PharmaSUG China 2025 - Paper SA-105

Implementing Multiple Imputation and ANCOVA with Rubin's Rule in R: A Comparative Study with SAS

Zifan Guo, AstraZeneca

ABSTRACT

Accurate handling of missing data is vital in clinical trials and research. Multiple imputation, a robust statistical method, is traditionally executed using SAS software. However, with R's growing popularity as an open-source alternative, implementing these processes in R is increasingly desirable for flexibility and accessibility. This paper details the implementation of multiple imputation and ANCOVA analysis using Rubin's rule in R, replicating established SAS procedures. Leveraging R's statistical libraries, we recreate the multiple imputation process and apply Rubin's rule to imputed datasets. Our comparative analysis, using dummy data, validates the consistency of R's results with those from SAS, attributing differences solely to inherent randomness in the imputation. Our findings confirm R's viability as an alternative to SAS, offering added flexibility without compromising accuracy. This work not only validates R for multiple imputation but also provides a practical guide for researchers transitioning from SAS to R. Key R packages used include "mice" and "norm" for imputation, and "stats" and "emmeans" for ANCOVA with Rubin's rule.

INTRODUCTION

In clinical trials and other research fields, handling missing data effectively is crucial to ensure accurate results. Multiple imputation, a robust statistical method, is commonly employed for this purpose. Traditionally, SAS has been the go-to software for executing multiple imputation and subsequent analysis. However, with the growing popularity of R, an open-source software environment, there is a need to implement these statistical processes in R to offer flexibility and accessibility to the exploratory analysis.

This paper focuses on implementing multiple imputation using both MCMC and monotone regression methods in R, followed by ANCOVA analysis using Rubin's rule—paralleling established SAS procedures. By leveraging R's extensive statistical libraries, we recreate the multiple imputation process and apply Rubin's rule to analyze the imputed datasets. Through a detailed comparison, we validate the consistency of results between R and SAS, demonstrating that any observed differences are attributed solely to inherent randomness in the imputation process rather than methodological discrepancies.

We utilize dummy data to perform our comparative analysis, illustrating the step-by-step process and ensuring reproducibility. Our findings confirm that R is a viable alternative to SAS for these statistical tasks, offering researchers additional flexibility without compromising accuracy. This work not only contributes to the validation of R as a tool for multiple imputation but also serves as a practical guide for researchers aiming to transition from SAS to R for their statistical analyses.

METHODS OVERVIEW: MULTIPLE IMPUTATION, ANCOVA, AND POOLING WITH RUBIN'S RULE

In clinical trials, multiple imputation (MI) is widely used to address missing data, with common methods including Markov Chain Monte Carlo (MCMC), monotone regression, and predictive mean matching, among others. This paper focuses on introducing the MCMC and monotone regression approaches, which are particularly relevant for handling various missing data patterns in longitudinal studies. In addition, we describe the use of analysis of covariance (ANCOVA) as a standard statistical method for comparing treatment effects after imputation, and the application of Rubin's Rule to appropriately combine estimates and account for the uncertainty associated with the imputation process.

MARKOV CHAIN MONTE CARLO (MCMC) IMPUTATION

Markov Chain Monte Carlo (MCMC) is a stochastic simulation method widely used for multiple imputation of missing data. MCMC generates plausible values for missing entries by iteratively drawing samples from the posterior distributions of the missing data, conditional on the observed data and model parameters. Typically, the data is assumed to follow a multivariate normal distribution, and the imputation process cycles through the variables with missing values, updating each in turn based on the current set of imputed values for the other variables. (Schafer 1997)

MCMC imputation is especially suitable in situations where data are missing in an arbitrary (non-monotone) fashion and the relationships among variables are complex, as it preserves multivariate structures and correlations present in the data. Key assumptions include the missing at random (MAR) mechanism and correct specification of the multivariate model. The strength of MCMC lies in its ability to handle complex data structures and provide proper uncertainty estimates for subsequent analysis. (Rubin 1987)

MONOTONE REGRESSION IMPUTATION

Monotone regression imputation is designed for datasets in which the missing data follow a monotone pattern-that is, if a value is missing at a certain time point, all subsequent values for that subject are also missing. In this case, regression models can be fit sequentially to impute missing values, with each model conditioning only on variables that are fully observed for the corresponding cases.

During imputation, each variable with missing data is regressed on previously observed variables, and missing values are then drawn from the resulting predictive distribution. This approach is computationally efficient and leverages the ordered nature of longitudinal or follow-up data, while explicitly handling the monotone structure of missingness. Monotone regression assumes the missing at random mechanism and requires the monotone missing data pattern for its direct application. (Little & Rubin 2019)

ANALYSIS OF COVARIANCE (ANCOVA)

Analysis of Covariance (ANCOVA) is a statistical technique commonly used in clinical trials to compare treatment effects while adjusting for baseline values or other covariates. After handling missing data through multiple imputation methods such as MCMC or monotone regression, ANCOVA can be applied separately to each imputed dataset. This helps to improve the precision of treatment effect estimates by accounting for variability explained by covariates and ensures more robust results in the context of partially observed data. (Vickers & Altman 2001)

RUBIN'S RULE FOR POOLING ESTIMATES

Rubin's Rule provides a framework for combining parameter estimates and standard errors across multiple imputed datasets, reflecting both within-imputation and between-imputation variability. Following separate analyses (e.g., ANCOVA) on each imputed dataset, Rubin’s Rule aggregates the estimates to produce valid statistical inferences that account for uncertainty due to missing data. This is essential for obtaining correct confidence intervals and p-values in multiple imputation analyses. (Rubin 1987; Schafer 1997)

IMPLEMENTATION IN SAS

In SAS, handling missing data in a dataset with covariates and follow-up values typically involves a two-step imputation process followed by statistical analysis. First, PROC MI with the MCMC option is used to convert a non-monotone missing pattern into a monotone structure. Then, monotone regression imputation in PROC MI completes the remaining missing values. Afterwards, ANCOVA is performed on each imputed dataset using PROC MIXED, and the results are combined with PROC MIANALYZE according to Rubin’s Rule. This process is illustrated in Figure 1.

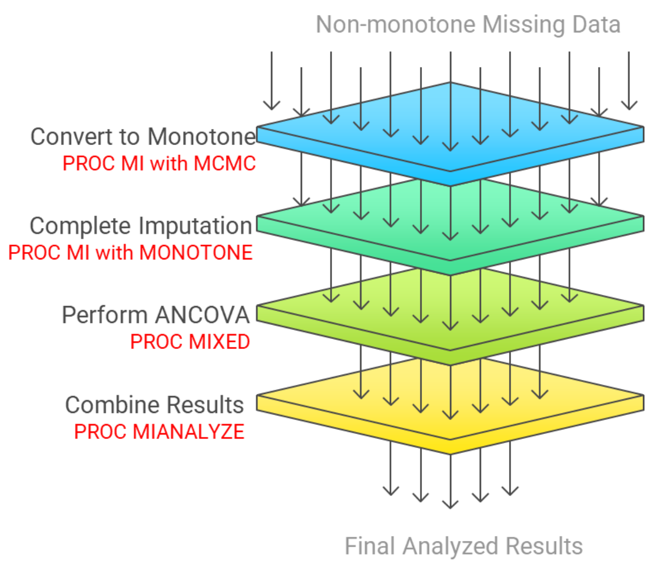

Figure 1. SAS Workflow for Multiple Imputation and ANCOVA Using MCMC and Monotone Regression

SAS EXAMPLE CODE FOR MULTIPLE IMPUTATION

The example below illustrates a two-step imputation workflow in SAS using generalized code. It is important to note that, unlike the typical BDS (Basic Data Structure) format defined by CDISC in which each row represents a single observation per visit per subject, the data used here must be in a wide format. In this structure, each subject corresponds to a single row, and the visit variables are derived by transposing the analysis values (e.g., AVAL) from different visits into separate columns (e.g., VISIT1, VISIT2, …, VISITn).

```
/* Step 1: Use MCMC to convert non-monotone missingness to a monotone structure */
PROC MI DATA=<input_data> NIMPUTE=<num_imputations> SEED=<mcmc_seed> OUT=<mcmc_output>;
  VAR <treatment_group> <categorical_covariate(s)> <baseline_variable> <visit_variables>;
  MCMC CHAIN=SINGLE NBITER=200 NITER=100 IMPUTE=MONOTONE;
RUN;

/* Step 2: Use monotone regression for final imputation */
PROC MI DATA=<mcmc_output> NIMPUTE=1 SEED=<reg_seed> OUT=<final_imputed_data>
(rename=(_imputation_=imputation_number));
  BY _imputation_;
  CLASS <treatment_group> <categorical_covariate(s)>;
  VAR <treatment_group> <categorical_covariate(s)> <baseline_variable> <visit_variables>;
  MONOTONE REG(<last_visit_variable> = <treatment_group> <categorical_covariate(s)>
<previous_visit_variables>);
RUN;
```

In the first step, the `PROC MI` procedure applies MCMC imputation to transform non-monotone missing patterns into a monotone structure across a set of baseline and follow-up variables. In the second step, monotone regression is used with `PROC MI` to complete the imputation process for the monotone segments of the dataset. Key variables such as treatment group, relevant covariates, baseline measurements, and repeated visit values are incorporated as predictors in both steps. This approach can be tailored to a range of clinical trial datasets by appropriately specifying variables and imputation parameters.

SAS EXAMPLE CODE FOR ANCOVA AND RUBIN'S RULE

The below example code demonstrates a typical workflow for conducting ANCOVA analysis on multiple imputed datasets in SAS.

```
/* Step 1: Fit ANCOVA model to each imputed dataset */
ods output lsmeans=lsmeans_out diffs=diffs_out;

proc mixed data=<imputed_data> method=REML noclprint;
  class <subject_id> <treatment_group> <categorical_covariates>;
  model <change_variable> = <treatment_group> <baseline_var> <other_covariates> / solution;
  lsmeans <treatment_group> / cl diff e obsmargins;
  by <imputation_index> <visit_variable>;
run;

/* Step 2: Pool lsmeans using Rubin's Rule */
proc mianalyze data=lsmeans_out;
  modeleffects estimate;
  stderr stderr;
  ods output ParameterEstimates=lsmeans_pooled;
  by <visit_variable> <treatment_group>;
run;

/* Step 3: Pool treatment differences using Rubin's Rule */
proc mianalyze data=diffs_out;
  modeleffects estimate;
  stderr stderr;
  ods output ParameterEstimates=diffs_pooled;
  by <visit_variable> <treatment_group>;
run;
```

For each imputed dataset, the `PROC MIXED` procedure is used to fit an ANCOVA model, adjusting for baseline and other relevant covariates, with treatment group as the primary factor of interest. Least squares means and treatment differences are saved for further analysis. The outputs are then sorted and processed using `PROC MIANALYZE`, which applies Rubin's Rule to pool estimates and standard errors across imputations, generating overall results that account for both within- and between-imputation variability. This process can be adapted to different analysis models, variables, and imputation schemes as needed for a variety of clinical trial study designs.

DEVELOPMENT AND IMPLEMENTATION IN R

Compared with SAS, which provides a standardized set of procedures for multiple imputation and subsequent analysis, implementing these steps in R often presents additional challenges. In R, there is no unified workflow; users must carefully select appropriate packages and functions, or develop custom code, to perform each step of the process. As a result, handling missing data and conducting pooled analyses require careful coordination between different tools and a clear understanding of their implementation details. In this section, we describe in detail how MCMC imputation, monotone regression, and the combined application of ANCOVA with Rubin's Rule can be achieved in R, highlighting practical considerations and illustrating suitable package options and coding development. For full reproducibility and to facilitate application, the complete R code used in this section is available at https://github.com/GZfisher/mi_ancova_pooling.

MCMC IMPUTATION IN R

By reviewing the literature and the SAS documentation, it is clear that the MCMC approach for multiple imputation is essentially a data augmentation procedure, which consists of two key steps: the I step (Imputation or Data Augmentation step) and the P step (Posterior step) (SAS Institute 2021; Schafer 1997). Specifically, the I step involves drawing plausible values for missing data given the observed data and current parameter values, while the P step updates the parameters based on the complete dataset obtained from the I step. This iterative process is the foundation of the MCMC algorithm for handling arbitrary missing patterns.

In R, the norm package (Schafer 2023) offers the function `mda.norm` for implementing data augmentation. However, practical experience and user reports indicate that `mda.norm` encounters significant limitations when the number of variables exceeds a certain threshold, preventing it from running successfully for medium- to large-scale problems. To address this, we examined and modified the underlying code of the norm package to bypass the bottleneck. Additionally, as the core computational logic of this package is implemented in Fortran, we translated the relevant Fortran code into R using AI-based tools (bing AI). This translation allowed us to better understand and trace the underlying algorithm, facilitating customized modifications.

In the sections below, we present the core R code that realizes the data augmentation algorithm, annotated to map clearly onto the I step (imputation) and P step (parameter updating) described in the MCMC framework. In order to directly implement the MCMC-based data augmentation algorithm in R, we developed a core function (`mda_r`) that alternately performs the **I-step** and **P-step** as described above. This function is based on, and extends, the `mda.norm` function from the norm package, with modifications designed to address the limitations observed when handling higher-dimensional data. The function takes as input the current imputed data and parameter estimates, and iteratively updates them. Specifically, the I-step is represented in the code by the call to the `is2n()` function, which generates plausible values for missing entries using the current parameter values. The P-step, implemented by the `ps2n()` function, updates parameter estimates based on the completed dataset just produced by the I-step. These two steps are repeated for a specified number of iterations to ensure adequate mixing and convergence of the Markov chain.

Here is the core implementation in R:

```r
# complete I-step and P-step
mda_r <- function (s, theta, steps = 1, showits = FALSE) {
 s$x <- .na.to.snglcode(s$x, as.double(999))
 tobs <- tobsmn(s$p, s$psi, s$n, s$x, s$npatt, s$r, s$mdpst,
         s$nmdp, s$last, integer(s$p), s$sj, s$layer, s$nlayer, s$d)
 if (showits)
  cat(paste("Steps of Monotone Data Augmentation:", "\n"))
 for (i in 1:steps) {
  if (showits)
   cat(paste(format(i), "...", sep = ""))
  # I-step: impute missing data given current parameters
  s$x <- is2n(s$d, theta, s$p, s$psi, s$n,
        s$x, s$npatt, s$r, s$mdpst, s$nmdp, s$sj, s$last,
        integer(s$p), integer(s$p), double(s$p), theta, rnorm(n=1))
  # P-step: update parameters given completed data
  theta <- ps2n(s$p, s$psi, s$n, s$x, s$npatt,
         s$r, s$mdpst, s$nmdp, integer(s$p), integer(s$p),
         s$nmon, s$sj, s$nlayer, s$d, tobs, numeric(s$d),
        numeric(s$d), numeric(s$p + 1), numeric(s$d))
 }
 if (showits)
  cat("\n")
 theta
}
```

This structure closely mirrors the theoretical underpinnings of data augmentation as described in the literature (SAS Institute 2021; Schafer 1997) and the original implementation in the norm package (Schafer 2023). By deconstructing the process into modular I-step and P-step functions, we also make future method development and troubleshooting more tractable for complex imputation scenarios. It is important to note that the output of the mda_r function is the set of parameter estimates obtained after the specified number of data augmentation iterations. These parameters are then subsequently used to impute the missing values in the dataset:

```
# MCMC imputation function
step1 <- function(data, nimpute, emmaxits, maxits, seed) {
 s <- prelim.norm.new(data)   # Prepare data for norm package
 thetahat <- em.norm(s, maxits = emmaxits)  # Get initial parameters using EM algorithm
 rngseed(seed)        # Set random seed
 theta <- mda_r(s, thetahat, steps = maxits, showits = TRUE)  # Run MCMC to update parameters

 all_mono_new <- data.frame(data)
 all_mono_new[,"impno"] <- 0  # Add imputation indicator for original data

 for (i in 1:nimpute) {
  all_mono_one_time <- data.frame(imp.norm(s, theta, data))  # Impute missing values
  # Ensure monotone structure by setting post-missing values to NA for each row
  for (j in 1:nrow(data)) {
   last_num <- max(which(!is.na(data[j,])))
   if (last_num < ncol(data)) {
    all_mono_one_time[j, (last_num+1):ncol(data)] <- "is.na<-"(all_mono_one_time[j,
(last_num+1):ncol(data)])
   }
  }
  all_mono_one_time$impno <- i  # Add imputation number
  all_mono_new <- rbind(all_mono_new, all_mono_one_time) # Append result
  print(paste0("MCMC imputation: ", i, "..."))
 }
 return(all_mono_new)
}
```

The parameters obtained from the `mda_r` function serve as the basis for generating multiple imputed datasets. As shown in the `step1` function above, we first compute the maximum likelihood estimates using the EM algorithm to obtain initial parameter values, then iteratively update them using our MCMC data augmentation routine. For each desired imputation, the most recent parameter estimates are used in conjunction with `imp.norm` to generate a new complete dataset with imputed values. This process is repeated multiple times (as specified by `nimpute`), producing a set of imputed datasets where the missing values are filled in based on the observed data structure and the uncertainty captured by the MCMC algorithm.

Additionally, this code `all_mono_one_time[j, (last_num+1):ncol(data)] <- "is.na<-"(all_mono_one_time[j, (last_num+1):ncol(data)])` sets all values following the last observed (non-missing) value in each row back to missing. This step ensures that MCMC imputation only fills in the internal missing values that are consistent with the MAR assumption, and does not impute values beyond a subject's last observed visit. In effect, this operation mirrors the behavior of the `IMPUTE=MONOTONE` option in SAS, transforming the dataset into a monotone missing data pattern in preparation for subsequent monotone regression imputation.

MONOTONE REGRESSION IMPUTATION IN R

To perform the monotone regression imputation step in R, we developed the `step2` function, which utilizes the `mice` package to impute the remaining missing values under a monotone pattern.

```
# Monotone regression imputation using mice
step2 <- function(data, nimpute, method, formula_list, seed) {

 reg <- list()
 for (i in 1:nimpute) {
  seed = seed + 1
  # Filter data for current imputation
  reg[[paste0("x", i)]] <- data %>% filter(impno == i)
  # Sequentially apply regression formulas
  for (j in 1:length(formula_list)) {
   x <- mice::mice(reg[[paste0("x", i)]], m = 1, method = method,
           formulas = formula_list[j], seed = seed)
   reg[[paste0("x", i)]] <- mice::complete(x)
  }
 }

 complete <- do.call(rbind, reg)  # Combine imputed datasets
 return(complete)
}
```

For each imputed dataset generated in the previous step, `step2` iterates over the specified regression formulas and applies single imputation (i.e., `m=1`) using the designated method (such as `"norm"`, `"norm.predict"`, etc.). Different seeds are set for each imputation to ensure reproducibility and variability across imputations. This approach allows flexible specification of regression models for each variable or time point, maintaining alignment with the underlying monotone missing data structure established previously. The resulting fully imputed datasets are then combined for downstream analysis.

A key aspect of the monotone regression imputation approach in R is the dynamic construction of regression formulas for each visit variable. In our workflow, we build a `formula_list` sequentially:

```
# Define covariate columns for modeling
cov_cols <- c('TRT01PN'...)
# Identify visit variables that need imputation or modeling
visit_cols = ...
```

```
# Initialize model formula list for each step
formula_list <- c()

# Dynamically construct the formula for each visit variable,
# each time adding the current visit as a predictor in subsequent formulas
for (i in 1:length(visit_cols)) {
 now_visit_col <- visit_cols[i]

 # If the current visit column has missing values:
 if (any(is.na(data[[now_visit_col]]))) {
  response <- now_visit_col
  formula_list <- c(
   formula_list,
   as.formula(paste0(response,"~",paste(cov_cols,collapse = '+')))
  )
  # Add this column to covariates for the next steps
  cov_cols <- c(cov_cols, now_visit_col)
 }
 # If the column is complete, just add it to covariates
 else {
  cov_cols <- c(cov_cols, now_visit_col)
 }
}
```
ensuring that each variable with missing values is imputed using all relevant covariates—including previously imputed or observed visit variables. Specifically, for each visit column, if missingness is detected, we create a regression formula with the current visit as the response and the available covariates (including baseline and previously constructed visits) as predictors. The current visit variable is then appended to the set of predictors for subsequent steps, so that later imputations are conditioned on a growing set of observed or imputed data. This stepwise updating aligns with the inherent structure of monotone regression and allows the `step2` function to flexibly model longitudinal relationships in the data during the imputation process.

PERFORMING ANCOVA & COMBINING RESULTS USING RUBIN'S RULE

In R, several common packages and functions are available for performing model fitting on multiply imputed datasets and combining results using Rubin's rule. Popular choices include the `pool()` function from the `mice` package and the `MIcombine()` function from the `mitools` package, both of which automate Rubin's rule pooling for regression results. However, these tools have notable limitations. For example, `pool()` primarily provides combined point estimates and standard errors, but does not always deliver confidence intervals or allow flexible specification of marginal means for group comparisons—features that are often needed in clinical trial analyses.

To overcome these limitations, we explored additional packages and found that the `emmeans` package integrates Rubin's rule functionality when applied to models fitted across multiply imputed datasets (such as those created by the `mice` workflow). Using `emmeans`, it is possible to obtain pooled estimates, confidence intervals, and hypothesis tests for adjusted group means and contrasts—all within a consistent and flexible framework. Importantly, our comparison of results from `emmeans` in R and the corresponding SAS procedures showed that differences were negligible, only occurring after the third decimal place (see following section).

Below is the core R code used for multiply imputed ANCOVA and Rubin's rule combination via `emmeans`:

```
ancova_res <- function(imp, formula, trt_var, ref) {
 lm_fit <- with(data=imp,exp=stats::lm(
  formula = stats::as.formula(formula)
 )) # Fit model based on each imputation
 emmeans_fit <- emmeans::emmeans(
  lm_fit,
  # Specify here the group variable over which EMM are desired.
  specs = trt_var,
  weights = "proportional"
 )
 emmeans_contrasts <- emmeans::contrast(
  emmeans_fit,
  # Compare dummy arms versus the control arm.
  method = "trt.vs.ctrl",
  # Take the arm factor from param "ref" as the control arm.
  ref = ref,
  level = 0.95
 )
 sum_contrasts <- summary(
  emmeans_contrasts,
  # Derive confidence intervals, t-tests and p-values.
  infer = TRUE,
  # Do not adjust the p-values for multiplicity.
  adjust = "none"
 )
 return(list(contrasts=sum_contrasts, lm_fit=lm_fit))
}
```

This approach enables a comprehensive, reproducible ANCOVA analysis in R with robust pooling of effect estimates and confidence intervals, closely aligning with the results produced by standard SAS workflows. In this workflow, the `imp` object supplied to the ANCOVA and pooling functions represents the completed, fully imputed dataset. After monotone regression imputation, the dataset (initially in wide format with separate columns for each visit) is first converted to long format so that each row corresponds to a subject and a visit. Derived variables, such as the analysis value (`AVAL`), are calculated as needed. The combined and reshaped data are then converted into the `mids` class using the `mice::as.mids` function, making them compatible with downstream functions such as `with()` and `emmeans()`. This ensures proper integration of imputed data structure for multiply imputed analyses.

For example, the following code snippet demonstrates how the completed imputed data is transformed for further analysis:

```
# Function to convert imputed and original (with missing values) data from wide to long
format,
# and compute analysis variables and a unique subject-visit ID
convert_long <- function(imputed_data, original_data, visit_prefix = "VISIT", subj_col =
"SUBJID",
            base_col = "BASE", target_var = "CHG", imp_col = "impno") {
 # Combine original data (with missing values) and imputed data
```

```
    combined <- rbind(original_data, imputed_data)

    # Pivot to long format: each row is a subject-visit combination
    long <- combined %>%
     pivot_longer(
      cols = starts_with(visit_prefix), # Visit columns (e.g., VISIT1, VISIT2, ...)
      names_to = "AVISIT",       # Name for visit variable
      values_to = target_var      # Name for change-from-baseline (or other measure)
     ) %>%
     mutate(
      AVAL = !!sym(base_col) + !!sym(target_var),         # Calculate analysis value
      id = paste0(!!sym(subj_col), "_", AVISIT)      # Create unique ID per subject-visit
     )
    return(long)
   }
   # Example usage:
   # original_data should be the initial dataset containing missing values
   # imputed_data is the completed data after imputation steps
   complete_long <- convert_long(imputed_data = complete100, original_data = ori_data,
                visit_prefix = "WEEK", subj_col = "SUBJID", base_col = "BASE", target_var =
   "CHG")
   # Convert to 'mids' object for multiply imputed analysis
   complete_imp <- mice::as.mids(long_data, .imp = "impno", .id = "id")
```

This prepares the imputed data for flexible analysis and reporting in R, ensuring compatibility with multiple imputation methods and facilitating the application of ANCOVA and Rubin's rule pooling.

COMPARATIVE ANALYSIS: R VS. SAS

Although the multiple imputation and analysis workflows in R and SAS are conceptually aligned—both involving MCMC imputation, monotone regression, ANCOVA, and Rubin's Rule for pooling results—practical equivalence must be established empirically. To rigorously evaluate the consistency between R and SAS implementations, our comparative process involved several steps. First, we constructed a complete dummy dataset and then introduced random missingness to simulate a realistic scenario for multiple imputation. Second, we applied the described imputation approach in R to fill in missing values, then performed ANCOVA with Rubin's Rule pooling in both R and SAS using the same imputed dataset. By comparing these analytic results, we were able to verify the consistency between the two software platforms in terms of ANCOVA modeling and pooling procedures. Third, we carried out the full multiple imputation workflow independently in both R and SAS—including imputation, ANCOVA, and Rubin's Rule pooling—then compared the results between the two systems as well as against the original complete dummy data (serving as the gold standard). The imputation quality was quantitatively assessed by calculating metrics such as mean squared error (MSE) and mean absolute error (MAE) between the imputed and true values.

The following subsections describe the comparative process step-by-step: construction of dummy data, comparing ANCOVA and Rubin's Rule pooling results across software, and finally evaluating the imputation results in detail.

GENERATION OF DUMMY DATA WITH SIMULATED MISSINGNESS

To enable a controlled and transparent comparison between R and SAS in both imputation and subsequent analysis, we first generated a comprehensive dummy dataset (`data_complete`) without any missing values. This synthetic dataset mirrors the typical structure of longitudinal clinical trial data, including covariates, a baseline measure, and repeated measures across multiple follow-up visits.

Data Generation Assumptions and Process

- **Sample size and visits:** The dataset simulates 500 subjects, each assessed at 29 evenly spaced timepoints (weeks 0 to 56, every 2 weeks).
- **Covariates:** We specified three categorical variables—treatment group (TRT01PN), geographic region (REGIONN), and a baseline binary variable (BLBMIG1N)—with predefined proportions reflecting realistic clinical trial scenarios. In addition, a continuous baseline measurement (BASE) was randomly assigned from a uniform distribution (5 to 7).
- **Regression structure:** The outcome for each subject and timepoint (AVAL) was generated using a prespecified linear model incorporating main effects for baseline, treatment, region, BMI, and visit, as well as normally distributed random noise. Each covariate's influence was set by predetermined coefficients to mimic plausible clinical effect sizes.
- **Score limits:** To ensure clinical plausibility and data integrity, simulated scores were restricted to the range of 0 to 10.
- **Data arrangement:** The data were structured in long format, with one row per subject per visit. Derived variables such as change from baseline (CHG) and visit indicators were also included.

After generating the complete dummy dataset, we deliberately introduced missingness following a controlled, hybrid pattern designed to reflect typical longitudinal clinical trial data challenges. Specifically, all 500 subjects were randomly assigned to one of three missing data patterns:

- Approximately 50% of subjects were designated as **completely observed** (all visits non-missing).
- About 30% of subjects were assigned a **monotone missing pattern**, where for each subject, a random follow-up point was selected and all subsequent visit values (`AVAL`, `CHG`) for that subject were set to missing. This simulates the common scenario where patients drop out and have no further data collected after a certain visit.
- The remaining ~20% of subjects received a **monotone plus intermittent missing pattern**. For these subjects, a monotone dropout was first applied as above, and then, among the visits before dropout, one to three additional missing values were randomly introduced, creating a mixture of monotone and missing-at-random (MAR) mechanisms.

This controlled and transparent assignment of missingness allowed us to precisely benchmark the imputation procedures under a range of realistic and challenging patterns. The resulting dataset with introduced missing values was structured to clearly indicate which subjects experienced dropout or intermittent missingness, and was utilized for all subsequent imputation and analytic comparisons between R and SAS.

CONSISTENCY CHECK OF ANCOVA AND RUBIN'S RULE POOLING BETWEEN R AND SAS

To assess the analytic consistency between R and SAS, we first performed multiple imputation on the dummy dataset with missing values using the workflow described previously in R. We then exported the fully imputed dataset and conducted ANCOVA with Rubin's Rule pooling on both R and SAS platforms, applying identical model specifications.

The comparison of results (Figure 2) revealed that the estimated mean differences, standard errors, and t-values were almost identical across R and SAS, indicating harmonious implementation of the key ANCOVA modeling and pooling steps. However, we observed substantial relative differences in the reported degrees of freedom (df), which led to slight but noticeable discrepancies in confidence intervals and p-values. Given that the degrees of freedom directly influence the selection of the t-distribution for interval and p-value calculations, this divergence warranted further investigation.

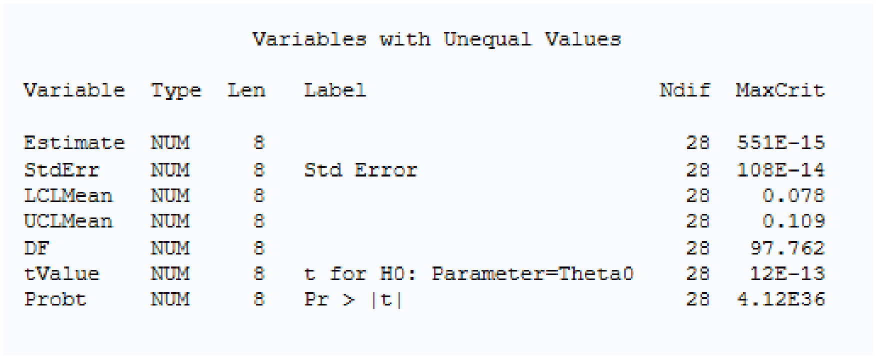
```
                    Variables with Unequal Values

Variable  Type  Len   Label                        Ndif  MaxCrit

Estimate  NUM     8                                  28  551E-15
StdErr    NUM     8   Std Error                      28  108E-14
LCLMean   NUM     8                                  28    0.078
UCLMean   NUM     8                                  28    0.109
DF        NUM     8                                  28   97.762
tValue    NUM     8   t for H0: Parameter=Theta0     28   12E-13
Probt     NUM     8   Pr > |t|                       28  4.12E36
```

Figure 2. Comparison Results when EDF is Null

By examining the SAS documentation and the source code of the `emmeans` package in R, we discovered the reason of this discrepancy. In SAS, the `PROC MIANALYZE` procedure includes an option named `EDF` (complete-data degrees of freedom), which is set to infinity by default unless specified manually (https://documentation.sas.com/doc/en/statug/15.2/statug_mianalyze_syntax01.htm#statug.mianalyze.edfopt). Conversely, the R implementation internally computes the complete-data degrees of freedom based on the number of subjects and the estimated parameters (https://github.com/rvlenth/emmeans/blob/master/R/multiple-models.R). In this case, with a sample size of 500 and seven model parameters (including three categorical covariates and the intercept), the correct complete-data degrees of freedom should be 500 - 7 = 493.

After manually specifying `EDF=493` in the SAS code, the results from R and SAS converged completely (Figure 3), with all relative differences reduced to below 1e-12—essentially machine precision. This demonstrates that, when properly aligned for degrees of freedom specification, R and SAS yield fully consistent results for ANCOVA with Rubin's Rule pooling.

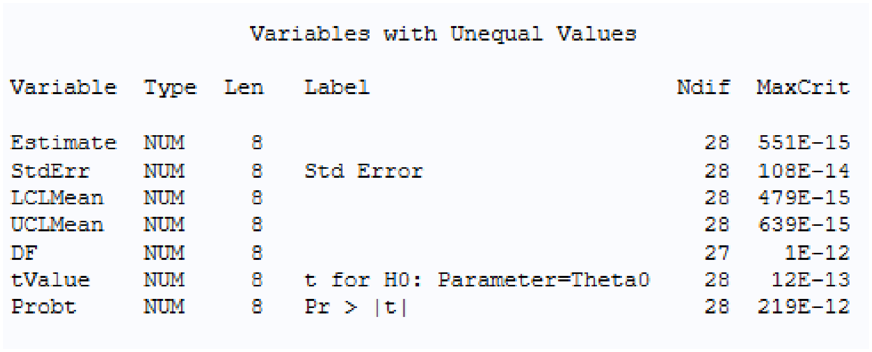
```
                    Variables with Unequal Values

Variable  Type  Len   Label                        Ndif  MaxCrit

Estimate  NUM     8                                  28  551E-15
StdErr    NUM     8   Std Error                      28  108E-14
LCLMean   NUM     8                                  28  479E-15
UCLMean   NUM     8                                  28  639E-15
DF        NUM     8                                  27    1E-12
tValue    NUM     8   t for H0: Parameter=Theta0     28   12E-13
Probt     NUM     8   Pr > |t|                       28  219E-12
```

Figure 3. Comparison Results when EDF = 493

COMPARATIVE EVALUATION OF IMPUTATION PERFORMANCE AND ANALYTICAL RESULTS

To further assess the practical alignment between R and SAS, we independently applied the MCMC and monotone regression imputation workflow in both software environments on the dataset with introduced missingness. In both cases, we maintained consistent model specifications and imputation parameters to ensure a fair comparison.

After generating multiply imputed datasets in R and SAS, we quantitatively evaluated the accuracy of the imputations by comparing each set of completed data to the original gold-standard dataset (`data_complete`). Performance was assessed using metrics such as mean absolute error (MAE) and mean squared error (MSE), providing a direct measure of how well each method was able to recover the true underlying values.

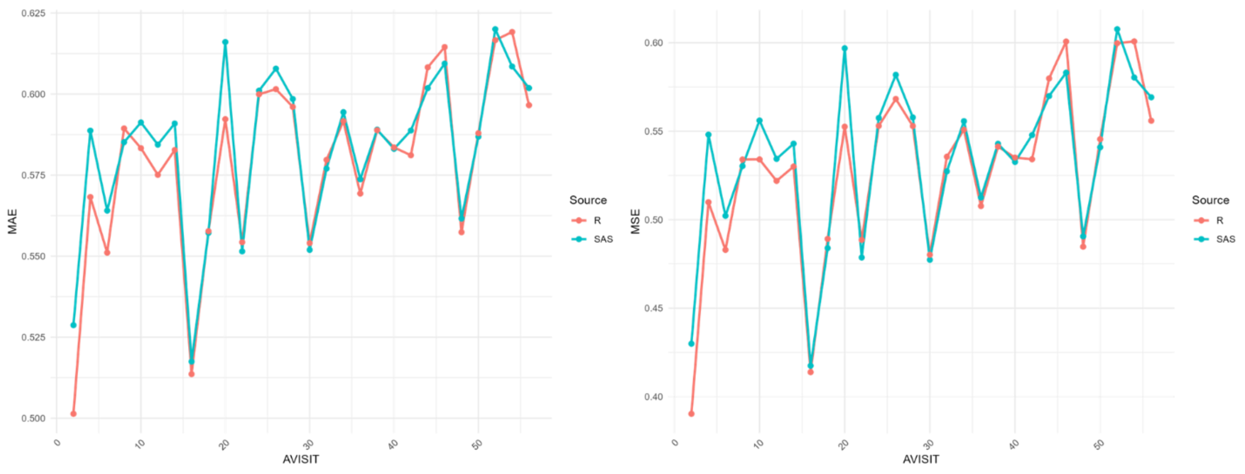

Figure 4. MAE and MSE Comparison between R and SAS

The MAE and MSE plots above (Figure 4) demonstrate that R and SAS achieved highly similar imputation performance throughout the full course of follow-up visits. The curves corresponding to both metrics overlap closely, showing only minor differences at individual timepoints. Notably, even when the same random seed was set in both R and SAS, the results are not expected to be perfectly identical due to internal differences in random number generation and algorithm implementation across the two platforms. Since our methods section has already established that the imputation procedures in R were developed to closely mirror the SAS process, the extremely small discrepancies observed here can reasonably be attributed to random variation inherent to the different software environments, rather than substantive differences in methodology. This further supports the conclusion that, under equivalent analytic strategies, R and SAS provide nearly indistinguishable multiple imputation accuracy in this simulated scenario.

Subsequently, we performed ANCOVA with Rubin's Rule pooling on the multiply imputed datasets from both R and SAS and compared the main analysis results. As shown in the forest plot (Figure 5), the point estimates and confidence intervals from R and SAS are overall quite close across all visit timepoints, despite some modest differences.

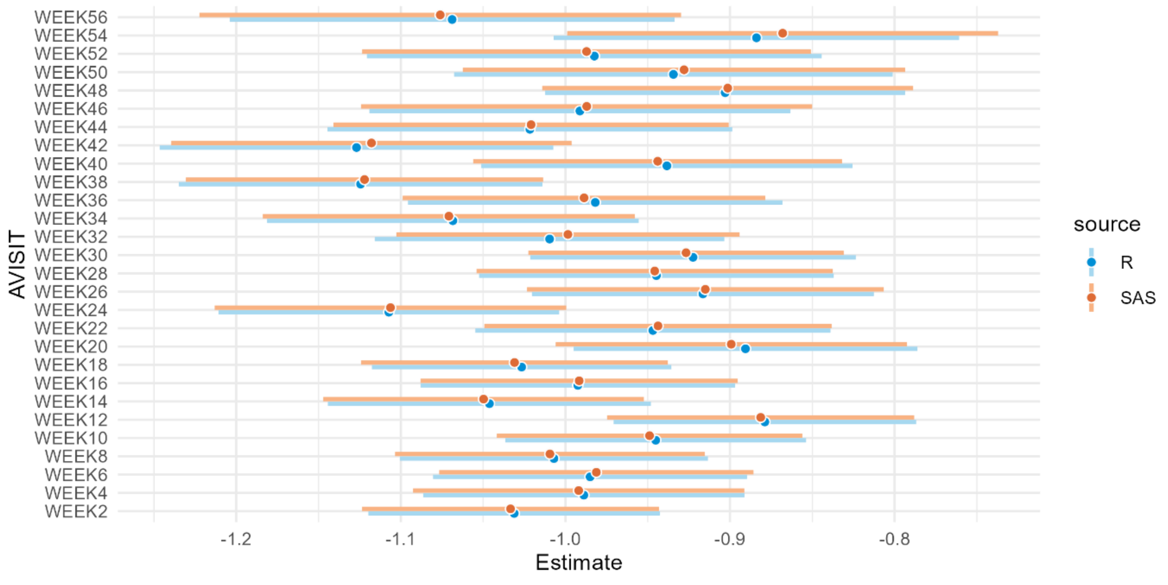

Figure 5. Comparison of Estimates and Cis between SAS and R

It is important to note that even though identical random seeds were used in both R and SAS, perfect replication between the two platforms was not possible due to differences in their underlying random number generation and imputation algorithms. These randomization mechanisms can lead to small discrepancies in the imputed datasets, which subsequently affect the analytic results. This effect becomes especially pronounced when confidence intervals are close to the null hypothesis value: in such cases, one software may produce a confidence interval that marginally includes the null value, while the other does not, potentially leading to divergent conclusions regarding statistical significance.

Overall, the observed differences are minor and are mainly attributed to the inherent randomness in the imputation process, rather than any substantive methodological or algorithmic discrepancies between R and SAS. This further highlights the importance of considering stochastic variation when interpreting multiply imputed analyses across platforms.

This dual evaluation, encompassing both imputation accuracy and downstream inferential consistency, confirms the overall equivalence of the R and SAS implementations when using harmonized model inputs and degree-of-freedom settings. Any minor differences observed were attributable to random variation or minor software default settings, rather than substantive methodological discrepancies.

CONCLUSION

This study provides a systematic comparison of multiple imputation and ANCOVA analysis using Rubin’s Rule as implemented in both R and SAS, focusing on datasets with complex longitudinal missing data patterns typical of clinical trials. Through the construction of a carefully controlled dummy dataset, and by introducing varied and realistic missing data mechanisms, we examined the ability of both platforms to impute missing data and conduct valid post-imputation inference.

Our results demonstrate that, when equivalent imputation methods and model specifications are used, R and SAS deliver highly similar performance. Metrics such as mean absolute error (MAE) and mean squared error (MSE) showed that the imputed values from both software were nearly indistinguishable across all timepoints, as illustrated by the strikingly overlapping error curves. The comparison of downstream ANCOVA results, including point estimates and confidence intervals, further confirmed this consistency: differences between R and SAS were minor and predominantly attributable to inherent stochasticity in the imputation process rather than methodological or computational discrepancies. While identical random seeds cannot fully synchronize random number generation across platforms, any resulting differences were negligible for practical purposes.

We also noted that in rare scenarios—such as when confidence intervals are very close to the null hypothesis—the minor imputation differences may lead to slightly divergent statistical significance conclusions. This observation highlights the impact of inherent stochastic variation in multiple imputation, an important aspect for researchers to consider in their interpretation.

In summary, our comparative evaluation confirms that both R and SAS provide robust, reliable, and methodologically consistent frameworks for multiple imputation and subsequent ANCOVA analyses in clinical trial settings. The choice between platforms may therefore be guided by other practical factors rather than imputation or analysis performance.

REFERENCES

Schafer, J. L. 1997. *Analysis of incomplete multivariate data*. Boca Raton, FL : Chapman & Hall/CRC.

Rubin, D. B. 1987. *Multiple imputation for nonresponse in surveys*. New York, NY : Wiley.

Little, R. J. A., & Rubin, D. B. 2019. *Statistical analysis with missing data (3rd edition)*. Hoboken, NJ : Wiley.

SAS Institute Inc. 2021. “Details about the MCMC Method.” SAS/STAT® 15.2 User’s Guide. n.d. Available at https://documentation.sas.com/doc/en/statug/15.2/statug_mi_details21.htm.

Schafer, Joseph L. 2023. “norm: Analysis of Multivariate Normal Datasets with Missing Values.” CRAN. Available at https://cran.r-project.org/web/packages/norm/norm.pdf.

SAS Institute Inc. “EDF= option.” SAS/STAT® 15.2 User’s Guide. n.d. Available at https://documentation.sas.com/doc/en/statug/15.2/statug_mianalyze_syntax01.htm#statug.mianalyze.edfopt.

Lenth, Russell V. “multiple-models.R.” emmeans GitHub repository. n.d. Available at https://github.com/rvlenth/emmeans/blob/master/R/multiple-models.R.

ACKNOWLEDGMENTS

The author would like to sincerely thank Menghao Xu, statistician at AstraZeneca, and Houmin Xing, then an intern, for their invaluable contributions to the initial stages of this work. Special thanks are also extended to Kevin Huang, programmer at AstraZeneca, and my supervisor Yiwen Wang for their thoughtful review and constructive support on this manuscript.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Zifan Guo
zifan.guo@astrazeneca.com or
gzfchong0815@163.com