STAT 529 Final Project: Evaluation of Imputation Methods Under Different Missing Data Conditions

Yehchan Yoo

April 2025

Introduction

Missing data is a growing concern in survey research. A 2018 study found that, on average, 38% of data were missing across 119 surveys from countries including the U.S., U.K., and Japan, often due to nonresponse such as skipped sensitive questions or survey dropouts (Dodeen, 2018). This trend has worsened over time, with response rates in epidemiological studies falling from over 90% in the 1950s to under 70% in the 2010s (Morton et al., 2012, pp. 106–107). While online surveys offer convenience and broader reach, they have not reversed this decline, partly due to the resulting oversaturation of surveys and the resulting survey fatigue (Shiyab et al., 2023, pp. 441–442). Even major U.S. government surveys like CPS and NHANES have experienced long-term response rate declines (Czajka & Beyler, 2017, pp. 15–19).

Low response rates don't necessarily invalidate a study, and strategies like incentives or varied survey modes can increase a survey's response rate (Czajka & Beyler, 2017; Shiyab et al., 2023). However, nonresponse often introduces bias, and many statistical models – such as linear regression -— require complete-case datasets (Kuhn & Johnson, 2023, Missing Data section). Simply improving response rates may not eliminate missing data entirely.

To address this, imputation methods are used to estimate missing values based on observed data (Lumley, 2010, pp. 185–186). These include mean, random, and regression imputation (Lohr, 1999, pp. 272–278). Their effectiveness can vary depending on the nature of missingness, typically categorized according to Mack et al. (2018) as:

- **MCAR** — Missingness unrelated to any data.
- **MAR** — Missingness related only to observed data.
- **MNAR** — Missingness related to unobserved data.

Unlike MCAR and MAR, MNAR leads to nonignorable bias and is more difficult to handle statistically (Lohr, 1999, pp. 264–265).

This context leads to my central research question:

> ***Question:*** **How does the performance of different imputation methods vary under MCAR, MAR, and MNAR conditions?**

To explore this, I conducted a simulation study with the following goal:

> ***Goal:*** **Simulate missing data in a given dataset and evaluate the performance of different imputation methods.**

Methods

Dataset

For this simulaton dataset, I used the American Community Survey (ACS) data. Being more specific, I used the 1-year ACS Public Use Microdata Sample (PUMS) for New York state in 2023 (United States Census

Bureau, 2024). I used the ACS PUMS data, because ACS PUMS data provides comprehensive individual-level microdata for custom tabulation and estimation for free – making the data perfect for use in simulation studies like mine (United States Census Bureau, n.d.-b). While the United States Census offers both 1-year and 5-year ACS PUMS data based on the desired window of data collection, I used the 1-year ACS PUMS dataset to better account for the annual nature of ACS surveys (United States Census Bureau, n.d.-b, 2025).

Now, the 1-year ACS PUMS datasets from 2023 were quite huge with the entire U.S. PUMS dataset being 570 megabytes in file size in `.csv` format after compression. Thankfully, ACS provides PUMS data state by state. I aimed to choose a state with which I had background knowledge in and that offered the largest possible PUMS dataset without being too demanding on my computer's processing capacity for data analysis. I eventually settled on using New York state's data, as I had background knowledge in New York state from having lived there in the past and as the New York state had the fourth largest ACS PUMS dataset by state – much larger in file size than most other states but much less in file size than datasets from more populated states like California and Texas (United States Census Bureau, 2024).

Variables

For the simulation study, I decided to focus on the following variables: `VALP` (property value) and `RNTP` (contract rent) (United States Census Bureau, n.d.-a, 2024).

I focused on these variables because the PUMS dataset for New York State used households as the unit of observation. Additionally, New York has long faced a housing shortage, leading to rising rents and increased homelessness throughout the 2010s and 2020s—especially in New York City (Horowitz & Staveski, 2023). Having personally experienced the difficulty of securing adequate housing in New York, I was particularly interested in how missing data might affect key housing variables. Therefore, I conducted a simulation study to examine how different imputation methods influence the estimation of property values and contract rents in New York state under different missingness conditions.

Steps

For each of the target variables (`VALP` and `RNTP`), I took the following steps for the simulation study:

1. I obtained a subset of the ACS NYS PUMS data with non-missing values for the target variable.
2. I then simulated MCAR, MAR, and MNAR for *only* the target variable (as discussed in a later section).
3. From here, I imputed the missing data *only* for the target variable using various imputation methods (as discussed in a later section).
4. Then, I calculated the means and quartiles of the target variable with corresponding standard errors using the `survey` package and the replication weights in the dataset in R (Lumley, 2024).

I should note that, outside of replication weight columns, the dataset had missing values in all columns except `RT` (record type), `SERIALNO` (housing unit/GQ person serial number), `DIVISION` (division code), `PUMA` (Public Use Microdata area code), `REGION` (region), `STATE` (state), `ADJHSG` (adjustment factor for housing dollar amounts), `ADJINC` (adjustment factor for income and earnings), `NP` (number of people in household), `TYPEHUGQ` (type of unit), and `FFSP` (yearly food stamp/SNAP recipiency allocation flag).

However, for each target variable, I avoided imputing other variables to simplify the imputation process and better isolate the effects of each method on that specific variable.

Simulation of Missing Data

As Dodeen (2018) found more than a third of data missing in prevalent surveys acorss the world, I followed on the research and aimed for an *expected* missing rate of 35% each target variable. I also followed the missingness simulation guidelines of Zhang (2021) to simulate MCAR, MAR, and MNAR conditions as usually done in statistical literature:

Missing Completely at Random (MCAR) Each value in the target variable had a 35% chance of being missing, independent of other values in the target variable and independent of other variables. That is, whether a value in the target variable was simulated to be missing follows a Bernoulli distribution with $p = 0.35$.

The code used can be found in the Appendix.

Missing at Random (MAR) Here, the probability of the target variable missing was set based on `RESMODE`, the mode of response (United States Census Bureau, n.d.-a). Past meta-analyses showed that online responses had a 12% lower response rate than non-online responses Shiyab et al. (2023). So, I simulated MAR condition by adjusting the probability of missing to reflect this reality while maintaining an overall expected missing rate of 35%. I should note that, while `RESMODE` had some missing values in the original dataset, it did not have any missing values in the `VALP` and `RNTP` subsets of the data, making MAR simulation based on `RESMODE` possible.

More information on the code and formulas used can be found in the Appendix.

Missing Not at Random (MNAR) In this case, the probability of the target variable was set based on the target variable itself. Peterson et al. (2021, p. 16) found based on the 2020 Household Pulse Survey that the average response rate was higher for higher quartiles of median household income and median house value. Based on this observation, I used the data from Peterson et al. (2021, p. 16) to simulate the trend of higher response rate or (equivalently) lower nonresponse rate for higher values of target variables `VALP` (property value) and `RNTP` (rent price) for their corresponding subsets of ACS data. This simulation was also done as to lead to an overall expected missing/nonresponse rate of 0.35.

Since Peterson et al. (2021, p. 16) lists out the nonresponse rates by quartile, I used the multiple cutoff method to set up the MNAR simulation in which the probability of a response in the target variable becoming missing was conditioned on which quartile the response was based in; it should be noted that such cutoff method is more realistic than the single cutoff method but is less realistic than the percentile or regression-based methods of simulating MNAR (Zhang, 2021, p. 122).

More information on the code and formulas used can be found in the Appendix.

Imputation Methods

The simulation study assesses the following imputation methods:

- **Random imputation:** Each missing value in the target variable is imputed with a randomly selected non-missing value from the same variable (Lohr, 1999, p. 275). All non-missing values in the target variable are equally likely to be selected for imputing each missing value.
- **Nearest neighbor (NN) imputation:** Each missing value in the target variable was imputed using the target variable values from the "nearest" neighbors of the corresponding row (Lohr, 1999, p. 275).
  - Beretta and Santaniello (2016) found that using three neighbors, selected with a relief-based feature selection method, yielded the best performance for nearest neighbor imputation in terms of balancing imputation accuracy and preservation of data structure. Based on this finding, I used three nearest neighbors with relief-based feature selection for the imputation procedure.
  - Since there may be missing values in the selected predictor features, I used a modified version of the Euclidean distance function that upscales non-missing values from other predictors in the presence of missing values in some predictors, as seen in scikit-learn developers (2025).
  - NN imputation is based on the k-nearest neighbors algorithm, which suffers from the curse of dimensionality for high-dimensional data for as few as 10-15 dimensions (Beyer et al., 1999). So, I limited the number of features to 10 for the NN imputation method.
- **Mean Imputation:** Each missing value in the target variable is imputed with the mean of the non-missing values from the same variable (Lohr, 1999, p. 275).

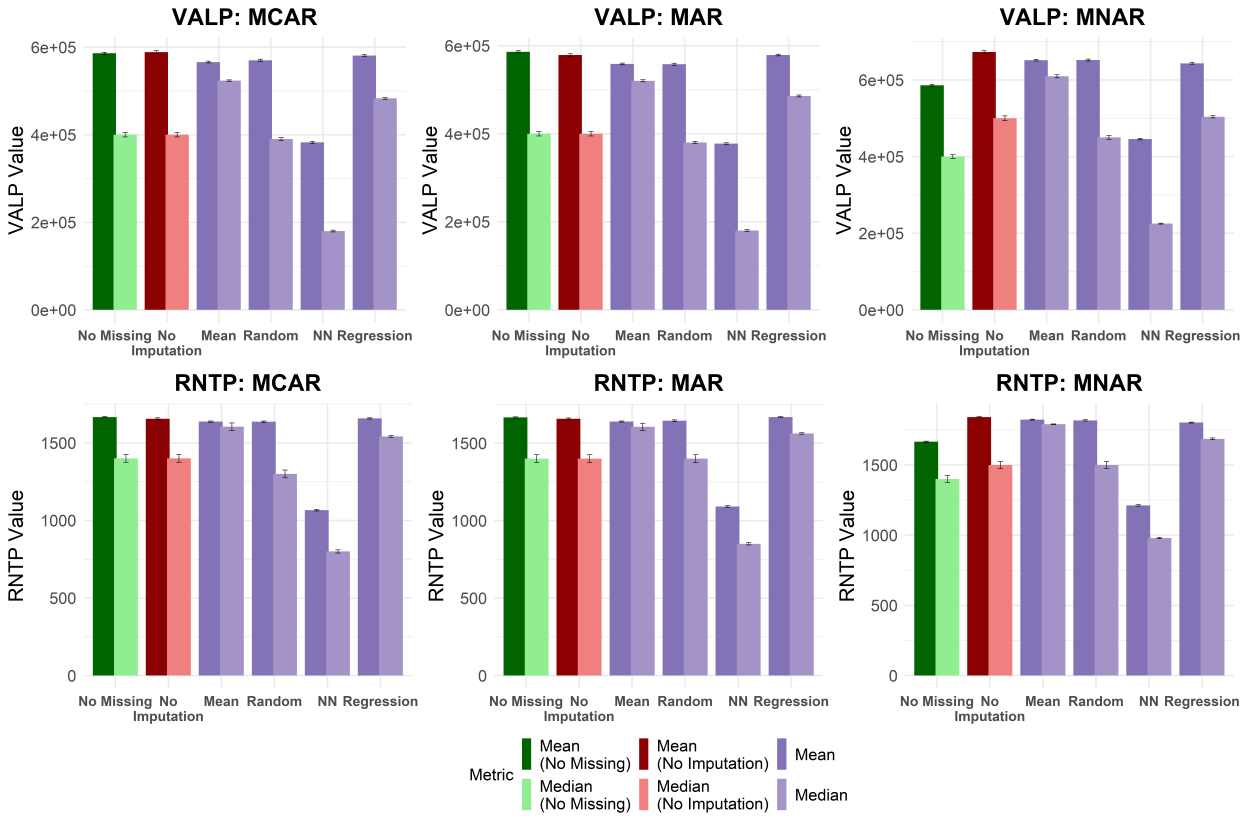


Figure 1: Bar Charts Comparing Mean and Median Estimations Between Imputation Methods
(Horizontal axes show imputation methods; vertical axes show values of the target variables.)
(Note that the bar charts do not include 25th and 75th quantile estimations.)

- **Regression Imputation:** Each missing value in the target variable is imputed using a multivariate regression model of the target variable on all other covariates that have no missing data and are not identifier, weight, adjustment, flag, regional, or Puerto Rico-specific columns (Lohr, 1999; United States Census Bureau, n.d.-a).

The simulation study also has two control group methods:

- **No missing:** This refers to the `VALP`/`RNTP` subset of the ACS dataset *before* simulating missingness.
- **No imputation:** This refers to the `VALP`/`RNTP` subset of the ACS dataset *after* simulating missingness and *before* implementing any imputation method.
  - Means and quartiles for target variables `VALP`/`RNTP` were calculated here by removing rows with missing `VALP`/`RNTP` values.

Results

First, looking at the control group methods and using the "no missing" values as the baseline, I found that not doing any imputation led to either the most accurate or the second most accurate mean and quartile estimates among all methods for both `VALP` and `RNTP` under MCAR and MAR conditions. However, under MNAR condition, not doing any imputation led to a gross overestimation of the means and quartiles for both `VALP` and `RNTP` – performing the second worst for mean estimation and the worst for estimating the 75th quantiles among the six methods. All other imputation methods except the NN method also led to overestimation of the mean, the 25th quantile, and the median for both target variables.

Second, as seen in Figure 1, the NN imputation method performs the worst overall in accuracy and compute time. While other imputation methods took less than a minute to run on my computer with 6-core 10th generation Intel Core i7, the NN imputation method took around one hour to run for each target variable with parallelization. The NN imputation method also generally severely or moderately underestimates the mean and quartiles for both `VALP` and `RNTP` for all randomness conditions. The NN imputation especially underestimates the 25th quantile for both variables under all randomness conditions, estimating this particular quartile to be less than 1 despite the baseline 25th quantile being 200,000 for `VALP` and 880 for `RNTP`.

Third, also as seen in Figure 1, random imputation performed better than all other imputation methods in accuracy for quartile estimates of both target variables. Random imputation's performance especially becomes notable under MNAR condition. While random imputation does lead to a more biased result for quartile estimations than no imputation under MCAR and MAR conditions, random imputation outperforms no imputation for quartile estimations under MNAR condition, even pushing back against the nonresponse bias and overestimating the quartiles *less* than when there is no imputation. However, random imputation does estimate the mean for either target variable less accurately than mean imputation or regression imputation under all missingness conditions, being off from the "no missing" baseline by about 3% under MCAR, about 5% under MAR, and about 11% under MNAR.

Fourth, mean and regression imputation consistently distort quartile estimates for both variables. Mean imputation overestimates the 25th quantile and the median while underestimating the 75th quantile for both target variables under all three missingness conditions. The distortion might be caused by the right-skewedness of the target variables, as seen in the extra histogram outputs in the Appendix. Regression imputation almost always overestimates the quartiles, including the median, for both target variables under all three missingness conditions.

Last but not least, there are a few interesting trends found with the standard errors. Outside of the baseline "no missing" method, having no imputation leads to the highest or the second highest standard error for all estimates for both variables under all missing conditions. The standard error for NN imputation is noticeably low for the 25th quantile and the median estimates for both target variables under all missing conditions, even having standard errors of close to 0 for the 25th quantile estimates. Compared to the NN imputation, other imputation methods generally lead to relatively high standard errors for median estimation. Random imputation especially had the highest standard error for median estimation for both target variables in all missingness conditions with mean imputation often having the next highest standard error. Additionally, except when estimating the 25th quantile for `RNTP`, regression imputation consistently decreases standard error from the baseline "no missing" method, though not as much as the NN imputation.

The detailed numerical results – especially on the 25th and 75th quantiles – can be found in the Appendix.

Discussion/Conclusion

Overall, the results first indicate that no imputation can lead to surprisingly accurate estimates under MCAR and MAR but highly biased estimates under MNAR; having no imputation also generally leads to larger standard errors for estimates than performing any imputation method. For the dataset here, the NN imputation always underestimates the means and quartiles with low standard errors while taking an unusually long period of time to run, making it the worst method for imputation. Random imputation performs the best for quartile estimation, especially under MNAR, though it performs less well than other imputation methods for mean estimation and suffers from high standard errors for median estimation. Mean and regression imputations tend to perform better for mean estimation, and regression imputation especially tends to make estimates with relatively low standard errors. However, mean and regression imputations also do not perform as well on quartile estimates.

There is potential room for improvement here. A future study could potentially look at how different missingness rates affect the quartile estimates and how the imputation methods perform on differently distributed data. A future study could also explore the efficacy of multiple imputation, which is usually the principled standard for missing data imputation (Lumley, 2010, pp. 185-186). While this study attempts to assess the performance of different imputation methods, future studies should continue on this attempt to better assess imputation methods over a more diverse range of conditions.

References

Beretta, L., & Santaniello, A. (2016). Nearest neighbor imputation algorithms: A critical evaluation. *BMC Medical Informatics and Decision Making*, *16*(3), 74. https://doi.org/10.1186/s12911-016-0318-z

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999, March). *When is "nearest neighbour" meaningful?* (Tech. rep.). University of Wisconsin-Madison. https://minds.wisconsin.edu/bitstream/handle/1793/60174/TR1377.pdf

Czajka, J. L., & Beyler, A. (2017). *Declining response rates in federal surveys: Trends and implications* (tech. rep.) (Accessed: 2025-05-21). Office of the Assistant Secretary for Planning, Evaluation, U.S. Department of Health, and Human Services. https://aspe.hhs.gov/sites/default/files/private/pdf/255531/Decliningresponserates.pdf

Dodeen, H. (2018). The prevalence of missing data in survey research. *International Journal for Innovation Education and Research*, *6*. https://doi.org/10.31686/ijier.vol6.iss3.978

Horowitz, A., & Staveski, A. (2023, May). *New york's housing shortage pushes up rents and homelessness*. The Pew Charitable Trusts. Retrieved June 8, 2025, from https://www.pew.org/en/research-and-analysis/articles/2023/05/25/new-yorks-housing-shortage-pushes-up-rents-and-homelessness

Kuhn, M., & Johnson, K. (2023). *Applied Machine Learning for Tabular Data*. Retrieved May 21, 2025, from https://aml4td.org

Lohr, S. L. (1999). *Sampling: Design and analysis*. Duxbury Press.

Lumley, T. (2010). *Complex surveys: A guide to analysis using r: A guide to analysis using r*. John Wiley; Sons.

Lumley, T. (2024). Survey: Analysis of complex survey samples [R package version 4.4].

Mack, C., Su, Z., & Westreich, D. (2018, February). *Types of missing data*. https://www.ncbi.nlm.nih.gov/books/NBK493614/

Morton, S. M., Bandara, D. K., Robinson, E. M., & Carr, P. E. A. (2012). In the 21st century, what is an acceptable response rate? *Australian and New Zealand Journal of Public Health*, *36*(2), 106–108. https://doi.org/https://doi.org/10.1111/j.1753-6405.2012.00854.x

Peterson, S., Toribio, N., Farber, J., & Hornick, D. (2021, March). *Nonresponse bias report for the 2020 household pulse survey* (tech. rep.). U.S. Census Bureau. https://www2.census.gov/programs-surveys/demo/technical-documentation/hhp/2020_HPS_NR_Bias_Report-final.pdf

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning*, *53*(1), 23–69. https://doi.org/10.1023/A:1025667309714

scikit-learn developers. (2025). *Sklearn.metrics.pairwise.nan_euclidean_distances* [Accessed: 2025-05-28]. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.nan_euclidean_distances.html

Shiyab, W., Ferguson, C., Rolls, K., & Halcomb, E. (2023). Solutions to address low response rates in surveys. *European Journal of Cardiovascular Nursing*, *22*(4), 441–444. https://doi.org/10.1093/eurjcn/zvad030

United States Census Bureau. (n.d.-a). Census data api: Variables in /data/2023/acs/acs1/pums/variables. https://api.census.gov/data/2023/acs/acs1/pums/variables.html

United States Census Bureau. (n.d.-b). Public use microdata sample (pums). https://www.census.gov/programs-surveys/acs/microdata.html

United States Census Bureau. (2024). American Community Survey (ACS) 2023 1-Year Public Use Microdata Sample (PUMS) [Accessed: 2025-05-21].

United States Census Bureau. (2025). *The importance of the american community survey and the decennial census* [Accessed: 2025-06-08].

Urbanowicz, R. J., Meeker, M., Cava, W. L., Olson, R. S., & Moore, J. H. (2018). Relief-based feature selection: Introduction and review [Epub 2018 Jul 18]. *Journal of Biomedical Informatics*, *85*, 189–203. https://doi.org/10.1016/j.jbi.2018.07.014

Zhang, X. (2021, July). Tutorial: How to generate missing data for simulation studies [Retrieved from https://doi.org/10.20982/tqmp.19.2.p100]. https://doi.org/10.20982/tqmp.19.2.p100

Appendix

R Code and Formulas

Here is the R code used to perform the simulation study. The R code below are meant to be run in order from top to bottom.

Introduction

The code starts with importing the needed libraries and dataset.

```r
# Import all necessary external libraries
library(survey)
library(dplyr) # For data manipulation
library(tidyr)
library(MASS)
library(RANN)
library(pbapply)
library(progressr)
library(car)
library(caret)
library(mice)
library(parallel)
library(doParallel)
library(foreach)
library(stringr)
library(ggplot2)
library(patchwork)

# Import the ACS NYS data
## Note: Run this code after checking the working directory;
## this code might take a while to run!
csv_pny <- read.csv("psam_h36.csv")
```

Imputation Functions – No Imputation, Mean Imputation, Random Imputation

The next set of code defines the imputation functions. The first two input parameters of all imputation functions are `data` (the dataset in the form of an R `data.frame` object) and `var_name` (the target variable). All imputation functions output a newer version of `data` with missing data all imputed for the target variable.

```r
# data: the dataset
# var_name: name of the target variable

# No imputation
no_imputation <- function(data, var_name) {
  # No changes to the data, just return it
  return(data)
}

# Mean imputation
mean_imputation <- function(data, var_name) {
  data[[var_name]][is.na(data[[var_name]])] <-
    mean(data[[var_name]], na.rm = TRUE)
  return(data)
```

```
}

# Random imputation
random_imputation <- function(data, var_name) {
  v <- data[[var_name]]
  if (any(na <- is.na(v))) {
    o <- v[!na]
    if (length(o) == 0) stop("Cannot impute: all values are NA")
    v[na] <- sample(o, sum(na), replace = TRUE)
    data[[var_name]] <- v
  }
  return(data)
}
```

Imputation Function – Nearest Neighbor Imputation

The following function defines the nearest neighbor imputation with relief-based feature selection based on Robnik-Šikonja and Kononenko (2003). Here, `max_num_of_features` is the maximum number of features to select for the nearest neighbor imputation, `K_relief` is the hyperparameter for relief-based feature selection, and `k_impute` is the number of neighbors to look at for the imputation process. I should note that, while `K_relief` is recommended to be set to 10 according to (Urbanowicz et al., 2018), I reduced it to 5 to reduce the computational intensity of the relief-based feature selection process.

```
k_NN_relief_imputation_parallel <- function(data, var_name,
                                            max_num_of_features = 10,
                                            K_relief = 5, k_impute = 3) {
    ## 0) Set up parallel processing
    ncores <- parallel::detectCores(logical = FALSE) - 1
    if (ncores < 1) ncores <- 1
    cl <- makeCluster(ncores)
    registerDoParallel(cl)
    on.exit({
        stopCluster(cl)
        registerDoSEQ()
    }, add = TRUE)
    print(sprintf("→ Using %d cores for parallel processing.", ncores))

    # 1) Define function for calculating Euclidean distance
    # with missing data in the predictors
    partial_euclidean <- function(x, Y, total_p) {
        m <- nrow(Y)
        dists <- numeric(m)
        for (j in seq_len(m)) {
          xj <- x; yj <- Y[j, ]
          shared <- which(!is.na(xj) & !is.na(yj))
          if (length(shared) == 0) {
            dists[j] <- Inf
          } else {
            dist_sq <- sum((xj[shared] - yj[shared])^2)
            dists[j] <- sqrt((total_p / length(shared)) * dist_sq)
          }
        }
        return(dists)
```

```r
  }
print("1/8) Partial Euclidean distance function defined.")

## 2) Identify continuous numeric predictors
# Finding columns to exclude first:
# Identifier columns
id_cols        <- c("SERIALNO", "RT")
# Replication weight columns
weight_cols    <- grep("^WGTP", names(data),   value = TRUE)
# Adjustment columns
adj_cols       <- c("ADJHSG", "ADJINC")
# Flag columns
flag_cols      <- grep("^F", names(data),      value = TRUE)
# Geographical columns + Puerto Rico-centric columns + the target variable/column
manual_exclude <- c("STATE", "DIVISION", "REGION", "HOTWAT", "PLMPRP", var_name)
# Columns with unusually high amounts of missing data (>90%)
high_missing   <- names(which(colMeans(is.na(data)) > 0.90))

excluded_cols  <- unique(
    c(id_cols, weight_cols, adj_cols, flag_cols,
        high_missing, manual_exclude)
)

# Finding all numeric columns that are not identified for exclusion
all_num_vars   <- names(data)[sapply(data, is.numeric)]
numeric_vars   <- setdiff(all_num_vars, excluded_cols)

# Identifying numeric columns that may be categorical in practice
# by finding numeric columns whose non-missing values are all integers
# and have fewer than 15 distinct non-NA values
is_cat_in_prac <- function(x) {
    is.numeric(x) &&
    all(na.omit(x) %% 1 == 0) &&
    (dplyr::n_distinct(na.omit(x)) < 15)
}
cat_vars       <- names(
    Filter(is_cat_in_prac, data[, numeric_vars, drop = FALSE])
)

# Excluding numeric columns that may be categorical in practice
continuous_vars<- setdiff(numeric_vars, cat_vars)

# Stop the function if no continuous predictor is left
if (length(continuous_vars) == 0) {
    stop("No continuous predictors found after filtering.")
}
print("2/8) Continuous predictors identified.")

## 3) Subset and normalize the remaining predictors
subset_df <- data[, c(var_name, continuous_vars), drop = FALSE]
all_cols  <- c(var_name, continuous_vars)
mins      <- sapply(subset_df[all_cols], function(col) min(col, na.rm = TRUE))
maxs      <- sapply(subset_df[all_cols], function(col) max(col, na.rm = TRUE))
```

```r
denom     <- pmax(maxs - mins, 1e-8)

norm_df <- as.data.frame(
    Map(function(col, mn, d) {(col - mn) / d},
        subset_df[all_cols], mins, denom),
    stringsAsFactors = FALSE
)

Xall <- as.matrix(norm_df[, continuous_vars, drop = FALSE])
Yall <- norm_df[[var_name]]   # may contain NAs
N    <- nrow(Xall)
p    <- ncol(Xall)

print("3/8) Data subsetted and normalized.")

## 4) Nearest-neighbor search for relief-based feature selection
print("4/8) Computing nearest neighbors in parallel...")
Xall_mat <- Xall

fast_partial_euclidean <- function(x_row, Y_mat, n_total) {
    # vectorized version
    shared_mask <- !is.na(x_row) & !is.na(Y_mat)
    sq_diffs <- (matrix(rep(x_row, each = nrow(Y_mat)), nrow = nrow(Y_mat),
                        byrow = FALSE) - Y_mat)^2
    sq_diffs[!shared_mask] <- NA
    num_shared <- rowSums(!is.na(sq_diffs))
    dist_sq    <- rowSums(sq_diffs, na.rm = TRUE)
    adjusted   <- sqrt((n_total / num_shared) * dist_sq)
    adjusted[num_shared == 0] <- Inf
    return(adjusted)
}

# Use pbapply with a cluster argument to show a progress bar + parallel compute
pboptions(type = "timer")  # or "txt" / "tk" as you prefer
nn_list <- pblapply(
    seq_len(N),
    function(i) {
        dists <- fast_partial_euclidean(Xall_mat[i, ], Xall_mat, p)
        dists[i] <- Inf
        order(dists, na.last = TRUE)[seq_len(K_relief)]
    },
    cl = cl
)
nn_idx_matrix <- do.call(rbind, nn_list)
print("4/8) Nearest neighbors computed.")

## 5) Compute |Y_i - Y_neighbor| for relief calculations
Y_neighbors <- matrix(NA_real_, nrow = N, ncol = K_relief)
for (i in seq_len(N)) {
    neigh <- nn_idx_matrix[i, ]
    Y_neighbors[i, ] <- abs(Yall[i] - Yall[neigh])
}
print("5/8) |Y_i - Y_neighbor| precomputed.")
```

```r
## 6) Compute ReliefF scores and select features with lowest scores
print("6/8) Computing ReliefF scores in parallel...")
handlers("progress")

clusterExport(
    cl,
    varlist = c("Xall_mat", "nn_idx_matrix", "Y_neighbors", "N", "K_relief"),
    envir   = environment()
)

relief_vec <- foreach(
    jfeat = seq_len(p),
    .combine  = "c",
    .packages = c("progressr")
) %dopar% {
    total_sum <- 0
    for (i in seq_len(N)) {
        neigh  <- nn_idx_matrix[i, ]
        dX_vec <- abs(Xall_mat[i, jfeat] - Xall_mat[neigh, jfeat])
        dY_vec <- Y_neighbors[i, ]
        valid  <- which(!is.na(dX_vec) & !is.na(dY_vec))
        if (length(valid) > 0) {
            total_sum <- total_sum + sum(dX_vec[valid] * dY_vec[valid])
        }
    }
    prog <- progressor(along = seq_len(p))
    prog(message = sprintf("Feature %d/%d done", jfeat, p))
    total_sum / (N * K_relief)
}

relief_scores <- setNames(relief_vec, continuous_vars)
print("6/8) ReliefF done.")

ranked_feats      <- names(sort(relief_scores, decreasing = FALSE))
selected_features <- head(ranked_feats, max_num_of_features)
print(paste0("   Selected features: ", paste(selected_features, collapse = ", ")))

## 7) Nearest neighbor imputation
print("7/8) Imputing missing target values in parallel...")
rows_missing <- which(is.na(Yall))
rows_obs     <- setdiff(seq_len(N), rows_missing)

# Now that selected_features is defined, build selected_indices:
selected_indices <- match(selected_features, continuous_vars)

X_obs     <- Xall_mat[rows_obs,      selected_indices, drop = FALSE]
Y_obs     <- Yall[rows_obs]
X_mis     <- Xall_mat[rows_missing,  selected_indices, drop = FALSE]
total_p_sel <- length(selected_features)

clusterExport(
    cl,
    varlist = c("X_obs", "Y_obs", "X_mis",
```

```
                    "k_impute", "partial_euclidean", "total_p_sel"),
        envir = environment()
    )

    imputed_vals_list <- pblapply(
        seq_len(nrow(X_mis)),
        function(idx) {
            d_m    <- partial_euclidean(
                        X_mis[idx, , drop = FALSE][1, ],
                        X_obs,
                        total_p_sel
                      )
            ord    <- order(d_m, na.last = TRUE)
            nn_idx <- ord[seq_len(k_impute)]
            mean(Y_obs[nn_idx], na.rm = TRUE)
        },
        cl = cl
    )

    imputed_vals          <- unlist(imputed_vals_list)
    data[[var_name]][rows_missing] <- imputed_vals

    print("7/8) Imputation complete.")

    ## 8) Return imputed dataset
    print("8/8) Returning imputed dataset.")
    return(data)
}
```

Imputation Function – Regression Imputation

The following function defines the regression imputation.

```
regression_imputation <- function(data, var_name){
  ## Find columns to not use for the regression model
  # Identifier columns
  id_cols        <- c("SERIALNO", "RT")
  # Replication weight columns
  weight_cols    <- grep("^WGTP", names(data), value = TRUE)
  # Adjustment columns
  adj_cols       <- c("ADJHSG", "ADJINC")
  # Flag columns
  flag_cols      <- grep("^F", names(data), value = TRUE)
  # Geographical columns + Puerto Rico-centric columns
  manual_exclude <- c("STATE", "DIVISION", "REGION", "HOTWAT", "PLMPRP")
  # Columns with unusually high amounts of missing data (>90%)
  high_missing   <- names(which(colMeans(is.na(data)) > 0.90))
  excluded_cols  <- unique(
    c(id_cols, weight_cols, adj_cols, flag_cols,
      high_missing, manual_exclude)
  )

  # Keep WGTP columns for later use with survey package
```

```r
data_from_weight_cols <- data[weight_cols]

# Exclude the excluded columns
included_cols <- setdiff(names(data), excluded_cols)
data <- data[included_cols]

# Find complete-case columns other than var_name
# for rows with non-missing values in var_name
# & only use those columns + var_name
complete_case_cols <- names(data)[
  sapply(data, function(col) all(!is.na(col)))
]
complete_case_cols <- setdiff(complete_case_cols, var_name)
complete_case_cols <- complete_case_cols[!is.na(complete_case_cols)]
data <- data[c(complete_case_cols, var_name)]

# If there are no complete-case columns, throw an error
if (length(complete_case_cols) == 0) {
  stop("No complete-case columns available for regression imputation.")
}

# Prep for one-hot encoding on categorical variables
# by converting factor col -> char
data <- data %>%
  mutate(across(where(is.factor), as.character))

# Removing columns with only one unique value
data <- data[ , sapply(data, function(col) length(unique(col)) > 1)]

# Looking only at numeric columns
data_numeric_cols <- data[, sapply(data, is.numeric)]
# Removing var_name
data_numeric_cols <- data_numeric_cols[, !names(data_numeric_cols) %in% var_name]
# Removing aliased numeric columns
if (length(data_numeric_cols) > 0){
  alias_cols <- findLinearCombos(data_numeric_cols)
  if (length(alias_cols$remove) > 0) {
    data <- data[, !names(data) %in% alias_cols$remove]
    warning(paste("Dropped columns with linear combinations:",
                  paste(alias_cols$remove, collapse = ", ")))
  }
}

# Check for strong multicollinearity
# and drop columns automatically
vif_values <- vif(lm(as.formula(paste(var_name, "~ .")), data = data))
vif_threshold <- 10
if (any(vif_values > vif_threshold)) {
  high_vif_cols <- names(vif_values[vif_values > vif_threshold])
  data <- data[, !names(data) %in% high_vif_cols]
  if (length(high_vif_cols) != 0) {
    warning(paste("Dropped columns with VIF > ", vif_threshold, ":",
                  paste(high_vif_cols, collapse = ", ")))
```

```
    }
  }

  # If there are no predictors left, throw an error
  if (ncol(data) == 1) {
    stop("No predictors left after filtering. Cannot perform regression imputation.")
  }
  # Fit a linear model
  lm_model <- lm(as.formula(paste(var_name, "~ .")), data = data)
  # Predict missing values
  missing_indices <- is.na(data[[var_name]])
  data[[var_name]][missing_indices] <- predict(lm_model,
                                               newdata = data[missing_indices, ])

  # Add back in weight columns
  data <- cbind(data, data_from_weight_cols)

  return(data)
}
```

Helper Functions for Imputation

Here is the R code defining functions `get_mean_quantiles()` and `get_histogram()` for obtaining the weighted mean, quantiles, and histogram of the target variable from the given data using the `survey` package.

```
# Defining a function for outputting mean and quantiles
# based on survey design and variable name & na_rm status (Default: FALSE)
# (If na_rm is true, the function will remove any missing data in the
# target variable while doing the calculations. Otherwise, the function
# will throw an error in the presence of any such missing data.)
get_mean_quantiles <- function(design, var_name, na_rm = FALSE) {
  mean_estimate <- svymean(as.formula(paste0("~", var_name)),
                           design = design, na.rm = na_rm)
  quantiles_estimate <- svyquantile(as.formula(paste0("~", var_name)),
                                    design = design,
                                    quantiles = c(0.25, 0.5, 0.75),
                                    na.rm = na_rm)

  return(list(mean = mean_estimate, quantiles = quantiles_estimate))
}

# Defining a function for getting histogram
# based on survey design and variable name
get_histogram <- function(design, var_name) {
  svyhist(as.formula(paste0("~", var_name)),
          design = design, breaks = 20,
          main = paste("Histogram of", var_name),
          xlab = var_name)
  }
```

The two functions defined above are used to create another massive helper function to streamline the simulation process. This function runs the given imputation function (`imputation_function`) for the target variable (`var_name`) in a given dataset (`data`). Afterwards, the function prints the summary statistics for

the target variable from the imputed dataset and plots a histogram of the target variable. The summary statistics and histogram are also saved as `.csv` and `.png` files, respectively. At the end, the function ends by invisibly returning the summary statistics.

The `na_rm` parameter determines whether to remove any missing data in the target variable while calculating the weighted means and quantiles. If `na_rm` is set to `FALSE`, the function will throw an error when the imputed data still has missing data in the target variable.

```
impute_calculate_and_visualize_PUMS <- function(data, var_name,
                                                imputation_function,
                                                na_rm = FALSE) {
  # 1) Run the imputation, get the new data frame
  imputed_data <- imputation_function(data, var_name)

  # 1.1) Save imputed data as csv file for backup
  write.csv(imputed_data,
            file = paste0("imputed_", var_name, "_",
                          deparse(substitute(imputation_function)), ".csv"),
            row.names = FALSE)

  # 2) Build the survey design on the imputed data
  acs_design <- svrepdesign(
    data       = imputed_data,
    weights    = ~WGTP,
    repweights = imputed_data[, grep("^WGTP[0-9]+$", names(imputed_data))]
  )

  # 3) Compute mean & quantiles for the target variable
  mean_quantiles <- get_mean_quantiles(
    design   = acs_design,
    var_name = var_name,
    na_rm    = na_rm
  )

  # 3.1) Write summary statistics to a CSV
  #   { mean:
  mean_val <- coef(mean_quantiles$mean)[[var_name]]
  se_mean  <- SE(mean_quantiles$mean)

  #   { quantiles (25th, 50th, 75th) and their SEs:
  q_obj      <- mean_quantiles$quantiles
  q_vals     <- as.numeric(q_obj[[var_name]])       # numeric vector length=3
  se_q_vec   <- as.numeric(SE(q_obj))    # same length=3

  #   Assign to named variables:
  q25  <- q_vals[1];  se_q25  <- se_q_vec[1]
  q50  <- q_vals[2];  se_q50  <- se_q_vec[2]
  q75  <- q_vals[3];  se_q75  <- se_q_vec[3]

  # ----- 5) Build a data frame of summary stats (including SEs) -----
  stats_df <- data.frame(
    variable    = var_name,
    mean        = mean_val,
    se_mean     = se_mean,
    q25         = q25,
```

```
    se_q25      = se_q25,
    q50         = q50,
    se_q50      = se_q50,
    q75         = q75,
    se_q75      = se_q75,
    row.names   = NULL
  )

  stats_csv_name <- paste0("stats_", var_name, "_",
                           deparse(substitute(imputation_function)), ".csv")
  write.csv(stats_df, file = stats_csv_name, row.names = FALSE)
  cat("→ Summary statistics (with SEs) written to:", stats_csv_name, "\n")

  # 3.2) Print the summary statistics to console
  cat("\n### Survey-weighted estimates for", var_name, "###\n")
  cat("Mean =", signif(mean_val, 6), "(SE =", signif(se_mean, 6), ")\n")
  cat("25th pct =", signif(q25, 6), "(SE =", signif(se_q25, 6), ")\n")
  cat("50th pct =", signif(q50, 6), "(SE =", signif(se_q50, 6), ")\n")
  cat("75th pct =", signif(q75, 6), "(SE =", signif(se_q75, 6), ")\n\n")

  # 6) Plot the histogram & save it as png file
  hist_png_name <- paste0("histogram_", var_name, "_",
                          deparse(substitute(imputation_function)), ".png")

  png(filename = hist_png_name,
      width    = 800,
      height   = 600,
      res      = 100)
  get_histogram(acs_design, var_name)
  dev.off()
  cat("→ Histogram saved to:", hist_png_name, "\n")

  # 7) Return the statistics invisibly
  invisible(mean_quantiles)
}
```

Getting Complete Case Subsets for Each Target Variable

Here is the R code for filtering the dataset for rows without missing values in `VALP` and `RNTP`, respectively, and create a complete case dataset for each variable.

```
# Filtering for rows without missing values
csv_pny_VALP_not_missing <-
  csv_pny %>%
  filter(!is.na(VALP))
csv_pny_RNTP_not_missing <-
  csv_pny %>%
  filter(!is.na(RNTP))
```

Missing Completely at Random (MCAR)

Here is the R code for simulating MCAR on each dataset by having each cell in the target variable become missing with a chance of 35%.

```r
# Simulating MCAR condition on each of these filtered datasets
# -- simulating 35% missingness
missing_prop <- 0.35

## VALP
csv_pny_VALP_mcar <- csv_pny_VALP_not_missing
csv_pny_VALP_mcar$VALP <- ifelse(
  runif(nrow(csv_pny_VALP_not_missing)) < missing_prop,
  NA,
  csv_pny_VALP_not_missing$VALP
)

## RNTP
csv_pny_RNTP_mcar <- csv_pny_RNTP_not_missing
csv_pny_RNTP_mcar$RNTP <- ifelse(
  runif(nrow(csv_pny_RNTP_not_missing)) < missing_prop,
  NA,
  csv_pny_RNTP_not_missing$RNTP
)

## Just in case, save the MCAR files as csv files
write.csv(csv_pny_VALP_mcar,
          file = "csv_pny_VALP_mcar.csv",
          row.names = FALSE)
write.csv(csv_pny_RNTP_mcar,
          file = "csv_pny_RNTP_mcar.csv",
          row.names = FALSE)
```

From here, the following R code runs the simulations and save the imputed datasets, the statistical summaries (with means, quartiles, and their standard errors), and the histograms as `.csv` or `.png` files. After the output files were generated, I moved these files to the `results_MCAR` directory for organized safekeeping. Just in case, I also saved the summary statistics in the R environment.

```r
# No imputation
no_imputation_VALP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mcar,
                                                               "VALP", no_imputation,
                                                               na_rm = TRUE)
no_imputation_RNTP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mcar,
                                                               "RNTP", no_imputation,
                                                               na_rm = TRUE)

# Mean imputation
mean_imputation_VALP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mcar,
                                                                 "VALP",
                                                                 mean_imputation)
mean_imputation_RNTP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mcar,
                                                                 "RNTP",
                                                                 mean_imputation)

# Random imputation
random_imputation_VALP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mcar,
                                                                   "VALP",
                                                                   random_imputation)
random_imputation_RNTP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mcar,
```

```
                                                                "RNTP",
                                                                random_imputation)

# Nearest neighbor imputation
kNN_imputation_VALP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mcar,
                                                                "VALP",
                                                                k_NN_relief_imputation_parallel)
kNN_imputation_RNTP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mcar,
                                                                "RNTP",
                                                                k_NN_relief_imputation_parallel)

# Regression imputation
regression_imputation_VALP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mcar,
                                                                       "VALP",
                                                                       regression_imputation)
regression_imputation_RNTP_mcar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mcar,
                                                                       "RNTP",
                                                                       regression_imputation)
```

Missing at Random (MAR)

Formulas First, let's define some variables. For each target variable, let

- p_{off} be the proportion of offline responses,
- $p_{\text{on}} = 1 - p_{\text{off}}$ be the proportion of online responses,
- r_{on} denote the simulated response rate for an online response,
- r_{off} denote the simulated response rate for an offline response,
- m_{on} be the simulated missingness/nonresponse rate for an online response, and
- m_{off} be the simulated missingness/nonresponse rate for an online response.

I found that Shiyab et al. (2023) did not specify whether the intended relationship in response rate between online and offline responses was (1) $r_{\text{on}} = (1 - 0.12)r_{\text{off}}$, or (2) $r_{\text{on}} = r_{\text{off}} - 0.12$. While both interpretations are plausible, I adopted interpretation (1) for the simulation. This formulation enabled me to scale the missingness probabilities in a way that achieved an overall expected missing rate of 0.35, even without knowing the exact values of r_{on} and r_{off}, which were not provided. Interpretation (2), while perhaps more intuitively suggested by the language in the original source, would not have worked in this context, as it does not allow the probabilities to scale proportionally without knowing the original values of r_{on} and r_{off}.

For each target variable subset of the ACS NYS data, I calculated r_{off} as follows to allow for an expected response rate of 0.65 or (equivalently) an expected nonresponse/missingness rate of 0.35:

$$\begin{aligned} r_{\text{on}}p_{\text{on}} + r_{\text{off}}p_{\text{off}} &= 1 - 0.35 \\ 0.88r_{\text{off}}(1 - p_{\text{off}}) + r_{\text{off}}p_{\text{off}} &= 0.65 \\ r_{\text{off}}(0.88 + 0.12r_{\text{off}}) &= 0.65 \\ r_{\text{off}} &= \boxed{\frac{0.65}{0.88 + 0.12p_{\text{off}}}} \end{aligned}$$

From here, I let r_{on} equal $\boxed{0.88r_{\text{off}}}$.

Then, I calculated the missingness/nonresponse rates by having $\boxed{m_{\text{on}} = 1 - r_{\text{on}}}$ and $\boxed{m_{\text{off}} = 1 - r_{\text{off}}}$.

R Code Here is the R code for calculating the missingness/nonresponse rates and using those rates to simulate the MAR condition. Note that, in this code, we have:

- p_{off} corresponding to `prop_VALP_not_internet` for `VALP` and `prop_RNTP_not_internet` for `RNTP`
- p_{on} corresponding to `1 - prop_VALP_not_internet` for `VALP` and `1 - prop_RNTP_not_internet` for `RNTP`
- r_{on} correspnding to `not_missing_prop_internet_VALP` for `VALP` and `not_missing_prop_internet_RNTP` for `RNTP`
- r_{off} correspnding to `not_missing_prop_not_internet_VALP` for `VALP` and `not_missing_prop_not_internet_RNTP` for `RNTP`
- m_{on} correspnding to `missing_prop_internet_VALP` for `VALP` and `missing_prop_internet_RNTP` for `RNTP`
- m_{off} correspnding to `missing_prop_not_internet_VALP` for `VALP` and `missing_prop_not_internet_RNTP` for `RNTP`

```
missing_prop <- 0.35
non_missing_prop <- 1 - missing_prop

# Looking at proportions of RESMODE
props_RNTP <- prop.table(table(csv_pny_RNTP_not_missing$RESMODE))
props_VALP <- prop.table(table(csv_pny_VALP_not_missing$RESMODE))
props_RNTP
props_VALP

prop_RNTP_not_internet <- 1 - props_RNTP[[3]]
prop_VALP_not_internet <- 1 - props_VALP[[3]]

c_val <- 0.88 # 12% less response in internet mode

# Setting probability of missingness for each group
not_missing_prop_not_internet_RNTP <- non_missing_prop / (c_val + (1 - c_val) *
                                                          prop_RNTP_not_internet)
not_missing_prop_not_internet_VALP <- non_missing_prop / (c_val + (1 - c_val) *
                                                          prop_VALP_not_internet)
not_missing_prop_internet_RNTP     <- not_missing_prop_not_internet_RNTP * c_val
not_missing_prop_internet_VALP     <- not_missing_prop_not_internet_VALP * c_val

missing_prop_not_internet_RNTP <- 1 - not_missing_prop_not_internet_RNTP
missing_prop_not_internet_VALP <- 1 - not_missing_prop_not_internet_VALP
missing_prop_internet_RNTP     <- 1 - not_missing_prop_internet_RNTP
missing_prop_internet_VALP     <- 1 - not_missing_prop_internet_VALP

#  Now draw ONE uniform per row, instead of two separate runif() calls

# 1) RNTP
csv_pny_RNTP_mar <- csv_pny_RNTP_not_missing

# draw one U(0,1) per row:
u_R <- runif(nrow(csv_pny_RNTP_not_missing))
```

```r
csv_pny_RNTP_mar$RNTP <- ifelse(
  csv_pny_RNTP_mar$RESMODE == 3,
  ifelse(
    u_R < missing_prop_internet_RNTP,    # use the same u_R vector
    NA,
    csv_pny_RNTP_not_missing$RNTP
  ),
  ifelse(
    u_R < missing_prop_not_internet_RNTP,  # and here too
    NA,
    csv_pny_RNTP_not_missing$RNTP
  )
)

# 2) VALP
csv_pny_VALP_mar <- csv_pny_VALP_not_missing

# draw one U(0,1) per row (fresh draw):
u_V <- runif(nrow(csv_pny_VALP_not_missing))

csv_pny_VALP_mar$VALP <- ifelse(
  csv_pny_VALP_mar$RESMODE == 3,
  ifelse(
    u_V < missing_prop_internet_VALP,    # use u_V here
    NA,
    csv_pny_VALP_not_missing$VALP
  ),
  ifelse(
    u_V < missing_prop_not_internet_VALP,  # and here
    NA,
    csv_pny_VALP_not_missing$VALP
  )
)

## Just in case, save the MAR files as csv files
write.csv(csv_pny_VALP_mar,
          file = "csv_pny_VALP_mar.csv",
          row.names = FALSE)
write.csv(csv_pny_RNTP_mar,
          file = "csv_pny_RNTP_mar.csv",
          row.names = FALSE)
```

From here, the following R code runs the simulations and save the imputed datasets, the statistical summaries (with means, quartiles, and their standard errors), and the histograms as `.csv` or `.png` files. After the output files were generated, I moved these files to the `results_MAR` directory for organized safekeeping. Just in case, I also saved the summary statistics in the R environment.

```r
# No imputation
no_imputation_VALP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mar,
                                                              "VALP", no_imputation,
                                                              na_rm = TRUE)
no_imputation_RNTP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mar,
                                                              "RNTP", no_imputation,
                                                              na_rm = TRUE)
```

```r
# Mean imputation
mean_imputation_VALP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mar,
                                                                "VALP",
                                                                mean_imputation)
mean_imputation_RNTP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mar,
                                                                "RNTP",
                                                                mean_imputation)

# Random imputation
random_imputation_VALP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mar,
                                                                  "VALP",
                                                                  random_imputation)
random_imputation_RNTP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mar,
                                                                  "RNTP",
                                                                  random_imputation)

# Nearest neighbor imputation
kNN_imputation_VALP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mar,
                                                               "VALP",
                                                               k_NN_relief_imputation_parallel)
kNN_imputation_RNTP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mar,
                                                               "RNTP",
                                                               k_NN_relief_imputation_parallel)

# Regression imputation
regression_imputation_VALP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mar,
                                                                      "VALP",
                                                                      regression_imputation)
regression_imputation_RNTP_mar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mar,
                                                                      "RNTP",
                                                                      regression_imputation)
```

Missing Not at Random (MNAR)

Formulas Let's initialize some variables. Let:

- r_1 denote the simulated response rate for the 1st quartile of the target variable,
- r_2 denote the simulated response rate for the 2nd quartile of the target variable,
- r_3 denote the simulated response rate for the 3rd quartile of the target variable,
- r_4 denote the simulated response rate for the 4th quantile of the target variable,
- m_1 be the simulated missingness/nonresponse rate for the 1st quartile,
- m_2 be the simulated missingness/nonresponse rate for the 2nd quartile,
- m_3 be the simulated missingness/nonresponse rate for the 3rd quartile, and
- m_4 be the simulated missingness/nonresponse rate for the 4th quantile.

From here, I used the average response rates provided by Peterson et al. (2021, p. 16) to simulate MNAR based on the values of the target variable (`VALP`, `RNTP`).

Domain	Average Response Rate			
	Quartile 1	**Quartile 2**	**Quartile 3**	**Quartile 4**
Median household income	5.0%	6.0%	6.9%	8.4%
Median house value	5.0%	6.1%	7.2%	8.3%
Mean response rate (across both domains)	5.0%	6.05%	7.05%	8.35%

Table 1: Comparison of Median Household Income and House Value (Peterson et al., 2021, p. 16)

As seen in Table 1, I calculated the mean response rate for each quartile from the two domains (median household income, median house value) and then obtained the value for r_1 as to get an expected overall missing/nonresponse rate of 0.35 from scaling up the mean response rates proportionally:

$$\begin{aligned}
\frac{1}{4}(r_1 + r_2 + r_3 + r_4) &= 1 - 0.35 \\
\frac{1}{4}\left(r_1 + \frac{6.05}{5}r_1 + \frac{7.05}{5}r_1 + \frac{8.35}{5}r_1\right) &= 0.65 \\
\left(\frac{1}{4}\right)\left(1 + \frac{6.05}{5} + \frac{7.05}{5} + \frac{8.35}{5}\right)r_1 &= 0.65 \\
r_1 &= \boxed{\frac{4 \times 0.65}{1 + \frac{6.05}{5} + \frac{7.05}{5} + \frac{8.35}{5}}}
\end{aligned}$$

From here, I calculated the other simulated response rates by having $r_2 = \frac{6.05}{5}r_1$, $r_3 = \frac{7.05}{5}r_1$, and $r_4 = \frac{8.35}{5}r_1$.

Then, I calculated the simulated missingness/nonresponse rates by having $m_i = 1 - r_i$ for all $i = \{1, 2, 3, 4\}$.

R Code Here is the R code for calculating the missingness/nonresponse rates and using those rates to simulate the MNAR condition. Note that, in this code, we have:

- r_1 corresponding to `non_missing_rate_for_first_quartile`,
- r_2 corresponding to `non_missing_rate_for_second_quartile`,
- r_3 corresponding to `non_missing_rate_for_third_quartile`,
- r_4 corresponding to `non_missing_rate_for_fourth_quartile`,
- m_1 corresponding to `missing_rate_for_first_quartile`,
- m_2 corresponding to `missing_rate_for_second_quartile`,
- m_3 corresponding to `missing_rate_for_third_quartile`, and
- m_4 corresponding to `missing_rate_for_fourth_quartile`.

```
missing_prop <- 0.35
non_missing_prop <- 1 - missing_prop

non_missing_rates_ratio <- c(1, 6.05/5, 7.05/5, 8.35/5)

non_missing_rate_for_first_quartile <- 4 * non_missing_prop / sum(non_missing_rates_ratio)
non_missing_rate_for_second_quartile <- non_missing_rate_for_first_quartile *
                                        non_missing_rates_ratio[2]
```

```r
non_missing_rate_for_third_quartile <- non_missing_rate_for_first_quartile *
                                       non_missing_rates_ratio[3]
non_missing_rate_for_fourth_quartile <- non_missing_rate_for_first_quartile *
                                       non_missing_rates_ratio[4]

missing_rate_for_first_quartile <- 1 - non_missing_rate_for_first_quartile
missing_rate_for_second_quartile <- 1 - non_missing_rate_for_second_quartile
missing_rate_for_third_quartile <- 1 - non_missing_rate_for_third_quartile
missing_rate_for_fourth_quartile <- 1 - non_missing_rate_for_fourth_quartile

#  Now draw ONE uniform per row, instead of two separate runif() calls
# 1) RNTP
csv_pny_RNTP_mnar <- csv_pny_RNTP_not_missing
u_R <- runif(nrow(csv_pny_RNTP_not_missing))
csv_pny_RNTP_mnar$RNTP <- ifelse(
  csv_pny_RNTP_mnar$RNTP <= quantile(csv_pny_RNTP_not_missing$RNTP, 0.25),
  ifelse(u_R < missing_rate_for_first_quartile, NA, csv_pny_RNTP_not_missing$RNTP),
  ifelse(
    csv_pny_RNTP_mnar$RNTP <= quantile(csv_pny_RNTP_not_missing$RNTP, 0.5),
    ifelse(u_R < missing_rate_for_second_quartile, NA, csv_pny_RNTP_not_missing$RNTP),
    ifelse(
      csv_pny_RNTP_mnar$RNTP <= quantile(csv_pny_RNTP_not_missing$RNTP, 0.75),
      ifelse(u_R < missing_rate_for_third_quartile, NA, csv_pny_RNTP_not_missing$RNTP),
      ifelse(u_R < missing_rate_for_fourth_quartile, NA, csv_pny_RNTP_not_missing$RNTP)
    )
  )
)
# 2) VALP
csv_pny_VALP_mnar <- csv_pny_VALP_not_missing
u_V <- runif(nrow(csv_pny_VALP_not_missing))
csv_pny_VALP_mnar$VALP <- ifelse(
  csv_pny_VALP_mnar$VALP <= quantile(csv_pny_VALP_not_missing$VALP, 0.25),
  ifelse(u_V < missing_rate_for_first_quartile, NA, csv_pny_VALP_not_missing$VALP),
  ifelse(
    csv_pny_VALP_mnar$VALP <= quantile(csv_pny_VALP_not_missing$VALP, 0.5),
    ifelse(u_V < missing_rate_for_second_quartile, NA, csv_pny_VALP_not_missing$VALP),
    ifelse(
      csv_pny_VALP_mnar$VALP <= quantile(csv_pny_VALP_not_missing$VALP, 0.75),
      ifelse(u_V < missing_rate_for_third_quartile, NA, csv_pny_VALP_not_missing$VALP),
      ifelse(u_V < missing_rate_for_fourth_quartile, NA, csv_pny_VALP_not_missing$VALP)
    )
  )
)

## Just in case, save the MNAR files as csv files
write.csv(csv_pny_VALP_mnar,
          file = "csv_pny_VALP_mnar.csv",
          row.names = FALSE)
write.csv(csv_pny_RNTP_mnar,
          file = "csv_pny_RNTP_mnar.csv",
          row.names = FALSE)
```

From here, the following R code runs the simulations and save the imputed datasets, the statistical summaries (with means, quartiles, and their standard errors), and the histograms as `.csv` or `.png` files.

After the output files were generated, I moved these files to the `results_MNAR` directory for organized safekeeping. Just in case, I also saved the summary statistics in the R environment.

```
# No imputation
no_imputation_VALP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mnar,
                                                               "VALP", no_imputation,
                                                               na_rm = TRUE)
no_imputation_RNTP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mnar,
                                                               "RNTP", no_imputation,
                                                               na_rm = TRUE)

# Mean imputation
mean_imputation_VALP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mnar,
                                                                 "VALP",
                                                                 mean_imputation)
mean_imputation_RNTP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mnar,
                                                                 "RNTP",
                                                                 mean_imputation)

# Random imputation
random_imputation_VALP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mnar,
                                                                   "VALP",
                                                                   random_imputation)
random_imputation_RNTP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mnar,
                                                                   "RNTP",
                                                                   random_imputation)

# Nearest neighbor imputation
kNN_imputation_VALP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mnar,
                                                                   "VALP",
                                                                   k_NN_relief_imputation_parallel)
kNN_imputation_RNTP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mnar,
                                                                   "RNTP",
                                                                   k_NN_relief_imputation_parallel)

# Regression imputation
regression_imputation_VALP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_VALP_mnar,
                                                                       "VALP",
                                                                       regression_imputation)
regression_imputation_RNTP_mnar <- impute_calculate_and_visualize_PUMS(csv_pny_RNTP_mnar,
                                                                       "RNTP",
                                                                       regression_imputation)
```

Creating Tables of Means, Quartiles, and SEs

The following R code creates an aggregated table of mean and quartile estimates (and their standard errors) for both target variables (`RNTP`, `VALP`) for each missingness condition (MCAR, MAR, MNAR). Each table is saved as a `.csv` file; and the three tables for the three missingness conditions are saved as `data_dfs`, a list of three `data.frame` objects.

```
# Defining data_dfs
# and making a list to match method marked in filename
# to the actual method
```

```r
data_dfs <- list()
filename_to_imputation_method <- list(
  "k" = "Nearest Neighbor Imputation",
  "mean" = "Mean Imputation",
  "random" = "Random Imputation",
  "no" = "No Imputation",
  "regression" = "Regression Imputation"
)

### Making VALP and RNTP mean and quartile estimates
### for no missing data situations
## Defining survey object
acs_design <- svrepdesign(
  data = csv_pny,
  weights = ~WGTP,
  repweights = csv_pny[, grep("^WGTP[0-9]+$", names(csv_pny))]
)
## VALP
# Mean estimate, removing NA values
VALP_mean <- svymean(~VALP, design = acs_design, na.rm = TRUE)
# Quartiles, i.e. 25th, 50th, and 75th percentiles
# removing NA values
VALP_quantile <- svyquantile(~VALP, design = acs_design,
            quantiles = c(0.25, 0.5, 0.75), na.rm = TRUE)
VALP_quantile_values <- as.numeric(VALP_quantile$VALP)
VALP_quantile_SEs <- as.numeric(SE(VALP_quantile))
## RNTP
# Mean estimate, removing NA values
RNTP_mean <- svymean(~RNTP, design = acs_design, na.rm = TRUE)
# Quartiles, i.e. 25th, 50th, and 75th percentiles
# removing NA values
RNTP_quantile <- svyquantile(~RNTP, design = acs_design,
            quantiles = c(0.25, 0.5, 0.75), na.rm = TRUE)
RNTP_quantile_values <- as.numeric(RNTP_quantile$RNTP)
RNTP_quantile_SEs <- as.numeric(SE(RNTP_quantile))
# "No missing" row
no_missing_row_df <- data.frame(
  "Imputation.Method" = "No Missing",
  "VALP.Mean" = VALP_mean[1],
  "VALP.Mean.SE" = SE(VALP_mean),
  "VALP.25th" = VALP_quantile_values[1],
  "VALP.25th.SE" = VALP_quantile_SEs[1],
  "VALP.Median" = VALP_quantile_values[2],
  "VALP.Median.SE" = VALP_quantile_SEs[2],
  "VALP.75th" = VALP_quantile_values[3],
  "VALP.75th.SE" = VALP_quantile_SEs[3],
  "RNTP.Mean" = RNTP_mean[1],
  "RNTP.Mean.SE" = SE(RNTP_mean),
  "RNTP.25th" = RNTP_quantile_values[1],
  "RNTP.25th.SE" = RNTP_quantile_SEs[1],
  "RNTP.Median" = RNTP_quantile_values[2],
  "RNTP.Median.SE" = RNTP_quantile_SEs[2],
  "RNTP.75th" = RNTP_quantile_values[3],
```

```
  "RNTP.75th.SE" = RNTP_quantile_SEs[3]
)

for(missing_type in c("MAR", "MNAR", "MCAR")){
  missing_type_df <- data.frame(
    "Imputation.Method" = c("No Imputation", "Mean Imputation", "Random Imputation",
                            "Nearest Neighbor Imputation", "Regression Imputation"),
    "VALP.Mean" = rep(NA,5),
    "VALP.Mean.SE" = rep(NA,5),
    "VALP.25th" = rep(NA,5),
    "VALP.25th.SE" = rep(NA,5),
    "VALP.Median" = rep(NA,5),
    "VALP.Median.SE" = rep(NA,5),
    "VALP.75th" = rep(NA,5),
    "VALP.75th.SE" = rep(NA,5),
    "RNTP.Mean" = rep(NA,5),
    "RNTP.Mean.SE" = rep(NA,5),
    "RNTP.25th" = rep(NA,5),
    "RNTP.25th.SE" = rep(NA,5),
    "RNTP.Median" = rep(NA,5),
    "RNTP.Median.SE" = rep(NA,5),
    "RNTP.75th" = rep(NA,5),
    "RNTP.75th.SE" = rep(NA,5)
  )
  folder_path <- paste0("results_", missing_type, "/")
  files <- list.files(folder_path, pattern = paste0("^stats_.*\\.csv$"),
                      full.names = TRUE)
  for(target_column in c("RNTP", "VALP")){
    target_files <- grep(target_column, files, value = TRUE)
    for(file in target_files){
      df <- read.csv(file)
      imputation_method <- filename_to_imputation_method[[sub(paste0(".*", target_column,
                                                                     "_([^_]+).*"), "\\1",
                                                              file)]]

      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".Mean")] <- df$mean
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".Mean.SE")] <- df$se_mean
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".25th")] <- df$q25
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".25th.SE")] <- df$se_q25
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".Median")] <- df$q50
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".Median.SE")] <- df$se_q50
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".75th")] <- df$q75
      missing_type_df[missing_type_df$Imputation.Method == imputation_method,
                                  paste0(target_column, ".75th.SE")] <- df$se_q75
    }
  }
```

```
  missing_type_df <- rbind(no_missing_row_df, missing_type_df)
  data_dfs[[missing_type]] <- missing_type_df
}

# Export data_dfs as file(s)
write.csv(data_dfs$MAR, file = "data_MAR.csv", row.names = FALSE)
write.csv(data_dfs$MNAR, file = "data_MNAR.csv", row.names = FALSE)
write.csv(data_dfs$MCAR, file = "data_MCAR.csv", row.names = FALSE)
```

Creating Figure 1

Here is the code used to generate Figure 1 and save it as a `.png` file.

```
# Function for cleaning labels
clean_labels <- function(label) {
  if (label == "No Imputation") {
    return("No\nImputation")
  } else if (label == "Nearest Neighbor Imputation") {
    return("NN")
  } else {
    return(str_replace(label, " Imputation", ""))
  }
}

# Custom theme for poster clarity
poster_theme <- theme_minimal(base_size = 28) +
  theme(
    axis.title    = element_text(size = 35),
    axis.text     = element_text(size = 28),
    plot.title    = element_text(size = 40, face = "bold", hjust = 0.5),
    legend.title  = element_text(size = 28),
    legend.text   = element_text(size = 28),
    axis.text.x   = element_text(size = 24, face = "bold")
  )

# Plot storage lists
valp_plots <- list()
rntp_plots <- list()

# Loop through each dataset and generate plots
i <- 1
for (name in names(data_dfs)) {
  data_df <- as_tibble(data_dfs[[name]])
  desired_order <- c(
    "No Missing",
    "No Imputation",
    "Mean Imputation",
    "Random Imputation",
    "Nearest Neighbor Imputation",
    "Regression Imputation"
  )

  #  VALP
```

```r
valp_long <- bind_rows(
  data_df %>%
    dplyr::select(Imputation.Method, Value = VALP.Mean, SE = VALP.Mean.SE) %>%
    mutate(Metric = "Mean"),
  data_df %>%
    dplyr::select(Imputation.Method, Value = VALP.Median, SE = VALP.Median.SE) %>%
    mutate(Metric = "Median")
) %>%
  mutate(
    Imputation.Method = factor(Imputation.Method, levels = desired_order),
    Metric = factor(Metric, levels = c("Mean", "Median"))
  ) %>%
  mutate(
    Imputation.Label = sapply(as.character(Imputation.Method), clean_labels),
    Imputation.Label = factor(
      Imputation.Label,
      levels = sapply(desired_order, clean_labels)
    ),
    FillGroup = case_when(
      Imputation.Method == "No Missing"    ~ paste0("NoMissing_", Metric),
      Imputation.Method == "No Imputation" ~ paste0("NoImputation_", Metric),
      TRUE                                 ~ as.character(Metric)
    ),
    FillGroup = factor(
      FillGroup,
      levels = c(
        "NoMissing_Mean", "NoMissing_Median",
        "NoImputation_Mean", "NoImputation_Median",
        "Mean", "Median"
      )
    )
  )

p_valp <- ggplot(valp_long, aes(x = Imputation.Label, y = Value, fill = FillGroup)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8)) +
  geom_errorbar(
    aes(ymin = Value - SE, ymax = Value + SE),
    position = position_dodge(width = 0.8),
    width = 0.2
  ) +
  scale_fill_manual(
    values = c(
      "NoMissing_Mean"      = "darkgreen",
      "NoMissing_Median"    = "lightgreen",
      "NoImputation_Mean"   = "darkred",
      "NoImputation_Median" = "lightcoral",
      "Mean"                = "#8372B5",
      "Median"              = "#A493C6"
    ),
    labels = c(
      "NoMissing_Mean"      = "Mean\n(No Missing)",
      "NoMissing_Median"    = "Median\n(No Missing)",
      "NoImputation_Mean"   = "Mean\n(No Imputation)",
```

```
      "NoImputation_Median" = "Median\n(No Imputation)",
      "Mean"                = "Mean",
      "Median"              = "Median"
    )
  ) +
  labs(
    title = paste("VALP:", name),
    x    = NULL,
    y    = "VALP Value",
    fill = "Metric"
  ) +
  poster_theme +
  theme(legend.position = "none")   # suppress individual legend

#  RNTP
rntp_long <- bind_rows(
  data_df %>%
    dplyr::select(Imputation.Method, Value = RNTP.Mean, SE = RNTP.Mean.SE) %>%
    mutate(Metric = "Mean"),
  data_df %>%
    dplyr::select(Imputation.Method, Value = RNTP.Median, SE = RNTP.Median.SE) %>%
    mutate(Metric = "Median")
) %>%
  mutate(
    Imputation.Method = factor(Imputation.Method, levels = desired_order),
    Metric = factor(Metric, levels = c("Mean", "Median"))
  ) %>%
  mutate(
    Imputation.Label = sapply(as.character(Imputation.Method), clean_labels),
    Imputation.Label = factor(
      Imputation.Label,
      levels = sapply(desired_order, clean_labels)
    ),
    FillGroup = case_when(
      Imputation.Method == "No Missing"    ~ paste0("NoMissing_", Metric),
      Imputation.Method == "No Imputation" ~ paste0("NoImputation_", Metric),
      TRUE                                 ~ as.character(Metric)
    ),
    FillGroup = factor(
      FillGroup,
      levels = c(
        "NoMissing_Mean", "NoMissing_Median",
        "NoImputation_Mean", "NoImputation_Median",
        "Mean", "Median"
      )
    )
  )

p_rntp <- ggplot(rntp_long, aes(x = Imputation.Label, y = Value, fill = FillGroup)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8)) +
  geom_errorbar(
    aes(ymin = Value - SE, ymax = Value + SE),
    position = position_dodge(width = 0.8),
```

```
      width = 0.2
    ) +
    scale_fill_manual(
      values = c(
        "NoMissing_Mean"      = "darkgreen",
        "NoMissing_Median"    = "lightgreen",
        "NoImputation_Mean"   = "darkred",
        "NoImputation_Median" = "lightcoral",
        "Mean"                = "#8372B5",
        "Median"              = "#A493C6"
      ),
      labels = c(
        "NoMissing_Mean"      = "Mean\n(No Missing)",
        "NoMissing_Median"    = "Median\n(No Missing)",
        "NoImputation_Mean"   = "Mean\n(No Imputation)",
        "NoImputation_Median" = "Median\n(No Imputation)",
        "Mean"                = "Mean",
        "Median"              = "Median"
      )
    ) +
    labs(
      title = paste("RNTP:", name),
      x    = NULL,
      y    = "RNTP Value",
      fill = "Metric"
    ) +
    poster_theme +
    theme(legend.position = "none")   # suppress individual legend

  valp_plots[[i]] <- p_valp
  rntp_plots[[i]] <- p_rntp
  i <- i + 1
}

# Ensure we have 3 plots each
if (length(valp_plots) < 3 || length(rntp_plots) < 3) {
  stop("`valp_plots` and `rntp_plots` must each have at least 3 elements.")
}

# Combine into 2-row × 3-column layout with a single, shared legend
final_plot <- (valp_plots[[3]] | valp_plots[[1]] | valp_plots[[2]]) /
              (rntp_plots[[3]] | rntp_plots[[1]] | rntp_plots[[2]]) +
              plot_layout(guides = "collect") &
              theme(
                legend.position   = "bottom",
                axis.title.x      = element_blank()
              )

# Show it on console
print(final_plot)

# Save high-res for poster
ggsave("combined_imputation_comparison.png", final_plot,
```

```
        width = 30, height = 20, dpi = 300)
```

Numerical Results

Here are tables of the numerical results from the simulation study with all values rounded to the nearest integer.

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	586,079 (2,363)	200,000 (2,512)	400,000 (5,024)	700,000 (6,280)
No Imputation	588,699 (3,090)	200,000 (2,512)	400,000 (5,024)	700,000 (6,280)
Mean Imputation	565,964 (2,000)	300,000 (5,024)	523,780 (1,562)	525,000 (4,074)
Random Imputation	570,015 (2,377)	180,000 (1,256)	390,000 (3,768)	700,000 (6,280)
Nearest Neighbor Imputation	382,530 (2,092)	0 (0)	180,000 (2,010)	525,000 (5,024)
Regression Imputation	580,772 (2,038)	240,000 (1,276)	482,986 (2,004)	732,174 (2,090)

Table 2: Summary Statistics for VALP under MCAR (SE values in parentheses)

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	1,666 (4)	880 (5)	1,400 (25)	2,000 (25)
No Imputation	1,655 (6)	860 (8)	1,400 (25)	2,000 (25)
Mean Imputation	1,637 (4)	1,100 (25)	1,604 (24)	1,604 (24)
Random Imputation	1,636 (5)	850 (5)	1,300 (25)	2,000 (25)
Nearest Neighbor Imputation	1,066 (6)	0 (0)	800 (10)	1,600 (25)
Regression Imputation	1,657 (4)	1,000 (6)	1,541 (6)	2,000 (3)

Table 3: Summary Statistics for RNTP under MCAR (SE values in parentheses)

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	586,079 (2,363)	200,000 (2,512)	400,000 (5,024)	700,000 (6,280)
No Imputation	578,792 (2,923)	190,000 (2,261)	400,000 (5,024)	700,000 (6,280)
Mean Imputation	558,416 (1,902)	295,000 (2,512)	520,223 (2,456)	520,223 (2,456)
Random Imputation	557,818 (2,301)	180,000 (1,256)	380,000 (2,512)	699,000 (5,024)
Nearest Neighbor Imputation	377,434 (1,996)	0 (0)	180,000 (2,261)	500,000 (6,280)
Regression Imputation	578,735 (1,882)	240,000 (1,256)	485,597 (2,182)	742,331 (2,115)

Table 4: Summary Statistics for VALP under MAR (SE values in parentheses)

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	1,666 (4)	880 (5)	1,400 (25)	2,000 (25)
No Imputation	1,657 (6)	880 (5)	1,400 (25)	2,000 (25)
Mean Imputation	1,639 (4)	1,100 (25)	1,605 (24)	1,700 (25)
Random Imputation	1,645 (5)	850 (5)	1,400 (25)	2,000 (25)
Nearest Neighbor Imputation	1,091 (5)	0 (0)	850 (8)	1,700 (25)
Regression Imputation	1,668 (4)	1,000 (9)	1,562 (5)	2,008 (4)

Table 5: Summary Statistics for RNTP under MAR (SE values in parentheses)

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	586,079 (2,363)	200,000 (2,512)	400,000 (5,024)	700,000 (6,280)
No Imputation	672,848 (3,245)	230,000 (2,512)	500,000 (6,280)	800,000 (10,048)
Mean Imputation	651,467 (2,132)	350,000 (3,768)	609,548 (3,882)	610,000 (3,882)
Random Imputation	652,018 (2,512)	220,000 (1,256)	450,000 (5,024)	750,000 (6,531)
Nearest Neighbor Imputation	445,576 (2,139)	0 (0)	225,000 (1,256)	610,000 (6,280)
Regression Imputation	643,306 (2,223)	273,137 (1,256)	503,751 (2,973)	800,000 (1,508)

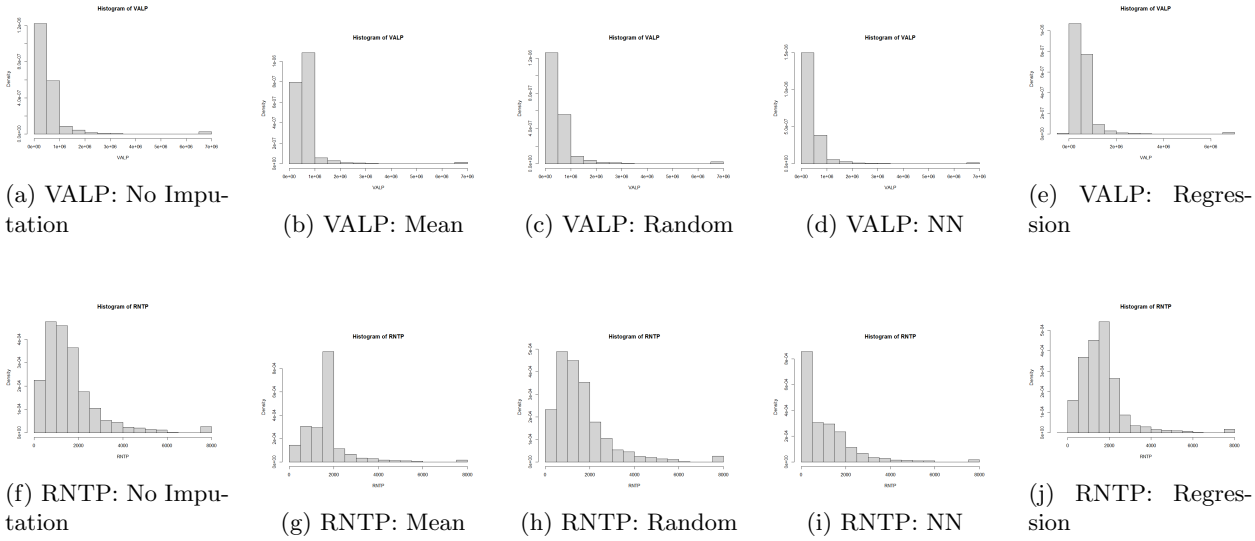
Table 6: Summary Statistics for VALP under MNAR (SE values in parentheses)

Imputation Method	Mean	25th Percentile	Median	75th Percentile
No Missing	1,666 (4)	880 (5)	1,400 (25)	2,000 (25)
No Imputation	1,841 (5)	1,000 (25)	1,500 (25)	2,300 (25)
Mean Imputation	1,824 (3)	1,300 (25)	1,790 (2)	1,800 (25)
Random Imputation	1,818 (5)	980 (5)	1,500 (25)	2,300 (25)
Nearest Neighbor Imputation	1,212 (5)	0 (0)	980 (5)	1,800 (25)
Regression Imputation	1,802 (3)	1,100 (7)	1,686 (7)	2,200 (3)

Table 7: Summary Statistics for RNTP under MNAR (SE values in parentheses)

Histogram Outputs

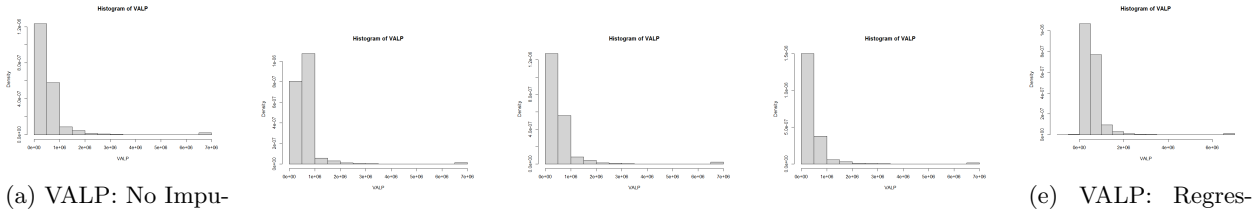
Outside of Figure 1, here are the histograms showing the distributions of the target variables under different imputation methods as generated by function `get_histogram()` run from `impute_calculate_and_visualize_PUMS()`.



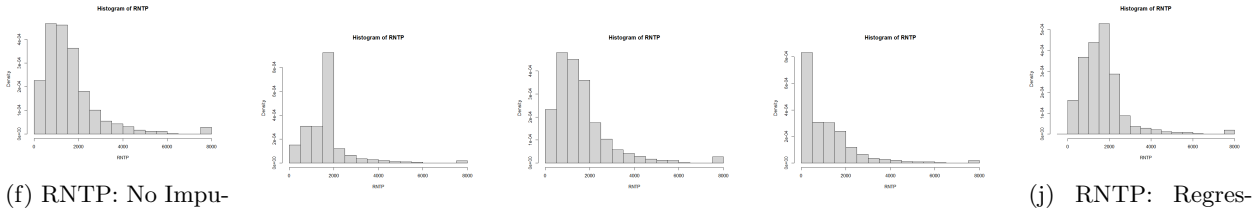
(a) VALP: No Imputation (b) VALP: Mean (c) VALP: Random (d) VALP: NN (e) VALP: Regression

(f) RNTP: No Imputation (g) RNTP: Mean (h) RNTP: Random (i) RNTP: NN (j) RNTP: Regression

Figure 2: Histograms of VALP and RNTP under different imputation methods (MCAR)

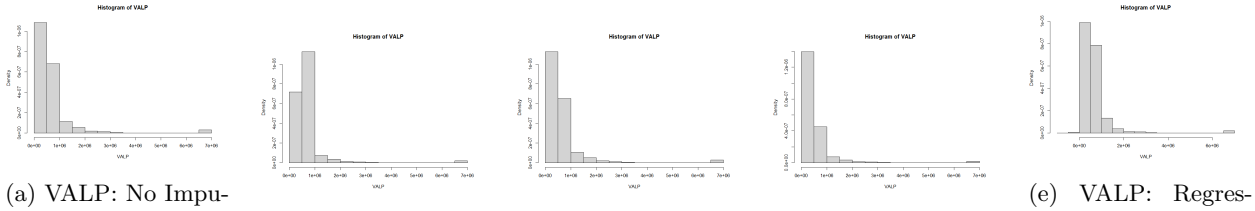


(a) VALP: No Imputation (b) VALP: Mean (c) VALP: Random (d) VALP: NN (e) VALP: Regression

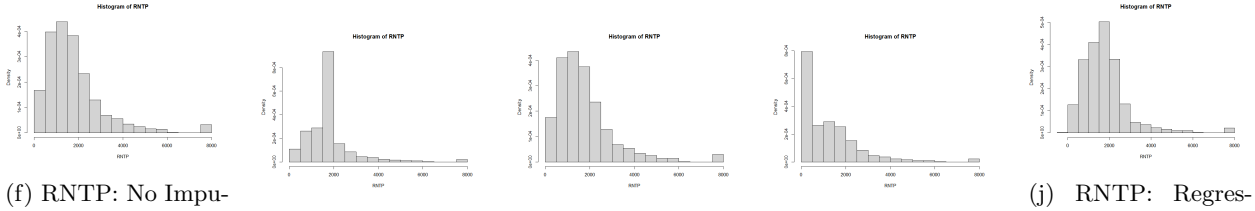


(f) RNTP: No Imputation (g) RNTP: Mean (h) RNTP: Random (i) RNTP: NN (j) RNTP: Regression

Figure 3: Histograms of VALP and RNTP under different imputation methods (MAR)



(a) VALP: No Imputation (b) VALP: Mean (c) VALP: Random (d) VALP: NN (e) VALP: Regression



(f) RNTP: No Imputation (g) RNTP: Mean (h) RNTP: Random (i) RNTP: NN (j) RNTP: Regression

Figure 4: Histograms of VALP and RNTP under different imputation methods (MNAR)