Gerbil: An R package for multiple imputation

Michael Robbins, Max Griswold, Pedro Nascimento de Lima

10/1/2020

Contents

Introduction to `gerbil`

The problem of missing data is common in scientific analysis. A variety of events might occur leading to missing data in a project: survey respondents might skip questions, data might be blinded for privacy reasons, or subjects might leave a study early. Restricting an analysis to complete cases can lead to biased inferences, biased parameters, larger standard errors, or reduced statistical power, depending on the reason data is missing.[1] Properly accounting for missing data in an analysis is critical for producing efficient and accurate results.

The `gerbil` software package introduces a contemporary solution for missing data problems using Generalized Efficient Regression-Based Multivariate Imputation with Latent processes (GERBIL). Through the use of a latent data-generating mechanism, the GERBIL method applies imputation through joint modeling on datasets which may contain variables with a variety of general structures (e.g. categorical data, continuous data, binary data), which is not easily accommodated by other joint modeling approaches. As such, the GERBIL method circumvents the use of fully conditional specification (FCS) in which sampling from an incoherent joint distribution may be applied. `gerbil` applies a novel method of joint imputation that uses regression-based modeling - sequences of conditional regression models are used to build the joint distribution. Therefore, `gerbil` provides its user flexibility to specify any dependencies within the conditional regression models.

`gerbil` is computationally efficient and ideal for application on high dimensional datasets with missingness. This is due to `gerbil`'s efficient use of the SWEEP operator for both sampling and modeling steps to minimize the amount of matrix inversion and due to `gerbil`'s avoidance of computationally expensive logistic and multinomial regression. More details on the specifics of GERBIL's approach are available in a scientific article introducing the method.[2] Therein, the procedure is demonstrated in simulations to recover information lost from missing data with less error and in less computational time than alternative approaches using FCS.

Using `gerbil` in an analysis requires a few assumptions (which are common to other imputation approaches) that users should keep in mind when using this package. Estimators using imputed data produced from `gerbil` are valid under the "missing at random" and "missing completely at random" assumptions. Additionally,

[1]Rubin, D. B. (1976). Inference and missing data. Biometrika, 63(3), 581-592.

[2]Robbins, M. W. (2020). A flexible and efficient algorithm for joint imputation of general data. arXiv preprint arXiv:2008.02243.

the GERBIL method assumes the data have a latent data-generating process and this latent process has a multivariate Gaussian distribution.

This vignette aims to demonstrate the use of the `gerbil` package. We'll use data from the India Human Development Survey-II to demonstrate `gerbil`'s functions and ability to recover missing data under both the MCAR and MAR assumptions.[3]. We'll generate missingness within the dataset, to create MCAR and MAR patterns. We'll then compare model predictions from a "complete-cases' model, with predictions using imputations produced by GERBIL with the MCAR and MAR datasets.

`gerbil` Example

Data and initial results

The India Human Development Survey-II is a nationally-representative survey on multiple topics, collected during 2010-2011. For example 1, we'll use data from the individual-module, pertaining to use a few select variables to predict income using a linear regression. To demonstrate the utility of `gerbil`, we'll drop 10% of observations in both MCAR and MAR patterns. We'll then use `gerbil` to recover those observations and compare the goodness-of-fit across four models: Complete cases, MAR, MCAR, and recovered cases.

```
library(gerbil)

#Replace with upload data, once that's finished. For now, read from local file
data(ihd)

#Specify all our variable types correctly:
ihd$job_field       <- factor(ihd$job_field, ordered = F)
ihd$education_level <- factor(ihd$education_level, ordered = T,
                              levels = sort(unique(ihd$education_level)))

# Peek at the initial dataset:
summary(ihd)
#>       sex              age        marital_status    job_field   
#>  Min.   :0.0000   Min.   : 1.00   Min.   :0.0000   1      :9627  
#>  1st Qu.:0.0000   1st Qu.:40.00   1st Qu.:1.0000   4      :8284  
#>  Median :0.0000   Median :49.00   Median :1.0000   8      :6345  
#>  Mean   :0.1431   Mean   :49.69   Mean   :0.8091   6      :4355  
#>  3rd Qu.:0.0000   3rd Qu.:60.00   3rd Qu.:1.0000   3      :4075  
#>  Max.   :1.0000   Max.   :99.00   Max.   :1.0000   11     :2917  
#>                                                    (Other):6522  
#>  farm_labour_days own_livestock   education_level     income        
#>  Min.   :  0.00   Min.   :0.000   0      :13453   Min.   :       0  
#>  1st Qu.:  0.00   1st Qu.:0.000   10     : 4521   1st Qu.:   39000  
#>  Median :  0.00   Median :0.000   5      : 3844   Median :   73500  
#>  Mean   : 47.73   Mean   :0.418   8      : 3111   Mean   :  128187  
#>  3rd Qu.: 70.00   3rd Qu.:1.000   9      : 3059   3rd Qu.:  144000  
#>  Max.   :365.00   Max.   :1.000   7      : 2276   Max.   :11360000  
#>                                   (Other):11861
```

We can see from the above summary table we have one dependent variable, "income", and 7 independent variables. The variables have the following data structures:

- The outcome, "income", is a continuous variable referring to net income in rupees, in the year 2010.

[3]Desai, Sonalde, and Vanneman, Reeve. India Human Development Survey-II (IHDS-II), 2011-12. Inter-university Consortium for Political and Social Research [distributor], 2018-08-08. https://doi.org/10.3886/ICPSR36151.v6

Value	Job Field
1	Cultivation
2	Allied agriculture
3	Agricultural labour
4	Non-agricultural wage labour
5	Artisan
6	Small business
7	Organized business
8	Salaried
9	Professional
10	Retired
11	Housework
12	Student
13	Unemployed
14	Unfit/Disabled
15	Other

Value	Educational Attainment
0	None
1	1st class
2	2nd class
3	3rd class
4	4th class
5	5th class
6	6th class
7	7th class
8	8th class
9	9th class
10	Secondary
11	11th class
12	High Secondary
13	1 year post-secondary
14	2 year post-secondary
15	Bachelors
16	Above Bachelors

- “sex” is a binary variable, corresponding to the head-of-household’s biological sex, with 0 = “male” and 1 = “female”.
- “age” is a continuous variable; we’ll drop observations with age less than 18, which are implausible observations for a head-of-household.
- “job_field” is a categorical variable in one of fifteen categories (see categories below).
- “education_level” is an ordinal categorical variable, referring to the highest level of education attained by the head of household (see categories below).
- “marital status” corresponds to 0 = “unmarried” and 1 = “married”.
- “farm_labour_days” is a semi-continuous variable, which equals zero if the head-of-household does not do farm work or is a continuous value in number of days if the head-of-household does do farm work.
- “own_livestock” is a binary variable indicating if the head-of-household owns livestock.

```r
percent_miss <- 0.35
drop_cols    <- c("income", "education_level", "age", "marital_status",
                  "job_field", "farm_labour_days", "own_livestock")

ihd_mcar <- data.frame(ihd)
ihd_mar  <- data.frame(ihd)

#Drop column cells completely at random
for (col in drop_cols){
  drop_rows <- sample(1:nrow(ihd), percent_miss*nrow(ihd), replace = F)
  ihd_mcar[drop_rows,][col] <- NA
}

#Drop column cells on the basis of age & sex (missing at random).
#This procedure is based on:

# Schouten, R. M., Lugtig, P., & Vink, G. (2018).
# Generating missing values for simulation purposes: a multivariate amputation procedure.
# Journal of Statistical Computation and Simulation, 88(15), 2909-2930.
```

```
#Make married individuals more likely to drop out from the study,
#along with females. Calculate a weighted sum score on this basis,
#then standarize the score.

wss <- 2*(ihd$sex) + 1.5*(abs(ihd$marital_status - 1))
wss <- (wss - mean(wss))/sd(wss)

#Transform the standardized weighted sum scores into a probability using
#a modified logistic function.
prob_drop <- plogis(wss, location = 0, scale = 3)

for (col in drop_cols){
  #For each column, randomly drop rows, with probability equal to
  #logistically-transformed standardized weighted sum scores
  drop_rows <- sample(1:nrow(ihd), percent_miss*nrow(ihd), replace = F, prob = prob_drop)
  ihd_mar[drop_rows,][col] <- NA
}
```

In the above code, we simulated two different missingness patterns: Missing completely at random (MCAR) and missing at random (MAR). In the MCAR case, we randomly dropped rows from the initial dataset. In the MAR case, we dropped rows on the basis of a hypothetical missingness pattern, determined by observed data with complete cases. In the example here, we dropped rows with female individuals at a greater probability than males, and unmarried individuals at a greater probability than married individuals.

Using `gerbil`

Running `gerbil` requires a single line of code. `gerbil` should ideally recognize the data structure of each variable included in our dataset. We can simply pass our dataset with missingness to `gerbil`, which will create a single imputation dataset, using five monte-carlo markov chains. Below, in the MAR example, we're reducing the amount of warnings produced by Gerbilby setting the option "printFlag" to false.

```
#Impute using Gerbil for both MCAR & MAR datasets:
imputed_mcar <- gerbil(ihd_mcar)
#> Variable Summary:
#>                      Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                         binary        42125        0     0.00%
#> age               continuous (EMP)        27382    14743    35.00%
#> marital_status              binary        27382    14743    35.00%
#> job_field              categorical        27382    14743    35.00%
#> farm_labour_days continuous (EMP)        27382    14743    35.00%
#> own_livestock               binary        27382    14743    35.00%
#> education_level            ordinal        27382    14743    35.00%
#> income            continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 0.95
#> Imp. 1: gerbil initialized.  Time = 3.89
#> Imp. 1: MCMC iteration 1 completed. Total time = 3.93, I-Step: 3.85, P-Step: 0.08
#> Imp. 1: MCMC iteration 2 completed. Total time = 3.59, I-Step: 3.51, P-Step: 0.08
#> Imp. 1: MCMC iteration 3 completed. Total time = 3.83, I-Step: 3.71, P-Step: 0.12
#> Imp. 1: MCMC iteration 4 completed. Total time = 3.83, I-Step: 3.75, P-Step: 0.08
#> Imp. 1: MCMC iteration 5 completed. Total time = 3.89, I-Step: 3.81, P-Step: 0.08
#> Completed untransformations for imputed dataset 1, Time = 0.71
imputed_mar  <- gerbil(ihd_mar)
#> Variable Summary:
```

```
#>                      Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                         binary        42125        0     0.00%
#> age               continuous (EMP)        27382    14743    35.00%
#> marital_status              binary        27382    14743    35.00%
#> job_field              categorical        27382    14743    35.00%
#> farm_labour_days continuous (EMP)        27382    14743    35.00%
#> own_livestock               binary        27382    14743    35.00%
#> education_level            ordinal        27382    14743    35.00%
#> income            continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 1.09
#> Imp. 1: gerbil initialized.  Time = 3.99
#> Imp. 1: MCMC iteration 1 completed. Total time = 3.83, I-Step: 3.76, P-Step: 0.07
#> Imp. 1: MCMC iteration 2 completed. Total time = 3.53, I-Step: 3.45, P-Step: 0.08
#> Imp. 1: MCMC iteration 3 completed. Total time = 3.62, I-Step: 3.53, P-Step: 0.09
#> Imp. 1: MCMC iteration 4 completed. Total time = 4.09, I-Step: 4.03, P-Step: 0.06
#> Imp. 1: MCMC iteration 5 completed. Total time = 3.78, I-Step: 3.70, P-Step: 0.08
#> Completed untransformations for imputed dataset 1, Time = 0.67

#Look at the results
summary(imputed_mcar)
#> Object Class: gerbil
#>
#> Includes 1 imputed dataset created using 5 iterations of MCMC.
#>
#> Predicted Variables, Types and Missing Rates:
#>                      Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                         binary        42125        0     0.00%
#> age               continuous (EMP)        27382    14743    35.00%
#> marital_status              binary        27382    14743    35.00%
#> job_field              categorical        27382    14743    35.00%
#> farm_labour_days continuous (EMP)        27382    14743    35.00%
#> own_livestock               binary        27382    14743    35.00%
#> education_level            ordinal        27382    14743    35.00%
#> income            continuous (EMP)        27382    14743    35.00%
#>
#> Predictor Matrix:
#>                  sex age marital_status job_field farm_labour_days
#> sex                0   0              0         0                0
#> age                1   0              0         0                0
#> marital_status     1   1              0         0                0
#> job_field          1   1              1         0                0
#> farm_labour_days   1   1              1         1                0
#> own_livestock      1   1              1         1                1
#> education_level    1   1              1         1                1
#> income             1   1              1         1                1
#>                  own_livestock education_level income
#> sex                          0               0      0
#> age                          0               0      0
#> marital_status               0               0      0
#> job_field                    0               0      0
#> farm_labour_days             0               0      0
#> own_livestock                0               0      0
#> education_level                 1                0       0
#> income                          1                1       0
```

We can see from the above summary table that the semi-continuous variable wasn't recognized as the correct data structure. Luckily, we can rectify this within `gerbil` by specifying this variable's data structure in the function, using the "type" argument.

```
imputed_mcar <- gerbil(ihd_mcar, type = c(farm_labour_days = "semicont"))
#> Variable Summary:
#>                      Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                         binary        42125        0     0.00%
#> age                continuous (EMP)        27382    14743    35.00%
#> marital_status              binary        27382    14743    35.00%
#> job_field              categorical        27382    14743    35.00%
#> farm_labour_days          semicont        27382    14743    35.00%
#> own_livestock               binary        27382    14743    35.00%
#> education_level            ordinal        27382    14743    35.00%
#> income             continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 0.34
#> Imp. 1: gerbil initialized.  Time = 3.97
#> Imp. 1: MCMC iteration 1 completed. Total time = 3.53, I-Step: 3.46, P-Step: 0.07
#> Imp. 1: MCMC iteration 2 completed. Total time = 3.55, I-Step: 3.50, P-Step: 0.05
#> Imp. 1: MCMC iteration 3 completed. Total time = 3.52, I-Step: 3.47, P-Step: 0.05
#> Imp. 1: MCMC iteration 4 completed. Total time = 3.48, I-Step: 3.42, P-Step: 0.06
#> Imp. 1: MCMC iteration 5 completed. Total time = 3.47, I-Step: 3.41, P-Step: 0.06
#> Completed untransformations for imputed dataset 1, Time = 0.09
imputed_mar  <- gerbil(ihd_mar, type = c(farm_labour_days = "semicont"))
#> Variable Summary:
#>                      Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                         binary        42125        0     0.00%
#> age                continuous (EMP)        27382    14743    35.00%
#> marital_status              binary        27382    14743    35.00%
#> job_field              categorical        27382    14743    35.00%
#> farm_labour_days          semicont        27382    14743    35.00%
#> own_livestock               binary        27382    14743    35.00%
#> education_level            ordinal        27382    14743    35.00%
#> income             continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 0.34
#> Imp. 1: gerbil initialized.  Time = 3.83
#> Imp. 1: MCMC iteration 1 completed. Total time = 3.45, I-Step: 3.39, P-Step: 0.06
#> Imp. 1: MCMC iteration 2 completed. Total time = 3.48, I-Step: 3.42, P-Step: 0.06
#> Imp. 1: MCMC iteration 3 completed. Total time = 4.19, I-Step: 4.15, P-Step: 0.04
#> Imp. 1: MCMC iteration 4 completed. Total time = 3.41, I-Step: 3.35, P-Step: 0.06
#> Imp. 1: MCMC iteration 5 completed. Total time = 3.52, I-Step: 3.46, P-Step: 0.06
#> Completed untransformations for imputed dataset 1, Time = 0.07
```

Alternatively, we could have set types using a slightly different method. Let's also increase the number of imputed datasets that `gerbil` produces using the "m" parameter (we could also increase the number of MCMC iterations using the "mcmciter" parameter). An increased number of imputations should lead to more accurate results later, when we pool models across imputation datasets. Similarly, increased MCMC iterations should lead to improved convergence in the imputation estimates.

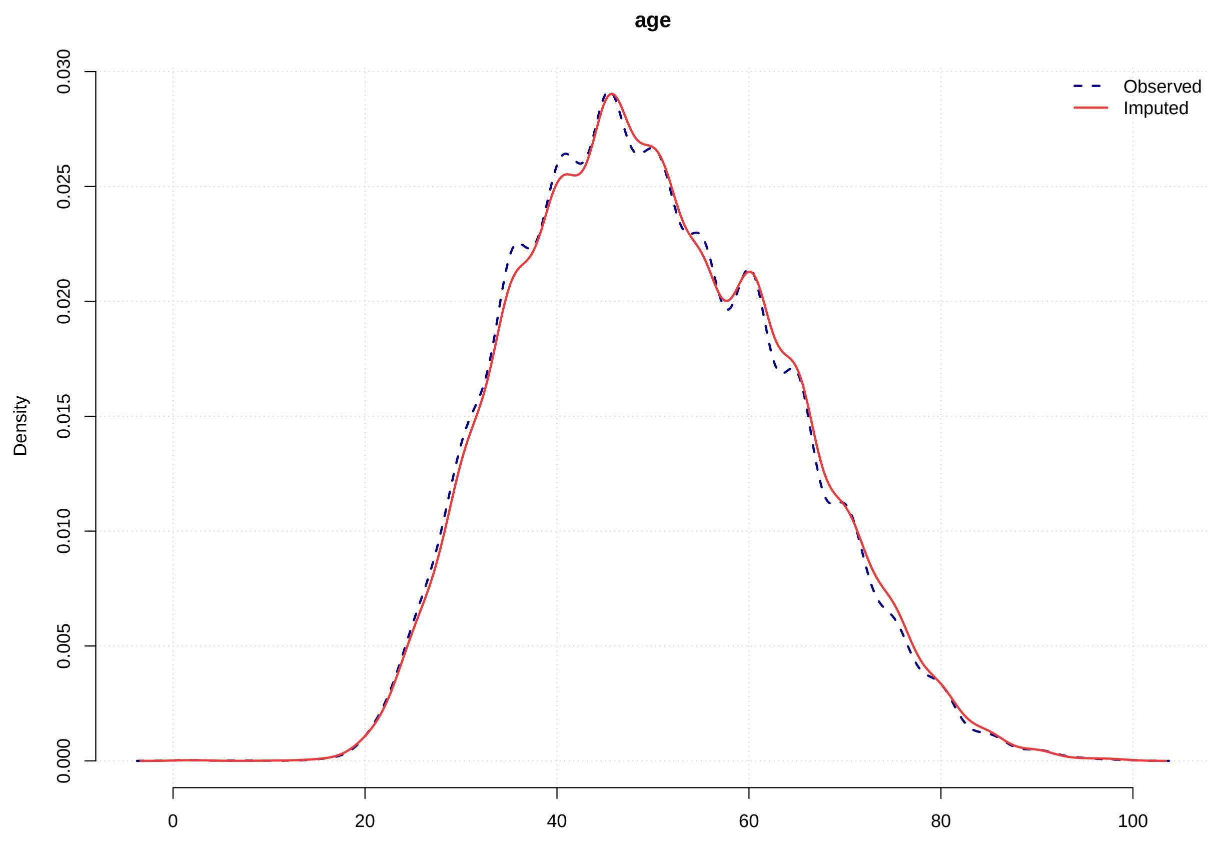
```
imported_mcar <- gerbil(ihd_mcar, semi = "farm_labour_days",
                        m = 5, mcmciter = 100, seed = 111675)
#> Variable Summary:
#>                       Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                          binary        42125        0     0.00%
#> age                continuous (EMP)        27382    14743    35.00%
#> marital_status               binary        27382    14743    35.00%
#> job_field               categorical        27382    14743    35.00%
#> farm_labour_days           semicont        27382    14743    35.00%
#> own_livestock                binary        27382    14743    35.00%
#> education_level             ordinal        27382    14743    35.00%
#> income             continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 0.35
#> Parallelizing with n.cores = 4...
#> Created 5 imputed datasets.  Total time = 861.00.
#> Completed untransformations and post-processing, Time = 2.06

imputed_mar  <- gerbil(ihd_mar, semi = "farm_labour_days",
                        m = 5, mcmciter = 100, seed = 111675)
#> Variable Summary:
#>                       Variable.Type Num.Observed Num.Miss Miss.Rate
#> sex                          binary        42125        0     0.00%
#> age                continuous (EMP)        27382    14743    35.00%
#> marital_status               binary        27382    14743    35.00%
#> job_field               categorical        27382    14743    35.00%
#> farm_labour_days           semicont        27382    14743    35.00%
#> own_livestock                binary        27382    14743    35.00%
#> education_level             ordinal        27382    14743    35.00%
#> income             continuous (EMP)        27382    14743    35.00%
#>
#> Completed transformations, Time = 0.35
#> Parallelizing with n.cores = 4...
#> Created 5 imputed datasets.  Total time = 851.85.
#> Completed untransformations and post-processing, Time = 1.87
```

We can see from the above summary that all of the variable types are being correctly identified by `gerbil` We also note that with an increased number of imputation datasets, `gerbil` is parallelizing computations. This behavior could be set off by setting "n.cores" equal to zero.

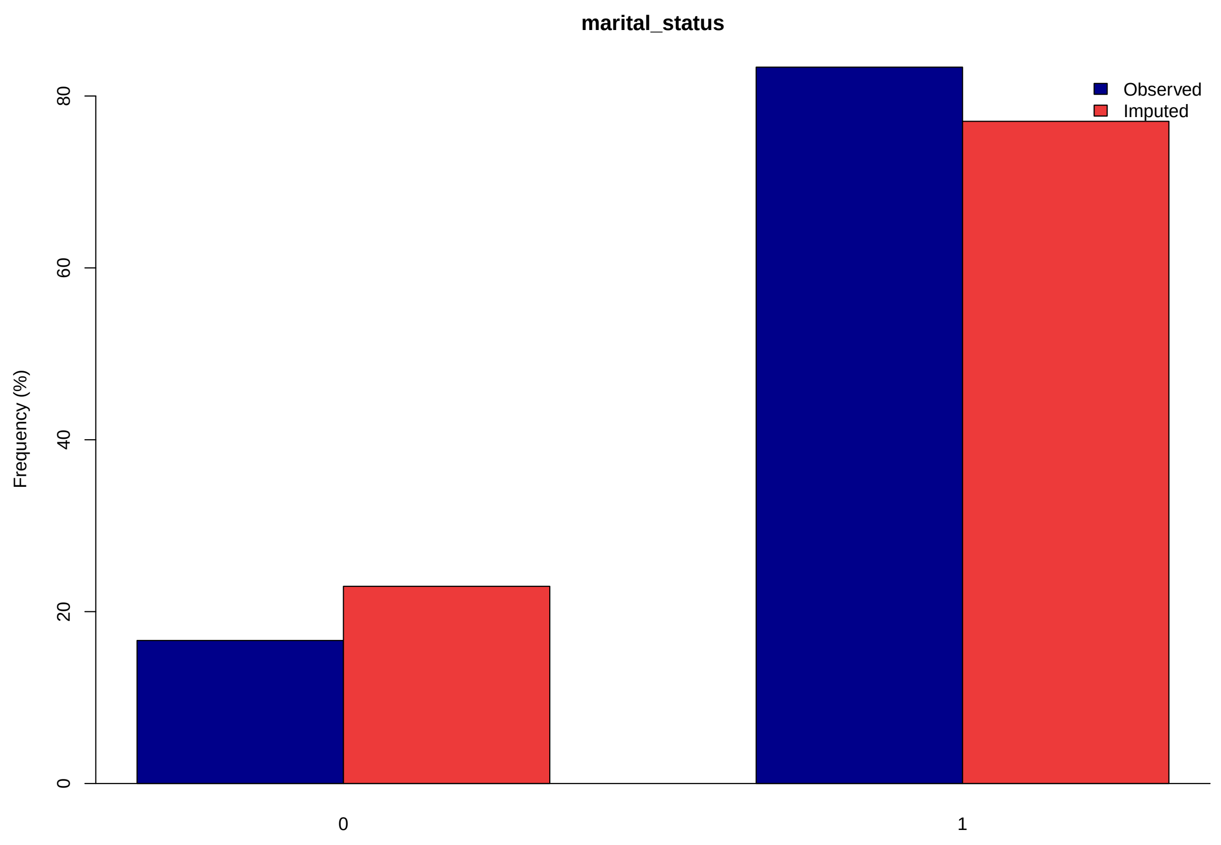
We'll now check the performance of the imputations by comparing the imputed estimates to our observed data.

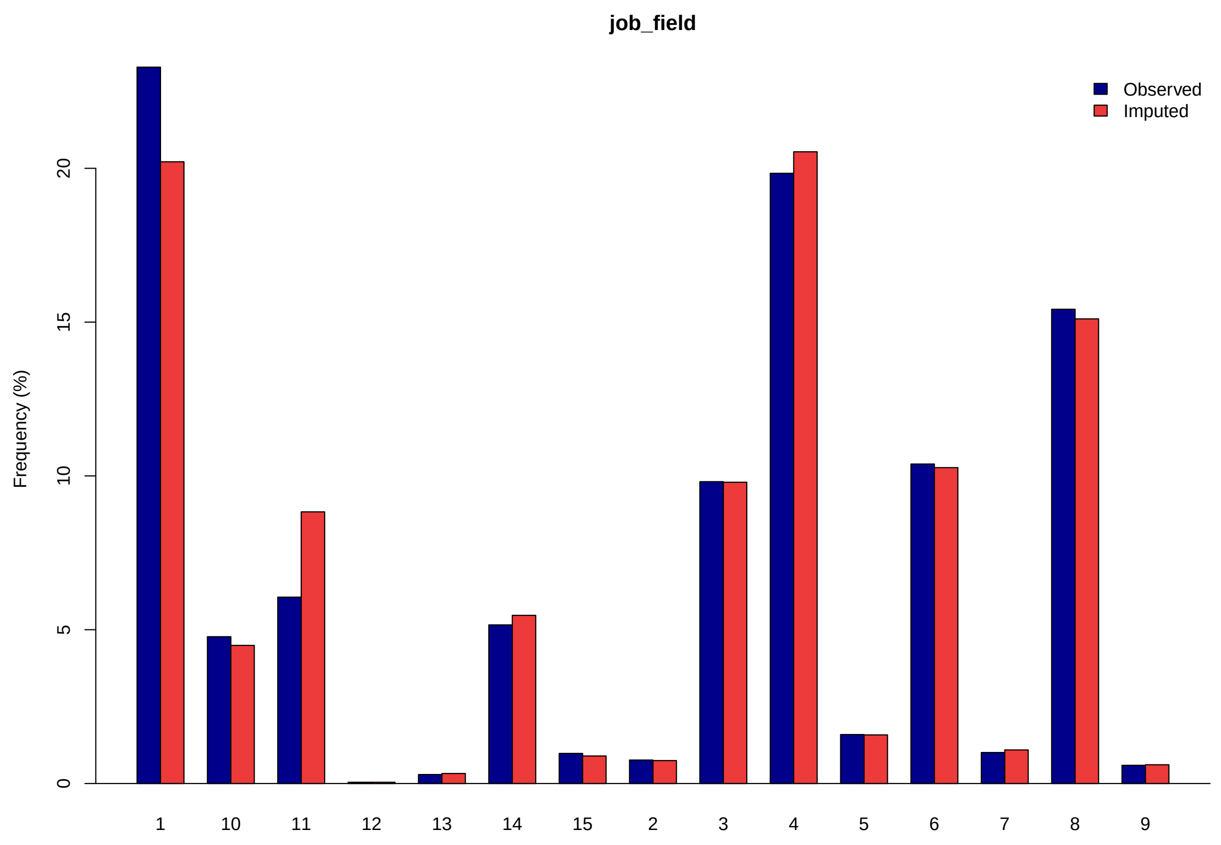
Diagnostic plots

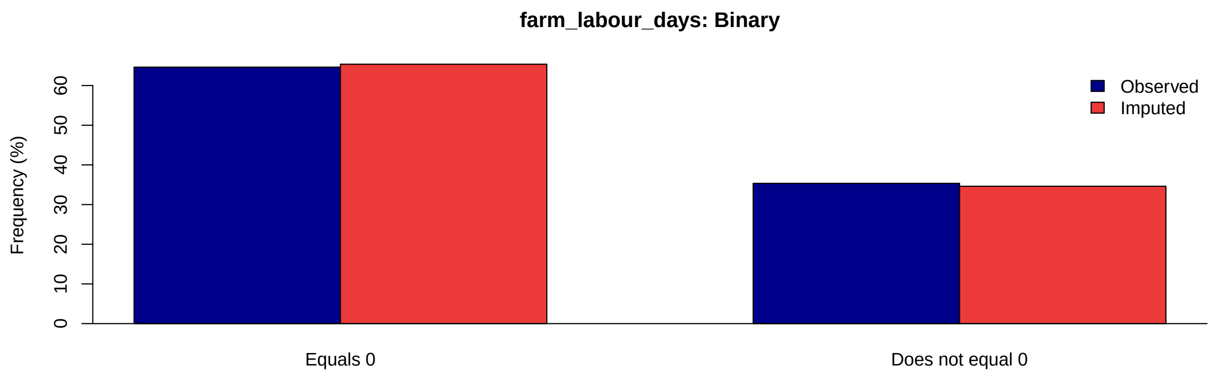
```
# Plotting functions aren't being sourced when loading gerbil, currently.
# Update this once they're fixed.

#Plot all univariate plots (e.g. type = 1) for the MAR dataset,
#and one example of a bivariate plot(e.g. type = 2):
par(bty = "n")
plot(imputed_mar, type = 1, mfrow = c(2, 1))
```

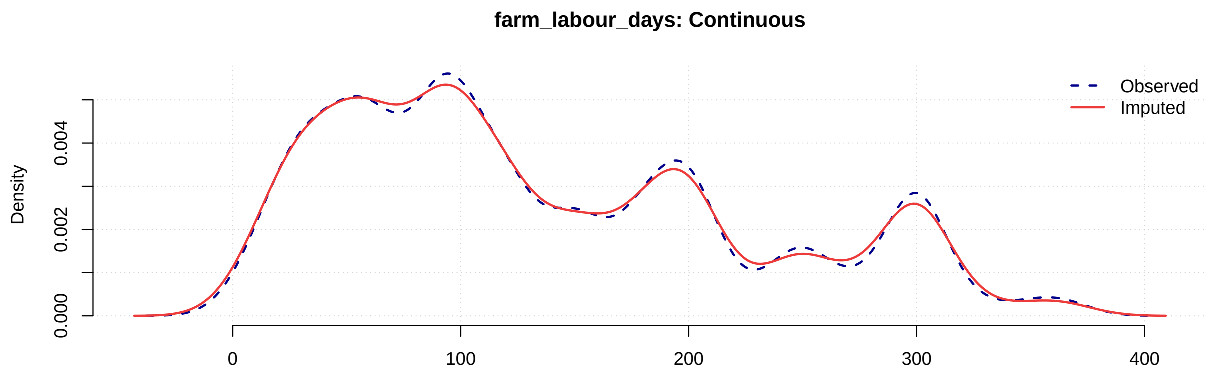


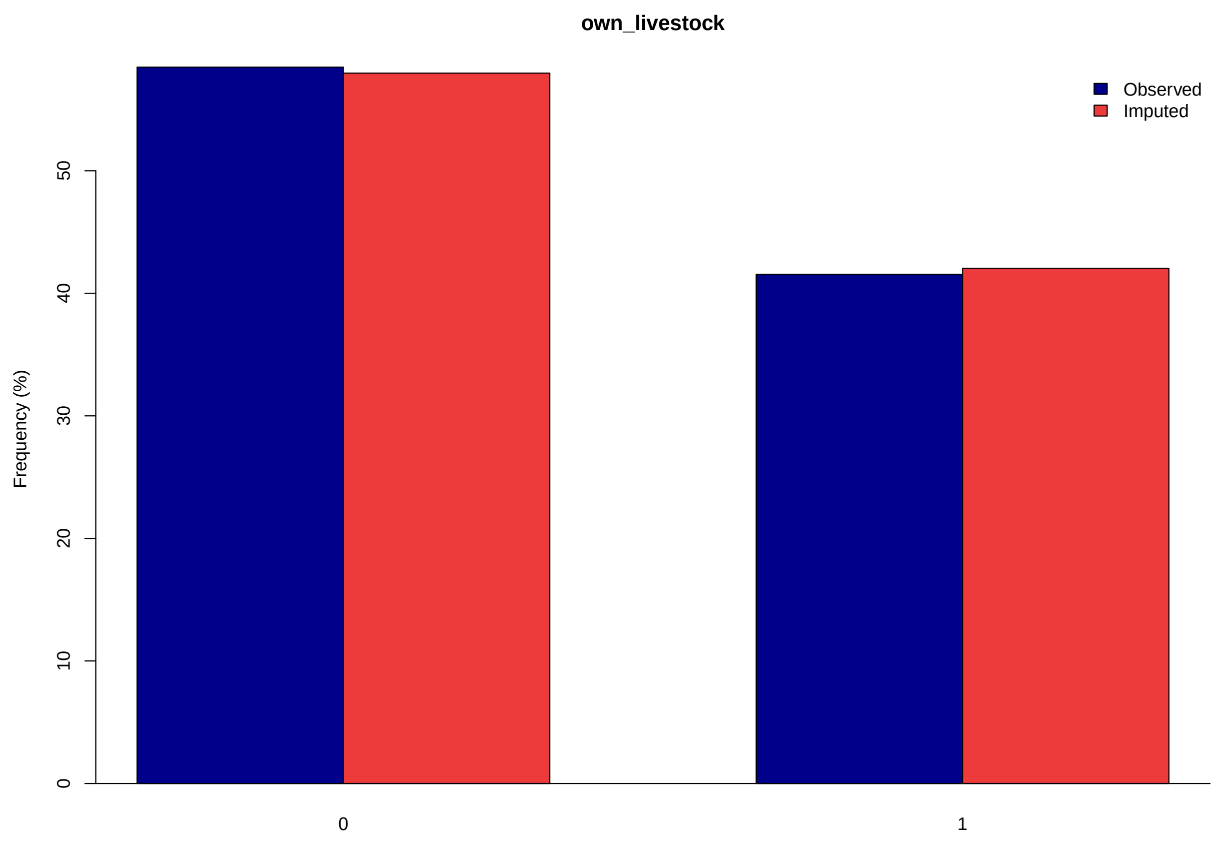
age
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
20
40
60
80
100
Observed
Imputed

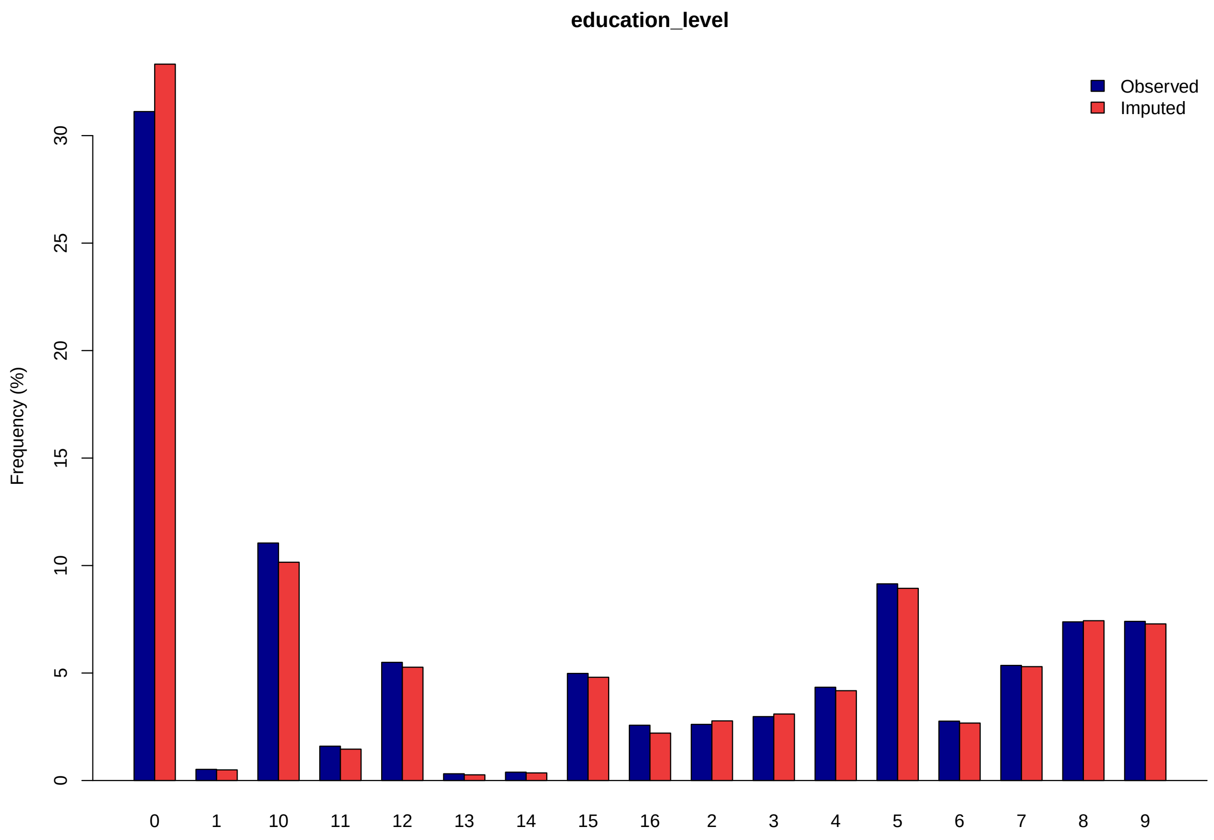

marital_status
Frequency (%)
Observed
Imputed
0
20
40
60
80
0
1

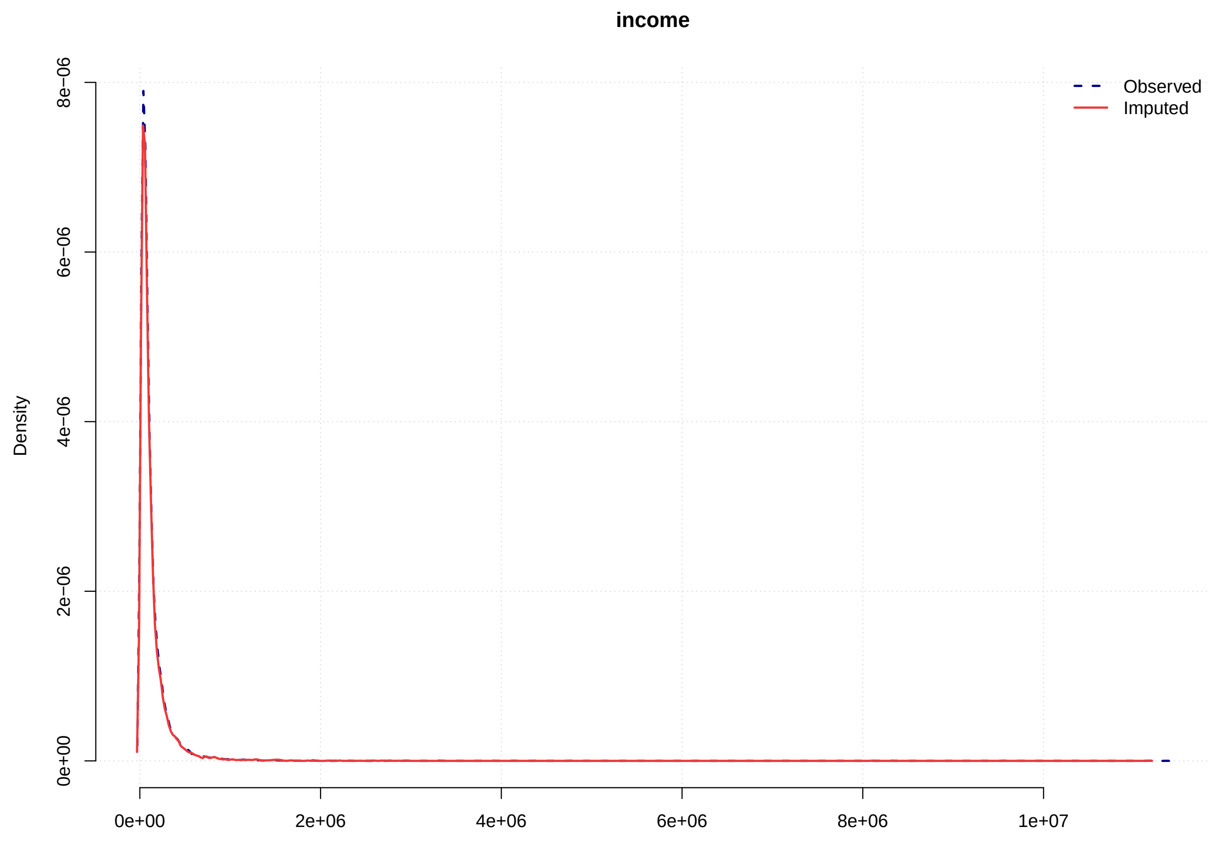

job_field
Observed
Imputed
Frequency (%)
0
5
10
15
20
1
10
11
12
13
14
15
2
3
4
5
6
7
8
9


farm_labour_days: Binary
Frequency (%)
0
10
20
30
40
50
60
Equals 0
Does not equal 0
Observed
Imputed


farm_labour_days: Continuous
Density
0.000
0.002
0.004
0
100
200
300
400
Observed
Imputed


own_livestock
Observed
Imputed
Frequency (%)
0
10
20
30
40
50
0
1


education_level
Observed
Imputed
Frequency (%)
0
5
10
15
20
25
30
0
1
10
11
12
13
14
15
16
2
3
4
5
6
7
8
9



```r
plot(imputed_mar, type = 2, y = c("sex", "marital_status"))
```

We can see from the above plots that GERBIL did pretty well matching imputed estimates with observed data. For the continuous variable “age”, the empirical distributions of imputed estimates matches the density of observed data fairly well (though the observed data has more clumping at 5-year intervals). The frequency of “marital_status” unmarried (i.e. == 0) is a little bit higher in the imputed data than in observed data but this is what we would hope happens, given the missingness pattern was initially predicated on being higher for unmarried individuals.

The patterns in “job_field” are pretty similar between observed data and imputed estimates, with the exception of housework (i.e. job_field == 11), which occurs at a higher frequency for the imputed estimates (again, this might be a function of the imposed missingness patterns, since we dropped “females” from the dataset at a higher rate than “males”; “females” in the dataset are coded as having the job_field “housework” more often than males.).

The “farm_labour_days” variable has a very similar frequency & density, between the imputed estimates and observed data. The empirical distribution matches well between the imputations and observations, despite the rather odd empirical distribution. We also see a close match between imputations and observations for the “own_livestock” variable.

The bivariate plot for sex & marital status allows follows a pattern we might expect. For males, we imputed similar estimates across both marital statuses. However, for females, we imputed estimates more often as “unmarried” than “married”, which makes sense given we generated more missingness within that category.

In addition to diagnostic plots, gerbil also has built-in functions for analyzing correlations between observed and imputed cases, through use of the “cor_gerbil” function:

```r
ihd_mar_imp_cors <- cor_gerbil(imputed_mar)
print(ihd_mar_imp_cors)
#> Summary analysis comparing correlations calculated between observed cases with
#>    corresponding correlations calcluated between imputed cases:
#>
#> Average absolute difference in correlation between observed and imputed cases:
#>
#> 0.0142
#>
#> Average value of the test statistic based on Fisher's z across all variable pairs:
#>
#> 1.3661
#>
#> The largest statistic (14.48) corresponds to variable pair sex and marital_status.
#>
#> Portion of p-values for the test based on Fisher's z that are less than 0.05:
#>
#> 0.2478
```

From the above statistics, we can see that the average absolute difference in pair-wise variable correlations between the imputed estimates and observed data is 0.0142, which indicates the variable correlations across imputed cases are, on average, very similar to variable correlations across the observed cases. For 75.2% of all pair-wise variable correlations, the correlation estimated with imputed cases does not test as being different from the correlation estimated with the observed cases at the 5% significance level. Note, however, that this test is based off of Fisher's z transformation and requires bivariate normality, which is not satisfied in these data

Now that we've seen the imputations look reasonable, we can run similar models to the ones we ran earlier, but using the imputation datasets rather than restricting the analysis to complete cases. We'll combine model estimates produced across imputation datasets by using Rubin's rules to calculate pooled predictions[4] We'll then compare predictions from the imputed models with the complete-cases model.

Regression models

```r
specification <- formula(income ~ sex + age + marital_status + job_field +
                         farm_labour_days + own_livestock + education_level)

#For illustrative purposes I'm using a glm with the poisson family
#but alternative estimators would likely perform better with this data &
#specification

model_bench <- glm(specification, data = ihd, family = poisson(link = "log"))
model_mcar  <- glm(specification, data = ihd_mcar, family = poisson(link = "log"))
model_mar   <- glm(specification, data = ihd_mar, family = poisson(link = "log"))

#Run model on each imputation dataset, then pool predictions.
impute_mcar <- lapply(imputed_mcar$imputed, glm, formula = specification,
                      family = poisson(link = "log"))
impute_mar  <- lapply(imputed_mar$imputed, glm, formula = specification,
                      family = poisson(link = "log"))

pool_predictions <- function(model_list){
```

[4]Rubin, D. B. (2004). Multiple imputation for nonresponse in surveys (Vol. 81). John Wiley & Sons.

Table 1: RMSE for imputation models

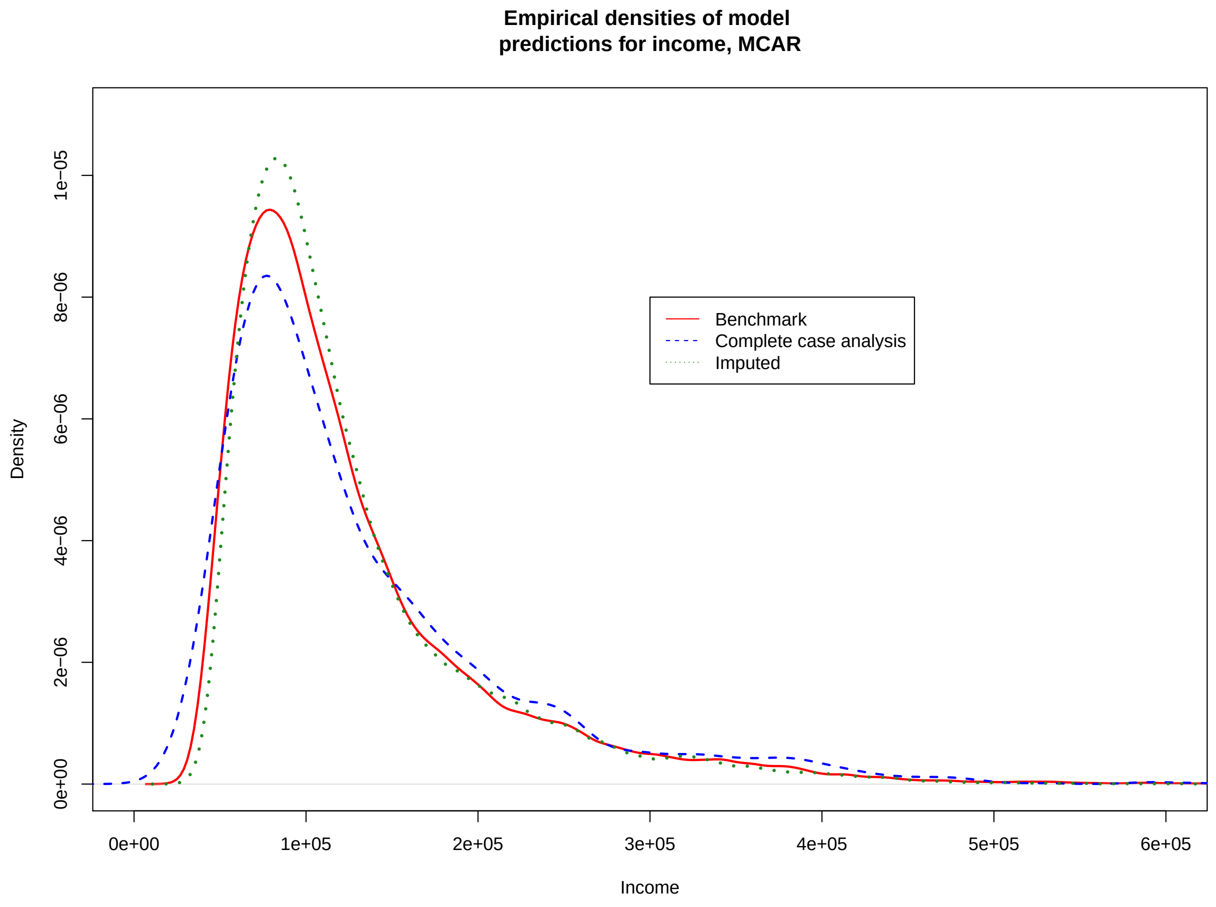
Model	RMSE (Benchmark)	RMSE (MCAR)	RMSE (MAR)
Complete case analysis	200000.4	205200.9	202145.7
Imputed analysis	200000.4	200928.4	200764.6

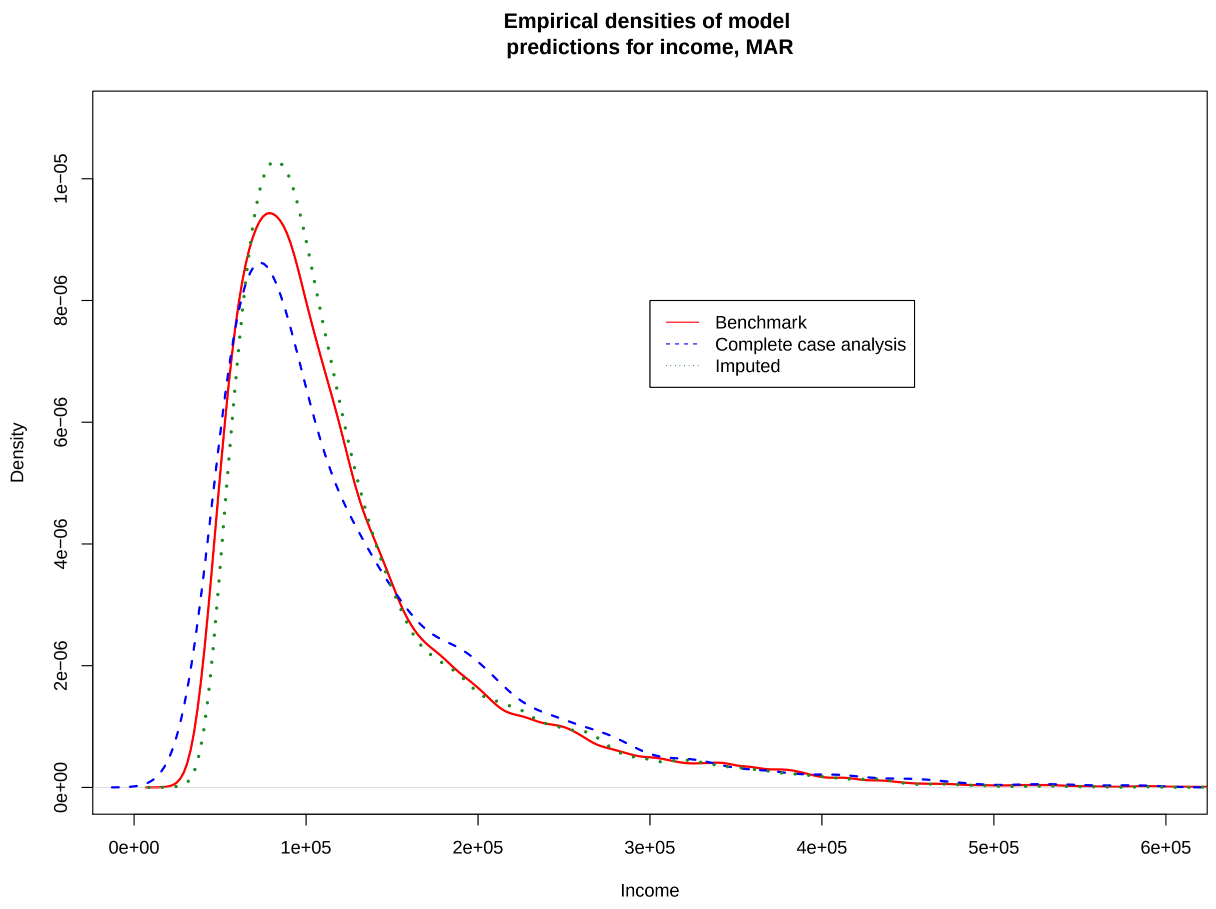
```r
  #Get coefficients from each model
  coefs <- lapply(model_list, function(x){x$coefficients})

  #Convert the list of coefficients into a data.frame
  coefs <- data.frame(matrix(unlist(coefs), nrow = length(coefs), byrow = T))

  #Calculate mean of each coefficient across models:
  coef_means <- apply(coefs, 2, mean)

  #Replace coefficient estimates in one of the model objects with those
  #produced by pooled predictions (so we can easily use the predict function)

  dummy_model <- model_list[[1]]
  dummy_model$coefficients <- coef_means

  pred_pool <- predict(dummy_model, type = "response", newdata = ihd)

  return(pred_pool)
}

bench_cca   <- predict(model_bench, type = "response")
mcar_cca    <- predict(model_mcar, type = "response", newdata = ihd)
mar_cca     <- predict(model_mar, type = "response", newdata = ihd)

mcar_pooled <- pool_predictions(impute_mcar)
mar_pooled  <- pool_predictions(impute_mar)

#Calculate RMSE for each model
rmse <- function(obs, pred){
  return(sqrt(mean((obs - pred)^2)))
}

rmse_bench    <- rmse(ihd$income, bench_cca)

rmse_mcar_cca <- rmse(ihd$income, mcar_cca)
rmse_mar_cca  <- rmse(ihd$income, mar_cca)

rmse_mcar_imp <- rmse(ihd$income, mcar_pooled)
rmse_mar_imp  <- rmse(ihd$income, mar_pooled)

rmse_results <- data.frame("models" = c("Complete case analysis", "Imputed analysis"),
                         "RMSE (Benchmark)" = c(rmse_bench, rmse_bench),
                         "RMSE (MCAR)" = c(rmse_mcar_cca, rmse_mcar_imp),
                         "RMSE (MAR)" = c(rmse_mar_cca, rmse_mar_imp))
```

We can see from the above table depicting each model's respective root-mean-squared error. In both the MCAR and MAR example, we would ideally want model error to closely match the error in the benchmark model.The benchmark model uses the initial dataset without any missingness and has root-mean-squared error (RMSE) of 20,0000 rupees (~$2750 USD).

However, we can see in both cases that restricting the MCAR and MAR model to a complete case analysis is leading to larger RMSE than the benchmark. We've reduced the discrepancy between the benchmark and MCAR/MAR by imputing for the missing observations using `gerbil`.

The models for MAR & MCAR using GERBIL imputations have RMSEs that are closer to the error observed in the benchmark model. Additionally, we can look at the predictions graphically by plotting their distributions across models:





Looking at these empirical density plots, we can also see that the predictions with imputations better match the predictions in the benchmark, compared to the complete case analysis. In both cases, GERBIL improved the analysis with missing data, compared with dropping incomplete cases within the model.