Simulation Study: Effectiveness of Imputation

Ian Strawn

2026-03-10

Introduction

Imputation is a widely-used method to manage missing data, and with the wide variety of options for how we might impute data, there are tools to cover many different situations. There are benefits and drawbacks to each of the methods; some are more accurate at predicting what the missing values would be, while others remain computationally easy, making them quicker to implement. With the variety of methods at our disposal, it would benefit us to see, as quantitatively as possible, how well each of the different methods perform under varying situations. As such, I have set up a simulation study in order to evaluate the effectiveness of certain imputation methods under a few different settings.

Throughout the course of this project, I aim to briefly explain the various imputation methods that are going to be used and types of missing data considered, lay out the structure of the simulation that I developed, evaluate how well each imputation method performs, and then compare and discuss the results.

Methods

Types of missingness

To begin, it benefits us to briefly go over the types of missingness present in the data. In this project, for the sake of simplicity, I'm going to assume that missingness (i.e. values that should be recorded in a data set but appear as missing or "NA") only applies to the response variable of our data, and not to the explanatory variables. There are three main mechanisms for missingness in our data:

Missing Completely At Random (MCAR)

This is the nicest kind of missingness to work with; it assumes that missingness in a variable does not depend on the values of either the explanatory or the response variables, and instead happens completely randomly. Mathematically, we would represent this as follows:

$$f_{M_i|Y_i}(M_i = \tilde{m}|Y_i = y_i, \phi) = f_{M|Y}(M_i = \tilde{m}|Y_i = y_i^*, \phi)$$

Functionally, the math here says that the likelihood of a data value being missing doesn't depend on the actual realization of each response, Y_i, or any other parameters, ϕ. This is the best-case scenario for missingness, as with MCAR, we have many tools at our disposal for handling the missingness (including just dumping the observations altogether), and the lack of a correlation structure removes potential confounding information. Unfortunately, we almost never see this in the real world - when data is missing, it's more likely to fall into the 2 other types of missingness.

Missing at Random (MAR)

Despite what the name seems to imply, there is a little bit of a dependence structure when we have data that is missing at random, but it's less restrictive than MNAR (discussed later). Mathematically, the likelihood for data that is missing at random can be expressed as

$$f_{M_i|Y_i}(M_i = \tilde{m}_i|Y_{(0)i} = \tilde{y}_{(0)i}, Y_{(1)i} = \tilde{y}_{(1)i}, \phi) = f_{M_i|Y_i}(M_i = \tilde{m}_i|Y_{(0)i} = \tilde{y}_{(0)i}, Y_{(1)i} = \tilde{y}^*_{(1)i}, \phi)$$

What this means in a practical sense is that the missingness of the response *can* depend on the values of any of the explanatory variables, but that its missingness doesn't depend on the value of the response itself. For example, say I was doing a study where I measured the height and weight of a certain number of respondents. If a patient is less likely to report their weight *if they were above a certain height* (but their actual weight doesn't change the likelihood of reporting that weight), we'd consider that MAR. MAR is not quite as nice as MCAR mathematically, but we do still have methods for handling it in ways that don't introduce bias into any estimates we may make. Unfortunately, the final missingness type is the trickiest one to deal with, and is also fairly common in day-to-day data analysis.

Missing Not At Random (MNAR)

Data that is missing not at random is tough to work with. Once again, the mathematical representation for the missingness is shown as follows:

$$f_{M_i|Y_i}(M_i = \tilde{m}_i|Y_{(0)i} = \tilde{y}_{(0)i}, Y_{(1)i} = \tilde{y}_{(1)i}, \phi) \neq f_{M_i|Y_i}(M_i = \tilde{m}_i|Y_{(0)i} = \tilde{y}_{(0)i}, Y_{(1)i} = \tilde{y}^*_{(1)i}, \phi)$$

Functionally, what this suggests is that the value that a response would take affects its likelihood of being missing. Returning to the height/weight example mentioned earlier, if somebody who weighed over 200 pounds was less likely to report their weight than somebody who was under 200 pounds, we would consider that to be MNAR, since the value of the response influences how likely it is to be missing. Unfortunately, this is one of the more common types of missingness in the real world. Even after imputation, MNAR data tends to lead to biased results, making it pretty difficult to draw inference from analysis.

Imputation Strategies

When missingness is present in our response, we oftentimes turn to imputation to fill in those missing values. There are a few different approaches for imputation, each of varying complexity.

Mean Imputation

This is the simplest of the imputation methods. Any time we have a missing response ($y_{i(1)}$), we simply take the mean of the observed responses ($y_{i(0)}$) and substitute that value in for the missingness:

$$y_{i(1)} = \frac{1}{n}\sum_j y_{j(0)}$$

for all j that have an observed response value. This is computationally quite simple, but not great for anything else - it tends to destroy the variance of the data set, changing the overall strucutre of the data, and also is not very useful if you actually care about the imputed values themselves.

Random Imputation

Random imputation is also a relatively simple method. Here, whenever we have missingness, we just randomly sample from the response values that we *do* have data for:

$$y_{i(1)} \in_R y_{j(0)}$$

This helps preserve the variance structure for our data and is not computationally intensive, which is useful, but beyond that, this still isn't a great method for trying to make accurate predictions on our missing y's, as it ignores the relationship between the predictors and the response. Fortunately, we have better methods.

Random Regression Imputation

This is where our predictions for missing y's begin to improve a bit. First, we take the complete observations that we do have and fit a linear regression line to those. Once we've done that, we take the explanatory values corresponding to our missing responses, plug them into our regression equation to get a predicted response value, and then *randomly select one of the residuals from the linear regression model to tack onto our prediction for the missing response*:

$$y_{i(1)} = \mathbf{X}\beta + \epsilon_i,$$

where $\epsilon_i \in_R (\hat{y}_j - y_j)$ for all j with an observed response. This method is more useful than the previous two - we now take into consideration the relationship between the covariates and the response, and we also preserve the variance structure of the data set. However, this doesn't fully solve the problem of having bias in our estimates, so it's still not as good as other options.

Hot Deck Imputation

Hot deck imputation is commonly used by Census Bureau for survey research. The gist of the technique involves creating a pool of responses that seem similar to what a missing response *should* look like, and then randomly selecting from that pool to impute. In the case of the synthetic data that I've created, the best way to measure the responses that should be included in our donor pool of size k is to calculate the euclidean distance between the covariates associated with the missing value and the covariates of every single other observed value. In other words, our Euclidian distance looks something like this:

$$d = \sqrt{(x_{1i} - x_{1j,obs})^2 + (x_{2i} - x_{2j,obs})^2 + (x_{3i} - x_{3j,obs})^2 + (x_{4i} - x_{4j,obs})^2}$$

where i corresponds to the indices for all missing response values, and j corresponds to the indices for all observed values. The k shortest distances and their corresponding observations are recorded, the k responses are placed in a pool, and one of them is randomly selected to serve as the imputed y value. This is conceptually a little more complicated, but this method also does a decent job of retaining the data's variance structure.

Multiple Imputation with Predictive Mean Matching (PMM)

Multiple imputation is the most complex of the 5 methods, but also is quite robust. Without getting too into the weeds of how the technique works, we attempt to randomly generate a certain number of "complete" datasets D, fit regression models on the complete observations of each of those datasets, and then average the coefficients that come from each of those regressions. Each regression estimate is of the familiar mean-form:

$$\bar{\theta}_D = \frac{1}{D}\sum_{d=1}^{D}\hat{\theta}_d$$

Predictive Mean Matching is how we handle the predictions for the missing values of the D data sets. Once we've created the linear regression model for each of the new data sets, we use those models to generate predictions for the missing responses. However, instead of directly imputing the values of the regression predictions in for the missing y's, we look at the k *observed* responses that are closest to that prediction, and then randomly draw one of those to serve as a "donor" for the missing response. The response takes that donor value as its imputed value instead.

This method is computationally intensive, but quite robust. It does a very good job of dealing with skewed or non-normal data, and is usually a very good way of getting close to the true values of any parameter values you're trying to estimate.

Simulation Structure

Now that the groundwork has been laid, we can move onto how the simulation itself is going to work. The ultimate goal for each simulation is going to be to test how well we can accurately measure a regression coefficient with datasets involving imputed data of each kind mentioned above. The steps for the simulation will be as follows:

1. Randomly generate a data set of sample size n = 200 that's constructed based off of 4 predictors. The response will take the form $y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \epsilon_i$, and the x_i's and ϵ_i's will be distributed as either normal or gamma. In one iteration of the simulation, they will be normally distributed to ensure that linear regression assumptions are met properly, but in a second iteration, they'll be generated from a gamma distribution. This violates standard linear regression assumptions and allows us to evaluate the robustness of imputation methods under model misspecification. Each of the β's will be randomly generated from an $N(0, 1)$ distribution, and stored for later comparison.
2. Once the full dataset has been created, 15% of the y values will be deemed as missing, and the method for which they'll be removed will depend on the missingness mechanism of that simulation loop (MCAR, MAR, or MNAR). MAR is implemented by making the missingness of the response depend on the value of x_3, while MNAR will depend on the value of y itself.
3. The data set with missing values will then have the missing values imputed using the 5 different methods mentioned above
4. With the imputed data sets, regression models will be created for each imputation strategy, and the β coefficients from those regression models will be stored for comparison.
5. The RMSE of the estimated regression coefficients is then calculated for each of the 5 imputation regression models using the actual β values as reference
6. Steps 1-5 are repeated 1000 times to get an "average" RMSE for each imputation type, along with a standard deviation.

Important to note: **for simplicity and interpretability, I am designating β_1 as our variable of interest, and only focusing on its predicted coefficients when calculating the error**.

All told, since we are dealing with 3 missingness mechanisms and 2 different data generating mechanisms (normal data and gamma-distributed data), we will run this process 6 times total. All of the code chunks required to do this are included in the **appendix**.

Results

The following tables display the results of the simulation study. The first two tables will be the results from the data that was generated using normal distributions, and the second two tables will be the results from the data that came from the gamma-generated data.

Table 1: Average RMSE For Predicted Regression Coefficient, Normal Data

	Mean	Random	Regression	Hot Deck	Multiple
MCAR	0.167	0.182	0.043	0.067	0.042
MAR	0.164	0.173	0.176	0.071	0.045
MNAR	0.387	0.395	0.396	0.161	0.063

Table 2: Standard Deviation of RMSE's, Normal Data

	Mean	Random	Regression	Hot Deck	Multiple
MCAR	0.0463	0.0559	0.0025	0.0068	0.0025
MAR	0.0420	0.0470	0.0523	0.0089	0.0026
MNAR	0.2158	0.2342	0.2337	0.0388	0.0069

Table 3: Average RMSE For Predicted Regression Coefficient, Gamma Data

	Mean	Random	Regression	Hot Deck	Multiple
MCAR	0.222	0.261	0.126	0.152	0.124
MAR	0.216	0.250	0.251	0.154	0.116
MNAR	0.461	0.475	0.477	0.252	0.152

Table 4: Standard Deviation of RMSE's, Gamma Data

	Mean	Random	Regression	Hot Deck	Multiple
MCAR	0.0835	0.1208	0.0234	0.0360	0.0245
MAR	0.0830	0.1160	0.1513	0.0401	0.0211
MNAR	0.3506	0.3744	0.3930	0.1161	0.0383

Discussion

There are quite a few interesting results worth mentioning, and we'll begin with the results from the normal data. Looking at table 1, we can see that, unsurprisingly, multiple imputation does the best job of predicting the regression coefficient pretty consistently across all 3 of the missing data types. Both multiple imputation and hot deck imputation do a good job of prediction across all 3 missingness types, but hot deck performs poorer more quickly than multiple does as the missingness pattern gets closer to MNAR. Both multiple and hot deck also consistently have the smallest standard deviations (table 2), being orders of magnitude smaller than their other imputation counterparts (which I found impressive).

Random imputation and regression imputation provided interesting results as well. As far as random imputation is concerned, not only does it have the highest average RMSE among each of the imputation methods, but it also has the highest standard deviation. This surprised me a little bit - going into the study, I had expected mean imputation to perform the worst among the methods, as it completely ignores the relationship between the variables and also ruins the variance structure of the data. It's still not a *great* way of imputing, but it does seem to perform marginally better than random imputation does, focusing strictly on the parameter estimation.

Additionally, the dropoff in performance for regression imputation is rather large as we move across the missing data types. Regression imputation performs nearly as well as multiple does in an MCAR setting, and then performs the *worst* of all of the methods in MAR and MNAR settings. This was probably the most surprising result that I saw; since the data was generated in a linear regression setting, I expected the regression imputation to perform quite well when imputing values. However, it seems that regression imputation begins to break down once we lose the optimistic assumption of MCAR.

When we misspecify the data-generating model a bit and begin treating both the covariates and the residuals as gamma instead of normal, we begin to see some dropoffs in performance. All 5 of the imputation methods have noticeably higher mean RMSE's than with the normal data, which makes sense - when we violate linear regression assumptions, we usually introduce some bias into our regression estimates, which would be reflected

in our higher RMSE values. Otherwise, the patterns for the gamma-distributed data are similar for those in the normal setting. Multiple still performs noticeably better than any of the other techniques, followed by hot deck. Additionally, regression is still great under MCAR settings, and then performs very poorly in MAR and MNAR situations. The standard deviations of all of the estimates have also increased noticeably with the non-normal data, unsurprisingly.

Conclusions

From these studies, it seems like any time we have the computational power to implement multiple or hot deck imputation strategies, it would be beneficial for us to do so. Both of these methods appear to perform great in situations where we have normal data across all types of missingness, and still appear to perform relatively robustly in the face of data that violates our linear regression assumptions. If we lack computational power but know that we're dealing with MCAR data, performing regression imputation is not a bad option (but MCAR data is nearly impossible to find in the wild, so betting on that is a bad idea).

I enjoyed the process of setting up and running the simulation, and the process has given me some ideas about where I might take future work on this project. First, and probably most interestingly, it would be fun to continue to screw around with the data generation process to see if we couldn't get the performance of the imputation methods to change even more. I chose the gamma distributions to fit the data because of the inherent asymmetry of gamma distributions, and exploiting that even further might be fun, perhaps with something like a poisson or exponential distribution. Additionally, I would probably also fix my β values instead of randomly generating them every single time, as that way I could potentially also create some confidence intervals for the predictions, making the analysis that much more thorough. Finally, I would also like to mess with the number of covariates in the data set; I don't see that making a massive difference in the analysis, but it would still be worth seeing. Regardless, the results of this project provide an interesting insight into the behavior of both imputation methods and how missingness mechanisms impact them.

Appendix

Data Generation

Hot Deck Function

Hot deck can be a little tricky to implement, so I wrote my own Hot Deck function to handle things for me:

```
#This is gonna create a hotdeck function that's gonna make my life easier later
hotdeck = function(x, y_miss, k){
  #First, identifying which values are missing and which are
  y_imp = y_miss
  obs = which(!is.na(y_miss))
  miss = which(is.na(y_miss))

  for(i in miss){
    #Getting Euclidian distances
    dist = sqrt((x[i,1] - x[obs,1])^2 +
                  (x[i,2] - x[obs,2])^2 +
                  (x[i,3] - x[obs,3])^2 +
                  (x[i,4] - x[obs,4])^2)

    #Give me the closest neighbors
    nearest = obs[order(dist)][1:k]

    #Pick one
    temp = sample(nearest, 1)

    #Assign it
    y_imp[i] = y_miss[temp]
  }

  return(y_imp)
}
```

Normal Data, 4 Variables

MCAR

```
set.seed(530)
#Setting up a rough loop for what a simulation might look like
nsim = 1000
n = 200
p = 4
#Keeping track of each of my betas
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
```

```r
beta_temp = betas_true[i,] = rnorm(4)

#Generate some x's and y's
x1 = rnorm(n, 5, 4)
x2 = rnorm(n, 1, 4)
x3 = rnorm(n, -1, 4)
x4 = rnorm(n, 3, 4)
y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
             beta_temp[3]*x3 + beta_temp[4]*x4 + rnorm(n, 0, 2)
ymean = mean(y)
ysd = sd(y)

#Determine which values of y are going to be missing. I want 15% of the
#values to be missing
nmiss = floor(0.15*n)
miss = sample(1:n, nmiss, replace = FALSE)
y_miss[miss] = NA

#Performing various kinds of imputation
#Mean
y_mean = y_miss
ybar = mean(y_miss, na.rm = TRUE)
y_mean[miss] = ybar

#Random
y_rand = y_miss
y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)

#Random Regression
y_randreg = y_miss
temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
y_randreg[miss] = predict(temp_lm, data.frame(x1, x2, x3, x4))[miss] +
                    sample(temp_lm$residuals, nmiss, replace = TRUE)

#Hot Deck
x = cbind(x1, x2, x3, x4)
y_hd = y_miss
y_hd = hotdeck(x,y_miss, k = 5)

#Multiple (With PMM)
data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
betas_mult[i,] = summary(pool(fit))$estimate[2:5]

#Fitting a regression model for each type of imputation and getting
#Coefficients
#Mean
mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

#Random
rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
```

```
  betas_rand[i,] = summary(rand_lm)$coefficients[2:5]

  #Regression
  reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
  betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

  #Hot Deck
  hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
  betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

  #No need for PMM - that's already done!

  #Getting the MSE
  #Mean
  mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2

  #Random
  rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

  #Random Reg
  randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

  #HD
  hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

  #Multiple (PMM)
  mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2

  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the RMSE's for each type
sqrt(mean(mean_mse))
sqrt(mean(rand_mse))
sqrt(mean(randreg_mse))
sqrt(mean(hd_mse))
sqrt(mean(mult_mse))

sd(mean_mse)
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```

MAR

```
set.seed(530)
nsim = 1000
n = 200
p=4
```

```r
#Keeping track of each of my betas
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
  beta_temp = betas_true[i,] = rnorm(p)

  #Generate some x's and y's
  x1 = rnorm(n, 5, 4)
  x2 = rnorm(n, 1, 4)
  x3 = rnorm(n, -1, 4)
  x4 = rnorm(n, 3, 4)
  y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
               beta_temp[3]*x3 + beta_temp[4]*x4 + rnorm(n, 0, 2)
  xmean = mean(x3)
  xsd = sd(x3)

  #Determine which values of y are going to be missing. Now, for MAR,
  #missing values are going to depend on the value of x3. We're going to
  #assume a structure where if x3 is extreme, the response is going to be
  #eligible to be missing.
  nmiss = floor(0.15*n)
  extreme = which(x3 > xmean + xsd | x3 < xmean - xsd)
  miss = sample(extreme, nmiss, replace = F)
  y_miss[miss] = NA

  #Performing various kinds of imputation
  #Mean
  y_mean = y_miss
  ybar = mean(y_miss, na.rm = TRUE)
  y_mean[miss] = ybar

  #Random
  y_rand = y_miss
  y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)

  #Random Regression
  y_randreg = y_miss
  temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
  y_randreg[miss] = predict(temp_lm, data.frame(x))[miss] +
                    sample(temp_lm$residuals, nmiss, replace = TRUE)

  #Hot Deck
  x = cbind(x1, x2, x3, x4)
  y_hd = y_miss
  y_hd = hotdeck(x,y_miss, k = 5)

  #Multiple (With PMM)
```

```
  data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
  imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
  fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
  betas_mult[i,] = summary(pool(fit))$estimate[2:5]

  #Fitting a regression model for each type of imputation and getting
  #Coefficients
  #Mean
  mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
  betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

  #Random
  rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
  betas_rand[i,] = summary(rand_lm)$coefficients[2:5]

  #Regression
  reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
  betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

  #Hot Deck
  hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
  betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

  #No need for PMM - that's already done!

  #Getting the MSE
  #Mean
  mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2

  #Random
  rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

  #Random Reg
  randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

  #HD
  hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

  #Multiple (PMM)
  mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2

  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the average MSE's for each type and their SD's
mean(mean_mse)
mean(rand_mse)
mean(randreg_mse)
mean(hd_mse)
mean(mult_mse)
```

```
print("")
sd(mean_mse)
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```

MNAR

```
set.seed(530)
nsim = 1000
n = 200
p=4

#Keeping track of each of my betas
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
  beta_temp = betas_true[i,] = rnorm(p)

  #Generate some x's and y's
  x1 = rnorm(n, 5, 4)
  x2 = rnorm(n, 1, 4)
  x3 = rnorm(n, -1, 4)
  x4 = rnorm(n, 3, 4)
  y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
               beta_temp[3]*x3 + beta_temp[4]*x4 + rnorm(n, 0, 2)
  ymean = mean(y)
  ysd = sd(y)

  #Determine which values of y are going to be missing. I want extreme values
  #to be more likely to be missing than average values, and try to keep it
  #consistent with the 15% that we saw in MCAR
  nmiss = floor(0.15*n)
  extreme = which(y > ymean + ysd | y < ymean - ysd)
  miss = sample(extreme, nmiss, replace = F)
  y_miss[miss] = NA

  #Performing various kinds of imputation
  #Mean
  y_mean = y_miss
  ybar = mean(y_miss, na.rm = TRUE)
  y_mean[miss] = ybar

  #Random
  y_rand = y_miss
  y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)
```

```r
#Random Regression
y_randreg = y_miss
temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
y_randreg[miss] = predict(temp_lm, data.frame(x))[miss] +
                    sample(temp_lm$residuals, nmiss, replace = TRUE)

#Hot Deck
x = cbind(x1, x2, x3, x4)
y_hd = y_miss
y_hd = hotdeck(x,y_miss, k = 5)

#Multiple (With PMM)
data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
betas_mult[i,] = summary(pool(fit))$estimate[2:5]

#Fitting a regression model for each type of imputation and getting
#Coefficients
#Mean
mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

#Random
rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
betas_rand[i,] = summary(rand_lm)$coefficients[2:5]

#Regression
reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

#Hot Deck
hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

#No need for PMM - that's already done!

#Getting the MSE
#Mean
mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2

#Random
rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

#Random Reg
randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

#HD
hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

#Multiple (PMM)
mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2
```

```
  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the average MSE's for each type and their SD's
mean(mean_mse)
mean(rand_mse)
mean(randreg_mse)
mean(hd_mse)
mean(mult_mse)
print("")
sd(mean_mse)
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```

Gamma-Distributed Data, 4 Variables

MCAR

```
set.seed(530)
#Setting up a rough loop for what a simulation might look like
nsim = 1000
n = 200
p = 4
#Keeping track of each of my betas
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
  beta_temp = betas_true[i,] = rnorm(4)

  #Generate some x's and y's
  x1 = rgamma(n, 4, 3)
  x2 = rgamma(n, 7, 3)
  x3 = rgamma(n, 1, 1)
  x4 = rgamma(n, 6, 1)
  y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
               beta_temp[3]*x3 + beta_temp[4]*x4 + rgamma(n, 1, 1)
  ymean = mean(y)
  ysd = sd(y)

  #Determine which values of y are going to be missing. I want 15% of the
  #values to be missing
  nmiss = floor(0.15*n)
```

```
miss = sample(1:n, nmiss, replace = FALSE)
y_miss[miss] = NA

#Performing various kinds of imputation
#Mean
y_mean = y_miss
ybar = mean(y_miss, na.rm = TRUE)
y_mean[miss] = ybar

#Random
y_rand = y_miss
y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)

#Random Regression
y_randreg = y_miss
temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
y_randreg[miss] = predict(temp_lm, data.frame(x1, x2, x3, x4))[miss] +
                    sample(temp_lm$residuals, nmiss, replace = TRUE)

#Hot Deck
x = cbind(x1, x2, x3, x4)
y_hd = y_miss
y_hd = hotdeck(x,y_miss, k = 5)

#Multiple (With PMM)
data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
betas_mult[i,] = summary(pool(fit))$estimate[2:5]

#Fitting a regression model for each type of imputation and getting
#Coefficients
#Mean
mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

#Random
rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
betas_rand[i,] = summary(rand_lm)$coefficients[2:5]

#Regression
reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

#Hot Deck
hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

#No need for PMM - that's already done!

#Getting the MSE
#Mean
mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2
```

```
  #Random
  rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

  #Random Reg
  randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

  #HD
  hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

  #Multiple (PMM)
  mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2

  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the RMSE's for each type
sqrt(mean(mean_mse))
sqrt(mean(rand_mse))
sqrt(mean(randreg_mse))
sqrt(mean(hd_mse))
sqrt(mean(mult_mse))

sd(mean_mse)
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```

MAR

```
set.seed(530)
nsim = 1000
n = 200
p=4

#Keeping track of each of my betas
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
  beta_temp = betas_true[i,] = rnorm(p)

  #Generate some x's and y's
  x1 = rgamma(n, 4, 3)
  x2 = rgamma(n, 7, 3)
```

```r
x3 = rgamma(n, 1, 1)
x4 = rgamma(n, 6, 1)
y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
             beta_temp[3]*x3 + beta_temp[4]*x4 + rgamma(n, 1, 1)
xmean = mean(x3)
xsd = sd(x3)

#Determine which values of y are going to be missing. Now, for MAR,
#missing values are going to depend on the value of x3. We're going to
#assume a structure where if x3 is extreme, the response is going to be
#eligible to be missing.
nmiss = floor(0.15*n)
extreme = which(x3 > xmean | x3 < xmean - xsd)
miss = sample(extreme, nmiss, replace = F)
y_miss[miss] = NA

#Performing various kinds of imputation
#Mean
y_mean = y_miss
ybar = mean(y_miss, na.rm = TRUE)
y_mean[miss] = ybar

#Random
y_rand = y_miss
y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)

#Random Regression
y_randreg = y_miss
temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
y_randreg[miss] = predict(temp_lm, data.frame(x))[miss] +
                    sample(temp_lm$residuals, nmiss, replace = TRUE)

#Hot Deck
x = cbind(x1, x2, x3, x4)
y_hd = y_miss
y_hd = hotdeck(x,y_miss, k = 5)

#Multiple (With PMM)
data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
betas_mult[i,] = summary(pool(fit))$estimate[2:5]

#Fitting a regression model for each type of imputation and getting
#Coefficients
#Mean
mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

#Random
rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
betas_rand[i,] = summary(rand_lm)$coefficients[2:5]
```

```
  #Regression
  reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
  betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

  #Hot Deck
  hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
  betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

  #No need for PMM - that's already done!

  #Getting the MSE
  #Mean
  mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2

  #Random
  rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

  #Random Reg
  randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

  #HD
  hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

  #Multiple (PMM)
  mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2

  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the average MSE's for each type and their SD's
sqrt(mean(mean_mse))
sqrt(mean(rand_mse))
sqrt(mean(randreg_mse))
sqrt(mean(hd_mse))
sqrt(mean(mult_mse))
print("")
sd(mean_mse)
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```

MNAR

```
set.seed(530)
nsim = 1000
n = 200
p=4

#Keeping track of each of my betas
```

```r
betas_true = betas_mean = betas_rand = betas_reg = betas_hd = betas_mult =
  matrix(0, nrow = nsim, ncol = p)

#Initializing
mean_mse = rand_mse = randreg_mse = hd_mse = mult_mse = rep(0, nsim)

#One full simulation loop
for(i in 1:nsim){
  #Generate a beta randomly
  beta_temp = betas_true[i,] = rnorm(p)

  #Generate some x's and y's
  x1 = rgamma(n, 4, 3)
  x2 = rgamma(n, 7, 3)
  x3 = rgamma(n, 1, 1)
  x4 = rgamma(n, 6, 1)
  y = y_miss = beta_temp[1]*x1 + beta_temp[2]*x2 +
               beta_temp[3]*x3 + beta_temp[4]*x4 + rgamma(n, 1, 1)
  ymean = mean(y)
  ysd = sd(y)

  #Determine which values of y are going to be missing. I want extreme values
  #to be more likely to be missing than average values, and try to keep it
  #consistent with the 15% that we saw in MCAR
  nmiss = floor(0.15*n)
  extreme = which(y > ymean + ysd | y < ymean - ysd)
  miss = sample(extreme, nmiss, replace = F)
  y_miss[miss] = NA

  #Performing various kinds of imputation
  #Mean
  y_mean = y_miss
  ybar = mean(y_miss, na.rm = TRUE)
  y_mean[miss] = ybar

  #Random
  y_rand = y_miss
  y_rand[miss] = sample(na.omit(y_miss), nmiss, replace = TRUE)

  #Random Regression
  y_randreg = y_miss
  temp_lm = lm(y_miss ~ x1 + x2 + x3 + x4, na.action = na.omit)
  y_randreg[miss] = predict(temp_lm, data.frame(x))[miss] +
                      sample(temp_lm$residuals, nmiss, replace = TRUE)

  #Hot Deck
  x = cbind(x1, x2, x3, x4)
  y_hd = y_miss
  y_hd = hotdeck(x,y_miss, k = 5)

  #Multiple (With PMM)
  data_miss = data.frame(y = y_miss, x1, x2, x3, x4)
  imp = mice(data_miss, m = 5, method = "pmm", print = FALSE)
```

```
  fit = with(imp, lm(y ~ x1 + x2 + x3 + x4))
  betas_mult[i,] = summary(pool(fit))$estimate[2:5]

  #Fitting a regression model for each type of imputation and getting
  #Coefficients
  #Mean
  mean_lm = lm(y_mean ~ x1 + x2 + x3 + x4)
  betas_mean[i,] = summary(mean_lm)$coefficients[2:5]

  #Random
  rand_lm = lm(y_rand ~ x1 + x2 + x3 + x4)
  betas_rand[i,] = summary(rand_lm)$coefficients[2:5]

  #Regression
  reg_lm = lm(y_randreg ~ x1 + x2 + x3 + x4)
  betas_reg[i,] = summary(reg_lm)$coefficients[2:5]

  #Hot Deck
  hd_lm = lm(y_hd ~ x1 + x2 + x3 + x4)
  betas_hd[i,] = summary(hd_lm)$coefficients[2:5]

  #No need for PMM - that's already done!

  #Getting the MSE
  #Mean
  mean_mse[i] = (betas_mean[i,1] - betas_true[i,1])^2

  #Random
  rand_mse[i] = (betas_rand[i,1] - betas_true[i,1])^2

  #Random Reg
  randreg_mse[i] = (betas_reg[i,1] - betas_true[i,1])^2

  #HD
  hd_mse[i] = (betas_hd[i,1] - betas_true[i,1])^2

  #Multiple (PMM)
  mult_mse[i] = (betas_mult[i,1] - betas_true[i,1])^2

  #Progress bar
  if(i %% floor(nsim/10) == 0){
    print((i/nsim)*100)
  }
}

#Spit out each of the average MSE's for each type and their SD's
sqrt(mean(mean_mse))
sqrt(mean(rand_mse))
sqrt(mean(randreg_mse))
sqrt(mean(hd_mse))
sqrt(mean(mult_mse))
print("")
sd(mean_mse)
```

```r
sd(rand_mse)
sd(randreg_mse)
sd(hd_mse)
sd(mult_mse)
```