettoink_f = exp(-0.5059772716 + ln(\text{gross income}) * 1.2084397177 + ln(\text{gross income})^2 * -0.0246538113$$
$$+(\text{age} > 64) * ln(\text{gross income}) * 0.0158967975 + (\text{household head}) * 0.0263173212 + (\text{paid employement}) * 0.2265126592$$
$$+(\text{paid employement}) * ln(\text{gross income}) * -0.0300171994$$

Where 'age $>$ 64', 'household head' and 'paid employement' are binary variables indicating whether the participant is strictly over 64, whether they are the head of their household and whether they are currently in paid employement, respectively.

Furthermore, since participants were also given the possibility to provide bracketed information on net income, the midpoints of these brackets were used as an estimate of exact net income. This was deemed acceptable by the researchers of the LISS panel as observed net income values in each bracket averaged their midpoints. This was however not relevant to the datasets at hand as none of the respondents provided only bracketed information. The following plots illustrate the final number of imputed values per year as well as the number of participants for which one year or more of net income is missing.

```
## Number of missing values on net income per year
```

```
##
##    0    1    2    3    4
## 7163  827 2835 1820  790
```

```
## Proportion of participants for which at least one value is missing
```

```
## [1] 0.4668403
```

```
## Number of imputed and remaining missing values per year
```

```
##                           2018 2019 2020 2021
## Imputed values              32   34   32   30
## Remaining missing values 3415 4444 3142 3988
```

Only a small proportion of missing values were imputed, whilst a considerable proportion (46.6%) of the participants failed to provide their net income on at least one occasion, according to the tables above. There thus remains a lot of missing values after addition of the imputations (labelled nettoink_f). The researchers admit in the documentation that other variables in the datasets could perhaps be used to impute values for respondents that did not provide any information about gross or net income. This was however not done as it was deemed too time-consuming.

A central objective of the work presented here is thus to improve on this imputation procedure by obtaining information about more of the missing values present in the datasets.

1.4 Longitudinal Imputation Methods

The current LISS imputation procedure has three considerable limitations. First, as imputations for the target variable net income are only performed when data regarding gross income is available, a large number of missing values is not imputed. Without further imputation, analysts and researchers will likely choose to ignore cases with item non-response in their analyses (list-wise or case-wide deletion). However, analysis of a similar social-economic longitudinal panel survey, the German Socio-Economic Panel Study (SOEP), showed that item non-response in survey questions regarding income was most prevalent in the tails of the income distribution, and particularly in the upper tale (Frick & Grabka, 2014). Consequently, ignoring non-response can introduce bias and underestimate variance.

Secondly, the current LISS imputation procedure only considers data from the same panel wave for the imputation procedure. As a result, valuable information that may have been obtained in previous waves are not considered in the imputation procedure.

Finally, the procedure is based on simple imputation, which does not account for the uncertainty caused by missing values. Observed values and imputed values are treated equally in analyses following single imputation, which may cause underestimation of the variance. The alternative to single imputation, multiple imputation, attempts to reflect the uncertainty of the missing data by creating multiple imputed datasets, analysing these, and combining the results (Buuren, 2018).

Many single imputation methods are available for longitudinal survey panel data, including hotdeck methods, carryover methods, nearest-neighbour regression, and row-and-column methods. Whereas hotdeck and regression methods tend to use cross-sectional information from the same wave to impute the missing data, carryover methods exclusively use longitudinal data from different waves to impute the data. Some examples of carryover methods are the last observation carried forward (LOCF), the next observation carried backward (NOCB), and taking the average of the last known and next known values (Engels & Diehr, 2003). The row-and-column method, also known as the Little and Su method, is a nearest neighbour technique that considers both cross sectional and longitudinal information. This method incorporates both information about the overall trend of the data and the single unit levels (Scholtus et al., 2014). Compared to other single imputation methods, it tends to have a higher predictive, distributional, and estimation accuracy (Watson & Starick, 2011). This method also tends to be robust against violations of the missing-at-random (MAR) assumption and therefore often a considered method when data is expected to be missing not at random (MNAR)). This Little and Su method is used in many national panel studies, including the German

Socio-Economic Panel Study (SOEP) (Frick & Grabka, 2014) and the Longitudinal Study of Australian Children (LSAC).

Although the Little and Su method seems to perform better than other single imputation methods, it still suffers from the disadvantages of being a single imputation method (Spiess et al., 2021). Furthermore, although imputed data with the Little and Su method seems to be very reliable when the data is used in cross-sectional analyses, the performance seems lower in longitudinal analyses. For example, when looking at income mobility over different panel waves the Little and Su method seems to somewhat overestimate short-term mobility and under-estimate long-term mobility (Lipps & Kuhn, 2023; Westermeier & Grabka, 2016). Although it might theoretically be possible to extend the Little and Su method to multiple imputation, there is little information available on how this would work in practice. More recent literature has therefore focused on the possibilities extending multiple imputation methods to longitudinal survey panel data (Huque et al., 2018; Van Buuren et al., 2011). The two most common methods are joint modelling (JM) and fully conditional specification (FCS). The last method is also known as sequential regression or the chained equations method. Joint modeling tends to work well on continuous data but assumes a joint normal distribution of the data. The FCS method imputes missing data on a variable-by-variable basis and therefore does not rely on the joint normality assumption. Although it is slower than joint modeling imputation and is more susceptible to converging issues, it is often the recommended method when dealing with different variable types. This method also tends to perform well for longitudinal wealth mobility analyses (Westermeier & Grabka, 2016). Both methods are available within the 'mice' library in R.

1.5 Aims and objectives

Given the constraints and limited use of the current LISS imputation procedure, the objectives of the current report include extending the LISS imputation procedure by incorporating longitudinal data and additional covariates, and exploring the merits and drawbacks of multiple imputation versus stepwise imputation on extensive datasets. Although the research is primarily exploratory, answering questions surrounding the use of various multiple imputation procedures, such as what advantages FCS holds over joint modelling, remains a key aspect of the project. In order to compare several imputation models, including the original LISS imputation framework, median income and income distribution across age will be treated as outcome variables and compared across models.

2 Imputations

In the following section, two different imputation methods will be applied to the LISS panel data from 2018 to 2021. The first method is a stepwise approach that directly builds upon the imputation method already developed by the LISS researchers. In the first step, available cross-sectional data is used to impute the target variable net income ('nettoink') with the equation developed by LISS. The second step also uses longitudinal data from different waves to impute missing values by applying the row-and-column or Little and Su method. As the Little and Su imputation method only works for units with at least one observation in the available waves, units with missing data in all available waves are not imputed. In an optional third step, these missing values are then imputed by multiple imputation with chained equations (MICE) or stochastic regression. Whether this last step is performed should be decided by the researcher based on their research questions, hardware limitations and the number of still missing values.

The second method exclusively uses multiple imputation with fully conditional specification (FCS) or the chained equations method. This imputation method has the advantage of using both longitudinal and cross-sectional data and allows for the specification of covariates to be used in the imputation. Furthermore, by creating multiple imputed datasets and combining these results, this imputation method better reflects the uncertainty introduced by missing data and can therefore more accurately estimate variances.

The main advantage of having two imputation methods available for the LISS panel data is that is allows the researcher to choose an approach that fits their research questions or other possible considerations. Based on the previous research into imputation methods for longitudinal data, we expect that the relatively simpler Little and Su imputation method will suffice for cross-sectional analysis, for example when interested in

population averages or income inequality. Strong expectations that the missing data is MNAR might also be a consideration that leads to choosing the stepwise approach. On the other hand, if the research question(s) require longitudinal data analysis, an FCS multiple imputation method will likely be preferred. Hardware limitations, running time, and size of the data (number of waves and/or covariates) may also be considerations that researchers would like to take into account when choosing an imputation method.

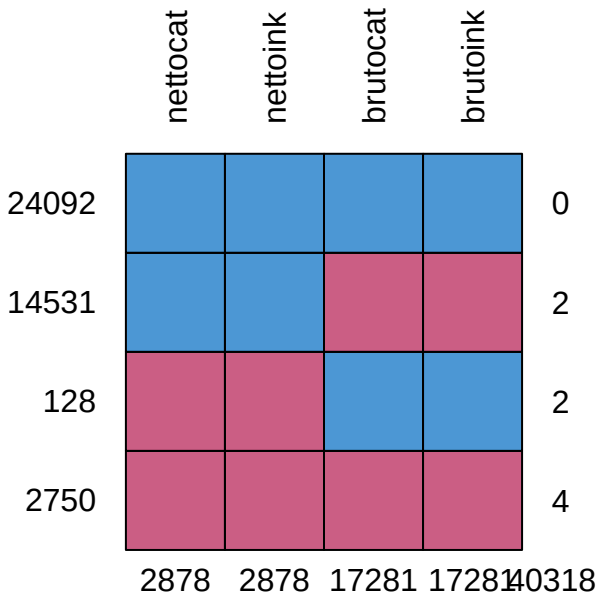
In the following section, the steps that were taken in pre-processing the LISS panel data are briefly discussed. This is followed by the explanations and implementations of the two imputation approaches, as well as the exploration of relevant considerations like the choice of covariates and the predictive power of previous waves.

2.1 Primary Data Pre-processing

In the initial general pre-processing procedure, we first imported the four .sav data files and combined them to form one dataframe in long format (i.e., with year as a variable). After this, several variables were judiciously excluded based on redundancy and relevance considerations. Specifically, "gebjaar" (Year of birth), "lftdcat" (Age in CBS categories), and "lftdhhh" (Age of the household head) were deemed redundant due to the presence of the variable "leeftijd" (Age of the household member). "Herkomstgroep" (Origin) faced removal due to an excessive amount of missing values and weak correlation with net income, as assessed through violin plots. Similarly, "sted" (Urban character of place of residence) was excluded owing to its weak association with income variables.

Given the pronounced correlation between "oplcat" (Level of education in CBS categories) and income, the variables "oplmet" (Highest level of education with diploma) and "oplzon" (Highest level of education irrespective of diploma) were removed to mitigate redundancy. Variables such as "brutoink_f" (Personal gross monthly income in Euros, imputed), "nettoink_f" (Personal net monthly income in Euros, imputed), "netinc" (Personal net monthly income in Euros, not including nettocat), "doetmee" (Household member participates in the panel or not), "werving" (From which recruitment wave the household originates), "simpc" (Does the household have a simPC), and "nohouse_encr" (Number of the household encrypted) were excluded due to their non-relevance in the imputation process. Furthermore, the variables brutocat (Personal gross monthly income in categories) and nettocat (Personal net monthly income in categories) were excluded as there were only 2 cases were these variables were non-missing and the variables nettoink and brutoink were not missing respectively.

```
# select data
subset <- dataL %>%
  select(nettocat, nettoink, brutocat, brutoink)
# plot the missing data pattern
md.pattern(subset, rotate.names = TRUE)
```




```
##       nettocat nettoink brutocat brutoink
## 24092        1        1        1        1     0
## 14531        1        1        0        0     2
## 128          0        0        1        1     2
## 2750         0        0        0        0     4
##             2878     2878    17281    17281 40318
```

Subsequently, "nettoink" and "brutoink" were transformed from the factor to character class and then to numeric class. This multi-step process was necessary to address potential issues that could arise when directly converting to numeric. Missing values in "nettoink" and "brutoink" were appropriately recoded to NA in accordance with the codebook. Empty levels of categorical variables were systematically removed.

We removed the variables gebjaar (Year of birth), lftdcat (Age in CBS categories) and lftdhhh (Age of the household head) because they are redundant considering the presence of the variable leeftijd (Age of the household member). Herkomstgroep (origin) was removed because it contained too much missingness. sted (Urban character of place of residence) is very weakly related to income variables so it was removed. Taking into account the strong relation of oplcat (level of education in CBS categories) and income, oplmet (Highest level of education with diploma) and oplzon (Highest level of education irrespective of diploma) were removed for redundancy. Since brutoink_f (Personal gross monthly income in Euros, imputed), nettoink_f (Personal net monthly income in Euros, imputed), netinc (Personal net monthly income in Euros, not including nettocat), doetmee (Household member participates in the panel or not), werving (From which recruitment wave the household originates), simpc (Does the household have a simPC), and nohouse_encr (Number of the household encrypted) are not relevant for imputation, they were also removed. Then nettoink and brutoink were first converted to character class and then to numeric class because they were originally saved as factor variables. Missing values in nettoink and brutoink were correctly recoded to NA according to the codebook. Levels of categorical variables that are empty were removed.

```r
### Data transformation ----

# Simplify the column for the year to each data frame
a2018$wave <- 2018
a2019$wave <- 2019
a2020$wave <- 2020
a2021$wave <- 2021

# Create a long dataset
dataL <- bind_rows(a2018, a2019, a2020, a2021)

# Transform the data
dataL <- dataL %>%
  # remove variables gebjaar, lftdcat and lftdhhh (are redundant considering "leeftijd")
  # Herkomstgroep (origin) has too many missing
  # sted (urban) is very weakly related to the income variables
  # oplmet and oplzon are redundant considering strong releation oplcat and income
  # brutoink_f, nettoink_f, nettoinc, doetmee, werving, simpc is not relevant for imputation
  select(-c("gebjaar", "lftdcat", "nohouse_encr", "herkomstgroep", "sted", "oplmet", "oplzon",
            "doetmee", "nettoink_f", "netinc", "brutoink_f", "nettohh_f", "brutohh_f", "werving",
            "simpc", "lftdhhh")) %>%
  # Fix the variable types
  mutate(nettoink = as.character(nettoink),
         nettoink = as.numeric(nettoink),
         brutoink = as.character(brutoink),
         brutoink = as.numeric(brutoink)) %>%
  # filter(nettoink < 15000) %>% # This was done by the original imputation as well (can be removed)
  # Fix the missing / don't know so that they are NA (according to codebook)
  # brutoink: 13 I don't know, 15 Unknown (missing)
  # nettoink: 13 I don't know, 14 Prefer not to say, 15 Unknown (missing)
  mutate(brutoink = ifelse(brutoink == 13 | brutoink == 15, NA, brutoink),
         nettoink = ifelse(nettoink == 13 | nettoink == 14 | nettoink == 15, NA, nettoink))

# remove the levels of categorical variables that are empty
dataL$aantalki <- droplevels(dataL$aantalki)
dataL$woning <- droplevels(dataL$woning)
dataL$woonvorm <- droplevels(dataL$woonvorm)
dataL$burgstat <- droplevels(dataL$burgstat)
dataL$positie <- droplevels(dataL$positie)
dataL$nettocat <- droplevels(dataL$nettocat)
dataL$brutocat <- droplevels(dataL$brutocat)
dataL$aantalhh <- droplevels(dataL$aantalhh)
dataL$aantalki <- droplevels(dataL$aantalki)
dataL$oplcat <- droplevels(dataL$oplcat)
dataL$geslacht <- droplevels(dataL$geslacht)
```

2.2 Stepwise approach

Building on what was previously done by the LISS panel, a stepwise approach to impute more missing values can be adopted. This approach is organised in several steps:

1. Implementing the initial LISS imputation procedure, detailed previously, to perform imputations in cases where gross income is provided but not net income. There is little reason to believe a better formula can be found in such cases, although its derivation could theoretically be verified through

replication.

2. Using the row-and-column or Little and Su method to impute missing values for units where at least one observation in a different wave is available. This builds upon the previous step by including longitudinal information from different waves in the imputation process.
3. Using multiple imputation with FCS to impute all item missing values that were not imputed in the previous steps.

2.2.1 Implementing the LISS procedure

Given the provided formula, the values in nettoink_f can be replicated using the following:

```
# Reshape the data to wide format
dataW <- dataL %>% reshape(
  idvar = "nomem_encr",  # specify the id column
  timevar = "wave",
  direction = "wide"
) %>% select(nomem_encr, sort(tidyselect::peek_vars())) #orders columns alphabetically

# Clean up variable names: replace dots with underscores
colnames(dataW) <- gsub("\\.", "_", colnames(dataW))

# List of dataset names
dataset_names <- c("a2018", "a2019", "a2020", "a2021")

# Loop through each dataset
for (dataset_name in dataset_names) {
  # Access the current dataset
  data <- get(dataset_name)

  # Create new variables for the imputation formula
  data$nettoink_f <- as.numeric(as.character(data$nettoink_f))
  data$nettoink <- as.numeric(as.character(data$nettoink))
  data$brutoink <- as.numeric(as.character(data$brutoink))
  data$lngross <- log(as.numeric(as.character(data$brutoink)))
  data$lngross2 <- data$lngross^2
  data$age65 <- as.numeric(ifelse(data$lftdcat == '65 years and older', 1, 0))
  data$lead <- as.numeric(ifelse(data$positie == 'Household head', 1, 0))
  data$employ <- as.numeric(ifelse(data$belbezig == 'Paid employment', 1, 0))

  # Create the imputation formula
  formula_terms <- c(
    "0.5059772716 * -1",
    "1.2084397177 * data$lngross",
    "0.0246538113 * -1 * data$lngross2",
    "0.0158967975 * data$age65 * data$lngross",
    "0.0263173212 * data$lead",
    "0.2265126592 * data$employ",
    "0.0300171994 * -1 * data$employ * data$lngross"
  )

  # Impute missing values for nettoink using the specified formula
  formula <- paste(formula_terms, collapse = " + ")

  data$nettoink_im <- ifelse(
    is.na(data$nettoink), eval(parse(text = paste("exp(", formula, ")", sep = ""))),
```

```
    as.numeric(as.character(data$nettoink))
  )

  # Store the imputed values in dataW based on matching nomem_encr
  col_name <- paste0('nettoink_im_', substr(dataset_name, 2, 5))

  for (i in 1:nrow(dataW)) {
    nomem_encr_val <- dataW[i, 'nomem_encr']

    if (nomem_encr_val %in% data$nomem_encr) {
      imputed_value <- data$nettoink_im[data$nomem_encr == nomem_encr_val]
      dataW[i, col_name] <- imputed_value
    }
  }
}

# Reordering the variables
dataW <- dataW %>% select(nomem_encr, sort(tidyselect::peek_vars())) #orders columns alphabetically
```

2.2.2 Using data from other years and the Little & Su method

2.2.2.1 Motivation A common method for imputation where longitudinal data is concerned is the Last observation carried forward (LOCF) method, where the last observed value of a variable is 'carried forward' to the current year. A major advantage of the LOCF method is that it maximizes the number of observations at any given year (Liu, 2016). In contrast, a drawback is that this may not make sense with the data, especially in situations where missing completely at random (MCAR) cannot be assumed. This is of particular concern in clinical trials where attrition may be due to participants' health deteriorating and thus imputing by LOCF does not provide adequate information on the participants' current situation (Liu, 2016). In the context of the LISS panel, this may be a case of participants being less likely to answer the question on net income if it has varied considerably from one month to the next. Furthermore, as with other single-point imputation methods, such as mean imputation, the LOCF method introduces bias in statistics of interest (Lachin, 2016).

The relevance of the LOCF procedure to the current problem can be assessed using the complete cases to measure the distance between the observed and predicted values imputed using the LOCF method. A t-test can then be conducted to test the hypothesis that the net income from one year to the next does not change. This can be performed on the data from 2019 to 2021 but not 2018 since no prior data was used in this project. Should the LOCF procedure prove to be useful, data from previous years could be retrieved online to impute values from 2018 in a similar way.

```
# Initialise vector for t values
tval <- matrix(NA, nrow = 3, ncol = 2)

for (target_year in 2018:2021){
  # Identify the net income variable for the current year
  target_var <- paste0("nettoink_im_", as.character(target_year))

  # Identify the net income variable for the previous year
  target_last <- target_year - 1
  target_varlast <- paste0("nettoink_im_", as.character(target_last))

  if (target_varlast %in% names(dataW)){
    tval[as.numeric(target_year) - 2018, 1] <- t.test(dataW[,target_var],
                                                      dataW[,target_varlast])$statistic # Calculate the
    tval[as.numeric(target_year) - 2018, 2] <- t.test(dataW[,target_var],
```

```
                                                    dataW[,target_varlast])$p.value # Calculate the p
  }
}

# Format column and row names
tval <- as.data.frame(tval)
row.names(tval) <- c('2019', '2020', '2021')
colnames(tval) <- c('t', 'p-value')

# Print the result
tval
```

```
##              t    p-value
## 2019 0.9419858 0.34621226
## 2020 2.0847098 0.03710879
## 2021 1.1589418 0.24649412
```

The hypothesis that net household income does not change from one year to the next cannot be rejected for 2019 and 2021 but is rejected for 2020. This is probably due to the Covid-19 pandemic that lasted through 2020 and 2021, which was accompanied by a decline in economic activity and resulted in many people losing their source of income (Han et al., 2020). Although this makes the use of LOCF harder to justify, using data from previous years may still be possible if trends are similar across participants. To examine this, cross-validation can be used to evaluate whether a linear model is successful at predicting net income values from the values of other years.

```
# Set the number of folds for cross-validation
num_folds <- 5

# Initialize a data frame to store results
cv_results <- data.frame(
  Year = character(),
  Fold = integer(),
  Adj_R_Squared = numeric(),
  stringsAsFactors = FALSE
)

# Loop through each year
for (target_year in c("2018", "2019", "2020", "2021")) {
  # Identify the net income variable for the current year
  target_var <- paste0("nettoink_", target_year)

  # Identify the predictor variables for the current year
  predictor_vars <- setdiff(paste0("nettoink_", c("2018", "2019", "2020", "2021")), target_var)

  # Create a formula for regression dynamically
  formula <- as.formula(paste(target_var, "~", paste(predictor_vars, collapse = " + ")))

  # Create a data frame with only complete cases for the current year
  complete_data <- na.omit(dataW[, c(target_var, predictor_vars)])

  # Set up k-fold cross-validation
  folds <- createFolds(complete_data[[target_var]], k = num_folds)

  # Loop through each fold
  for (fold in 1:num_folds) {
```

```
    # Extract the indices for training and testing data
    train_indices <- unlist(folds[-fold])
    test_indices <- unlist(folds[fold])

    # Subset the data for training and testing
    train_data <- complete_data[train_indices, ]
    test_data <- complete_data[test_indices, ]

    # Save the observed values for the current year
    observed_values <- test_data[[target_var]]

    # Fit the linear regression model
    lm_model <- lm(formula, data = train_data)

    # Predict the "missing" values using the regression model
    predicted_values <- predict(lm_model, newdata = test_data)

    # Calculate the adjusted R-squared
    adj_r_squared <- summary(lm_model)$adj.r.squared

    # Append the results to the cv_results data frame
    cv_results <- rbind(cv_results, data.frame(Year = target_year, Fold = fold,
                                               Adj_R_Squared = adj_r_squared))
  }
}

# Use tidyr's spread function to pivot the data frame
cv_results_pivot <- spread(cv_results, Fold, Adj_R_Squared)

# Print the pivoted results
print(cv_results_pivot)
```

```
##   Year         1         2         3         4         5
## 1 2018 0.9758068 0.9764595 0.9768109 0.9772475 0.9338912
## 2 2019 0.9809361 0.9798790 0.9817041 0.9853360 0.9494178
## 3 2020 0.9761449 0.9275732 0.9771566 0.9751343 0.9847891
## 4 2021 0.9662247 0.9668287 0.9199999 0.9743405 0.9662566
```

The adjusted R^2 is high for each fold and year, indicating that a model that predicts net income based on net income from other years is adequate for imputation. Thus, although the LOCF method may be limited and ill-fitted given the years the data was collected in, other imputation models based on previous data are the most straight-forward way of improving on the LISS imputation procedure.

2.2.2.2 Little & Su Method Implementation The Little & Su method consists of determining a column effect for each wave, a row effect for each unit, and a residual effect based on the nearest neighbour matching for each missing value. Consequently, it uses both cross-sectional trend information as well as individual longitudinal information into account in the imputation process and includes a stochastic component in the process by adding the residual. The imputed values are a product of the column effect, row effect, and residual. These values were calculated using the equations provided in a paper on longitudinal imputation methods by Westermeier and Grabka (Westermeier & Grabka, 2016). These equations were then cross-checked in a different technical paper by Mullan et al. on the imputation strategies used for imputing income in the LSAC, including the Little and Su approach (Mullan et al., 2015). These equations are as follows:

$$c_t = \frac{k \cdot \overline{y}_t}{\sum_t \overline{y}_t}$$

where c_t is the column effect for each separate wave t (where $t = 2018, 2019, 2020, 2021$);

$$r_i = \frac{1}{m_i} \cdot \sum_t \frac{y_{it}}{c_t}$$

where r_i is the row effect calculated for each unit in the sample, m_i the number of waves the individual's income was observed in, and y_{it} is the observed income for the individual in wave t;

$$residual\,term = \frac{y_{lt}}{r_l \cdot c_t}$$

where y_{lt} is the income of the nearest neighbour l and r_l the row effect of the nearest neighbour l. The nearest neighbour is found by reordering the data based on the calculated row effect. Finally, the imputed value is calculated by multiplying the column effect, row effect and residual term for each missing value.

One of the limitations of the Little & Su method is that an individual's missing value(s) can be only imputed if at least one non-missing value has been observed in one of the different waves. During the implementation of the method on the LISS panel data, we found that missing values within units with a calculated row effect of 0 could not be imputed as well. This was the case in instances where an individual had at least one missing value in a wave and an income value of 0 in the other waves.

```r
# smaller dataset with income data and to be imputed columns
dataW_inc <- dataW %>%
  select(nettoink_2018, nettoink_2019, nettoink_2020, nettoink_2021) %>%
  mutate(id = row_number()) %>%
  relocate(id) %>%
  add_column(nettoink_2018_imp = NA,
             nettoink_2019_imp = NA,
             nettoink_2020_imp = NA,
             nettoink_2021_imp = NA)
k <- 4 # number of waves

# create separate df for reordering
LS <- dataW_inc[, 1:(k+1)]

# calculate column effect for each wave
column_effects <- rep(0, k)
for(i in 1:length(column_effects)){
  yt <- mean(dataW_inc[,i+1], na.rm=T) # mean income for each wave
  yk <- sum(colMeans(dataW_inc[,2:(k+1)], na.rm=T))
  column_effects[i] <- (k * yt) / yk
}

# calculate row effect for each row
row_effects <- rep(0, nrow(dataW_inc))
for(i in 1:length(row_effects)){
  m <- sum(!is.na(dataW_inc[i, 2:(k+1)])) # number of waves each person was in
  row_effects[i] <- (1/m) * sum((dataW_inc[i, 2:(k+1)] / column_effects), na.rm=T)
}

# order responses by row effects
```

```
LS <- LS %>%
  mutate(row_effect = row_effects) %>%
  arrange(row_effect)

# determine complete case with highest row effect
highest_complete <- LS %>% na.omit() %>% filter(row_effect == max(row_effect)) %>% select(row_effect)

# start imputation loop (rows)
for(i in 1:nrow(dataW_inc)){

  # check if row has NA values and at least 1 observation
  mna <- sum(is.na(dataW_inc[i, 2:(k+1)])) # number of NA's per person
  if(mna > 0 & mna < k){

    # find row id and corresponding row effect in LS ordered data
    LS_row <- which(LS$id == dataW_inc[i, 1])
    row_effect <- LS[LS_row, 'row_effect']

    # find previous complete case
    prev_row_effect <- NA
    prev <- 1
    while(is.na(prev_row_effect) & LS_row != 1){
      prev_row <- LS[LS_row - prev,]
      if(sum(is.na(prev_row[2:(k+1)])) == 0){
        prev_complete <- prev_row
        prev_row_effect <- prev_row['row_effect']
      }
      prev <- prev + 1
    }

    # find next complete case
    next_row_effect <- NA
    nex <- 1
    while(is.na(next_row_effect) & row_effect < highest_complete){
      next_row <- LS[LS_row + nex,]
      if(sum(is.na(next_row[2:(k+1)])) == 0){
        next_complete <- next_row
        next_row_effect <- next_row['row_effect']
      }
      nex <- nex + 1
    }

    # determine nearest complete case
    if(LS_row == 1){
      nearest_row_effect <- next_row_effect
      nearest_complete <- next_complete
    } else if(row_effect > highest_complete) {
      nearest_row_effect <- prev_row_effect
      nearest_complete <- prev_complete
    } else if((next_row_effect - row_effect) < (row_effect - prev_row_effect)){
      nearest_row_effect <- next_row_effect
      nearest_complete <- next_complete
    } else {
```

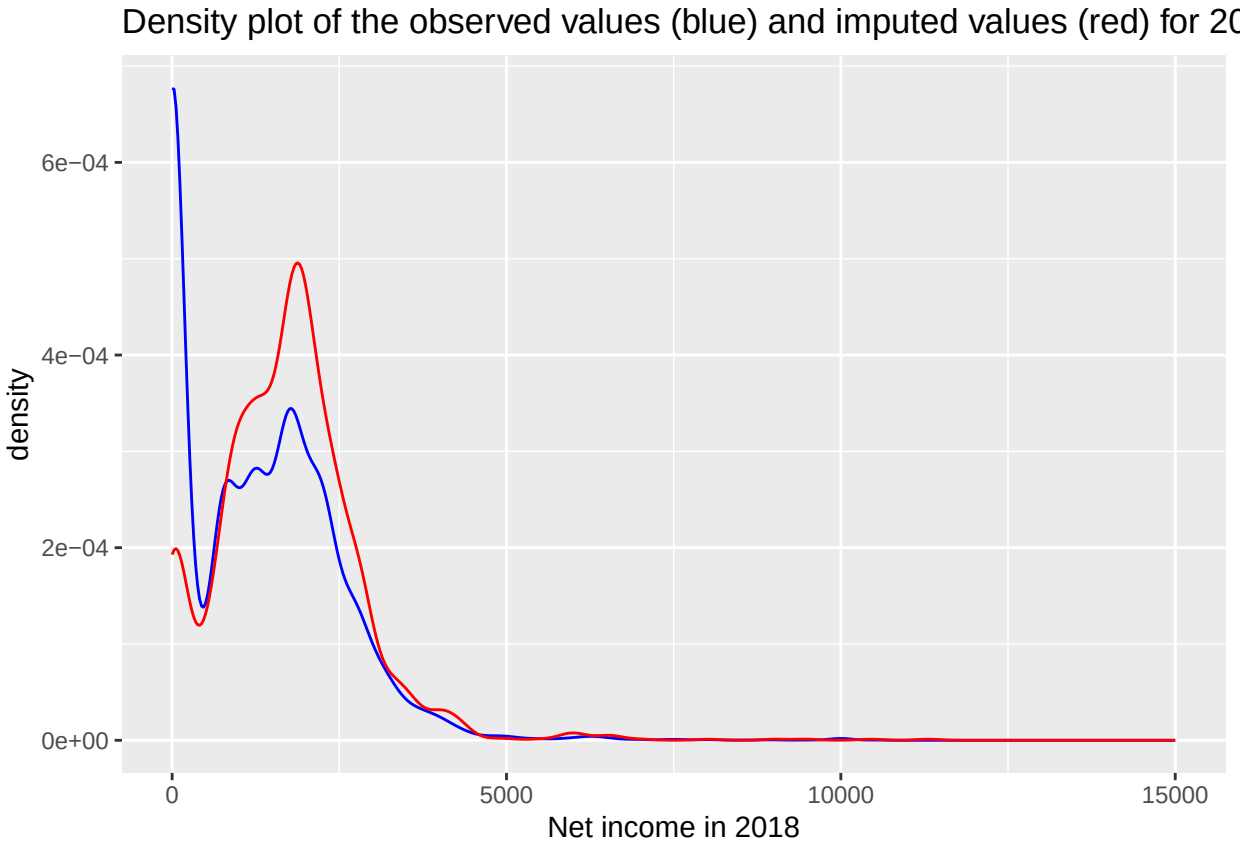
```
      nearest_row_effect <- prev_row_effect
      nearest_complete <- prev_complete
    }

    # loop over columns and check for NA values
    for(j in 2:(k+1)){
      if(is.na(dataW_inc[i,j])){

        # assign residual term from nearest complete case
        nearest_inc <- nearest_complete[j] # income of nearest complete case in same wave
        resid <- nearest_inc / (nearest_row_effect * column_effects[j-1])

        # impute missing value (column effect * row effect * residual)
        imputed_value <- row_effect * column_effects[j-1] * resid
        dataW_inc[i, j+k] <- imputed_value
      }
    }
  }
}

# set NaN imputed values to NA
dataW_inc[sapply(dataW_inc, is.nan)] <- NA
```

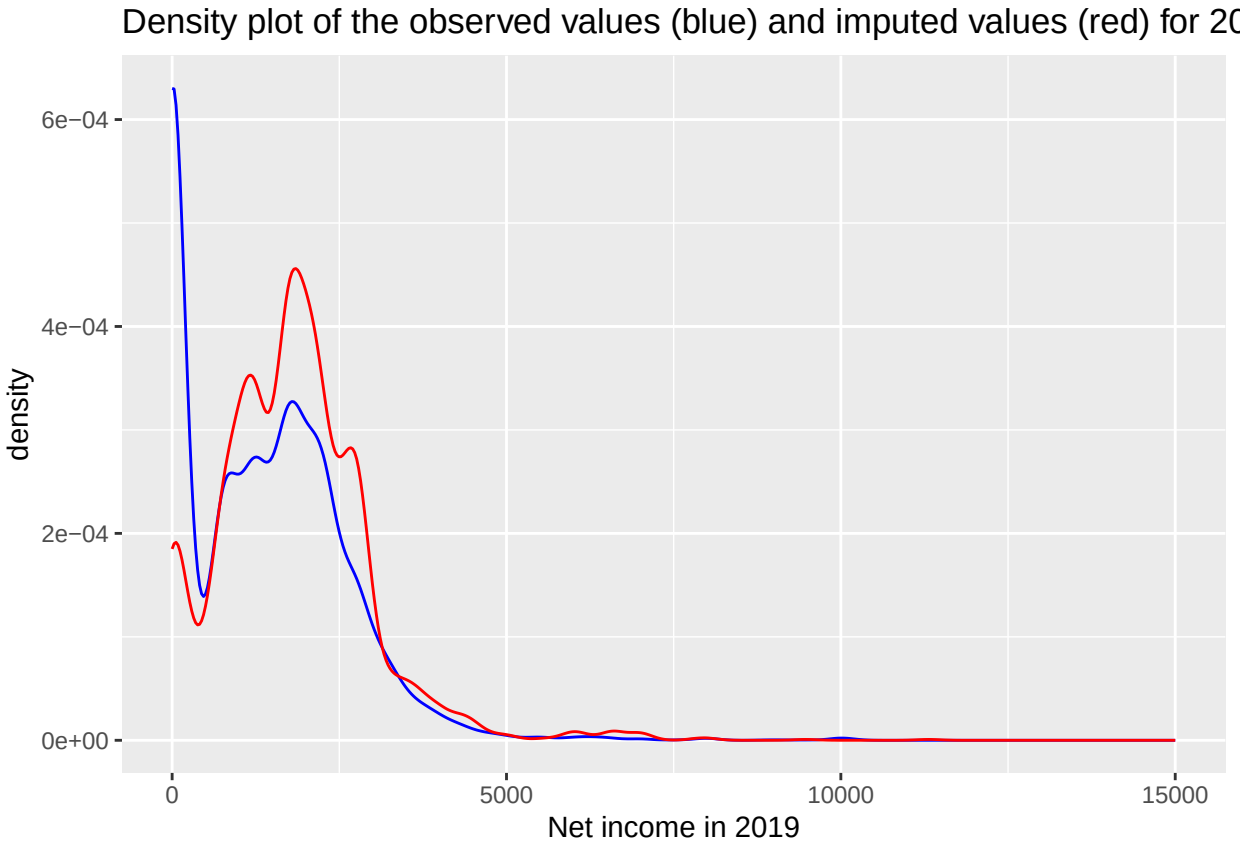
The table below shows the number of values that were imputed with the Little & Su method for each year. It also shows the total number of imputed values up to the this step of the stepwise approach and the number of values that are still missing.

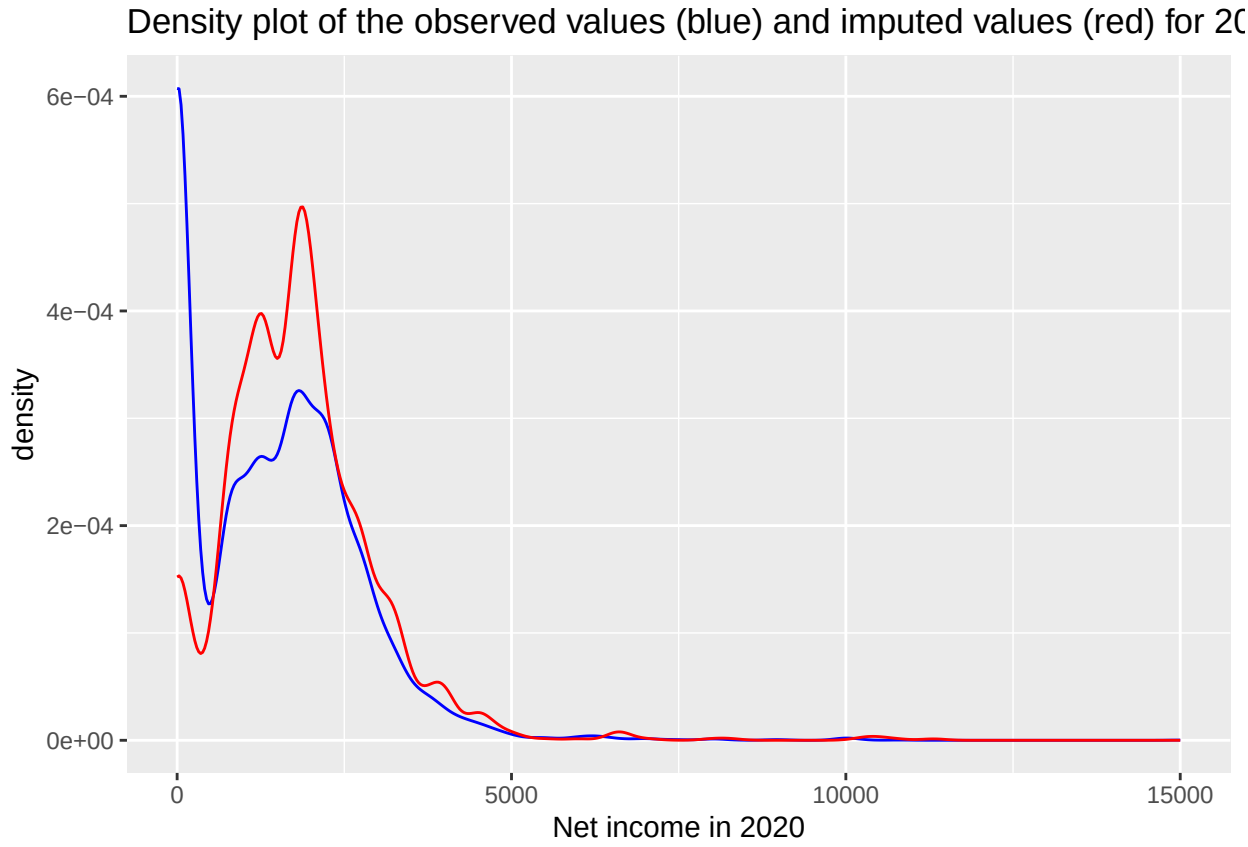
```
## Number of imputed and remaining missing values per year
```

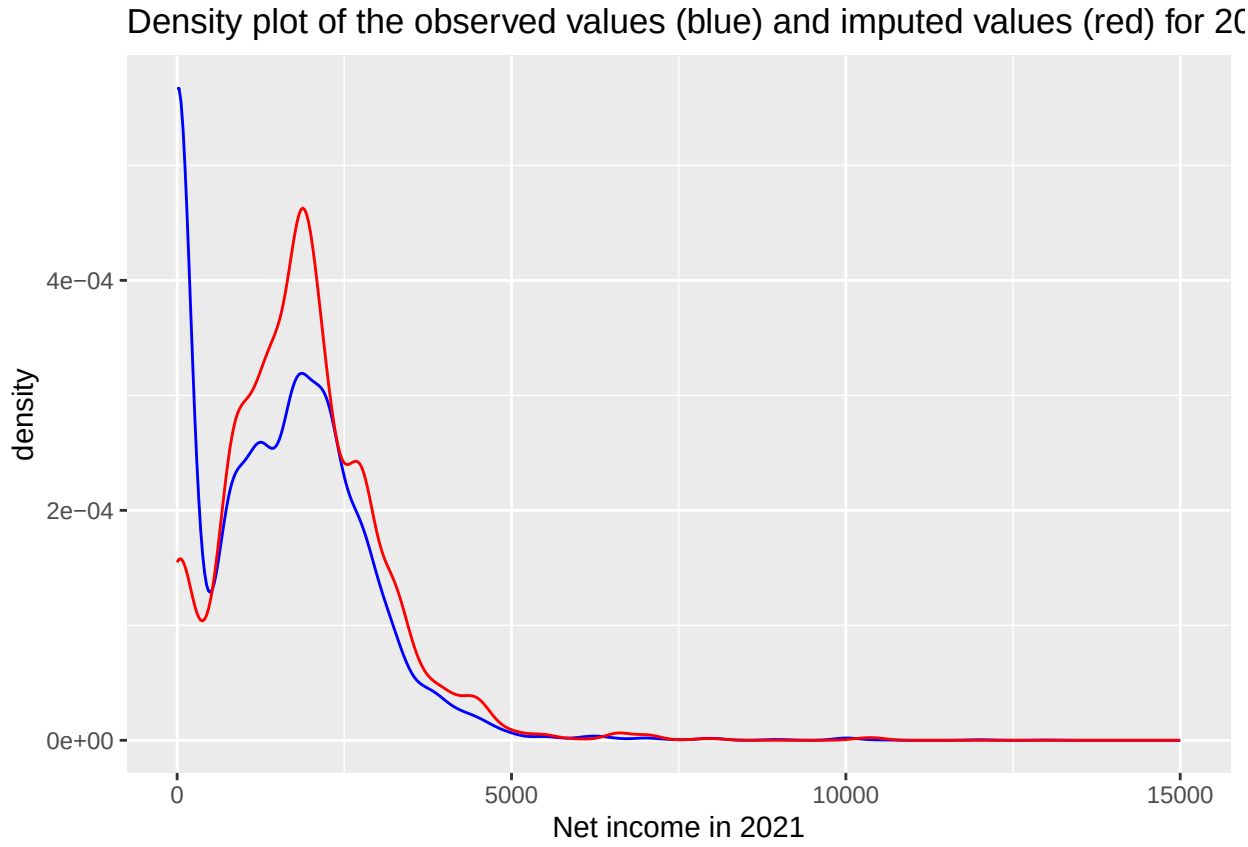
```
##                                 2018 2019 2020 2021
## Imputed values in the last step 1884 2666 1671 2352
## Total imputed values            1896 2676 1681 2360
## Remaining missing values        1551 1803 1494 1658
```

The graphs below show the density plots for the net income variable 'nettoink' in each year, where the blue line represents the original observed values and the red line the imputed values. In each year, the number of imputed values with a net income of 0 is relatively low compared to the number of observed values with a net income of 0. Consequently, the relative number of imputed values with a net income above 0 is considerably higher. This can also be seen in the accompanying table, which shows that the median income is slightly increased and the mean income is considerably increased when comparing the data after the Little & Su imputation with the data from after the LISS imputation. This could be an indication of overestimation of income, where the implemented Little & Su method has difficulties with the imputation of 0-value incomes. During the implementation of the method, we found that missing values within units with a combination of 0-values and missing values could often not be imputed due to the resulting 0-value row effect. However, it may also be the case that many of the missing values come from individuals that did not know their net income at the time of the survey, possibly resulting in a higher number of nonzero imputed values. Another consideration is that it may be possible that the missing values are distributed similarly over the net income variable as they are in the German SOEP, where missing values are most prevalent on the higher end of the income distribution (Frick & Grabka, 2014). However, with the LISS panel data currently available to us, we cannot make any conclusions on whether the Little & Su method has overestimated income or whether the imputations are in line with expectations.


Density plot of the observed values (blue) and imputed values (red) for 2(
6e−04
4e−04
2e−04
0e+00
density
0
5000
10000
15000
Net income in 2018


Density plot of the observed values (blue) and imputed values (red) for 2(
6e−04
4e−04
2e−04
0e+00
density
0
5000
10000
15000
Net income in 2019


Density plot of the observed values (blue) and imputed values (red) for 2(
6e−04
4e−04
2e−04
0e+00
density
0
5000
10000
15000
Net income in 2020




```
## # A tibble: 4 x 9
##   Statistic `LISS 2018` `L&S 2018` `LISS 2019` `L&S 2019` `LISS 2020` `L&S 2020`
##   <chr>           <dbl>      <dbl>       <dbl>      <dbl>       <dbl>      <dbl>
## 1 median          1250       1300        1250       1421.       1400       1480
## 2 mean            1354.      1439.       1393.      1514.       1471.      1545.
## 3 missing         3427       1551        4456       1803        3153       1494
## 4 nonmissi~      10008      11884        8979      11632       10282      11941
## # i 2 more variables: `LISS 2021` <dbl>, `L&S 2021` <dbl>
```

2.2.3 Imputing the remaining missing values with multiple imputation

After the two previous steps, there remains missing values for participants that either provided net income on none of the years or only ever provided 0 as their net household income. The number of remaining missing values for each year can first be tabulated.

```
## Remaining missing values per year, including and excluding unit non-response
```

```
##   Year Missing_Values_Including Missing_Values_Excluding
## 1 2018                     1551                      627
## 2 2019                     1803                      567
## 3 2020                     1494                      654
## 4 2021                     1658                      583
```

Since the aim of the current project is to correct for item non-response, the number of remaining missing values per year that can plausibly be imputed are in the right column that excludes unit non-response. This is also because imputation using longitudinal data was performed in the previous step, using the Little & Su method. This final step aims at using cross-sectional data to impute some of the remaining missing data, for which neither gross nor net household income is available. As such, the initial LISS imputation procedure

cannot be used as it relies on gross income data. This step can however be viewed as an extension of it, that is alluded to in the panel documentation but was not seen as a 'quick solution'. In order to identify variables for imputation, missing counts for the remaining missing values can be tabulated.

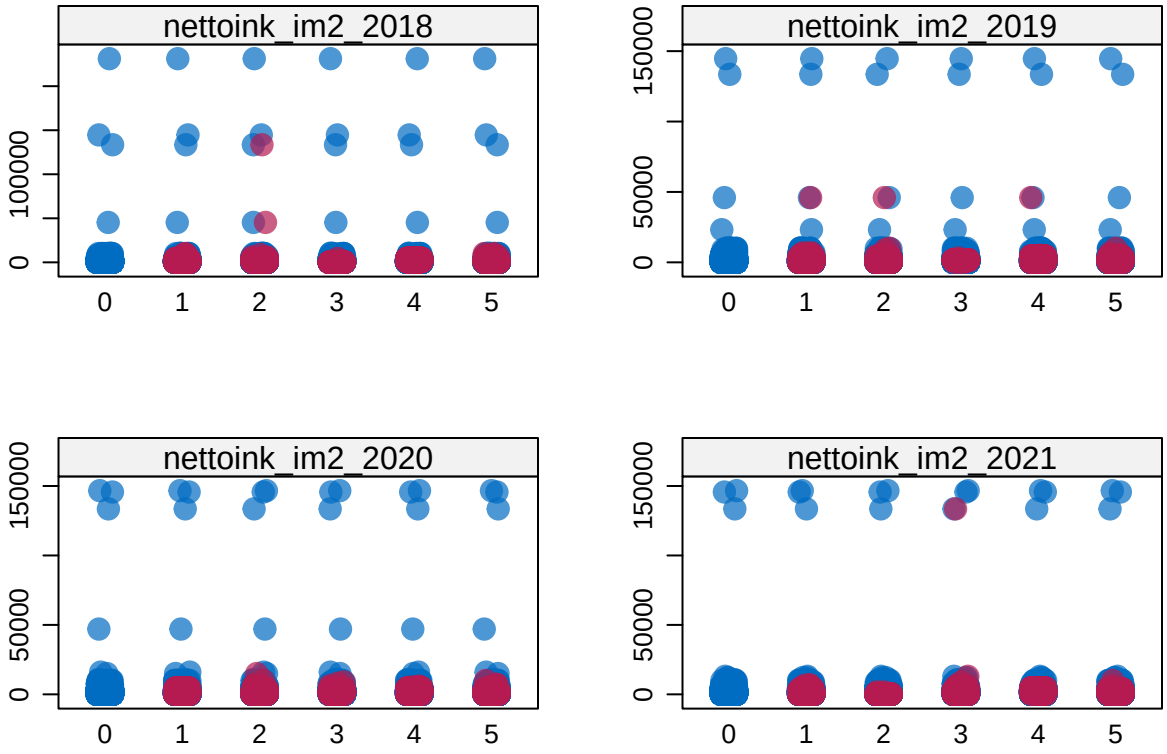
```
#Transform the data by removing all variables unusable for prediction
data2 <- dataW %>% select(-c("nomem_encr", "brutocat_2018", "nettocat_2018", "nettoink_2018", "nettoink
                             "brutocat_2019", "nettocat_2019", "nettoink_2019", "nettoink_im_2019",
                             "brutocat_2020", "nettocat_2020", "nettoink_2020", "nettoink_im_2020",
                             "brutocat_2021", "nettocat_2021", "nettoink_2021", "nettoink_im_2021",
                            "brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2021"))


#Initialize table to store missing values
missing_count <- matrix(NA, nrow = 4, ncol = 12)

for (year in 2018:2021) {
  # Extract the net income variable for the current year
  column_for_year <- grep(paste0("nettoink_im2_", year), colnames(data2), value = TRUE)

  # Choose only selected year
  data3 <- data2[, grepl(year, names(data2))] # Check columns from selected year only
  cnt_na <- apply(data3, 1, function(z) sum(is.na(z))) # Count the number of NAs in each row
  rows_with_missing <- which(cnt_na < 11) # Find rows from the correct year

  # Subset the data to obtain data from only one dataset
  data3 <- data2[rows_with_missing, grepl(year, names(data2))]

  # Count the number of missing values per column
  na_count <-sapply(data3, function(y) sum(length(which(is.na(y)))))
  missing_count[year - 2017,] <- na_count
}

# Naming the columns and rows
missing_count <- as.data.frame(missing_count) # Transforming the output

# Retrieve original variable names from the loop
colnames(missing_count) <- sub("_2021", "", names(na_count))

# Naming the rows
rownames(missing_count) <- 2018:2021

# Displaying the table
missing_count[,-12] # Remove net income from the output
```

```
##      aantalhh aantalki belbezig burgstat geslacht leeftijd oplcat partner
## 2018        0        0        4        0        0        0    549       0
## 2019        0        0        3        0        0        0    499       0
## 2020        0        0        0        0        0        0    611       0
## 2021        0        0        0        0        0        0    580       0
##      positie woning woonvorm
## 2018     130     20        0
## 2019     133     22        0
## 2020     122     15        0
## 2021     140     21        0
```

For performing imputation using linear or stochastic regression, Hardt et al. note that utilising too many auxiliary variables may lead to an over-parameterisation of the imputation model and thus introduce bias, whilst simultaneously lengthening processing times (Hardt et al., 2012). Hence, since there are comparatively a lot of missing values on the variables oplcat (level of education) and positie (position in the family) for the remaining missing data, they are excluded from the imputation. In the case of cross-sectional imputations on longitudinal data, multiple imputation has been shown to yield values close to true parameter values (Ferro, 2014). Including variables highly correlated with item missingness also reduces bias from differences between MAR and MNAR (Ferro, 2014). Since most of the remaining items have been shown in section 1.2 to be correlated to the missingness on net income, it was decided here to attempt to impute the few remaining missing values using multiple imputation. This is done for each year separately to avoid introducing error from the repeated-measure structure of the data. Performing 2 iterations is chosen here for ease of computation but could be extended to 20 to conform with the minimum suggested in the literature when looking for standard error estimates (**Hippel?**). Five imputed datasets are computed, in line with recommendations from the literature that the number of iterations should match the average proportion of missing data (Bodner, 2008). All imputations were performed with the mice package, developed by van Buuren & Groothuis-Oudshoorn (van Buuren & Groothuis-Oudshoorn, 2011).

```
# Removing variables
data3 <- data2 %>%
  select(-c("oplcat_2018", "positie_2018",
       "oplcat_2019", "positie_2019",
       "oplcat_2020", "positie_2020",
       "oplcat_2021", "positie_2021"))

for (year in 2018:2021) {
  # Extract the net income variable for the current year
  column_for_year <- grep(paste0("nettoink_im2_", year), colnames(data3), value = TRUE)

  # Choose only selected year
  data4 <- data3[, grepl(year, names(data3))] # Check columns from selected year only
  cnt_na <- apply(data4, 1, function(z) sum(is.na(z))) # Count the number of NAs in each row
  rows_with_missing <- which(cnt_na < 9) # Find rows from the correct year panel

  # Subset the data to obtain data from only one dataset
  data4 <- data3[rows_with_missing, grepl(year, names(data3))]

  # Create the imputations
  imputed_data <- mice(data4, m = 5, maxit = 2, seed = 54433, print = FALSE)

  # Extract the imputed datasets
  imputed_datasets <- complete(imputed_data, "long", include = TRUE)

  # Create variables to store the imputations and imputed datasets
  imputed_data_name <- paste0("imputed_data_", year)
  imputed_datasets_name <- paste0("imputed_datasets_", year)

  # Assign the results to the newly created variables on each loop
  assign(imputed_data_name, imputed_data)
  assign(imputed_datasets_name, imputed_datasets)
}
```

This code has thus imputed every value that was missing from the original data sets. A more in-depth discussion of multiple imputation methods, including arguments that the mice function can take, is presented in the next section but a quick glance at plots of imputed against observed data reveals that the new values for net household income are plausible, where the observed data is in blue, the imputed is in red.



2.3 Multiple imputation Approach

2.3.1 Secondary Data Pre-Processing

Following the initial primary data pre-processing, we conducted additional data pre-processing to optimize the dataset for the subsequent multiple imputation process. This step was essential given the dataset's size and the computational complexity associated with imputing categorical variables using algorithms such as "polyreg" and "polr."

The variables "aantalki" (Number of living-at-home children in the household) and "aantalhh" (Number of household members) underwent transformation from character string entries to numeric values. This conversion significantly reduced computation time, as the "pmm" algorithm runs more efficiently on numeric values compared to the "polr" algorithm on ordinal variables.

```
# Transforming the categorical variables "aantalki" and "aantalhh" to continuous

# changing the character strings to numeric values for variable aantalki
dataL$aantalki <- as.character(dataL$aantalki)
# with "None" = 0
# with "One child" = 1
# with "Two children" = 2
# with "Three children" = 3
# with "Four children" = 4
# with "Five children" = 5
# with "Six children" = 6

dataL$aantalki[dataL$aantalki == "None"] <- 0
```

```r
dataL$aantalki[dataL$aantalki == "One child"] <- 1
dataL$aantalki[dataL$aantalki == "Two children"] <- 2
dataL$aantalki[dataL$aantalki == "Three children"] <- 3
dataL$aantalki[dataL$aantalki == "Four children"] <- 4
dataL$aantalki[dataL$aantalki == "Five children"] <- 5
dataL$aantalki[dataL$aantalki == "Six children"] <- 6

dataL$aantalki <- as.numeric(dataL$aantalki) # Convert to numeric

# Make a binary variable from aantalki indicating whether or not someone has children (simplied variabl
# dataL <- dataL %>%
#   mutate(has_children = ifelse(aantalki == 0, 0, 1)) # 0 = no children, 1 = has children

# Do the same for variable aantalhh
dataL$aantalhh <- as.character(dataL$aantalhh)
# with "One person" = 1
# with "Two persons" = 2
# with "Three persons" = 3
# with "Four persons" = 4
# with "Five persons" = 5
# with "Six persons" = 6
# with "Seven persons" = 7
# with "Eight persons" = 8
# with "Nine persons or more" = 9

dataL$aantalhh[dataL$aantalhh == "One person"] <- 1
dataL$aantalhh[dataL$aantalhh == "Two persons"] <- 2
dataL$aantalhh[dataL$aantalhh == "Three persons"] <- 3
dataL$aantalhh[dataL$aantalhh == "Four persons"] <- 4
dataL$aantalhh[dataL$aantalhh == "Five persons"] <- 5
dataL$aantalhh[dataL$aantalhh == "Six persons"] <- 6
dataL$aantalhh[dataL$aantalhh == "Seven persons"] <- 7
dataL$aantalhh[dataL$aantalhh == "Eight persons"] <- 8
dataL$aantalhh[dataL$aantalhh == "Nine persons or more"] <- 9

dataL$aantalhh <- as.numeric(dataL$aantalhh) # Convert to numeric
```

Subsequently, to address sparsity and cardinality issues in the categorical variables "belbezig" and "burgstat," we collapsed categories based on certain criteria. First, "belbezig" (Primary occupation) underwent categorization, where categories with proportions below a 10% threshold were collapsed. This process resulted in four refined levels: "Attends school or is studying," "Is pensioner," "Paid employment," and "Others." Second, in the case of "burgstat" (Civil status), the categories "Divorced" and "Separated" were merged into a unified category named "Divorced or separated." This amalgamation was informed by a synthesis of theoretical and empirical factors, as both categories exhibited analogous relationships with net income.

```r
### Dealing with high cardinality in belbezig and burgstat by collapsing categories
# table(dataL$belbezig)
# prop.table(table(dataL$belbezig)) * 100

## Any category has a percentage smaller than 10 is going to be into "others"
belbezig_percentages <- prop.table(table(dataL$belbezig)) * 100
categories_to_collapse <- names(belbezig_percentages[belbezig_percentages < 10])
dataL$belbezig_collapsed <- ifelse(dataL$belbezig %in% categories_to_collapse, "Others", as.character(da
dataL$belbezig_collapsed <- factor(dataL$belbezig_collapsed)
```

```r
# table(dataL$belbezig_collapsed) / length(dataL$belbezig_collapsed) * 100 # Print the updated table of
# summary(aov(nettoink ~ belbezig_collapsed, dataL)) # still significant result
# lm(nettoink ~ belbezig_collapsed, dataL) # still significant result)

### Combine divorced and separated into one level of burgstat
dataL$burgstat_collapsed <- ifelse(dataL$burgstat %in% c("Divorced", "Separated"), "Divorced or separat
dataL$burgstat_collapsed <- factor(dataL$burgstat_collapsed)

# summary(lm(nettoink ~ burgstat, dataL))
# summary(lm(nettoink ~ burgstat_collapsed, dataL)) # Practically no difference in variance accounted f

# Remove the original variables
dataL <- dataL %>%
  select(-belbezig, -burgstat)
```

Following these adjustments to categorical variables, we reshaped the data from a long to a wide format, facilitating the multiple imputation procedure through the mice package.

```r
# Reshape the data to wide format
dataW <- reshape(
  data = dataL,
  idvar = "nomem_encr",  # specify the id column
  timevar = "wave",
  direction = "wide"
)

# Clean up variable names: replace dots with underscores
colnames(dataW) <- gsub("\\.", "_", colnames(dataW))
```

Finally, the representation of gender and age variables are simplified to prevent multicollinearity violations and unnecessary clutter in the predictor matrix. A new variable, "geslacht," was created by consolidating gender information from any available year (2018 to 2021) to form a unified gender variable for each observation. Additionally, a baseline age variable, "leeftijd_2018," is generated based on the available data from the respective years. The code utilizes conditional logic to assign the age value for each year. If the age information is present, it is assigned accordingly. Otherwise, the age is imputed by subtracting a predetermined value based on the temporal distance from the most recent year. It should be noted that after this procedure, both variables had no missing values, removing the need for imputation.

```r
# Simplifying the sex and age variables
dataW <- dataW %>%
  # Create a single gender variable using whichever year is available
  mutate(geslacht = coalesce(geslacht_2018, geslacht_2019, geslacht_2020, geslacht_2021)) %>%
  # Create a baseline age variable using whichever year is available
  mutate(leeftijd_2018 = case_when(
    !is.na(leeftijd_2018) ~ leeftijd_2018,
    !is.na(leeftijd_2019) ~ leeftijd_2019 - 1,
    !is.na(leeftijd_2020) ~ leeftijd_2020 - 2,
    !is.na(leeftijd_2021) ~ leeftijd_2021 - 3,
    TRUE ~ NA_real_))
```

2.3.2 Covariate exploration

In the preliminary covariate exploration outlined in the interim report, we employed violin plots and conducted pairwise correlations. These yielded important insights into the relatedness to the net income variable. We first plotted the missingness across potential categorical variables to exclude any that was associated with too

much missingness. First, it became clear that approximately 7% of values was missing across all categorical variables. On top of that, a visual inspection revealed that the variable "herkomstgroep" (i.e., origin) contributed to about 30% additional missing values. To prevent complications with dealing with a lot of missingness in a categorical variable, this variable was excluded from further analysis. To check the suitability of categorical variables for the imputation model, we plotted the distribution of net income for each level of the categorical variable of interest. Violin plots can be insightful for exploring the distribution of a continuous variable across different levels of a categorical variable. By comparing the shapes of the violins, you can quickly assess whether there are significant differences in the distributions. The extent to which the violins are separated from each other provides an indication of how well the categorical variable differentiates the continuous variable. A clear separation suggests that the categorical variable may be a good covariate. Relatively small differences between the levels of the independent variable were identified for the variables already excluded variable origin and the variables household head living with a partner and urban character. Consequently, these variables were excluded from the imputation model. Moderate differences were found for gender, number of household members, number of living-at-home children, civil status, domestic situation, type of dwelling, and primary occupation. From these it can be concluded that men have a greater net income than women, that having more household members is associated with lower income, that having children is associated with lower income, that being married is associated with higher income, that living with children is associated with lower income, that living in a rental property is associated with lower income, and that being employed is associated with higher income. Great differences were found for position within the household: household heads and unwedded partners tend to earn the highest income. All variables with moderate to large differences were included in the imputation model.

Additionally, ANOVAs were utilized to explore the relationship between categorical variables and net income, while multiple linear regression models were fitted to investigate the relationship between continuous variables and net income. To gain further insights into the association between variables and net income, along with its associated missingness, we generated plots depicting variables against both the net income variable and its corresponding missingness. The findings from these analyses informed the selection of variables for the subsequent imputation model.

In the context of imputing net income, the LISS data offered several suitable continuous predictors. Through the examination of linear regression models, pairwise correlations, and unique variance considerations, we identified key variables: "brutoink," "aantalki" (converted), and "aantalhh" (converted). For categorical variables, the selected set included: "leeftijd," "geslacht," "oplcat," "burgstat" (collapsed), "woning," "belbezig" (collapsed), and "partner."

```
# Check the relationship between age and income
summary(lm(nettoink ~ leeftijd, data = dataL)) # Explains 4% of variance
# ... nettoink and aantalki
summary(lm(nettoink ~ aantalki, data = dataL)) # Explains 2% of variance
plot(nettoink ~ aantalki, data = dataL)
summary(lm(nettoink ~ has_children, data = dataL)) # Explains a little less than 2% of variance
# ... nettoink and aantalhh
summary(lm(nettoink ~ aantalhh, data = dataL)) # Explains 2% of variance
plot(nettoink ~ aantalhh, data = dataL)
summary(lm(nettoink ~ lives_alone, data = dataL)) # Explains a little less than 1% of variance (more in

# Check the multiple linear regression model
summary(lm(nettoink ~ brutoink + leeftijd + aantalki + aantalhh, data = dataL)) # aantalki and aantalhh

# Now include all variables that I also want to use for the actual imputation model
summary(lm(nettoink_2018 ~ brutoink_2018 + leeftijd_2018 + has_children_2018 + aantalhh + geslacht + bel

# Imputation model using continuous income variables (gross + net)
cont_var_2018 <- lm(nettoink_2018 ~ nettoink_2019 + nettoink_2020 + nettoink_2021 + brutoink_2018 + brut
summary(cont_var_2018)
```

```r
cont_var_2019 <- lm(nettoink_2019 ~ nettoink_2018 + nettoink_2020 + nettoink_2021 + brutoink_2018 + bru
summary(cont_var_2019)
cont_var_2020 <- lm(nettoink_2020 ~ nettoink_2018 + nettoink_2019 + nettoink_2021 + brutoink_2018 + bru
summary(cont_var_2020)
cont_var_2021 <- lm(nettoink_2021 ~ nettoink_2018 + nettoink_2019 + nettoink_2020 + brutoink_2018 + bru
summary(cont_var_2021)

# In cases where the gross income and net income are both non-missing, a linear model consisting of the
# of the other years accounts for more than 99% of the variance in the net income of the year in questi
# the net income of other years or the gross income is available, adding more predictors may be pointle

# Imputation model using continuous income variables (net)
cont_var_2018 <- lm(nettoink_2018 ~ nettoink_2019 + nettoink_2020 + nettoink_2021, data = dataW)
summary(cont_var_2018)
cont_var_2019 <- lm(nettoink_2019 ~ nettoink_2018 + nettoink_2020 + nettoink_2021, data = dataW)
summary(cont_var_2019)
cont_var_2020 <- lm(nettoink_2020 ~ nettoink_2018 + nettoink_2019 + nettoink_2021, data = dataW)
summary(cont_var_2020)
cont_var_2021 <- lm(nettoink_2021 ~ nettoink_2018 + nettoink_2019 + nettoink_2020, data = dataW)
summary(cont_var_2021)

# In cases where the net income of all years are non-missing, a linear model where a person's net incom
# by their net income in the other years accounts for more than 96%/97% of the variance in the net inco

# Plot cardinalities of categorical variables
categorical_vars <- c("geslacht", "oplcat", "partner", "brutocat", "nettocat",
                      "positie", "burgstat", "woonvorm", "woning", "belbezig")

plot_list <- list()

for (var in categorical_vars) {
  p <- ggplot(dataL, aes(x = !!sym(var))) +
    geom_bar(fill = "skyblue", color = "black") +
    labs(title = paste("Barplot of", var),
         x = var,
         y = "Count") +
    theme_minimal()

  plot_list[[var]] <- p
}

combined_plot <- grid.arrange(grobs = plot_list, ncol = 4)

print(combined_plot)

### Make a plot showing the missingness for variable "nettoink" across different levels of categorical

for (i in 1:length(categorical_vars)) {
  # Selecting the categorical variable for analysis
  categorical_variable <- categorical_vars[i]

  # Creating a subset of the data with selected variables
  subset_data <- dataL[c("nettoink", categorical_variable)]
```

```r
  # Creating an indicator variable for missingness in 'nettoink'
  subset_data$nettoink_missing <- is.na(subset_data$nettoink)

  # Calculating the percentage of missingness and non-missing data
  missingness_summary <- subset_data %>%
    group_by(!!as.symbol(categorical_variable), nettoink_missing) %>%
    summarise(count = n()) %>%
    group_by(!!as.symbol(categorical_variable)) %>%
    mutate(percentage = count / sum(count) * 100)

  # Plotting combined bar chart and printing it
  print(ggplot(missingness_summary, aes(x = !!as.symbol(categorical_variable), y = percentage, fill = fa
          geom_bar(stat = "identity", position = "stack") +
          labs(title = paste("Missingness in nettoink Across", categorical_variable),
               x = categorical_variable, y = "Percentage") +
          scale_fill_manual(values = c("TRUE" = "darkred", "FALSE" = "lightgreen"), labels = c("Non-Miss
          theme_minimal())
}

### Check how high the scores are

# Create a bar plot
ggplot(dataL, aes(x = oplcat, y = nettoink)) +
  geom_bar(stat = "summary", fun = "mean", fill = "skyblue", position = "dodge") +
  labs(title = "Nettoink Scores",
       y = "Mean Nettoink Score") +
  theme_minimal()
```

2.3.3 Creating the Multiple Imputation Model

The generation of multiply imputed data involved an iterative approach. Initially, a rudimentary imputation model was established. We then scrutinized diagnostic plots, such as traceplot, stripplot, and densityplot. Subsequently, we followed a systematic procedure: 1) assessed warning logs, 2) fine-tuned specific parameters, and 3) evaluated if these adjustments led to enhancements. This iterative cycle was repeated until we derived the definitive version of the imputation model outlined in the subsequent steps.

2.3.3.1 Step 1: Data Preparation In this step, we selected all relevant variables for imputation, including demographic information, income, and several categorical and continuous covariates across the different years (see covariate exploration).

```r
### Step 1: Prepare the data

# All years
allyears <- dataW %>%
  select(leeftijd_2018,
         aantalki_2018, aantalki_2019, aantalki_2020, aantalki_2021,
         aantalhh_2018, aantalhh_2019, aantalhh_2020, aantalhh_2021,
         brutoink_2018, brutoink_2019, brutoink_2020, brutoink_2021,
         nettoink_2018, nettoink_2019, nettoink_2020, nettoink_2021,
         # add the categorical variables (4 or less levels)
         geslacht,
         burgstat_collapsed_2018, burgstat_collapsed_2019, burgstat_collapsed_2020, burgstat_collapsed_
         partner_2018, partner_2019, partner_2020, partner_2021,
```

```
          woning_2018, woning_2019, woning_2020, woning_2021,
          # add the categorical variables (more than 4 levels)
          # woonvorm_2018, woonvorm_2019, woonvorm_2020, woonvorm_2021,
          belbezig_collapsed_2018, belbezig_collapsed_2019, belbezig_collapsed_2020, belbezig_collapsed_
          oplcat_2018, oplcat_2019, oplcat_2020, oplcat_2021,
          # add the ID variable
          nomem_encr)
```

2.3.3.2 Step 2: Predictor Matrix Specification In this step, we constructed a predictor matrix using the `quickpred` function, automatically specifying variables to be predicted and those to be excluded. This function drastically speeds up the process of having to select predictors (e.g., having a partner) for the imputation model for the predictors (e.g., number of children) used for imputation of the target variables (i.e., net income). Through extensive exploration, we determined that setting mincor to 0.2 struck a balance, providing an optimal number of predictors. This choice avoids oversimplification and overcomplication of the model. Notably, the ID variable (nomem_encr) was intentionally excluded from both the predictor and predicted variables to maintain model integrity.

```
# make the predictor matrix (quickpred)
pred <- quickpred(allyears, mincor = 0.2)

# Also exclude nomem_encr (id variable) as predictor and to-be predicted
pred[,"nomem_encr"] <- 0
pred["nomem_encr",] <- 0
```

Subsequently, for each of the four years (2018-2021), we manually specified the predictors for the nettoink variable. Among these, geslacht (sex), leeftijd_2018 (Age at baseline) and brutoink (2018, 2019, 2020, 2021; gross income of each year) were used as predictors for each of the four years. In addition, for the net income of a specific year (e.g., 2018), we selected the following variables of that corresponding year: oplcat (education), aantalki (number of children), aantalhh (number of household members), burgstat_collapsed (urbanization), "woning" (type of housing), "belbezig_collapsed" (employment status). Last, we included the "nettoink" (net income) variables of the remaining years. Last, for the "brutoink" variable of a specific year (e.g., 2018), we removed the "brutoink" variables of the remaining years (e.g., 2019, 2020, 2021). This was done to avoid multicollinearity that would otherwise occur.

```
### First: 2018 data

# Variables to include as predictors for nettoink_2018
incl_pred1 <- c("oplcat_2018", "aantalki_2018", "aantalhh_2018", "geslacht",
                          "burgstat_collapsed_2018", "woning_2018",
                          "belbezig_collapsed_2018", "leeftijd_2018", "brutoink_2018",
                          "nettoink_2019", "nettoink_2020", "nettoink_2021",
                          "brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2020", "brutoink

# Set all values in the row to 0
pred["nettoink_2018", ] <- 0

# Set these variables to 1
for (i in incl_pred1) {
  pred["nettoink_2018",i] <- 1
}

# Variables to exclude as predictors for brutoink_2018
excl_pred1 <- c("brutoink_2019", "brutoink_2020", "brutoink_2021")
```

```r
# Set these variables to 0
for (i in excl_pred1) {
  pred["brutoink_2018",i] <- 0
}

### Second: 2019 data

# Variables to include as predictors for nettoink_2019
incl_pred2 <- c("oplcat_2019", "aantalki_2019", "aantalhh_2019", "geslacht",
                          "burgstat_collapsed_2019", "woning_2019",
                          "belbezig_collapsed_2019", "leeftijd_2018", "brutoink_2019",
                          "nettoink_2018", "nettoink_2020", "nettoink_2021",
                          "brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2020", "brutoink

# Set all values in the row to 0
pred["nettoink_2019", ] <- 0

# Set these variables to 1
for (i in incl_pred2) {
  pred["nettoink_2019",i] <- 1
}

# Variables to exclude as predictors for brutoink_2019
excl_pred2 <- c("brutoink_2018", "brutoink_2020", "brutoink_2021")

# Set these variables to 0
for (i in excl_pred2) {
  pred["brutoink_2019",i] <- 0
}

### Third: 2020 data

# Variables to include as predictors for nettoink_2020
incl_pred3 <- c("oplcat_2020", "aantalki_2020", "aantalhh_2020", "geslacht",
                          "burgstat_collapsed_2020", "woning_2020",
                          "belbezig_collapsed_2020", "leeftijd_2018", "brutoink_2020",
                          "nettoink_2018", "nettoink_2019", "nettoink_2021",
                          "brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2020", "brutoink

# Set all values in the row to 0
pred["nettoink_2020", ] <- 0

# Set these variables to 1
for (i in incl_pred3) {
  pred["nettoink_2020",i] <- 1
}

# Variables to exclude as predictors for brutoink_2020
excl_pred3 <- c("brutoink_2018", "brutoink_2019", "brutoink_2021")

# Set these variables to 0
for (i in excl_pred3) {
  pred["brutoink_2020",i] <- 0
```

```
}

### Fourth: 2021 data

# Variables to include as predictors for nettoink_2021
incl_pred4 <- c("oplcat_2021", "aantalki_2021", "aantalhh_2021", "geslacht",
                          "burgstat_collapsed_2021", "woning_2021",
                          "belbezig_collapsed_2021", "leeftijd_2018", "brutoink_2021",
                          "nettoink_2018", "nettoink_2019", "nettoink_2020",
                          "brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2020", "brutoink

# Set all values in the row to 0
pred["nettoink_2021", ] <- 0

# Set these variables to 1
for (i in incl_pred4) {
  pred["nettoink_2021",i] <- 1
}

# Variables to exclude as predictors for brutoink_2021
excl_pred4 <- c("brutoink_2018", "brutoink_2019", "brutoink_2020")

# Set these variables to 0
for (i in excl_pred4) {
  pred["brutoink_2021",i] <- 0
}
```

2.3.3.3 Step 4: Method Specification In this step, we systematically define imputation methods for different types of variables in our multiple imputation model. After exploring various estimation methods, we opted for the default methods in mice for each scale type (Buuren, 2018), as they demonstrated superior performance in our experimentation and fit the characteristics of our data very well. We started by automatically generating a set of imputation methods for the selected subset. We then manually adjusted these methods to ensure that they are appropriate for the variables in our dataset. First, for the binary variables (e.g., partner) we used logistic regression ("logreg"). Second, for the ordered categorical (i.e., ordinal) variables we utilized the proportional odds model ("polr"). Third, for the unordered categorical variables we employed polytomous logistic regression ("polyreg"). In the realm of multiple imputation, the challenge of *separation* poses a significant concern for categorical variables (Albert & Anderson, 1984). This issue can result in infinite estimates, leading to biased or unstable outcomes if not addressed adequately (Buuren, 2018). The mice package's three specified methods ("logreg", "polr", "polyreg") employ a solution to this technical problem by leveraging data augmentation. This approach mitigates the risk of infinite estimates by adding pseudo-observations to the data with a negligible weight. Furthermore, as the likelihood of encountering separation approaches zero with an expanding sample size, the presence of a substantial dataset should effectively mitigate any concerns related to this issue (Albert & Anderson, 1984). In addition, given that the number of events for each fitted parameter significantly surpasses the established threshold of 10, these methods are expected to contribute to a robust imputation process (Buuren, 2018) . Fourth, for the continuous variables we employed predictive mean matching ("pmm"), as this method tends to yield imputations that closely mirror the attributes of the observed data in large samples (Buuren, 2018). An additional advantage of predictive mean matching over alternative methods, such as Bayesian linear regression ("norm"), lies in its ability to generate realistic and plausible values. Utilizing a nearest neighbor hot-deck approach, predictive mean matching ensures that the imputed values align with the observed data distribution. In contrast, Bayesian linear regression may produce negative values for net income, which is both theoretically implausible and inconsistent with the survey response distribution. Last, predictive mean matching does not require the variables to be normally distributed, further contributing to the robustness by accommodating

various data distributions.

```
# Make methods automatically
meth <- make.method(allyears)

# Binary variables: (logreg)
# geslacht and partner have all observations
meth[c("partner_2018", "partner_2019", "partner_2020", "partner_2021"
       )] <- "logreg"
# Ordered categorical variables: (polr)
# oplcat, brutocat, nettocat
meth[c("oplcat_2018", "oplcat_2019", "oplcat_2020", "oplcat_2021"
       )] <- "polr"
# Unordered categorical variables: (polyreg)
# burgstat, woonvorm, woning, belbezig
meth[c("woning_2018", "woning_2019", "woning_2020", "woning_2021",
       "belbezig_collapsed_2018", "belbezig_collapsed_2019", "belbezig_collapsed_2020", "belbezig_colla
       "burgstat_collapsed_2018", "burgstat_collapsed_2019", "burgstat_collapsed_2020", "burgstat_colla
       "oplcat_2018", "oplcat_2019", "oplcat_2020", "oplcat_2021"
       )] <- "polyreg"
# Continuous variables: (pmm)
# leeftijd, brutoink, nettoink
meth[c("brutoink_2018", "brutoink_2019", "brutoink_2020", "brutoink_2021",
       "nettoink_2018", "nettoink_2019", "nettoink_2020", "nettoink_2021",
       "aantalki_2018", "aantalki_2019", "aantalki_2020", "aantalki_2021",
       "aantalhh_2018", "aantalhh_2019", "aantalhh_2020", "aantalhh_2021"
       )] <- "pmm"
```

2.3.3.4 Step 5: Perform Multiple Imputation We utilized the mice function to generate multiple plausible datasets with imputed values, providing a robust framework for subsequent analyses. During model building, we set the number of imputations (m) as well as the numer of iterations (maxit) to 5. In our final model, we followed the guidelines of van Buuren (2018) and set the m to 20. Further increasing this value would significantly extend computation time, a resource allocation that some argue may not yield worthwhile benefits (Schafer, 1997). Additionally, we set the number of iterations (maxit) to 5. Empirical findings suggest that a relatively low number of imputations is often sufficient to ensure the convergence of the imputation model, as the introduction of noise in the multiple imputation by chained equations (MICE) algorithm tends to accelerate the convergence process (Buuren, 2018). Last, a seed was established for reproducibility. Subsequently, we thoroughly examined the imputed data for errors and warnings, successfully resolving all issues in the final model.

```
# Impute the data (m = 20 imputations, seed = 123)
imp.allyears <- mice(allyears, pred = pred, meth = meth, seed = 123, maxit = 20, m = 5)
imp.allyears$loggedEvents # check for errors
```

2.3.3.5 Step 6: Inspect the Imputed Data We create plots to check convergence (traceplot), distribution (stripplot), box-and-whisker plots (bwplot), and density plots (densityplot) for the imputed variables across all years.

We created a traceplot to visually assess the convergence of the imputation process across all iterations. This plot can aid in determining whether the imputation model has reached a stable state, if there is convergence, which is said to be present when the different streams freely intermingle with one another, without definite trends (Buuren, 2018). The traceplots indicate the presence of trends in the tracelines of the means of the nettoink variables, especially in the earlier iterations. Namely, it could be observed that most streams run parallel and largely remain where they start of (e.g., low). At the final iterations there seems to be more convergence. This may suggest that increasing the number of iterations would improve the convergence.

Similarly, the standarddeviations of the nettoink variables are also not without issues: there seems to be substantial trends, which is most apparent in the 2020 variable.

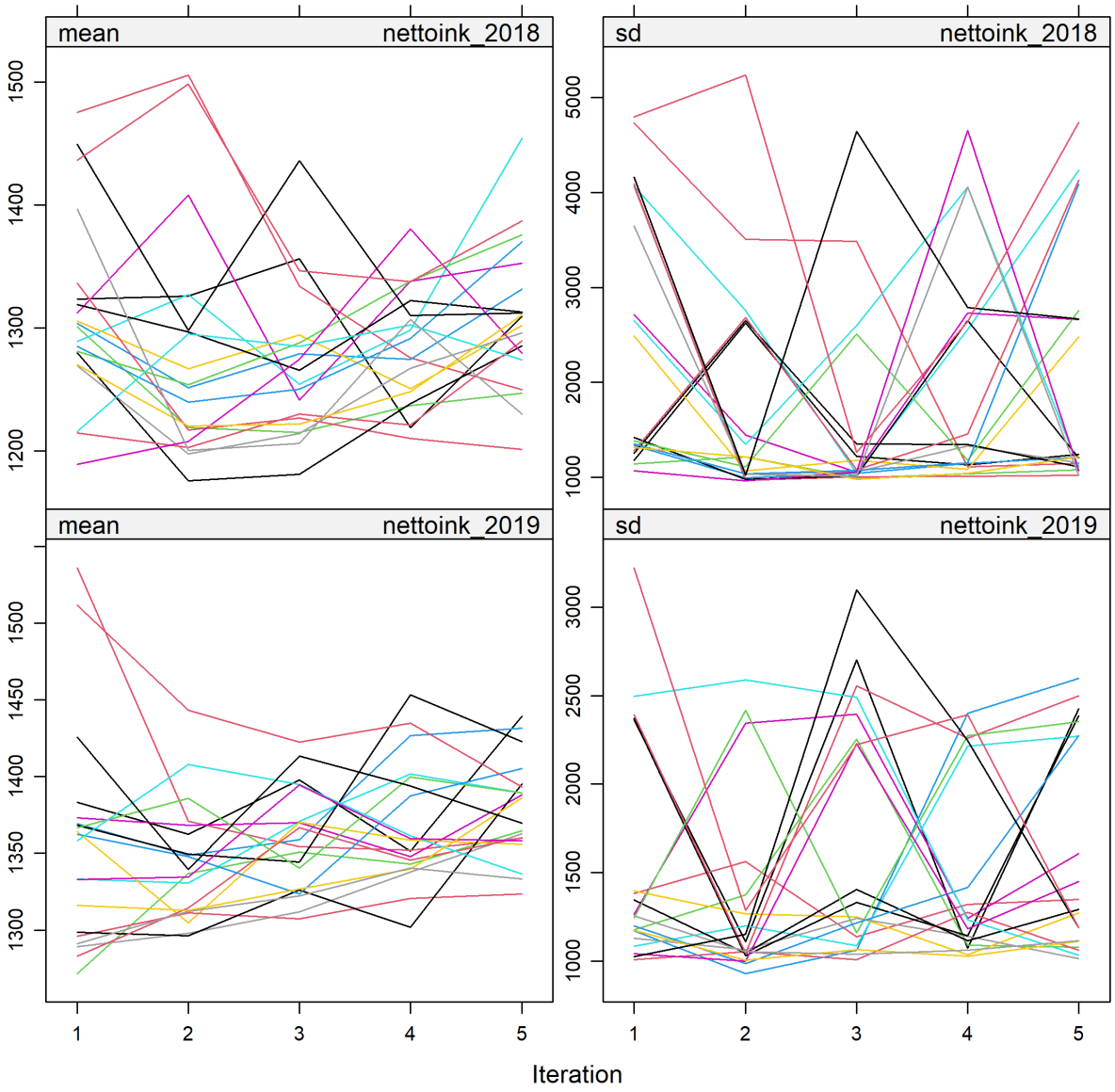


Figure 1: Traceplot

To evaluate the imputation quality, we generated Box-and-whisker plots and density plots to compare the distribution of observed and imputed values for net income across different years. The Box-and-whisker plots demonstrated that the distribution of the imputed values closely matches that of the imputed data, even including the outliers. This is desirable, since extreme values are also real data (Buuren, 2018).

Because of the outliers, however, it has become more challenging to see the distribution of the lower and also majority of values. Therefore, we created a second one to give more insight into the main part of the distribution.

The kernel density plot below indicates that the distribution of observed and imputed values are closely

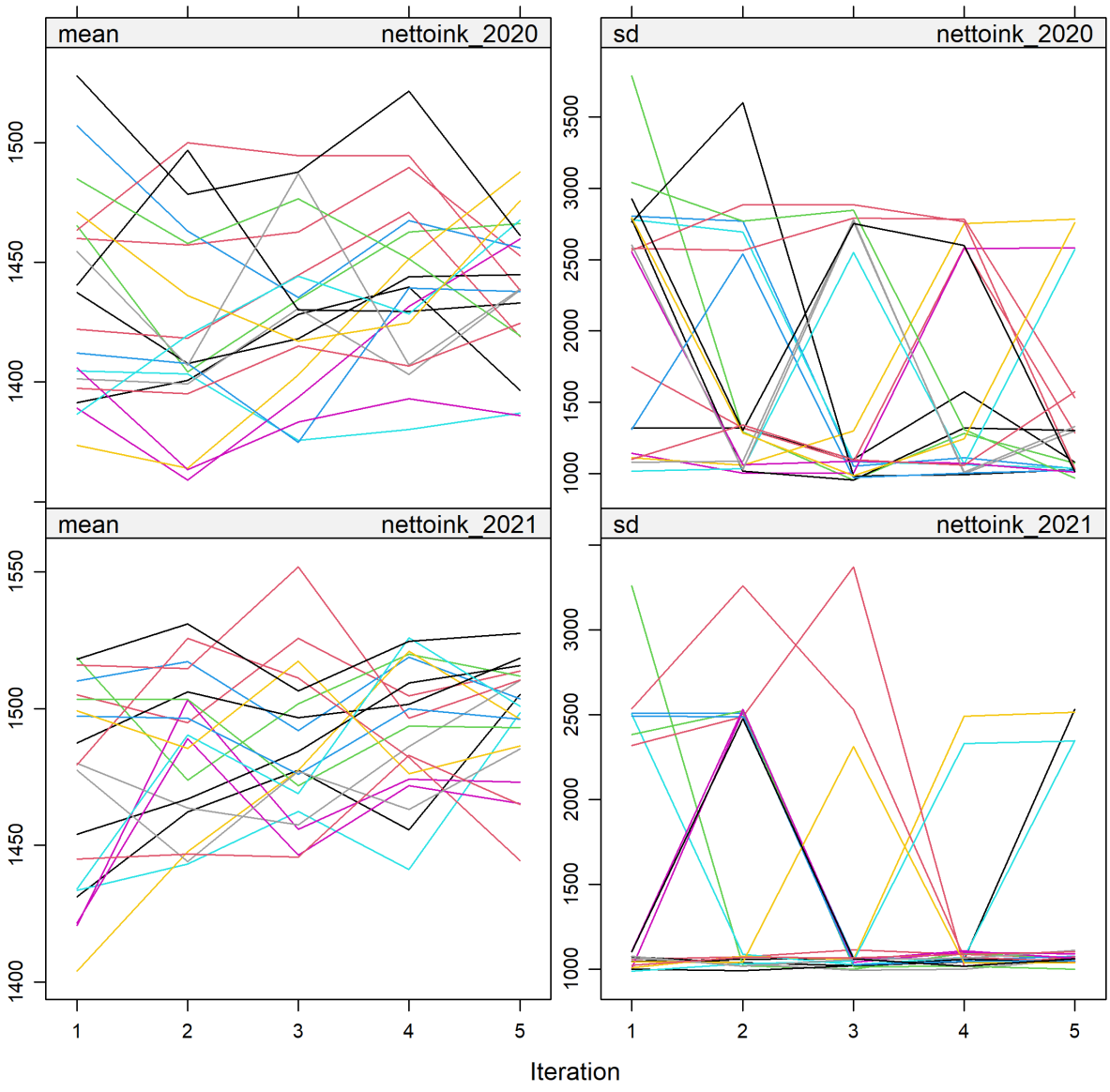


Figure 2: Traceplot

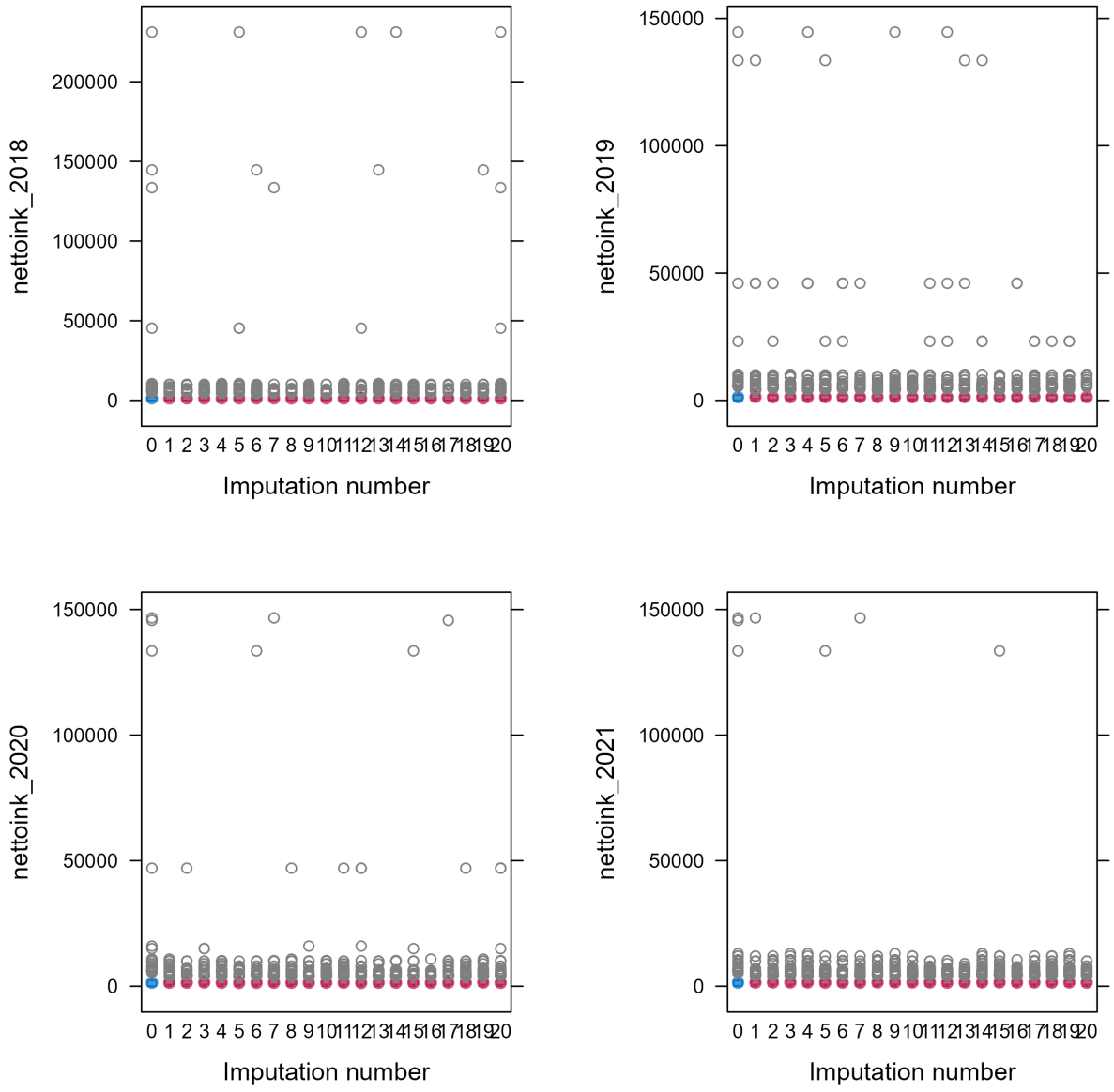


Figure 3: Netto Income Boxplot

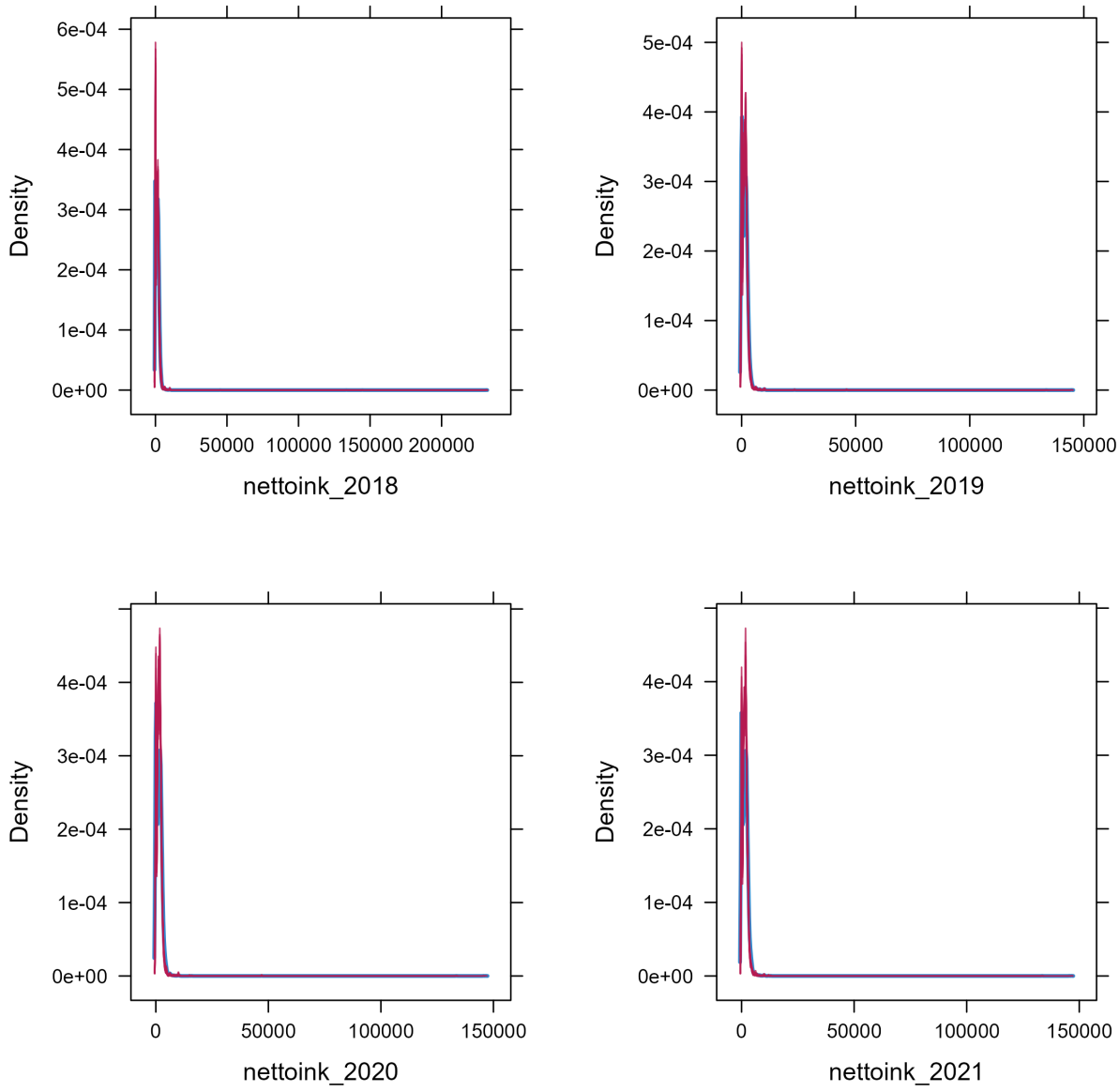


Figure 4: Netto Income Densityplot

alligned. In particular for the 2018 and 2019 variables. The discrepancy between the two distributions was greater for the variables associated with the years 2020 and 2021.

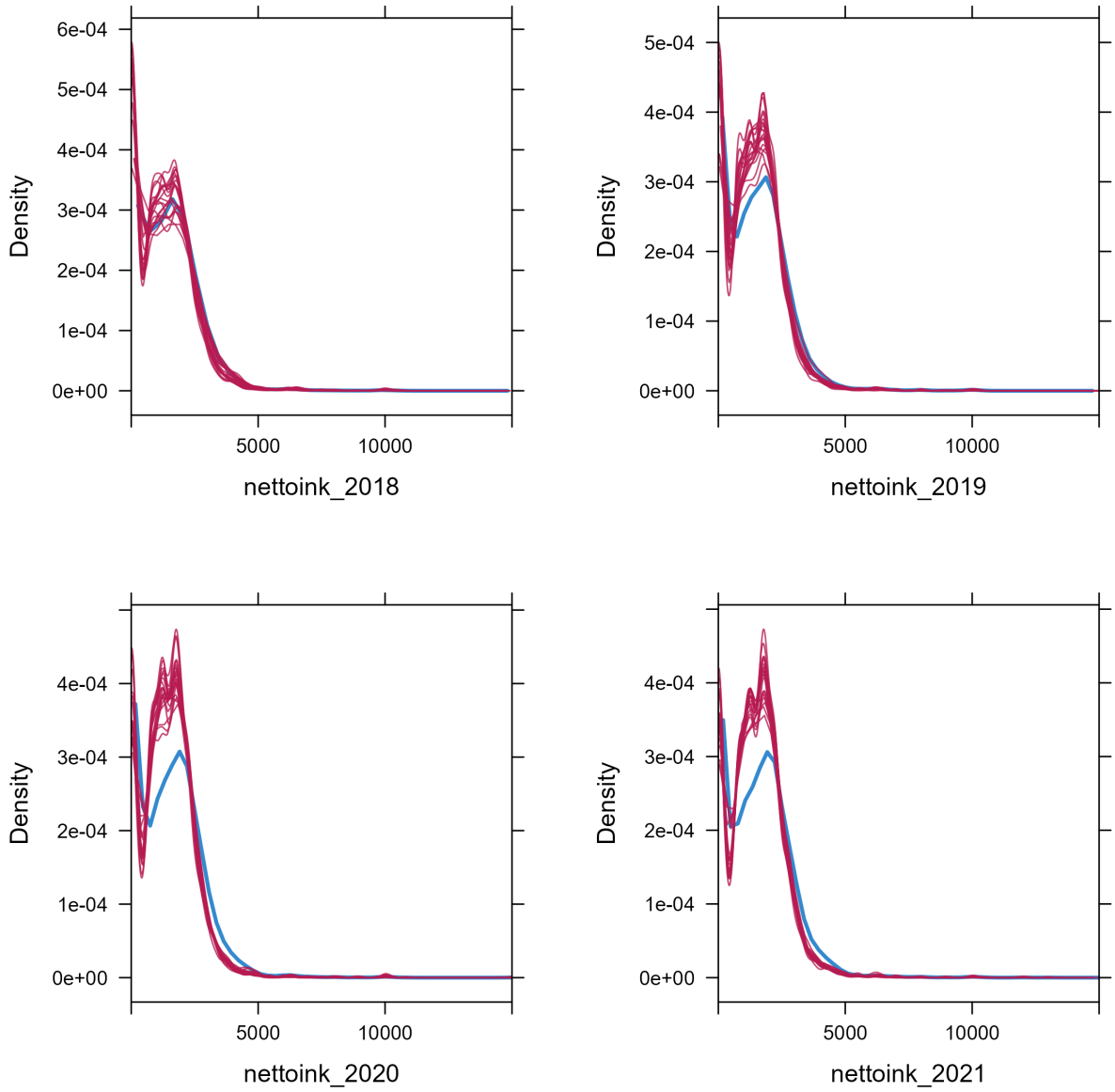


Figure 5: Netto Income Densityplot

3 Discussion

3.1 Comparing results

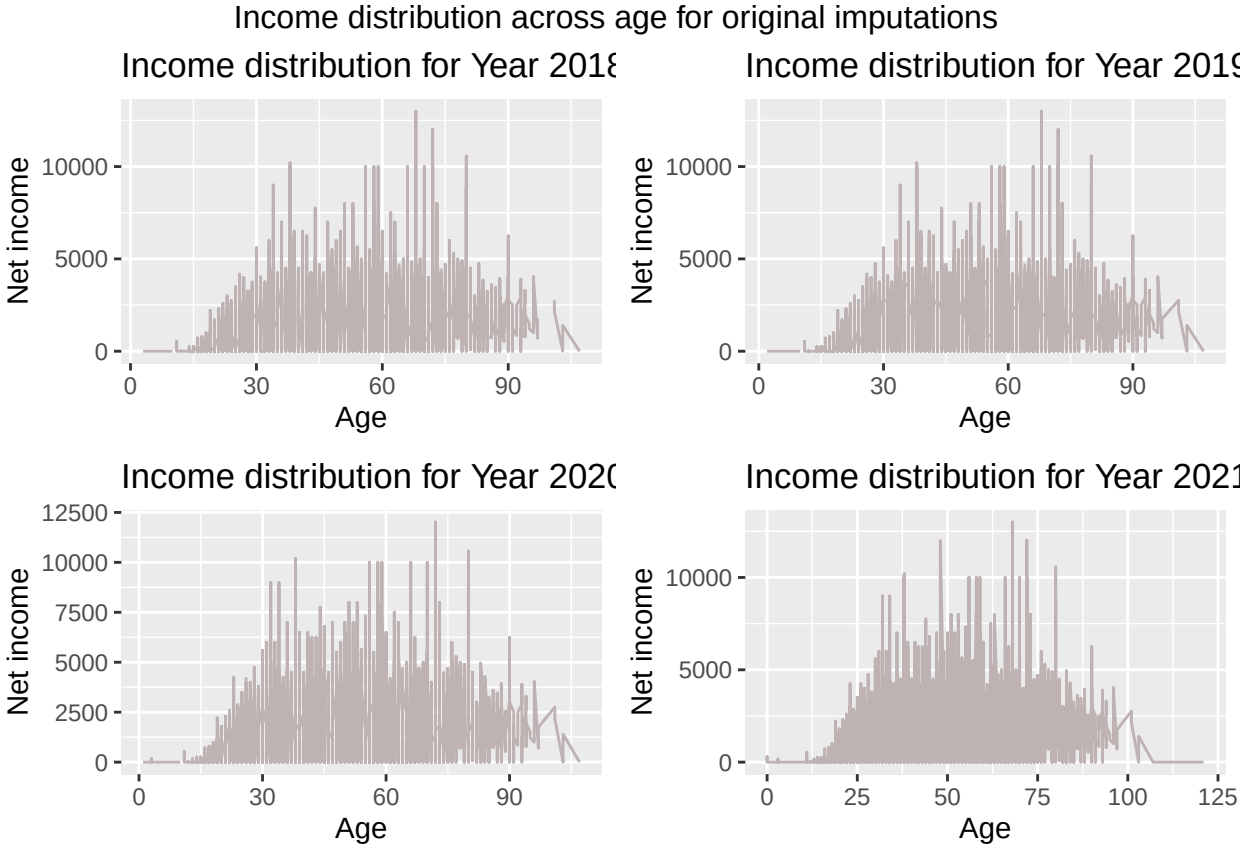
3.1.1 Median Household Income

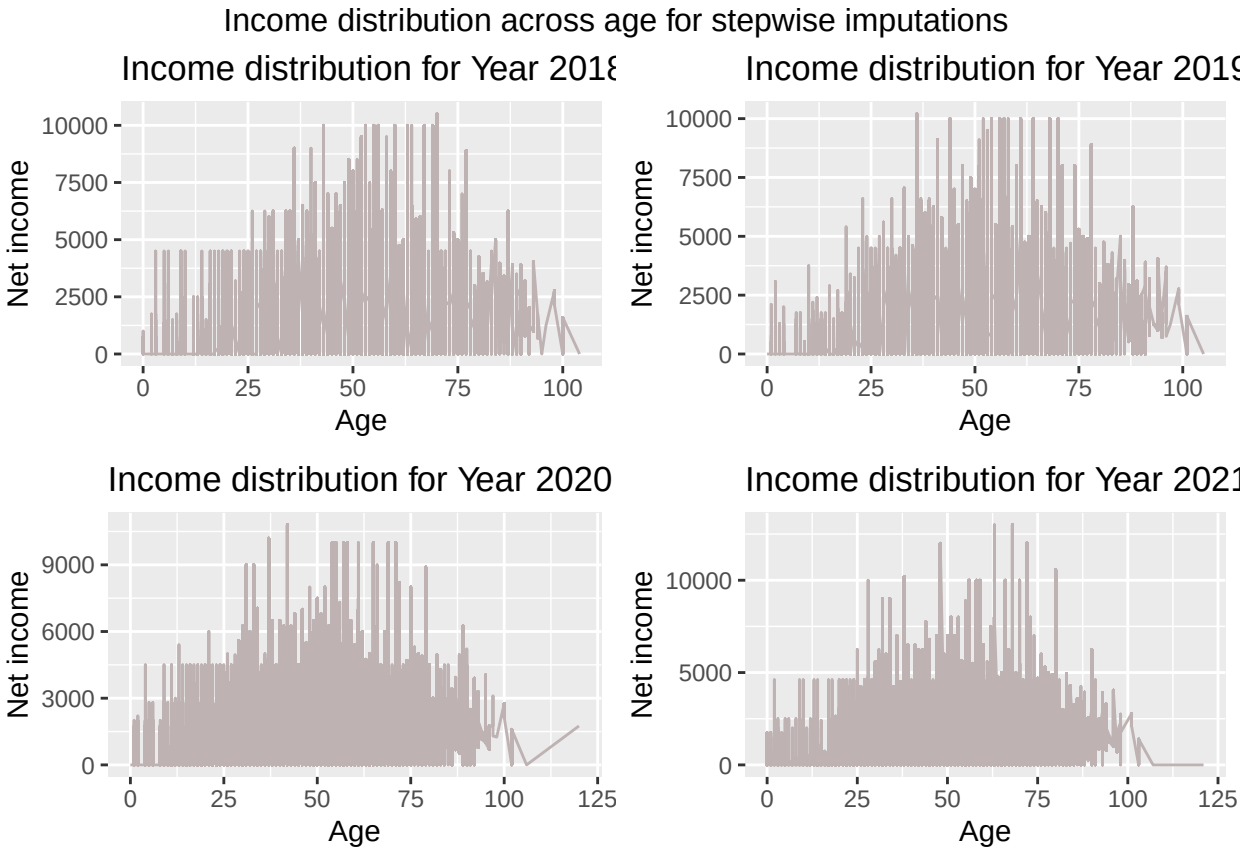
In order to compare the results of implementing the three different imputation methods on the LISS panel data, we first look at the median net income at each year after (a part of) the missing values have been imputed. The median net income after the stepwise imputation process is close to the median income from

the LISS imputation process for each year, with a slightly lower value in 2018 in 2021, a higher value in 2019, and the same value in 2020. It should be noted that the median income from the stepwise process was calculated by taking the mean of the different median incomes for each of the generated imputation models (m = 5). Interestingly, we saw previously that the Little & Su method in the second step of the stepwise approach resulted in a higher median income. This indicates that the final step in the stepwise approach, the stochastic regression, had a relatively large number of low or zero-value income imputations and thereby countered some of the overestimation of the Little and Su method. Ultimately, the stepwise imputation did not have a large effect on median income compared to the original LISS imputation procedure.

3.1.2 Household income distribution across age

The income distribution across age can be plotted for the single imputation LISS procedure and for the pooled stepwise multiple imputation method presented in this project. Only one of the two methods is presented in order to illustrate the resulting variance. This way, outliers both for age and net household income can be identified and trends amongst age groups can be easily analysed.





There is a clear increase in higher income values for younger and older participants in the stepwise imputation compared to the original LISS imputation. This is likely due to most missing values coming from these age groups but perhaps also because the variables used in the final step were not accurate predictors. This may however indicate that the stepwise imputations are improbable as the observed data shows less variation in income between participants of the same age, as one would expect at identical stages of their life. The plot also suggests an unlikely upward trend in household income in 2020 after the age of one hundred.

3.2 Limitations

It should be noted that the current investigation is associated with several limitations. Some of these have to do with questionable values and a lack of documentation concerning the dataset. For instance, it is not thoroughly explained why the authors chose to exclude values for net income that exceeded 15000 euros a month. If these are indeed considered invalid values, deletion makes sense. However, if these are true values, they should be included in the imputation model, which is what we opted for. Furthermore, during data exploration we discovered that there were several values of age under the supposed threshold of 18. However, it is unclear what this means (there is no elaboration in the codebook). While we at some point obtained a model that yielded reasonable imputations, we would have liked to do more testing and comparisons. One of the main issues here was the mere size of the dataset. The dataset is very large and the imputation process is very time-consuming (e.g., the multiple imputation model with m = 20 took approximately 3 hours to run). Not only this, but also the experience with crashing (R is terminated) after making plots made it difficult to test more scenarios and have the opportunity to compare different these scenarios. This problem indicates some of the limitations with applying the mice package to a dataset of this size. While we could simplify the predictor matrix and exclude variables to deal with a smaller mice.mids object, doing so would negatively affect the distribution of the imputed variables and the overall quality of the imputation model.

4 Conclusion

The two methods presented in this project, stepwise and multiple imputation, highlight the difficulties faced when trying to impute large amounts of missing data. Although they succeeded at imputing more values than were originally imputed by the LISS panel, the new values demonstrate a higher variance in the median and across age groups. The lengthy process of obtaining these values also provided evidence that the board's original opinion that imputing more values would be too time-consuming was justified. Furthermore, the benefits of FCS compared to joint modeling were unfortunately difficult to ascertain given the volume of the data.

Nevertheless, using longitudinal data was proven to be useful and easy to implement, allowing for a more flexible approach to imputation dependent on how many missing values were present for each participant. An extension of the stepwise approach presented here would be to separate the cases further depending on missing patterns, such as perhaps only using longitudinal data in cases where at least two years of data were available. Given the sheer amount of data, this approach was not feasible here in order to adhere to deadlines. The lack of information concerning the reasons behind the data being missing also made this problem harder and underlines the importance of providing the missing data mechanism in a survey's documentation to facilitate imputation.

Bibliography

Aaron, R., Caldwell, & aut. (2022). Exploring equivalence testing with the updated TOSTER r package. *PsyArXiv*. https://doi.org/10.31234/osf.io/ty8de

Albert, A., & Anderson, J. A. (1984). On the Existence of Maximum Likelihood Estimates in Logistic Regression Models. *Biometrika*, *71*(1), 1–10. https://doi.org/10.2307/2336390

Ammar, R. (2019). *randomcoloR: Generate attractive random colors*. https://CRAN.R-project.org/package=randomcoloR

Auguie, B. (2017). *gridExtra: Miscellaneous functions for "grid" graphics*. https://CRAN.R-project.org/package=gridExtra

Aust, F., & Barth, M. (2023). *papaja: Prepare reproducible APA journal articles with R Markdown*. https://github.com/crsh/papaja

Ballings, M., & Van den Poel, D. (2015). *Dummy: Automatic creation of dummies with support for predictive modeling*. https://CRAN.R-project.org/package=dummy

Barth, M. (2023). *tinylabels: Lightweight variable labels*. https://cran.r-project.org/package=tinylabels

Bates, D., Maechler, M., & Jagan, M. (2023). *Matrix: Sparse and dense matrix classes and methods*. https://CRAN.R-project.org/package=Matrix

Bodner, T. (2008). What improves with increased missing data imputations? *Structural Equation Modeling*, *15*, 651–675. https://doi.org/10.1080/10705510802339072

Bolker, B., & Robinson, D. (2022). *Broom.mixed: Tidying methods for mixed models*. https://CRAN.R-project.org/package=broom.mixed

Buuren, S. van. (2018). *Flexible Imputation of Missing Data, Second Edition* (2nd ed.). Chapman; Hall/CRC. https://doi.org/10.1201/9780429492259

Daniel, Lakens, & aut. (2017). Equivalence tests: A practical primer for t-tests, correlations, and meta-analyses. *Social Psychological and Personality Science*, *1*, 1–8. https://doi.org/10.1177/1948550617697177

de Clercq, M., D'Haese, M., & Buysse, J. (2023). Economic growth and broadband access: The european urban-rural digital divide. *Telecommunications Policy*, *47*(6), 102579. https://doi.org/https://doi.org/10.1016/j.telpol.2023.102579

Engels, J. M., & Diehr, P. (2003). Imputation of missing longitudinal data: A comparison of methods. *Journal of Clinical Epidemiology*, *56*(10), 968–976.

Ferro, M. A. (2014). Missing data in longitudinal studies: Cross-sectional multiple imputation provides similar estimates to full-information maximum likelihood. *Annals of Epidemiology*, *24*(1), 75–77. https://doi.org/https://doi.org/10.1016/j.annepidem.2013.10.007

Fortran code by Alan Miller, T. L. based on. (2020). *Leaps: Regression subset selection*. https://CRAN.R-project.org/package=leaps

Frick, J. R., & Grabka, M. M. (2014). *Missing income data in the german SOEP: Incidence, imputation and*

its impact on the income distribution. SOEP Survey Papers.

Friedman, J., Tibshirani, R., & Hastie, T. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Grolemund, G., & Wickham, H. (2011). Dates and times made easy with lubridate. *Journal of Statistical Software*, *40*(3), 1–25. https://www.jstatsoft.org/v40/i03/

Han, J., Meyer, B. D., & Sullivan, J. X. (2020). *Income and poverty in the COVID-19 pandemic* (Working Paper 27729; Working Paper Series). National Bureau of Economic Research. https://doi.org/10.3386/w27729

Hardt, J., Herke, M., & Leonhart, R. (2012). Auxiliary variables in multiple imputation in regression with missing x: A warning against including too many in small sample research. *BMC Medical Research Methodology*, *12*(1), 184–184.

Harrison, E., Drake, T., & Pius, R. (2023). *Finalfit: Quickly create elegant regression results tables and plots when modelling*. https://CRAN.R-project.org/package=finalfit

Henry, L., & Wickham, H. (2023). *Rlang: Functions for base types and core r and 'tidyverse' features*. https://CRAN.R-project.org/package=rlang

Hippel. (2020). How many imputations do you need? A two-stage calculation using a quadratic rule. *Sociological Methods & Research*, *49*(3), 699–718. https://doi.org/10.1177/0049124117747303

Huque, M. H., Carlin, J. B., Simpson, J. A., & Lee, K. J. (2018). A comparison of multiple imputation methods for missing data in longitudinal studies. *BMC Medical Research Methodology*, *18*(1), 1–16.

Knoef, M., & Vos, K. (2008). *The representativeness of LISS, an online probability panel*.

Kuhn, & Max. (2008). Building predictive models in r using the caret package. *Journal of Statistical Software*, *28*(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Lachin, J. M. (2016). Fallacies of last observation carried forward analyses. *Clinical Trials*, *13*(2), 161–168. https://doi.org/10.1177/1740774515602688

Lipps, O., & Kuhn, U. (2023). Income imputation in longitudinal surveys: A within-individual panel-regression approach. *Survey Research Methods*, *17*, 159–175.

Liu, X. (2016). Chapter 14 - methods for handling missing data. In X. Liu (Ed.), *Methods and applications of longitudinal data analysis* (pp. 441–473). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-801342-7.00014-9

Mullan, K., Daraganova, G., & Baker, K. (2015). *Imputing income in the longitudinal study of australian children*.

Müller, K., & Wickham, H. (2023). *Tibble: Simple data frames*. https://CRAN.R-project.org/package=tibble

Oberman, H. (2023). *Ggmice: Visualizations for 'mice' with 'ggplot2'*. https://CRAN.R-project.org/package=ggmice

R Core Team. (2022). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

R Core Team. (2023). *Foreign: Read data stored by 'minitab', 's', 'SAS', 'SPSS', 'stata', 'systat', 'weka', 'dBase', ...* https://CRAN.R-project.org/package=foreign

Revilla, M., & Saris, W. E. (2010). *A comparison of surveys using different modes of data collection: European social surveys versus LISS panel*.

Rich, B. (2023). *table1: Tables of descriptive statistics in HTML*. https://CRAN.R-project.org/package=table1

Sarkar, D. (2008). *Lattice: Multivariate data visualization with r*. Springer. http://lmdvr.r-forge.r-project.org

Schafer, J. L. (1997). *Analysis of Incomplete Multivariate Data*. Chapman; Hall/CRC. https://doi.org/10.1201/9780367803025

Scherpenzeel, A. (2011). Data collection in a probability-based internet panel: How the LISS panel was built and how it can be used. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, *109*(1), 56–61. https://doi.org/10.1177/0759106310387713

Scholtus, S., Laar, R. van de, & Willenborg, L. (2014). *The memobust handbook on methodology for modern business statistics*. Eurostat.

Simon, N., Friedman, J., Tibshirani, R., & Hastie, T. (2011). Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, *39*(5), 1–13. https://doi.org/10.18637/jss.v039.i05

Spiess, M., Kleinke, K., & Reinecke, J. (2021). Proper multiple imputation of clustered or panel data.

Advances in Longitudinal Survey Methodology, 424–446.

Tay, J. K., Narasimhan, B., & Hastie, T. (2023). Elastic net regularization paths for all generalized linear models. *Journal of Statistical Software*, *106*(1), 1–31. https://doi.org/10.18637/jss.v106.i01

Van Buuren, S. et al. (2011). Multiple imputation of multilevel data. *Handbook of Advanced Multilevel Analysis*, *10*, 173–196.

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, *45*(3), 1–67. https://doi.org/10.18637/jss.v045.i03

Watson, N., & Starick, R. (2011). Evaluation of alternative income imputation methods for a longitudinal survey. *Journal of Official Statistics*, *27*(4), 693.

Wei, T., & Simko, V. (2021). *R package 'corrplot': Visualization of a correlation matrix*. https://github.com/taiyun/corrplot

Westermeier, C., & Grabka, M. M. (2016). Longitudinal wealth data and multiple imputation: An evaluation study. *Survey Research Methods*, *10*, 237–252.

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H. (2022). *Stringr: Simple, consistent wrappers for common string operations*. https://CRAN.R-project.org/package=stringr

Wickham, H. (2023). *Forcats: Tools for working with categorical variables (factors)*. https://CRAN.R-project.org/package=forcats

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Wickham, H., François, R., Henry, L., Müller, K., & Vaughan, D. (2023). *Dplyr: A grammar of data manipulation*. https://CRAN.R-project.org/package=dplyr

Wickham, H., & Henry, L. (2023). *Purrr: Functional programming tools*. https://CRAN.R-project.org/package=purrr

Wickham, H., Hester, J., & Bryan, J. (2023). *Readr: Read rectangular text data*. https://CRAN.R-project.org/package=readr

Wickham, H., Vaughan, D., & Girlich, M. (2023). *Tidyr: Tidy messy data*. https://CRAN.R-project.org/package=tidyr

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.org/knitr/

Xie, Y. (2016). *Bookdown: Authoring books and technical documents with R markdown*. Chapman; Hall/CRC. https://bookdown.org/yihui/bookdown

Xie, Y., Allaire, J. J., & Grolemund, G. (2018). *R markdown: The definitive guide*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown

Xie, Y., Dervieux, C., & Riederer, E. (2020). *R markdown cookbook*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook

Yan, T., Curtin, R., & Jans, M. (2010). Trends in income nonresponse over two decades. *Journal of Official Statistics*, *26*, 145–164.