eatRep: a package to analyze multiple imputed data in complex survey designs

Sebastian Weirich
Humboldt University Berlin, Germany

March 4, 2014

Abstract

Estimation of simple descriptive statistics becomes cumbersome, if the sample cannot be considered to be a (completely) random draw from the population for which descriptives should be interpreted. This occurs in weighted samples or clustered samples. The same is true if the variables of interest stem from a multiple imputation process and occur, for example, as plausible values. This tutorial describes some basic analyses to compute descriptives in complex survey designs using the R package `eatRep`, which was designed mainly to supply replications methods in R. To date, only the Jackknife-2 (JK2) method is supported. Some functions overlap with methods provided in the computer software WesVar (Westat, 2000)—in this case the package only allows for executing these analyses in R, which may be easier to implement due to a syntax related interface. Some methods in WesVar are not implemented in `eatRep` yet, for example methods of balanced repeated replicates (BRR) or bootstrapping or even JK1. However, some methods are only implemented in `eatRep`, for example analyses for nested imputed data or linear logistic regression models.

`eatRep` heavily relies on the `survey` package (Lumley, 2012) which functions has been extended by methods for multiple imputed data. While the functional principle of `survey` is based on replication of conventional analyses, `eatRep` is based on replication of `survey` analyses to take multiple imputed data into account.

1 Introduction

In a completely random sample, the mean

$$\bar{x} = n^{-1} \sum_{i=1}^{n} (x_i) \tag{1}$$

is an unbiased estimate for the corresponding mean

$$\mu = N^{-1} \sum_{i=1}^{N} (x_i) \tag{2}$$

of the underlying population the sample was drawn from. This does not hold for dispersion measures (variance and standard deviation), as the variance in a sample is always less than the variance in the population the sample was drawn from. The transformation, however, is very easy made: The variance in a sample is multiplied by $n/(n-1)$ to obtain population variance, where n is the sample size. Based on

$$\sigma^2 = N^{-1} \sum_{i=1}^{N} (x_i - \mu)^2 \tag{3}$$

for the population with N elements, we apply

$$s^2 = (n-1)^{-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{4}$$

to estimate population variance from a sample of size n. In a weighted sample, i.e. if the population weights differ between examinees in the sample, mean and variance may be estimated by incorporating these population weights. (In a completely random sample, these weights equal 1 for each examinee.)

$$\bar{x}_w = \sum_{i=1}^{n} \left(\frac{w_i}{W} x_i \right), \tag{5}$$

$$s_w^2 = \sum_{i=1}^{n} \frac{w_i}{W-1} (x_i - \bar{x})^2, \tag{6}$$

where w_i is the case weight of the ith person, and W is the sum over all case weights, i.e. $W = \sum w_i$. To summary, the crucial point in the estimation

of population variance estimates is the factor $n/(n-1)$. Unfortunately, this factor only applies when we sample (conditionally) independently from the population, as in completely random samples or weighted random samples. In a clustered sample, however, where schools or classes are sampling units instead of single persons, the relationship between sample and population variance is not so clear at all. The reason is that persons *within* a cluster (for example pupils in a class) often share a common variance. The sample variance underestimates the population variance, but more severely than indicated by the factor $n/(n-1)$. To estimate the relationship between sample and population variance, it is necessary to estimate the variance explained by the cluster.

Without taking the cluster structure into account, we would not only obtain biased variance estimates but biased standard errors, too (Luke, 2009). This problem occurs in the same way for estimation of frequency tables, quantiles or estimates of (linear) regression models. To gain unbiased estimates, several replication methods were introduced, which based on the same principle: To estimate the proportion by which the variance in the sample is underestimated due to a clustered structure (Lumley, 2004). In the Jackknife-2 (JK2) procedure which is the only procedure implemented in `eatRep` to date, this is implemented by reproducing the original sample to several replicates. In each replicate one sampling unit (e.g. one class) is replaced by another class, which therefore occurs two times in the sample. Each replicate is analyzed if it would have been a completely random sample. Recognize what is to be expected then: If the variance is explained partially by the clusters, removing one sampling unit should decrease the variance of the sample slighty. Conversely, the point estimates of each replication sample should vary slightly. The variance in the point estimates between the replicates is used to estimate unbiased parameters. Otherwise, if there is no variance between clusters, removing one cluster would have no effect on the variance estimate, and the point estimates between replicates would have no variance. In this case replication methods will result in exactly the same variance estimates and standard errors as they would follow from conventional analysis.

For the purpose of illustration, assume a simple population mean which has to be estimated from a completely random sample of $N = 1000$. To estimate the standard error of this mean, we may apply a rather laborious

method: to draw 100 samples (with replacement) from our original sample, each of $N = 1000$, and compute the mean in each sample. The standard deviation of the 100 means is the standard error of the mean estimate. Of course, this bootstrap method is far to cumbersome, as in a random sample the standard error can be estimated in a much more easier way. However, in a clustered sample, an extension of this bootstrap method is appropriate indeed. Several software (Westat, 2000) and free R packages such as `survey` (Lumley, 2012) do allow for several replication methods.

The situation is becoming still more complicated when the variables in the data to be analysed occur as (multiple) imputed data, for example as plausible values. Where missing values may cause biased parameters, analyses are conducted with imputed data. Often, the original data which includes missing values is reproduced several times, whereas the missing entries are filled with a set of plausible values, which results in several imputed data sets. To gain unbiased parameter estimates, the analyses are conducted for each data set separately and pooled afterwards according to Rubin (1987).

If we have both, a clustered sample with multiple imputed data, both methods have to be combined. This leads to a replication of replications. Analyses have to be repeated to account for the clustered structure, and the results of these replications have to be repeated to account for multiple imputed data. In the following, we refer to "cluster replicates" and "imputation replicates" to differentiate between both.

2 Estimate some population descriptives

In this example, we use some artificial data from the context of educational research. We may think of a stratified clustered sample of German fourth-grade primary school students whose reading and writing competencies are measured. Proficiency estimates obtained from a Item response Theory (IRT) marginal model are included as plausible values. Each plausible value may be recognized as an imputation of the latent competence construct.

```
> str(reading_writing)
'data.frame':        4619 obs. of  25 variables:
 $ idstud         : chr  "LandA01010401" "LandA01010402" "LandA01010403" "LandA01010404" ...
 $ wgtSTUD        : num  60 60 60 60 60 ...
```

```
$ sex            : Factor w/ 2 levels "female","male": 1 2 1 1 2 2 1 1 2 2 ...
$ country        : Factor w/ 3 levels "LandA","LandB",..: 1 1 1 1 1 1 1 1 1 1 ...
$ JKZone         : num  40 40 40 40 40 40 40 40 40 40 ...
$ JKrep          : num  0 0 0 0 0 0 0 0 0 0 ...
$ reading_score1 : num  631 614 549 663 522 ...
$ reading_score2 : num  708 588 586 618 534 ...
$ reading_score3 : num  758 613 579 613 443 ...
$ writing_score1 : num  636 618 587 682 567 ...
$ writing_score2 : num  707 579 557 568 593 ...
$ writing_score3 : num  672 612 642 493 522 ...
$ passed_reading1: num  1 1 1 1 1 1 0 0 1 1 ...
$ passed_reading2: num  1 1 1 1 0 1 1 0 1 1 ...
$ passed_reading3: num  1 1 1 1 0 1 1 0 1 1 ...
$ passed_writing1: num  1 1 1 1 1 1 0 1 1 1 ...
$ passed_writing2: num  1 1 1 1 1 1 1 0 1 1 ...
$ passed_writing3: num  1 1 1 0 0 1 1 1 1 0 ...
$ zehisei1       : num  5 3 3 3 4 3 5 1 4 3 ...
$ zehisei2       : num  5 3 3 3 4 3 5 3 4 4 ...
$ zehisei3       : num  5 3 3 3 4 3 5 3 4 3 ...
$ zehisei4       : num  5 3 3 3 4 3 5 4 4 4 ...
$ zehisei5       : num  5 3 3 3 4 3 5 2 4 3 ...
$ income1        : num  2154 1987 2109 2067 2039 ...
$ income2        : num  2136 1863 1994 2023 2147 ...
```

Requesting the data structure provides us with information about the number and type of variables and the number of examinees. `"idstud"` is a unique person identifier of the 4,619 examinees, `"wgtSTUD"` a person weight, `"country"` denotes the country the person comes from. `"JKZone"` and `"JKrep"` denote jackknifing variables which contains information about which unit has to be replaced by which other unit in which replicate of the original data. We may think of `"reading_score1"`, `"reading_score2"` and `"reading_score3"` as three plausible values for the reading competence. Please note that the three imputations of the reading competence occur as three different variables whereas they conceptually belong to one common competence measure. This is quite usual if multiple imputed data is presented in a wide-format dataset. `eatRep` strictly requires the wide format which becomes apparent when dealing with nested multiple imputations. Please note further that the dataset does not contain any replicates, only the information required for generating them, captured in the `"JKZone"` and `"JKrep"` variables.

2.1 Populations means, standard deviations, variances and mean differences

We now want to compute the means by each country, considering the clustered structure as well as the multiple imputed data structure. The replicates

do not need to be created separately, as they will be generated in each analysis automatically. Even in large data sets this takes only a few seconds.

```
> means <- jk2.mean(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+          JKrep = "JKrep", groups = list(federalState = "country"),
+          dependent = list(reading = c("reading_score1", "reading_score2", "reading_score3")))
Create 81 replicate weights.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replication(s) overall.
...
Pooling Standard errors. Assume no nested structure.
```

One may ask whether the `groups` argument has to be specified in this laborious list format. The answer is: If the grouping variable is multiple imputed as well, its imputations have to be specified in a character vector of corresponding variable names as well as in the `dependent` argument. A statement like `groups = list(federalState = c("country1", "country2"))` then would refer to two imputations of the group variable.

While the function is operating, some additional information is displayed on console. First we see that 81 replicate weights are created due to 81 distinct jackknifing zones in the JKZone variable. This information refers to the "cluster replicates" and implies that the subsequent analysis has to be repeated 81 times *for each imputation*. In each of the 81 replication samples, one unit (e.g. school) of a certain jackknifing zone is missing and the weights of the other unit of the same zone are doubled. The data in all other zones remain unchanged. Each analysis revealed slighty different results. This variation is used to estimate the sampling variance. But why it is talking about 3 replications overall? This refers to the "imputation replicates", because we have defined three plausible values of the dependent variable. Whereas we only change the weights in the "cluster replicates" and work with identical variables, we change the variables (e.g. the imputations) between the "imputation replicates". To sum up, for each of the three imputations, 81 analyses are executed, which results in $3 \times 81 = 243$ analyses overall. The little dots continuously appearing on the console therefore refer to "imputation replicates" and are intended to work as a rough progress bar. Each dot represents one replication. When the procedure finished, the results are pooled in the case of more than one imputation.

```
> means[c(1:4,19:20),]
```

```
   group  depVar        modus parameter coefficient        value federalState
1  LandA reading jk2.weighted    Ncases         est 1.187318e+05        LandA
2  LandA reading jk2.weighted    Ncases          se 1.726642e+03        LandA
3  LandB reading jk2.weighted    Ncases         est 5.398038e+04        LandB
4  LandB reading jk2.weighted    Ncases          se 1.135124e+03        LandB
19 LandA reading jk2.weighted        sd         est 1.041336e+02        LandA
20 LandA reading jk2.weighted        sd          se 2.071056e+00        LandA
```

The output is a data frame in the long format with 30 rows. To keep the overview, only a few selected rows are displayed here. For each subpopulation denoted by the `groups` statement (here: `LandA`, `LandB` and `LandC`), each dependent variable (here: only the reading competence), each parameter (we requested mean, variance, standard deviation and sample size or population size) and each coefficient (i.e., the estimate and the corresponding standard error) the corresponding value is given. To display the results in the more common wide format, use the `reshape2` package. A possible call would be `reshape2::dcast(means, group ~ parameter+coefficient, value.var = "value")`. Alternatively, an abbreviated display of the results is provided by the `dM` function (whereas dM stands for "display means"). You may think of `dM` as a simple summary function which is not intended for saving results or further processing as the results are displayed in an abbreviated (i.e., rounded) manner to offer clear arrangement on console. The `dM` function has an additional argument to omit displaying parameters or coefficients you are not interested at the moment.

```
> dM(means, omitTerms = c("var", "Ncases","NcasesValid", "meanGroupDiff") )
   depVar federalState mean_est mean_se  sd_est sd_se
1 reading        LandA  515.749   5.320 104.134 2.071
5 reading        LandB  492.773   5.627 103.384 4.199
9 reading        LandC  511.811   4.030 103.330 3.080
```

Is it possible to see how the results would change if we do not consider the clustered structure? Sure is is! Simply leave out the jackknifing arguments `JKZone` and `JKrep`. The results will be pooled only due to multiple imputed data:

```
> means <- jk2.mean(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD",
+       groups = list(federalState = "country"),
+           dependent = list(reading = c("reading_score1", "reading_score2", "reading_score3")))
No jacknifing variables. Assume no cluster structure.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replication(s) overall.

Pooling Standard errors. Assume no nested structure.
```

```
> dM(means, omitTerms = c("var", "Ncases","NcasesValid", "meanGroupDiff") )
   depVar federalState mean_est mean_se  sd_est sd_se
1 reading        LandA  515.749   2.665 101.224    NA
5 reading        LandB  492.773   3.218 101.277    NA
9 reading        LandC  511.811   2.745 100.405    NA
```

We see that the means are completely unaffected, but the standard deviation now is lower. Consequently, also the standard errors for the mean estimates are considerably lower. (Standard errors for standard deviations and for variances are not implemented yet.) If we decide to leave out the weights as well, we would additionally expect to receive different means now:

```
> means <- jk2.mean(dat = reading_writing, ID = "idstud", groups = list(federalState = "country"),
+           dependent = list(reading = c("reading_score1", "reading_score2", "reading_score3")))
No jacknifing variables. Assume no cluster structure.
No weights specified. Use weight of 1 for each case.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replication(s) overall.

Pooling Standard errors. Assume no nested structure.
> dM(means, omitTerms = c("var", "Ncases","NcasesValid", "meanGroupDiff") )
   depVar federalState mean_est mean_se  sd_est sd_se
1 reading        LandA  518.783   2.810 101.224    NA
5 reading        LandB  493.894   2.940 101.277    NA
9 reading        LandC  514.113   2.775 100.405    NA
```

Two possible interesting features should be emphasized in the following. First assume that we do not have one, but two grouping variables, namely `federalState` and `gender`. As we have three federal states and two gender values, the whole population is splitted into $3 \times 2 = 6$ subpopulations for which descritives can be requested. If we additionally are interested in the descriptives of the whole population or the descriptives *within each state, but together for both gender groups*, we can use the `group.splits` argument to particularly specify the groups we are interested in. Let us consider for example the two group variables `federalState` and `gender`. If `group.splits` equals 2 (the default, i.e., the number of grouping variables), descriptives for the $3 \times 2 = 6$ subpopulations are computed. If `group.splits` is `1:2`, descriptives for each state (e.g., across gender) and each gender group (e.g. across federal states) additionally are computed. If `group.splits` is `0:2`, descriptives also for the whole population (e.g. across gender *and* federal states) are computed.

The second feature is about mean differences. Suppose you are interested in the gender difference *within* each federal state. The grouping variable for which mean differences should be computed has to be specified in the `group.differences.by` argument. For a grouping variable with K levels, all $K!/(2! * (K-2)!)$ comparisons are computed. It is important that the group defined in `group.differences.by` also has to occur in the `groups` statement, otherwise `group.differences.by` will be ignored. To estimate gender differences *within each federal state*, gender and federal state have to be part of the `groups` statement, whereas only gender has to be used in the `group.differences.by` argument. Both features are illustrated in the following example:

```
> means <- jk2.mean(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+           JKrep = "JKrep", groups = list(federalState = "country", gender = "sex"),
+           group.splits = c(0,2), group.differences.by = "gender",
+            dependent = list(reading = c("reading_score1", "reading_score2", "reading_score3")))
Create 81 replicate weights.
No group(s) specified. Analyses will be computed only for the whole sample.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replication(s) overall.
...
Pooling Standard errors. Assume no nested structure.
Create 81 replicate weights.
Found 2 grouping variable(s).
Run 1 analyses overall.
Use 3 replication(s) overall.
...
Pooling Standard errors. Assume no nested structure.
```

First note the `group.splits` is set to `c(0,2)`, which means that we request descriptives for the whole population and the 6 subpopulations. Consequently, two analyses are conducted. The `group.differences.by` only applies for the second analysis, as the gender group is not considered relating to the whole population analysis. Please note that you have to use the name of the group variable, not the group variable itself: `group.differences.by = "gender"` instead of `group.differences.by = "sex"`! To estimate gender differences *across all federal states*, only gender has to be part of the `group` statement, and only gender has to be used in the `group.differences.by` argument. The output of the analysis is nearly the same as we would have omitted the `group.differences.by` argument, but now, some additional lines have joined. Again, we may use the `dM` function to display the part of the results we are interested in—note that now we do *not* exclude `meanGroupDiff`

from the results to summarize:

```
> dM(means, omitTerms = c("var","Ncases", "NcasesValid"))
    depVar                                    group mean_est mean_se
1  reading                           LandA_female  534.061   6.030
5  reading                             LandA_male  499.532   6.799
9  reading                           LandB_female  500.607   6.071
13 reading                             LandB_male  485.300   6.820
17 reading                           LandC_female  521.785   4.849
21 reading                             LandC_male  502.114   5.623
25 reading federalState=LandA____female.vs.male       NA      NA
27 reading federalState=LandB____female.vs.male       NA      NA
29 reading federalState=LandC____female.vs.male       NA      NA
31 reading                             wholeGroup  508.857   3.675
   meanGroupDiff_est meanGroupDiff_se  sd_est sd_se
1                 NA               NA  99.255 3.744
5                 NA               NA 105.682 3.866
9                 NA               NA  98.768 4.378
13                NA               NA 107.116 5.420
17                NA               NA 100.097 4.166
21                NA               NA 105.533 3.596
25           -34.529            7.232      NA    NA
27           -15.307            6.318      NA    NA
29           -19.671            6.872      NA    NA
31                NA               NA 104.336 1.702
```

The output now changed slightly: Instead of several group columns, only one column for group membership is provided. The last line labelled `wholeGroup` provides results concerning the whole population. The line labelled `LandC_male` contains values for the males in LandC. Moreover, three mean differences were computed. In each federal state, the difference between males and females is given.

2.2 Frequency tables

Computation of frequency tables works in the same manner as in the examples mentioned before. Representative for several possible analyses only one example is given below. First of all, let's have a look at the backmost columns in our example data:

```
> reading_writing[1:5,19:23]
      zehisei1 zehisei2 zehisei3 zehisei4 zehisei5
18916        5        5        5        5        5
18917        3        3        3        3        3
18918        3        3        3        3        3
18919        3        3        3        3        3
18920        4        4        4        4        4
```

The Hisei variables "zehisei1", "zehisei2", "zehisei3", "zehisei4" and "zehisei5" indicates high versus low socio-economical status which is clustered in five groups. Hence, each variable consists of five distinct values. Categorical variables are often represented as factors in R, which is quite straightforward. However, the "zehisei" variables are of class numeric. This is an inconsistency which may cause annoying misinterpretations when such variables are called in functions related to the generalized linear model like aov(), glm() etc. For the computations of frequency tables it is not necessary to convert the variable class to factor.

We now are interested in the relative frequencies of this groups in the different countries and within each country for different groups of gender. As before, we want to take the cluster structure and multiple imputations into account.

```
> freqs <- jk2.table(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD",
+       JKZone = "JKZone", JKrep = "JKrep",
+           groups = list(federalState = "country", gender = "sex"),
+           dependent = list(Hisei = paste("zehisei",1:5,sep="")))
Create 81 replicate weights.
Found 2 grouping variable(s).
Run 1 analyses overall.
Use 5 replication(s) overall.
.....
Pooling Standard errors. Assume no nested structure.
> dT(freqs)
   depVar federalState gender 1_est  1_se 2_est  2_se 3_est  3_se 4_est  4_se
1   Hisei        LandA female 0.042 0.011 0.258 0.023 0.359 0.019 0.263 0.021
11  Hisei        LandA   male 0.037 0.008 0.279 0.027 0.374 0.020 0.240 0.020
21  Hisei        LandB female 0.061 0.011 0.270 0.019 0.359 0.021 0.251 0.019
31  Hisei        LandB   male 0.047 0.012 0.290 0.022 0.365 0.023 0.224 0.021
41  Hisei        LandC female 0.048 0.010 0.367 0.018 0.398 0.021 0.158 0.014
51  Hisei        LandC   male 0.055 0.011 0.349 0.023 0.369 0.020 0.165 0.017
   5_est  5_se
1  0.077 0.011
11 0.070 0.013
21 0.058 0.011
31 0.075 0.013
41 0.028 0.007
51 0.062 0.010
```

The output is a single data frame in the long format. To make the output more pleasing to the eye, a short summary function dT is just waiting to do her job, to summarize the results. For each gender group in each country, the relative frequency of each Hisei category is given with its standard error. The first column refers to the dependent variable for which we want to

compute frequencies. The next two columns refer to the groups specified in the analysis (in our example: `federalState` and `gender`). The "labels" of the dependent variable now are captured in the column names of the summary table. We see that in federal state `"LandA"` the first Hisei category labelled `"1"` has a relative frequency of about 4.2 percent in the female group and 3.7 percent in the male group.

As in the examples mentioned before, these analyses may be conducted without considering clustered structure. See whether the standard errors will change. Technically, this function even works when the results are theoretically questionable. First, let's imagine we have no "5" – values in the first and second imputation of the Hisei:

```
> reading_writing2 <- reading_writing
> reading_writing2[which(reading_writing2[,"zehisei1"] == 5),"zehisei1"] <- 4
> reading_writing2[which(reading_writing2[,"zehisei2"] == 5),"zehisei2"] <- 4
```

Here, we replaced the "5" values by "4" values. Moreover, we assume missings on the Hisei imputations 2, 3 and 4. Missings on an imputed variable? Yes, for example, when we want to estimate the proportion of missingness. (Conceptually, however, it makes more sense to define the categories by certain categories, for example `c("hisei_group1", "hisei_group2", "hisei_group3", "hisei_group4", "hisei_group5", "no_anwer")`.)

```
> cols <- paste("zehisei", 2:4, sep = "")
> for (i in cols) {
+      casesToNA <- sample(x = c(1:nrow(reading_writing2)), size = 12, replace = FALSE )
+      reading_writing2[ casesToNA ,i ] <- NA
+ }
```

For each of the Hisei imputations 2, 3 and 4, we choose 12 random cases to insert a missing value. Will the function have to surrender when the data structure is of this kind?

```
> freqs2 <- jk2.table(dat = reading_writing2, ID = "idstud", wgt = "wgtSTUD",
+       separate.missing.indikator = TRUE, JKZone = "JKZone", JKrep = "JKrep",
+           groups = list(federalState = "country", gender = "sex"),
+           dependent = list(Hisei = paste("zehisei",1:5,sep="")))
Create 81 replicate weights.
Found 2 grouping variable(s).
Run 1 analyses overall.
Use 5 replication(s) overall.
.....
Pooling Standard errors. Assume no nested structure.
```

```
> dT(freqs2)
   depVar federalState gender 1_est  1_se 2_est  2_se 3_est  3_se 4_est  4_se
1   Hisei        LandA female 0.042 0.011 0.258 0.023 0.359 0.019 0.292 0.051
13  Hisei        LandA   male 0.037 0.008 0.279 0.027 0.373 0.020 0.267 0.051
25  Hisei        LandB female 0.061 0.011 0.270 0.019 0.359 0.021 0.274 0.041
37  Hisei        LandB   male 0.047 0.012 0.289 0.021 0.364 0.022 0.252 0.048
49  Hisei        LandC female 0.048 0.010 0.367 0.018 0.398 0.021 0.169 0.026
61  Hisei        LandC   male 0.055 0.011 0.349 0.023 0.368 0.020 0.190 0.040
   5_est  5_se missing_est missing_se
1  0.047 0.047       0.003      0.004
13 0.042 0.044       0.002      0.002
25 0.035 0.036       0.001      0.001
37 0.046 0.047       0.001      0.002
49 0.017 0.018       0.002      0.003
61 0.037 0.038       0.001      0.002
```

Without bothering you unduly with the output, I only want to mention two details: First a new category has joined to the output, which is labelled `"missing"`. Secondly, remember what we have done with our data disturbance. We replaced the "5" values from the first and second imputation. Now, as group "5" no longer appears in both imputations, the distribution of the categories now vary more widely between imputations. In the concept of multiple imputation this reflects the uncertainty due to missing data. Consequently, the standard errors especially for group "5" now have grown up, compared to the preceding analysis.

2.3 Quantiles

Estimation of quantiles for numerical variables is possible using the function `jk2.quantile`. All related analyses mentioned up to this point apply in the same way. Note that these analyses apply for numerical dependent variables. See the examples in the help file of `jk2.quantile()`.

3 Generalized linear models

Considering multiple imputations and clustered structure in the estimation of generalized linear models is based on the same principes mentioned before. However, some additional comments due to specific characteristics of regression models have to be made. First we now have another type of

variable—independent variables, which may occur as multiple imputed variables, too. Second, we (optionally) have to specify the regression expression, what about we may are confused, as one variable occurs in different labels in multiple imputed data sets. Third, we will have to specify the kind of regression we propose to estimate, for example linear or logistic regression. We start with a simple example using the same data as before.

```
> mod1  <- jk2.glm(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+          JKrep = "JKrep", groups = list(country = "country"),
+          dependent = list(reading = paste("reading_score", 1:3, sep = ""),
+                           writing = paste("writing_score", 1:3, sep = "")),
+          independent = list(gender = "sex", INCOME = c("income1", "income2")),
+          complete.permutation = "no", glm.family = gaussian(link="identity") )
Create 81 replicate weights.
Found 1 grouping variable(s).
Run 2 analyses overall.
Use 3 replications overall.
...
Pooling Standard errors. Assume no nested structure.
Use 3 replications overall.
...
Pooling Standard errors. Assume no nested structure.
```

As we might have expected, the outcome is a single data frame in the long format. And long really means long! For our purpose, it may be sufficient to content ourself with the summary provided by `dG`. But beforehand let us consider how many regression analyses are conducted and how many results we expect to find. The message on the console speaks of about "2 analyses overall" according to two dependent variables, reading and writing. But strictly speaking, we have estimated six regression analyses, as the model is fitted in each group separately. As we specified one group variable dividing the data into three distinct groups, for which we instruct `jk2.glm()` to fit the regression model separately, we find results of the three models for each of the two dependent variables in the results. More specifically, for each country, an intercept and two regression coefficients according to `gender` and `INCOME` are estimated. The `dG` function allows us to have a look only at a specific result out of the 6 analyses. `analyses = 1:2` advises the function to display the results of the first and second analysis. First we should consider that each single analyses is characterized by two variables, the group for which the model is fitted, and the dependent variable. In the heading we find information about both. The actual regression results are displayed underneath.

```
> dG(mod1, analyses = 1:2)
```

```
           groups: country = LandA
dependent Variable: reading

    parameter     est     se t.value p.value
1 (Intercept) -82.461 52.569  -1.569   0.117
2      INCOME   0.306  0.025  12.033   0.000
3  gendermale -31.851  7.025  -4.534   0.000

           R-squared: 0.148; SE(R-squared): 0
Nagelkerkes R-squared: 0.597; SE(Nagelkerkes R-squared): 0.002
1659 observations and 1656 degrees of freedom.
------------------------------------------------------------------
           groups: country = LandB
dependent Variable: reading

    parameter     est     se t.value p.value
1 (Intercept) -72.903 77.878  -0.936   0.349
2      INCOME   0.286  0.038   7.567   0.000
3  gendermale -15.644  5.976  -2.618   0.009

           R-squared: 0.107; SE(R-squared): 0
Nagelkerkes R-squared: 0.501; SE(Nagelkerkes R-squared): 0.001
1573 observations and 1570 degrees of freedom.
```

Remember what was said about factors in the chapter about frequency tables: The gender variable now has to be defined explicitly to be of class factor! Otherwise, albeit gender variable may be coded as 0/1, it would be treated to be a continuous numeric variable. With only two levels—male and female—this may have no effect on the results, but consider a factor variable with three levels, which may be coded 0, 1 and 2. We are interested in two coefficients which correspond to the effect of level 1 vs. level 0 and the effect of level 2 vs. level 0. If we miss to define the variable to be of class factor, only one coefficient is computed, and the variable is assumed to be continuous. What we see additionally is that R implicitly defined the female group to be the reference—the regression parameter was labelled `gendermale`.

Now we try something different. First we define gender to be our dependent variable. Secondly, we use country as a predictor. This is to test whether the proportions of gender vary between countries. To simplify displaying the results, we use the same workaround as in the example before.

```
> mod1  <- jk2.glm(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+          JKrep = "JKrep", dependent = list(gender = "sex"),
+          independent = list(federalState = "country") ,
+          complete.permutation = "no", glm.family = binomial(link="logit") )
Create 81 replicate weights.
No group(s) specified. Analyses will be computed only for the whole sample.
```

```
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 1 replications overall.
.
> dG(mod1)
            groups:
dependent Variable: gender

          parameter    est    se t.value p.value
1       (Intercept)  0.121 0.049   2.458   0.014
2 federalStateLandB -0.074 0.077  -0.964   0.335
3 federalStateLandC -0.093 0.068  -1.377   0.168

            R-squared: 0; SE(R-squared): NA
Nagelkerkes R-squared: 0; SE(Nagelkerkes R-squared): NA
4619 observations and 4616 degrees of freedom.
```

As we have no imputed variables, only one replication is run. No pooling has taken place. Although we have only defined one independent variable, we obtain two regression coefficients for the two categories of the country variable. Again, R choosed its favorite reference group by itself. The effects are expressed in relation to `LandA`. To interpretate the effects, the coefficients may be transformed to odds ratios:

```
> exp(mod1[3:5,"value"])
[1] 1.1291068 0.9285035 0.9109296
```

In `LandB` the odds ratio to be male is 0.93 times the corresponding odds ratio in `LandA`. The following subsections address three little questions one might ask oneself.

3.1 How to change reference group at costumer's option

As we saw in the preceding section, R choosed the reference group of factor variables by itself. Persuading R to meet our needs is easier said than done. The essentially easiest way is a rather dummy method: recode the variable so that your favourite reference group occurs at first when ordered alphabetically. We will demonstrate this procedure about the gender variable in our fictitious data set. Remember the first example in section 3—R choosed the females to be the reference. Why? Simply because female comes before male in the alphabet. Let's use the `recode()` function from the `car` package to define a new variable `sexRecoded`:

```
> library(car)
> reading_writing[,"sexRecoded"] <- car::recode(reading_writing[,"sex"], "'male' = '_male'")
```

The simple trick of adding a `"_"` sign to the `"male"` label provokes R to sort `"_male"` before `"female"` alphabetically. Consequently, `"male"` is used as reference group when repeating the first example analysis of section 3.

```
> mod1  <- jk2.glm(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+          JKrep = "JKrep", groups = list(country = "country"),
+          dependent = list(reading = paste("reading_score", 1:3, sep = "")),
+          independent = list(gender = "sexRecoded", INCOME = c("income1", "income2")),
+          complete.permutation = "no", glm.family = gaussian(link="identity") )
Create 81 replicate weights.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replications overall.
...
Pooling Standard errors. Assume no nested structure.
> dG(mod1, analyses = 1)
            groups: country = LandA
dependent Variable: reading

     parameter      est     se t.value p.value
1  (Intercept) -114.313 51.188  -2.233   0.026
2       INCOME    0.306  0.025  12.033   0.000
3 genderfemale   31.851  7.025   4.534   0.000

            R-squared: 0.148; SE(R-squared): 0
Nagelkerkes R-squared: 0.597; SE(Nagelkerkes R-squared): 0.002
1659 observations and 1656 degrees of freedom.
```

3.2 How to modify the regression statement

When we call `glm()`, we are advised to specify the formula statement of the regression model. In `jk2.glm()`, however, the regression statement is created automatically by the independent variables connected by a `"+"` symbol. As `jk2.glm()` is advised to accomplish multiple imputations, the names of one and the same variable change between imputations. Consequently, the regression statement is adapted between replications. However, regression models which go beyond simple addition of predictors, have to be specified manually. The right side of the regression formula has to defined via the `reg.statement` argument as a string, whereas the names of the independent variables occurring in the `independent` argument have to be used. We illustrate this procedure by seizing on the preceding example. We have two independent variables, `gender` and `INCOME`. Without specifying any `reg.statement` argument, `jk2.glm()` implicitly uses `gender + INCOME`. Modelling an additional

interaction of both variables needs to be defined explicitly:

```
> mod1  <- jk2.glm(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD", JKZone = "JKZone",
+          JKrep = "JKrep", groups = list(country = "country"),
+          dependent = list(reading = paste("reading_score", 1:3, sep = "")),
+          independent = list(gender = "sexRecoded", INCOME = c("income1", "income2")),
+          reg.statement = "gender * INCOME",
+          complete.permutation = "no", glm.family = gaussian(link="identity") )
Create 81 replicate weights.
Found 1 grouping variable(s).
Run 1 analyses overall.
Use 3 replications overall.
...
Pooling Standard errors. Assume no nested structure.
```

Please not that the regression statement—if specified—must contain all variables defined in the `independent` argument. Note further that the names of the variables occurring in the `independent` argument are used instead of single imputation variable names. Don't try your luck with `reg.statement = "sexRecoded * income1"`! This logic implies that the list of variables in the `independent` argument has to be named. Consequently, `independent = list("sexRecoded", c("income1", "income2"))` will only work as long as no `reg.statement` argument is specified. Let's have a look at the output of the first analysis:

```
> dG(mod1, analyses = 1)
            groups: country = LandA
dependent Variable: reading

             parameter      est      se t.value p.value
1          (Intercept) -167.849  79.191  -2.120   0.034
2               INCOME    0.333   0.039   8.494   0.000
3         genderfemale  146.080 107.127   1.364   0.173
4 genderfemale:INCOME   -0.057   0.052  -1.086   0.278

            R-squared: 0.149; SE(R-squared): 0
Nagelkerkes R-squared: 0.6; SE(Nagelkerkes R-squared): 0.003
1659 observations and 1655 degrees of freedom.
```

3.3 Which of both determination coefficients should I pay attention?

The output of each `jk2.glm()` analysis also contains the pooled determination coefficient, R^2, most frequently the conventional R^2 and Nagelkerke's R^2. However, in linear regression models, i.e. if the identity link is used, assuming

normally distributed errors, the conventional R^2 should be used to interpret explained variance. In log-linear regression models, i.e. if the binomial link function is used, Nagelkerke's R^2 should be used. The reason for reporting both coefficients is the programming disability of the package developer.

4 Nested imputations

The next to last chapter of this little tutorial is reserved to the problem of nested imputation. The general concept is described in Rubin (2003). At this point, only some specific aspects which are relevant in large scale assessments, are mentioned. Suppose you want to estimate IRT proficiencies (often denoted θ) in a specific domain. Applying an extensive marginal model which comprehends of items responses and background information as well, the posterior distribution of each examinees' θ is specified. Without any certain proficiency value of a specific examinee, plausible values are drawn from the posterior of each examinee. Conceptually, plausible values are multiple imputations of the inherently missing variable θ and may analyzed in standard statistic procedures according to the generalized linear model. To obtain valid estimates and standard errors, the results have to be pooled according to Rubin (1987).

Suppose you have missing data in the background variables as well, which have to be imputed in the first step, which may result in $M = 5$ data sets. For each data set a marginal IRT model is specified and $N = 20$ plausible values are drawn. Overall $5 \times 20 = 100$ plausible values in a dependency structure will result from the analysis. Formally, we now have nested imputed data. To pool the results, the formulas in Rubin (1987) cannot be applied, as the plausible values do not stem from a common "nest". The interdependence has to be taken into account. Whereas the conventional pooling formulas split the overall variance in the variance within imputation and the variance between imputation, where the latter one is used to estimate the uncertainty due to imputation, the formulas for nested imputation extend the old ones by splitting the variance between imputation in the within-nest variance and the variance between nests. See Rubin (2003) for further details. These varied formulas are also implemented in `eatRep`.

If the data analysed with `eatRep` stem from a nested multiple imputation

structure, this structure has to be specified. More specifically, `eatRep` has to know the number of nests and the number of imputations in each nest. Moreover, it has to be specified which variable belongs to which imputation in which nest. The above procedure sounds more complicated than it hopefully is.

4.1 Example: Compute descriptives from a nested imputation structure

The `reading_writing` data set ist not very proper for instructional purposes concerning nested imputations, but we will do our best. First consider the variables `"income1"` and `"income2"`. Suppose that these are only two imputations of an `INCOME` variable which originally contains missing values. Therefore, we have $M = 2$ nests. Now we assume that we have measured only one domain in our IRT analysis, which we call "ability". Suppose we draw three plausible values in each nest, therefore $N = 3$. The corresponding variables are `"reading_score1"`, `"reading_score2"` and `"reading_score3"` for the first nest, and `"writing_score1"`, `"writing_score2"` and `"writing_score3"` for the second nest. (I am aware that the variable labelling is not disarmingly intuitive that way.) Overall, we have $3 \times 2 = 6$ plausible values in a dependency structure. We want to estimate descriptives for groups of high vs. low income, and for each country separately. First we create an indicator of high vs. low income, where we set the threshold arbitrarily at 2000.

```
> reading_writing[,"income_ind1"] <- factor(ifelse(reading_writing[,"income1"]>2000,"high","low"))
> reading_writing[,"income_ind2"] <- factor(ifelse(reading_writing[,"income2"]>2000,"high","low"))
```

We than will have to specify two groups for which we want to compute descriptives. Remember that one group appears as a nested imputed variable:

```
> means  <- jk2.mean(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD",
+           JKZone = "JKZone", JKrep = "JKrep",
+           groups = list(country = "country", INCOME = c("income_ind1","income_ind2")),
+           dependent = list(ability = list(nest1 = paste("reading_score",1:3,sep=""),
+                                           nest2 = paste("writing_score",1:3,sep="") )),
+           complete.permutation = "no" )
Create 81 replicate weights.
Found 2 grouping variable(s).
Run 1 analyses overall.
Use 6 replication(s) overall.
......
Pooling Standard errors. Underlying nested structure is assumed.
```

The nested structure has to be specified in the `dependent` and `group` argument. The `dependent` argument now is no longer a list of character vectors but a list of lists. As `dependent` is a list of length 1, only one analysis is conducted. As the first and only element of `dependent` is a list instead of a character vector, a nested imputation structure is implied. As the length of the first element of `dependent` is 2, $M = 2$ nests are assumed. Therefore, the function expects to find group variables either without any imputations or with exactly $M = 2$ imputations. Moreover, the plausible values of the first nest (labelled `"reading_score1"`, `"reading_score2"` and `"reading_score3"`) are assumed to belong to the first imputation of `INCOME`, labelled `"income1"`. Correspondingly, the plausible values of the second nest (labelled `"writing_score1"`, `"writing_score2"` and `"writing_score3"`) are assumed to belong to the second imputation of `INCOME`, labelled `"income2"`.

Conceptually, the `complete.permutation` argument should be set to `"nothing"` if a nested imputation structure is assumed. The output reveals rather high standard errors, as in our fictitious example the between-nest variance is rather high.

```
> dM(means, omitTerms = c("var", "Ncases", "NcasesValid", "meanGroupDiff"))
    depVar country INCOME mean_est mean_se  sd_est sd_se
1  ability   LandA   high  542.281   6.213  94.346 5.042
5  ability   LandA    low  484.866   6.186 102.186 2.922
9  ability   LandB   high  521.928   5.310  93.385 4.272
13 ability   LandB    low  464.114   7.357 107.599 6.171
17 ability   LandC   high  523.521  13.633  97.717 5.415
21 ability   LandC    low  483.562  12.777 108.226 4.475
```

4.2 Example: Fit a linear regression model in a nested imputation structure

The principles of considering the nested structure are quite the same as in the preceding example. For each country and the both `income` groups we want to predict "ability" by `gender` and `INCOME`. Using `income` as group variable likewise allows for investigating whether a potential effect of `INCOME` on "ability" differs in groups of high vs. low income. Additionally, we model a possible interaction between `gender` and `INCOME`.

```
> mod1  <- jk2.glm(dat = reading_writing, ID = "idstud", wgt = "wgtSTUD",
+           JKZone = "JKZone", JKrep = "JKrep",
+           groups = list(country = "country", INCOME.group = c("income_ind1","income_ind2")),
```

```
+           dependent = list(ability = list(nest1 = paste("reading_score",1:3,sep=""),
+                                           nest2 = paste("writing_score",1:3,sep="") )),
+           independent = list ( gender = "sex", INCOME = c("income1", "income2")),
+           reg.statement = c("gender * INCOME"), glm.family = gaussian(link = "identity"),
+           complete.permutation = "no" )

Create 81 replicate weights.
Found 2 grouping variable(s).
Run 1 analyses overall.
Use 6 replications overall.
......
Pooling Standard errors. Underlying nested structure is assumed.

> dG(mod1, analyses = 1)

            groups: country = LandA; INCOME.group = high
dependent Variable: ability

          parameter     est      se t.value p.value
1       (Intercept) -22.362 142.139  -0.157   0.875
2            INCOME   0.272   0.065   4.209   0.000
3        gendermale  78.689 221.332   0.356   0.722
4 gendermale:INCOME  -0.042   0.102  -0.414   0.679

            R-squared: 0.049; SE(R-squared): 0
Nagelkerkes R-squared: 0.387; SE(Nagelkerkes R-squared): 0.004
843.5 observations and 839.5 degrees of freedom.
```

References

Douglas˜A. Luke. *Multilevel modeling*. Sage, Thousand Oaks, CA, 2009.

Thomas Lumley. Analysis of complex survey samples. *Journal of Statistical Software*, 9(1):1–19, 2004.

Thomas Lumley. *Survey: analysis of complex survey samples*. R package version 3.28-2, 2012.

Donald˜B. Rubin. *Multiple imputation for no nonresponse in surveys*. Wiley, New York, 1987.

Donald˜B. Rubin. Nested multiple imputation of nmes via partially incompatible mcmc. *Statistica Neerlandica*, 51(1):3–18, 2003.

Westat. *WesVar*. Westat, Rockville, MD, 2000.