Bayesian Missing Data - Multiple Imputation Methods

Presented to

CSUF

COLLEGE OF

Natural Sciences and Mathematics

California State University, Fullerton

Math 538 Fall 2023

Dr. Poynor

Prepared by

Paul LOPEZ

Emilio VASQUEZ

December 07, 2023

Contents

1 Introduction

Missing data is an unavoidable issue that arises in many real-world datasets across various fields, including finance, healthcare, and social sciences. When data are missing, it can introduce biases and lead to invalid statistical inferences if the missing values are simply ignored or eliminated. Multiple imputation is a principled approach for handling missing data that involves creating multiple complete versions of the incomplete dataset, with the missing values imputed through simulated draws from an appropriate model. The key insight is that missing data uncertainty can be represented by generating multiple imputed datasets.

In the Bayesian approach to multiple imputation, prior distributions are specified for the model parameters and missing data, representing initial beliefs before examining the data. The posterior distributions of the parameters and missing values are estimated by fitting the model to the observed data using Markov chain Monte Carlo (MCMC) methods. This allows simulating multiple imputed datasets by drawing missing values from their posterior predictive distributions. The completed datasets can then be analyzed using standard complete-data methods, with final estimates pooled across the imputed datasets to incorporate missing data uncertainty. Bayesian multiple imputation provides a flexible framework for handling missing data while properly representing imputation uncertainty.

A key advantage of the Bayesian approach to multiple imputation is that it allows incorporating appropriate prior information and modeling complex relationships between variables. For example, hierarchical priors can be used to share information between related parameters or models, improving estimates for variables with limited data. Bayesian models like mixtures and nonparametric models provide flexibility to adapt to complex patterns in the data. MCMC provides a convenient computational approach for fitting Bayesian models with missing data, avoiding analytical intractability. The posterior predictive distribution for the missing values conditions on both the observed data and model parameters, ensuring proper imputation uncertainty. Practical implementations utilize packages like `mice` in R and `miceforest` in Python, which automate iterative Bayesian modeling, imputation, and analysis. Overall, Bayesian multiple imputation provides a robust approach for handling missing data that accounts for uncertainty and allows incorporating flexible modeling and prior information about the data structure.

2 Motivation

2.1 Missing Data

Unlike working with data in a classroom setting, real-life data is messy and almost never "clean" out of the box. Often times one can have records that have non-uniform inputs captured. Maybe someone left in a free-form field for entering in a birthday so all the birthdays entered in by the users may all be in different formats. The biggest dilemma that many data professionals will face in their data science careers is the missing data. Missing data come in the following flavors:

2.1.1 Missing Completely At Random (MCAR)

Missing data can come in the form of missing completely at random. This is when the missing data is completely random. For example, imagine students are taking a survey regarding study habits. At random, some of these students spilled liquid on their survey accidentally making these unable or some portion unusable. In this scenario, missing of the data is completely at random as there is not a systematic pattern related to the missing data. This scenario can be accounted for.

2.1.2 Missing at Random (MAR)

Missing at random is when the rate of missing data can be explained if one knew of some other factor. Missing data is common in financial datasets especially due to non-response on surveys or forms. For example, customers may not fill out all the fields on a loan application or investors may skip questions on an investment risk tolerance questionnaire. However, this missing data might be able to be explained by some other question that they answered on the survey, perhaps maybe on occupation. So in theory if one observed that customers who happened to be teachers skipped questions at a higher rate than those who are not teachers, this missing data could be solved if by knowing the occupation.

2.1.3 Missing Not At random (MNAR)

Missing not at random is when the missing data is related to the variable of interest. For example, suppose that one is collecting data on income, age, and home ownership status in which some individuals refuse to disclose their income. You notice that individuals who refuse to disclose their income happen to make more than those who disclose. Missing data of this nature cannot be necessarily accounted for.

For the types of missing data above, it becomes imperative for a data scientist or statistician to have tools to overcome the obstacle of missing data. Depending on the scenario such as having a small dataset, throwing out incomplete records could result in losing valuable information and bring potential biases into any estimates provided to a stakeholder. Multiple imputation is one

way to predict missing values based on patterns in the observed data. This provides a principled way to fill in missing values while accounting for uncertainty, rather than ad-hoc methods like mean imputation. Some models that can be used are multivariate normal imputation or chained equations/sequential regression approaches. The multiple imputed datasets can then be used to train the predictive models, with results appropriately combined across the imputed datasets which allow building more robust models on messy real-world data.

3 Structure of Model/Process

The model structure is a standard Bayesian regression model relating a continuous response Y to predictor variables X:

$$Y \sim N(\mu, \sigma)$$

$$\mu = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ...$$

where:

- Y is the response variable with some values missing
- X_1, X_2, etc are the predictor variables
- β_0 is the intercept
- β_1, β_2, etc are the regression coefficients
- σ is the error standard deviation

In addition, we have missingness indicator variables R for each observation, where R = 1 if Y is observed and R = 0 if Y is missing for that observation.

The priors on the parameters $\beta_0, \beta_1, \beta_2,$ and σ are weakly informative normals or half-normals:

$$\beta_0 \sim N(0, a_1)$$

$$\beta_1 \sim N(0, a_2)$$

$$\beta_2 \sim N(0, a_3)$$

$$\sigma \sim HalfNormal(0, c)$$

Where $a_1, a_2, a_3, ..., c$ are constants. For the missing Y values, we treat them as parameters to be estimated, with priors based on the observed data likelihood:

$$Y_{miss} \sim N(\mu, \sigma)$$

where μ and σ come from fitting the model on the observed data. This makes the missing values exchangeable with the observed data.

Multiple imputation is done by drawing missing Y values from their posterior predictive distribution based on the fitted model. Multiple datasets are created by repeating this process and used to account for missing data uncertainty.

In summary, the model leverages Bayesian regression, weakly informative priors, and treats missing values as parameters to plausibly fill in gaps while quantifying uncertainty. The multiple imputed datasets integrate over the posterior distribution of the missing data.

4 Example

The model being fit in this analysis is a Bayesian linear regression model with time-series data. The formula for the model is:

$$close_t = \beta_0 + \beta_1 close_{t-1} + \epsilon_t$$

- $close_t$ is the dependent variable, representing the close of the AAPL stock on day t.
- $close_{t-1}$ is the independent variable, which is the close of the AAPL stock on the previous day t-1.
- β_0 is the intercept of the regression line, which represents the expected value of r_t, when r_{t-1} is zero.
- β_1 is the slope coefficient, indicating how much r_t is expected to increase when r_{t-1} increases by one unit.
- ϵ_t is the error term, which accounts for the variability in $return_t$ that is not explained by r_{t-1}.

The Bayesian aspect of this model comes from the use of priors and the Bayesian inference process. Instead of just finding point estimates for β_0 and β_1 as in classical regression, the Bayesian approach

estimates the entire posterior distributions for these parameters based on the prior distributions and the observed data.

The following sections will delve into the steps taken for this analysis.

4.1 Data Acquisition

The `tq_get` function from the `tidyquant` package retrieves historical stock price data for Apple Inc (APPL) from an online source.

```
set.seed(555)

# Get Apple stock data
stocks <- tq_get("AAPL", get = "stock.prices", from = "2013-01-01")

# Help make sure we don't accidentally add new data
stocks <- stocks[stocks$date <= as.Date('2023-11-17'),]

# Store pre-amp
stocks$close_preamp <- stocks$close
```

4.2 Exploratory Data Analysis

4.2.1 Examining APPL

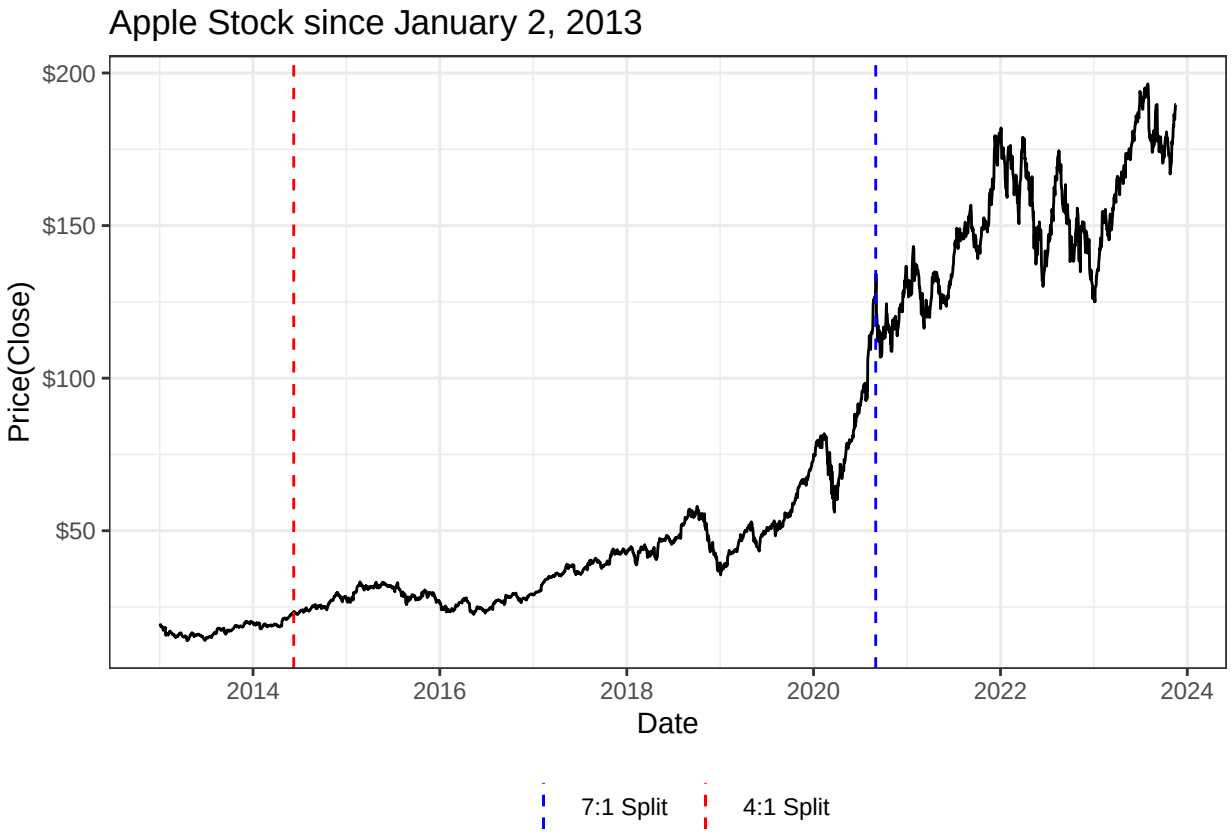


Figure X. Historical trend of APPL

With the help of Tidyquant, historical results can be provided for APPL since January 2, 2013. Overall there has been an upward trend in the daily high stock price. Within this time frame, the stock price started off at $19.60 and closed at $189.69 on November 17, 2023 which is a 162.54% increase in only a decade of data.

During this time frame, the stock split twice. The first occurred on June 9, 2014 where the stock split 7:1. So the number of shares was multiplied by 7 and saw a share price divided by 7. Following this split, there was not much of a jump in price in the time following this split. The second stock 4:1 stock split on August 31, 2020 saw an upward surge in price in the weeks leading up to and after the stock split.

4.3 Data Amputation Prep and Procedure

A subset of the retrieved data is selected, focusing on the 'close' (closing price) and 'volume' (number of shares traded) columns.

```
# Select multiple columns for the amputation process
stocks_for_ampute <- stocks[, c("close", "volume")]
```

The `set.seed` function sets the random number generator's seed to ensure reproducibility of the results. The mice::ampute function artificially introduces missing values into the dataset to simulate incomplete data, which is common in real-world scenarios. The few key arguments are used as follows:

- The prop argument specifies the proportion of data to be made missing
- The mech argument specifies the missingness mechanism as 'MAR' (Missing At Random)

```
# Introduce missing values
amputed_data <- mice::ampute(stocks_for_ampute, prop = 0.3, mech = "MAR")

# Update the stocks with the amputed data for 'close'
stocks$close <- amputed_data$amp[, "close"]

# Add flag for complete data.
# 1 - complete data, 0 - incomplete data
stocks$r <- ifelse(is.na(stocks$close), 0, 1)

# Ensure all necessary columns are present after amputation
stocks <- as.data.frame(stocks)
```

4.4 Data Preparation Post-Amputation

The stocks dataframe is updated with the amputated 'close' prices.

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```

Figure X. Visualization of the amputated data

In the above figure, one can visually see how the mice package amputated the data. Although it could be hard to see due to the visualization, it appears there is a sizable gap in data a few months prior to the year 2021. Again, this was done at random.

4.5 Data Imputation via Predictive Mean Matching

The `mice` function is used to impute missing values in the dataset. It uses Predictive Mean Matching ('pmm') method to fill in missing 'close' prices. Multiple imputed datasets are generated (specified by m = 5) to account for the uncertainty of the imputation process.

```
# Impute missing values using Predictive Mean Matching

stock_subset <- stocks[, c("symbol", "date", "close")]

imputed <- mice(data = stock_subset,
                m = 5,
                maxit = 25,
```

```
                method = 'pmm',
                seed = 555,
                printFlag = FALSE)
```

```
## Warning: Number of logged events: 1
```

```
# Extract imputed datasets
imp_datasets <- lapply(1:5, function(i) complete(imputed, i))

View(imp_datasets)
```

The PMM Process for this mice (Multivariate Imputation by Chained Equations) function works as follows:

i) Fitting of the initial model: We have our variables generically as X (predictors) and Y(variable with missing values). We fit a regression model:

$$\hat{Y} = f(X;\theta), \text{ where } \theta \text{ are the estimated parameters}$$

ii) For a missing Y_i, we find the set of cases(called donors in the package) D_i where $\hat{Y}_d \in D_i$ are closest to $\hat{Y}_i$.

iii) Randomly select a Y_d from D_i to impute Y_i. There is the option to do a straight imputation or choose n Y_d and take an average. Straight imputation was performed in this analysis and the selection from this set D gives each candidate an equal chance of being selected. Upon exploring the mice documentation, this probability process was akin to a uniform probability distribution over the pool of potential donors.

The PMM method is a standalone, non-Bayesian imputation method. The imputed datasets generated are then used further using Bayesian methods in this project. By creating multiple imputations (m = 10), `mice` acknowledges and represents the uncertainty inherent in the imputation process.

4.5.1 Analyzing Imputated Data

First, we will add the results of the 5 imputed datasets back into the main 'stocks' dataframe.

```
stocks$imp1 <- imp_datasets[[1]]$close
stocks$imp2 <- imp_datasets[[2]]$close
stocks$imp3 <- imp_datasets[[3]]$close
stocks$imp4 <- imp_datasets[[4]]$close
stocks$imp5 <- imp_datasets[[5]]$close
```

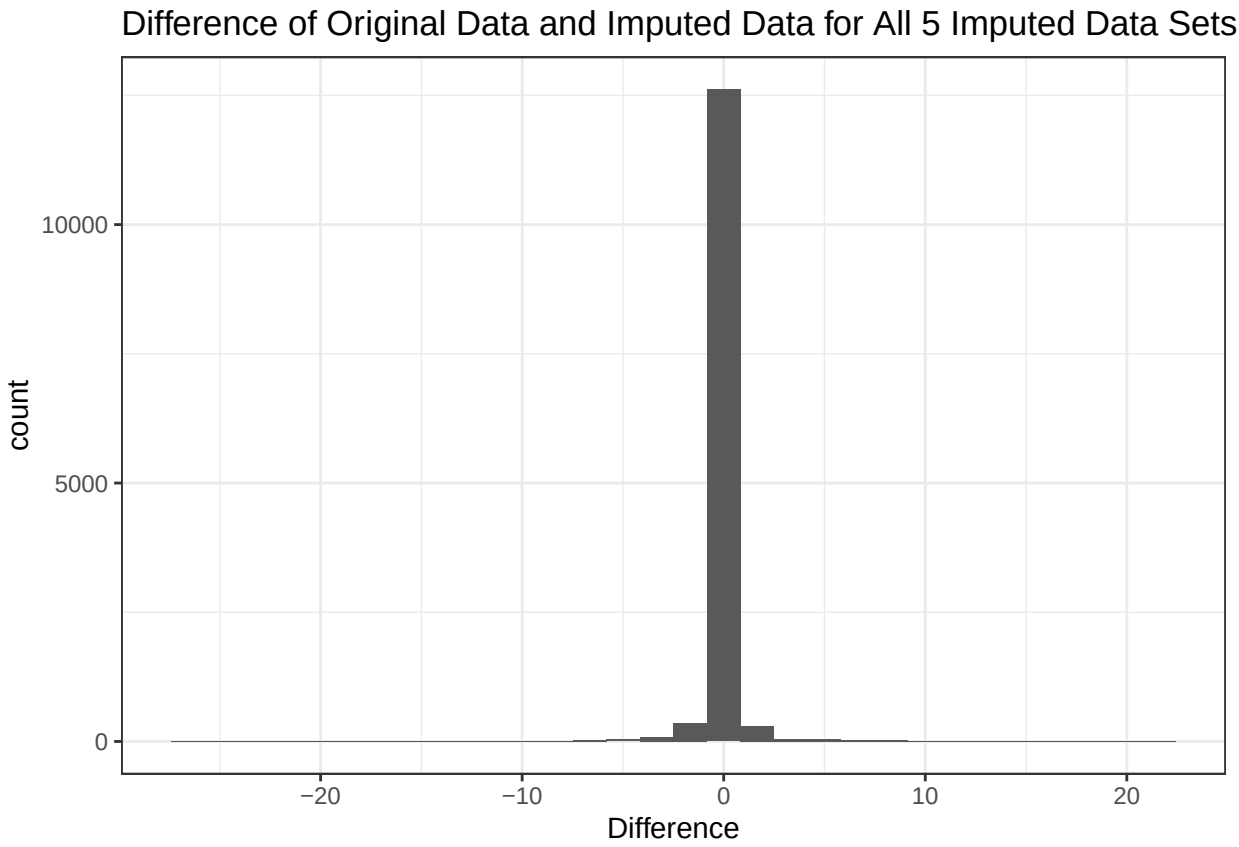


Figure X. Histogram of the difference of the actual close price vs the imputated data points

Figure X shows the difference from the original score vs the imputated score for all 5 multiply imputed datasets combined into one visualiztion. As one can see, the vast majority of points had a very small difference between the actual and the imputated, showing that the imputation was a success. The appendix hosts histograms of differences for all 5 data sets individually.

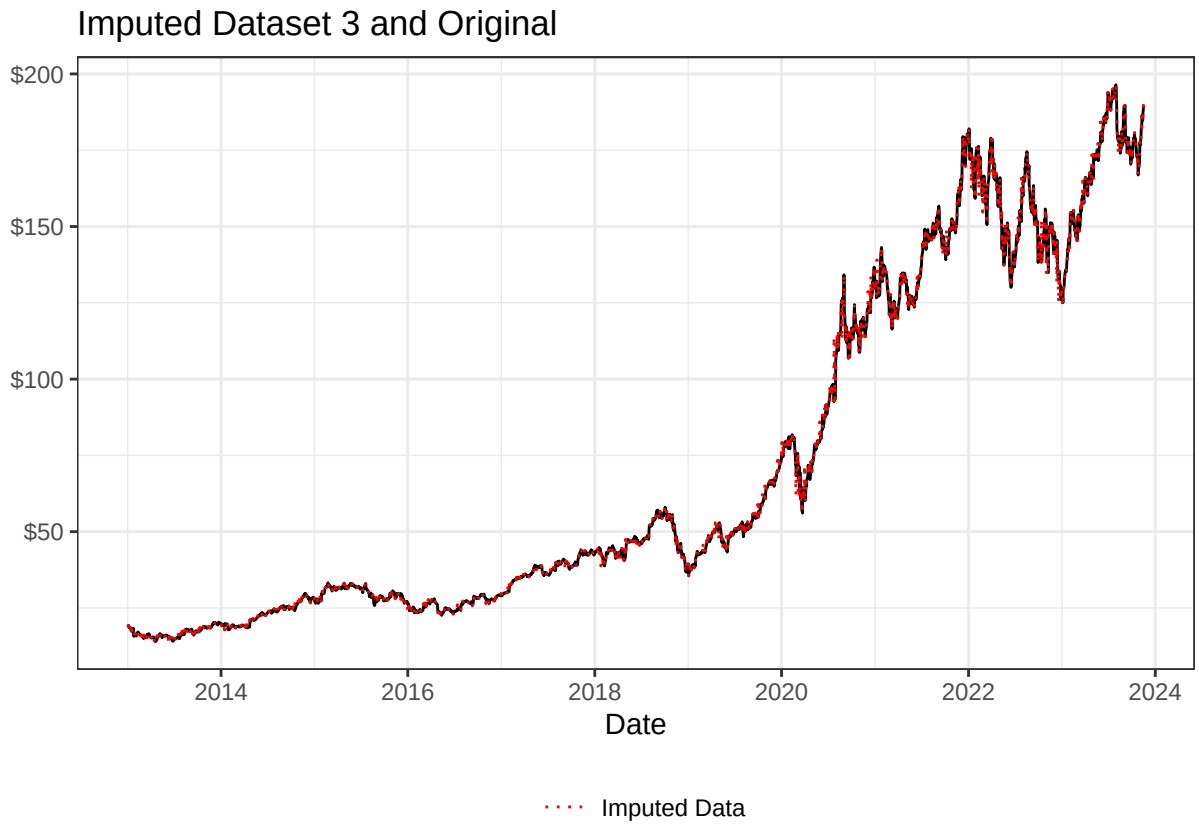


Figure X. Imputed results for dataset 3 overlayed the actual close price

Figure X shows the original data overlayed with the imputed data results for the third imputed dataset. The third dataset happened to be the imputed dataset with the absolute difference between the actual data and the imputed dataset being the third largest. This means there exists results where the difference was actually smaller and worse than this. The other 4 imputed dataset results can be viewed in the appendix.

4.6 Bayesian Regression Model

4.6.1 Setting Priors

Priors are defined for the Bayesian regression model using the `prior` function. The priors reflect our beliefs about the parameters before seeing the data. In this case, normal priors are set for the intercept and slope (return and lagged_return coefficients).

```
# Define the priors for the Bayesian regression model
priors <- c(
  prior(normal(0, 2), class = "Intercept"), # Intercept Prior
  prior(normal(0, 1), class = "b") # Slope Prior
```

```
)
```

- Intercept Prior: A normal distribution with a mean of 0 and a standard deviation of 2. This is specified by prior(normal(0, 2), class = "Intercept"). A prior such as this indicates that before looking at the data, an analyst might believe the average daily return when the lagged return is zero is likely to be around 0 but would like to reflect some uncertainty, which is captured by the standard deviation of 2.
- Slope Prior: A normal distribution with a mean of 0 and a standard deviation of 1. This is specified by prior(normal(0, 1), class = "b"). It suggests that before analyzing the data, an analyst might expect the impact of the previous day's return on today's return to be small, as indicated by the mean of 0. The standard deviation of 1 reflects uncertainty about this expectation.

4.6.2 Fitting a Bayesian Regression Model

The `brm` function from the `brms` package fits a Bayesian regression model to each imputed dataset. The models predict 'return' based on 'lagged_return'. The `iter`, `warmup`, and `chains` arguments control the Markov Chain Monte Carlo (MCMC) sampling process used to estimate the posterior distribution of the model parameters.

The posterior distribution combines the prior distribution with the likelihood of the observed data to update our beliefs about the model parameters. After fitting the model using `brm`, which internally uses Hamiltonian Monte Carlo (HMC) sampling, a type of MCMC, one can obtain a distribution for each parameter that reflects all the information from the priors and the data.

```
# Initialize an empty list to store the models
fits <- vector("list", length = 5)

# Fit a Bayesian regression model to each imputed dataset and check for errors
for (i in seq_along(imp_datasets)) {
  dataset <- imp_datasets[[i]]
  # Create lag variable
  dataset$lagged_close <- c(NA, head(dataset$close, -1))
  fit <- tryCatch(
    {
      model <- brm(close ~ lagged_close, data = dataset,
                   iter = 750, warmup = 500, chains = 2, prior = priors)
      model # Return the model
    },
```

```
    error = function(e) {
      cat(sprintf("Error in fitting model %d: %s\n", i, e$message))
      NULL # Return NULL if there was an error
    }
  )
  fits[[i]] <- fit
}
```

```
## Warning: Rows containing NAs were excluded from the model.

## Compiling Stan program...

## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.53 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 1: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 1: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 1: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 1: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 1: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 1: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 1: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 1: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 1: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 1: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 1: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.095 seconds (Warm-up)
## Chain 1:                0.04 seconds (Sampling)
## Chain 1:                0.135 seconds (Total)
## Chain 1:
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.8e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 2: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 2: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 2: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 2: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 2: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 2: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 2: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 2: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 2: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 2: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 2: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.095 seconds (Warm-up)
## Chain 2:                0.085 seconds (Sampling)
## Chain 2:                0.18 seconds (Total)
## Chain 2:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and me
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances ar
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess

## Warning: Rows containing NAs were excluded from the model.

## Compiling Stan program...
## Start sampling

##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.44 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 1: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 1: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 1: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 1: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 1: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 1: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 1: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 1: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 1: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 1: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 1: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.113 seconds (Warm-up)
## Chain 1:                0.04 seconds (Sampling)
## Chain 1:                0.153 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 5.6e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.56 seconds
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 2: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 2: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 2: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 2: Iteration: 300 / 750 [ 40%]  (Warmup)
```

```
## Chain 2: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 2: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 2: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 2: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 2: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 2: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 2: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.098 seconds (Warm-up)
## Chain 2:                0.052 seconds (Sampling)
## Chain 2:                0.15 seconds (Total)
## Chain 2:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and me
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances ar
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess

## Warning: Rows containing NAs were excluded from the model.

## Compiling Stan program...
## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 6.4e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.64 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 1: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 1: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 1: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 1: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 1: Iteration: 375 / 750 [ 50%]  (Warmup)
```

```
## Chain 1: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 1: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 1: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 1: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 1: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 1: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.116 seconds (Warm-up)
## Chain 1:                0.062 seconds (Sampling)
## Chain 1:                0.178 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 2: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 2: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 2: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 2: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 2: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 2: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 2: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 2: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 2: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 2: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 2: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.108 seconds (Warm-up)
## Chain 2:                0.048 seconds (Sampling)
## Chain 2:                0.156 seconds (Total)
## Chain 2:

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances ar
```

```
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess

## Warning: Rows containing NAs were excluded from the model.

## Compiling Stan program...
## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 7.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.77 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 1: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 1: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 1: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 1: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 1: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 1: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 1: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 1: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 1: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 1: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 1: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.111 seconds (Warm-up)
## Chain 1:                0.045 seconds (Sampling)
## Chain 1:                0.156 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1.9e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds
## Chain 2: Adjust your expectations accordingly!
```

```
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 2: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 2: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 2: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 2: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 2: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 2: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 2: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 2: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 2: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 2: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 2: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.1 seconds (Warm-up)
## Chain 2:                0.049 seconds (Sampling)
## Chain 2:                0.149 seconds (Total)
## Chain 2:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and me
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess


## Warning: Rows containing NAs were excluded from the model.

## Compiling Stan program...
## Start sampling

##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.8e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 1: Iteration:  75 / 750 [ 10%]  (Warmup)
```

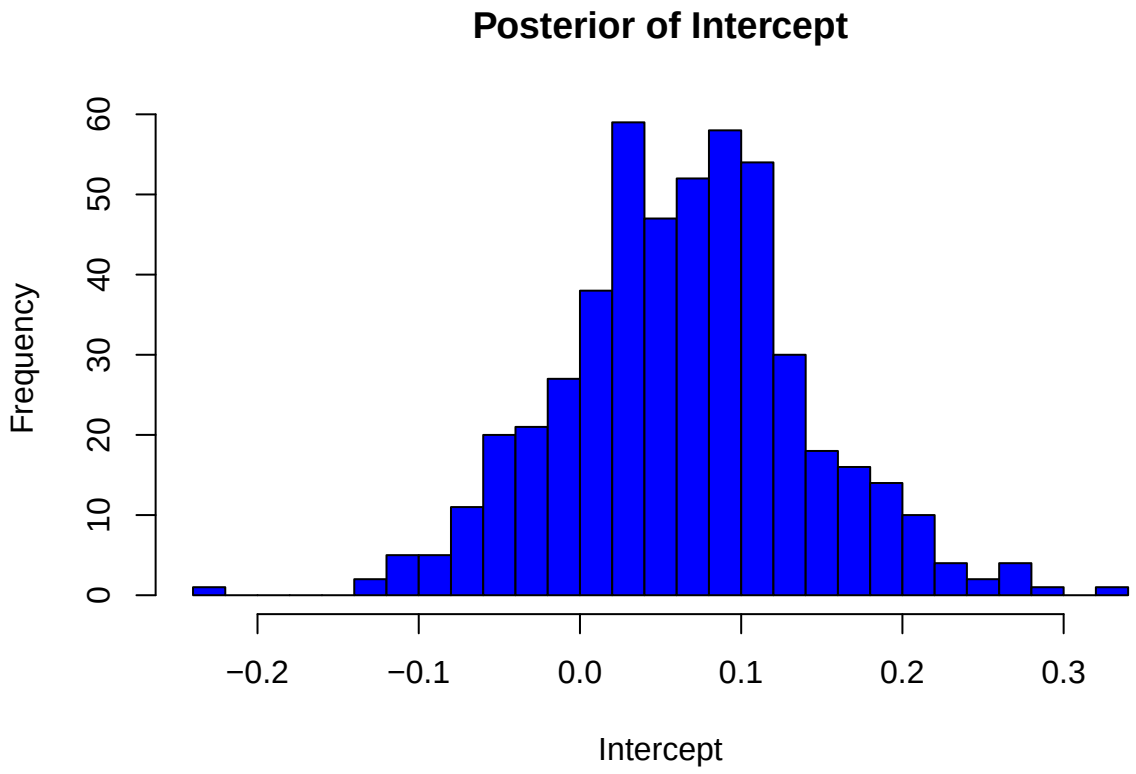
```
## Chain 1: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 1: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 1: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 1: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 1: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 1: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 1: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 1: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 1: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 1: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.078 seconds (Warm-up)
## Chain 1:                0.039 seconds (Sampling)
## Chain 1:                0.117 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.2 seconds
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 750 [  0%]  (Warmup)
## Chain 2: Iteration:  75 / 750 [ 10%]  (Warmup)
## Chain 2: Iteration: 150 / 750 [ 20%]  (Warmup)
## Chain 2: Iteration: 225 / 750 [ 30%]  (Warmup)
## Chain 2: Iteration: 300 / 750 [ 40%]  (Warmup)
## Chain 2: Iteration: 375 / 750 [ 50%]  (Warmup)
## Chain 2: Iteration: 450 / 750 [ 60%]  (Warmup)
## Chain 2: Iteration: 501 / 750 [ 66%]  (Sampling)
## Chain 2: Iteration: 575 / 750 [ 76%]  (Sampling)
## Chain 2: Iteration: 650 / 750 [ 86%]  (Sampling)
## Chain 2: Iteration: 725 / 750 [ 96%]  (Sampling)
## Chain 2: Iteration: 750 / 750 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.075 seconds (Warm-up)
```

```
## Chain 2:                0.03 seconds (Sampling)
## Chain 2:                0.105 seconds (Total)
## Chain 2:
```

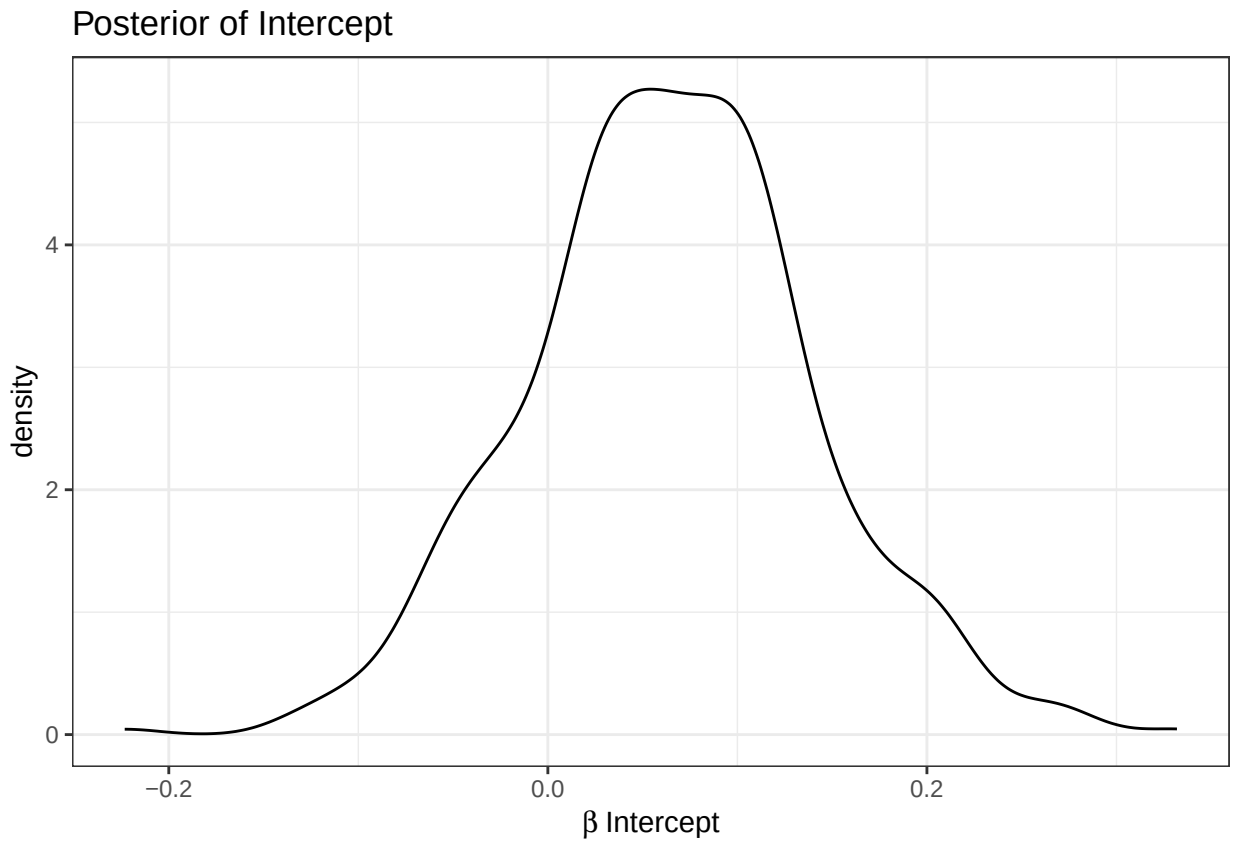
```
# Extract posterior samples for the first fitted model
post_samples <- posterior_samples(fits[[1]], pars = c("Intercept", "lagged_close", "sigma"
```

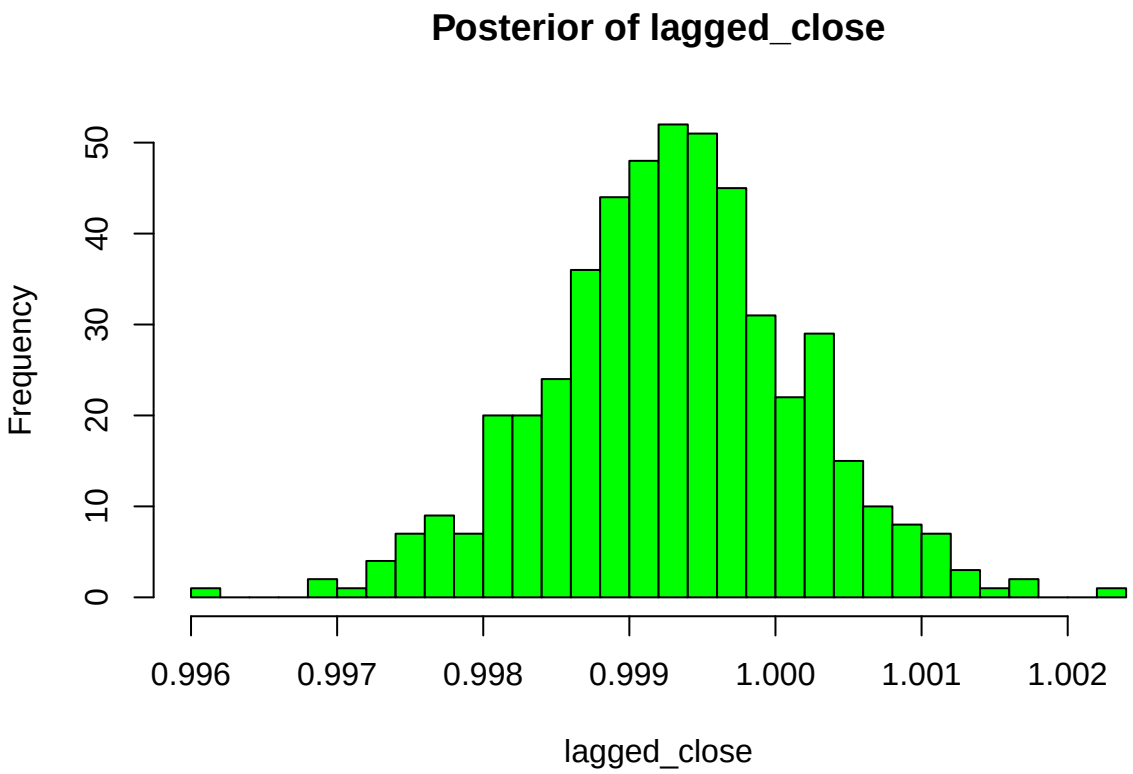
```
## Warning: Method 'posterior_samples' is deprecated. Please see ?as_draws for
## recommended alternatives.
```

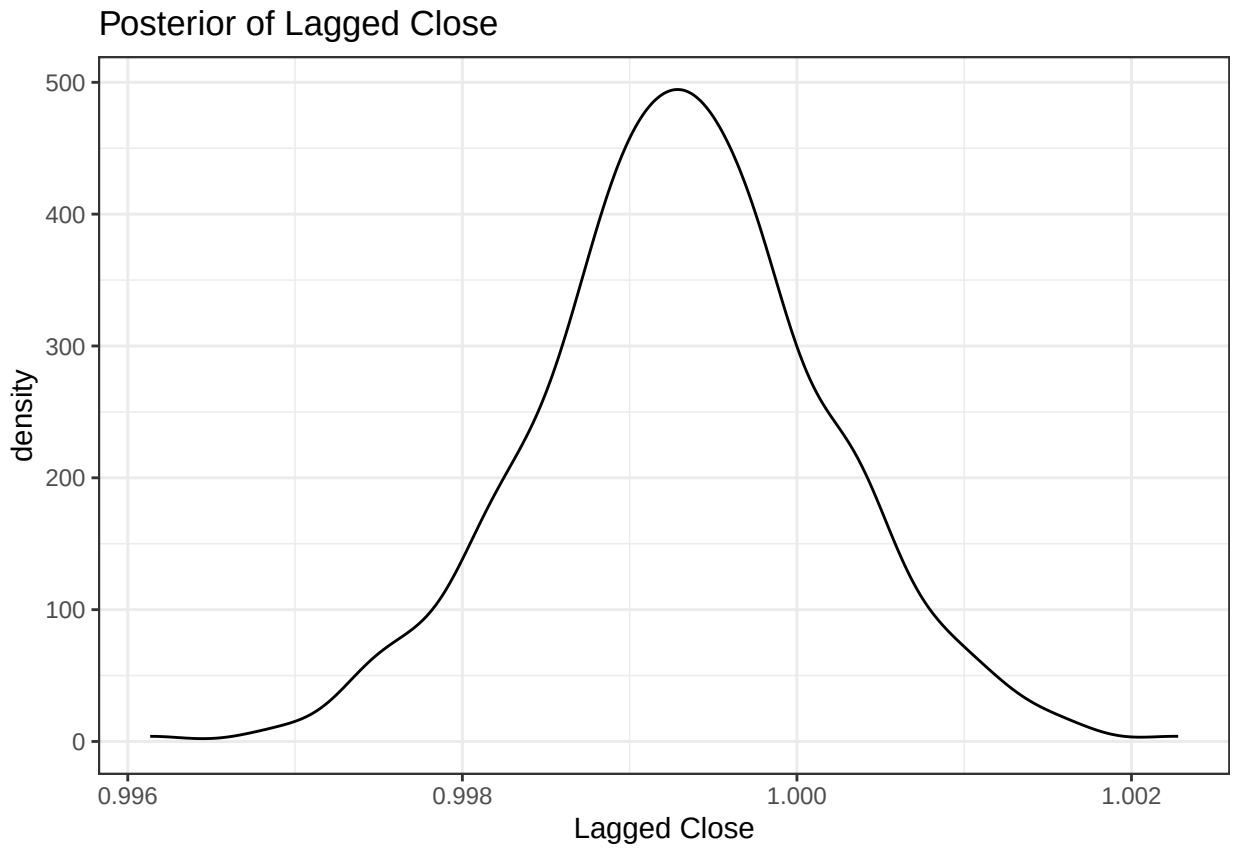
```
# Histogram for the Intercept
hist(post_samples$b_Intercept, breaks = 30, main = "Posterior of Intercept",
     xlab = "Intercept", col = "blue")
```

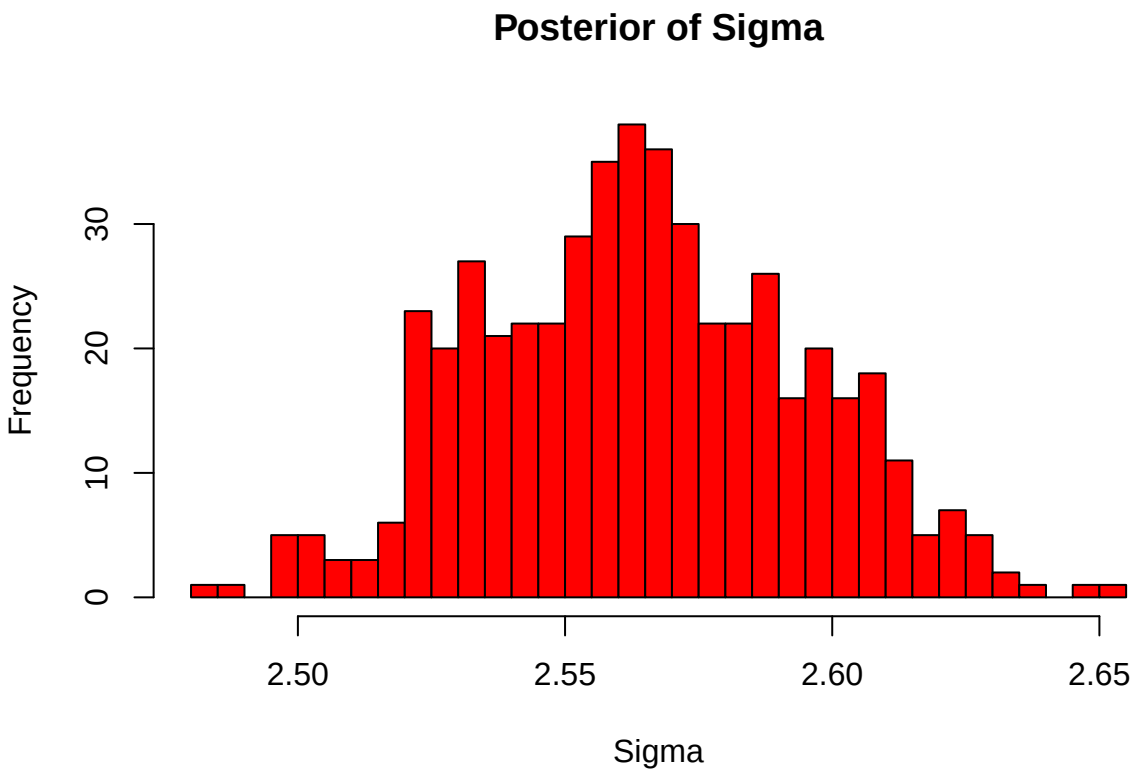


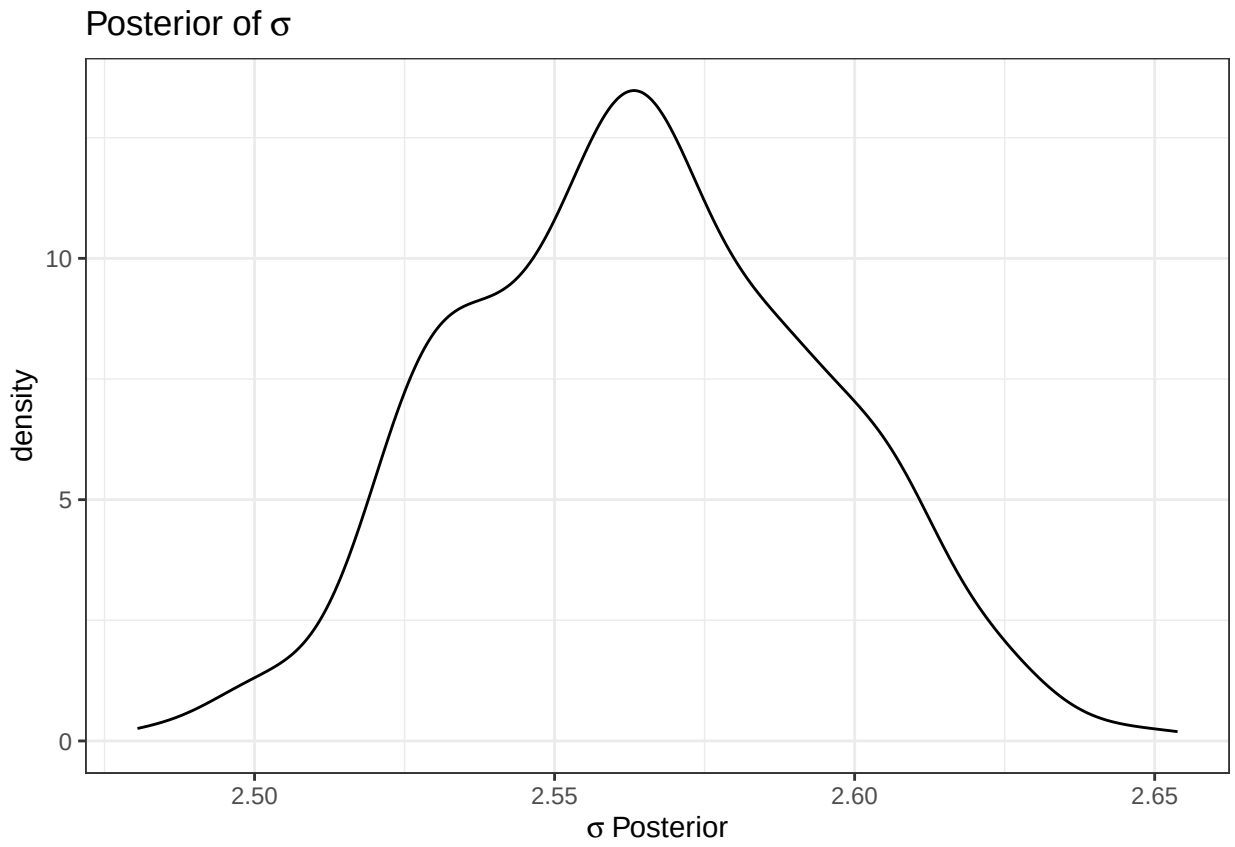

```
density_gg(post_samples, b_Intercept, "Posterior of Intercept", expression(beta ~ 'Interce
```


Posterior of Intercept
density
4
2
0
−0.2
0.0
0.2
β Intercept


Posterior of lagged_close
Frequency
0
10
20
30
40
50
0.996
0.997
0.998
0.999
1.000
1.001
1.002
lagged_close


Posterior of Lagged Close
density
500
400
300
200
100
0
0.996
0.998
1.000
1.002
Lagged Close


Posterior of Sigma
Frequency
0
10
20
30
2.50
2.55
2.60
2.65
Sigma




```r
# Initialize lists to store parameter estimates and standard errors
param_estimates <- list()
param_se <- list()

# Extract parameter estimates and standard errors from each model
for (i in seq_along(fits)) {
  if (!is.null(fits[[i]])) {
    # Extracting estimates
    est <- as.data.frame(summary(fits[[i]])$fixed)[, "Estimate"]
    param_estimates[[i]] <- est

    # Extracting standard errors
    se <- as.data.frame(summary(fits[[i]])$fixed)[, "Est.Error"]
    param_se[[i]] <- se
  }
}

# Calculate the mean of the parameter estimates
combined_estimates <- Reduce("+", param_estimates) / length(fits)
```

```r
# Calculate the pooled standard error if necessary
combined_se <- sqrt(Reduce("+", lapply(param_se, `^`, 2))) / length(fits)

# Combine into a data frame
combined_results <- data.frame(Estimate = combined_estimates, Std.Error = combined_se)

# Print combined results
#print(combined_results)

# Initialize lists to store CIs
ci_lower <- list()
ci_upper <- list()

for (i in seq_along(fits)) {

  if (!is.null(fits[[i]])) {

    # Extract CIs
    ci <- as.data.frame(summary(fits[[i]])$fixed)[, c("l-95% CI", "u-95% CI")]

    ci_lower[[i]] <- ci[,1]
    ci_upper[[i]] <- ci[,2]

  }

}

# Combine CIs

ci_lower_combined <- Reduce("+", ci_lower) / length(fits)
ci_upper_combined <- Reduce("+", ci_upper) / length(fits)

combined_results$ci_lower <- ci_lower_combined
combined_results$ci_upper <- ci_upper_combined

print(combined_results)
```

```
##     Estimate    Std.Error    ci_lower  ci_upper
## 1 0.05693082 0.0364115112 -0.09265884 0.2163004
## 2 0.99943648 0.0004039464  0.99766967 1.0011488
```

```
cat("point estimate for sigma: ", 100)
```

```
## point estimate for sigma:  100
```

```
cat("ci_lower for sigma: ", quantile(post_samples$sigma, 0.025))
```

```
## ci_lower for sigma:  2.508397
```

```
cat("ci_upper for sigma: ", quantile(post_samples$sigma, 0.975))
```

```
## ci_upper for sigma:  2.622535
```

4.7 Comparing with OLS

```
data_all <- tq_get("AAPL", get = "stock.prices", from = "2013-01-01")

data_all$lagged_close <- c(NA, head(data_all$close, -1))



# OLS model
fit_ols <- lm(data_all$close ~ data_all$lagged_close)
summary(fit_ols)
```

```
##
## Call:
## lm(formula = data_all$close ~ data_all$lagged_close)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5970  -0.3942  -0.0234   0.3983  11.9222
##
## Coefficients:
##                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)            0.0451154  0.0513770    0.878     0.38
## data_all$lagged_close 1.0002426  0.0005672 1763.392   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.654 on 2746 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9991, Adjusted R-squared:  0.9991
## F-statistic: 3.11e+06 on 1 and 2746 DF,  p-value: < 2.2e-16
```

4.8 Combining of Model Results

- Parameter estimates and standard errors are extracted from each fitted model. These estimates are combined by calculating the mean estimate and pooled standard error for each parameter across all imputed datasets.

4.9 Summary and Analysis

The combined results are printed out for interpretation.

This analysis allows for Bayesian inference on time-series data with missing values. The choice of prior values should be justified based on domain knowledge or previous research. Since normal priors are used for both the intercept and slope, it implies a belief that the coefficients are likely to be around 0 but allowing for some variation.

The mice::ampute function is crucial because it enables researchers to understand how well their imputation and analysis methods work when some data is missing, which is a common occurrence in real-world datasets.

The brm function is central to the Bayesian approach, as it fits a model within the Bayesian framework using Hamiltonian Monte Carlo, a type of MCMC sampling. The function returns a wealth of information about the model, including estimates of the posterior distribution of the model parameters, which reflects both the data and the priors.

5 Conclusion

This analysis demonstrated a Bayesian approach for handling missing data, specifically using multiple imputation to fill gaps while quantifying uncertainty. The mice R package enabled missing data simulation on the AAPL stock dataset. Five imputed complete datasets were then generated using predictive mean matching. Bayesian regression models were fitted to predict daily returns based on prior day returns, combining prior beliefs about the coefficients with likelihood from the observed data to estimate posterior distributions. Hamilton Monte Carlo sampling enabled convenient fitting of the posteriors. Parameter estimates were finally pooled across the multiple imputations to obtain overall inference, incorporating missing data uncertainty.

The multi-step process provides a blueprint for principled missing data handling - imputing while preserving relationships in the data, fitting appropriate Bayesian models, and aggregating estimates for final inference. The methodology can generalize across applications with missing observations, from financial time series forecasting to political polls, epidemiology, econometrics, or climate data analysis. Missingness often arises when merging multiple datasets as well. The multiple imputation framework allows proceeding with modeling and prevents relying only on complete cases. The Bayesian approach allows flexible incorporation of complex relationships and prior knowledge to strengthen imputations and parameter estimates despite gaps. Use of MCMC sampling handles high-dimensional posteriors despite analytic intractability. Thus Bayesian missing data techniques serve as broadly applicable tools for overcoming an omnipresent analysis challenge to enable robust modeling from incomplete real-world data.

6 Reference

1. Lai, Mark. 2019. "Course Handouts for Bayesian Data Analysis Class." Class handouts, December 13, 2019. https://bookdown.org/marklhc/notes_bookdown/missing-data.html
2. Little, Roderick. 2011. "Calibrated Bayes, for Statistics in General, and Missing Data in Particular." Statistical Science 26 (2): 162–74. https://doi.org/10.1214/10-STS318.

7 Appendix

7.1 Additional Exploratory Data Analysis

7.1.1 Raw Returns Analysis

```
# Calculate raw returns
returns =diff(log(stocks$close_preamp))

# Plot time series
plot(returns, type="l", main="Raw Apple Returns Since January 2, 2013")
```

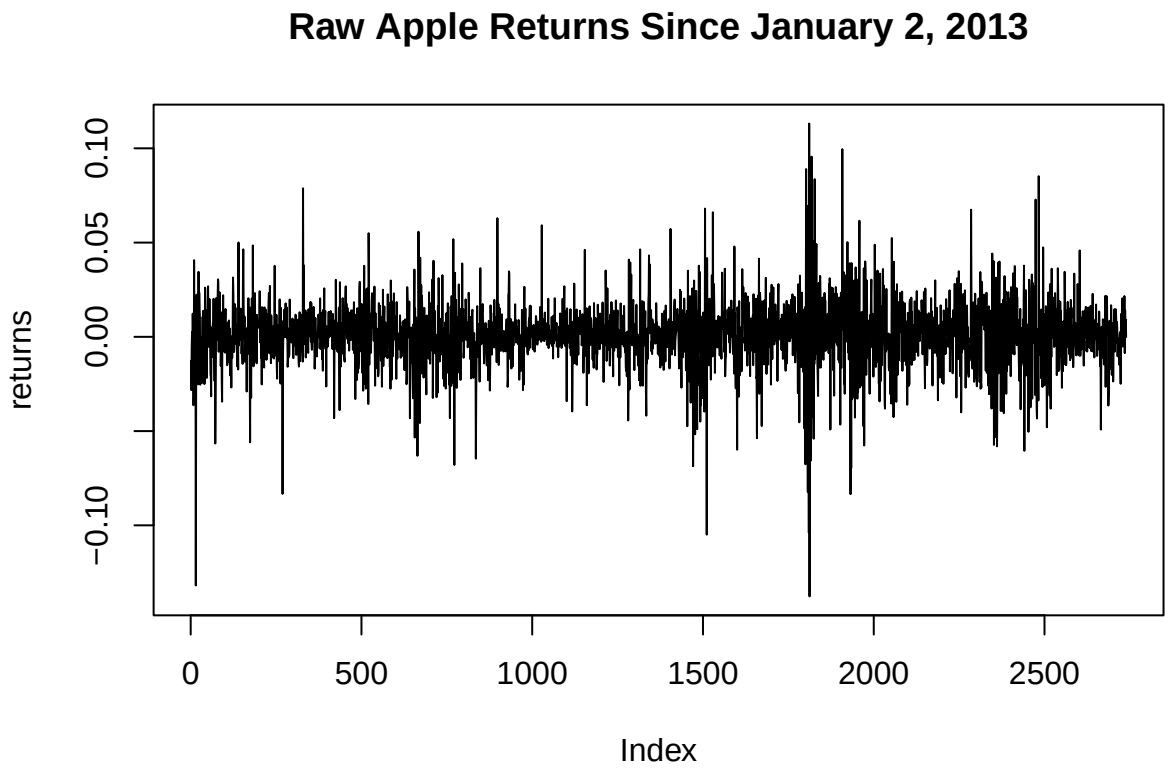


Figure X. Historical Trend of Raw Returns for APPL since January 1, 2013

In the above figure, one can see the historical trend of raw returns for APPL since January 1, 2013. Taking the log of the returns is a common application in fincance used for volatility analysis. Applying the 'diff' function calcualtes the difference between consecutive element. Within the context of this analysis, the function will compute the daily log returns by calculating the difference between each day's log closing price and the previous day closing's prices.

```
# Calculate autocorrelations
acf_returns = acf(returns, lag.max=5, plot=FALSE)

# Plot ACF
plot(acf_returns, main="ACF Plot of Raw Returns")
```

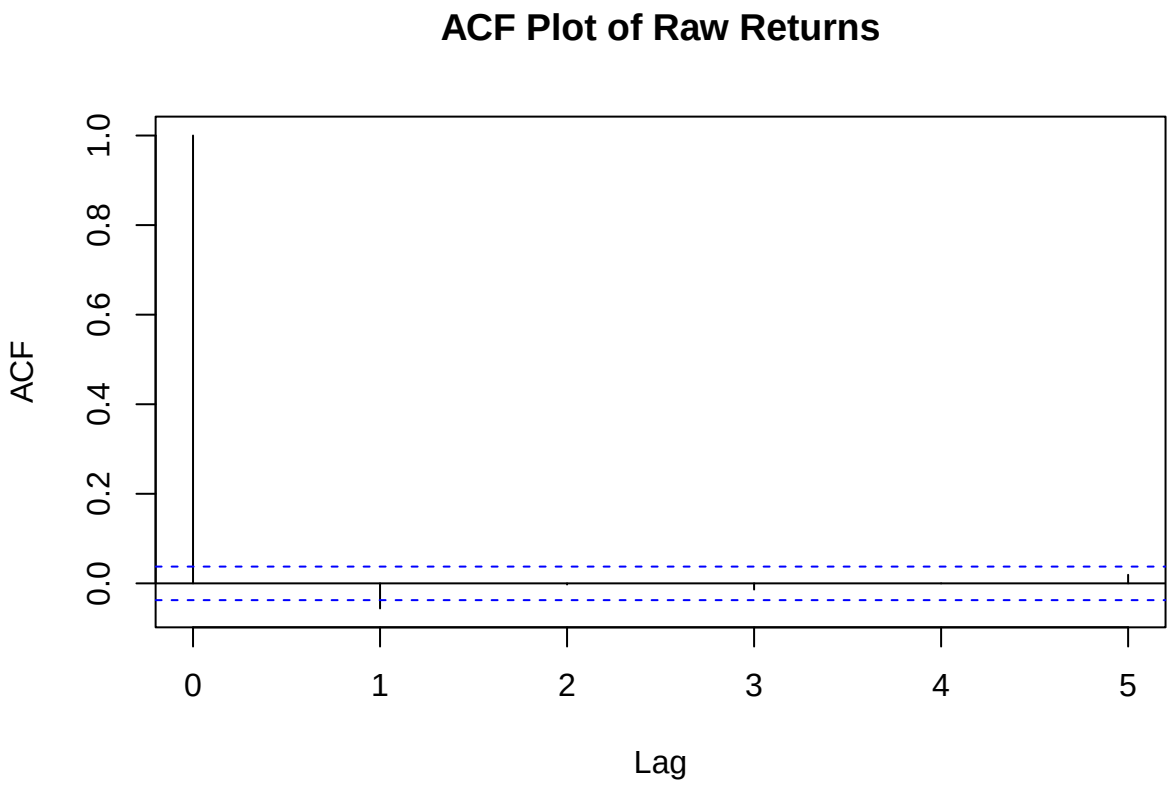


Figure X. ACF Plot of raw returns

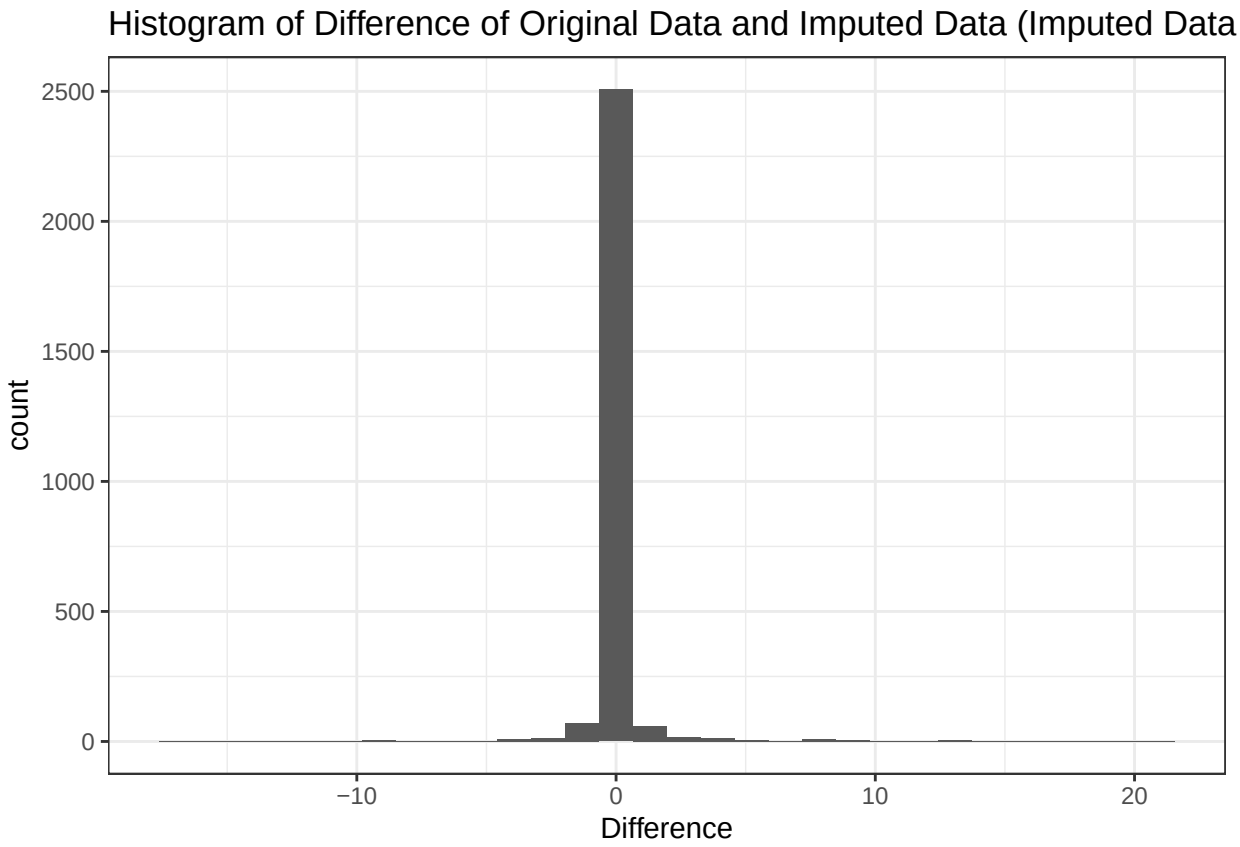
With the ACF plots shown in the above figure, the autocorrelation structure of the raw returns. With the autocorrelation falling within the blue horizontal bands, this would mean that the there is independence within the log returns.

7.2 Imputed Data Results

7.2.1 Histograms for the Difference of Original Data and Imputed Data

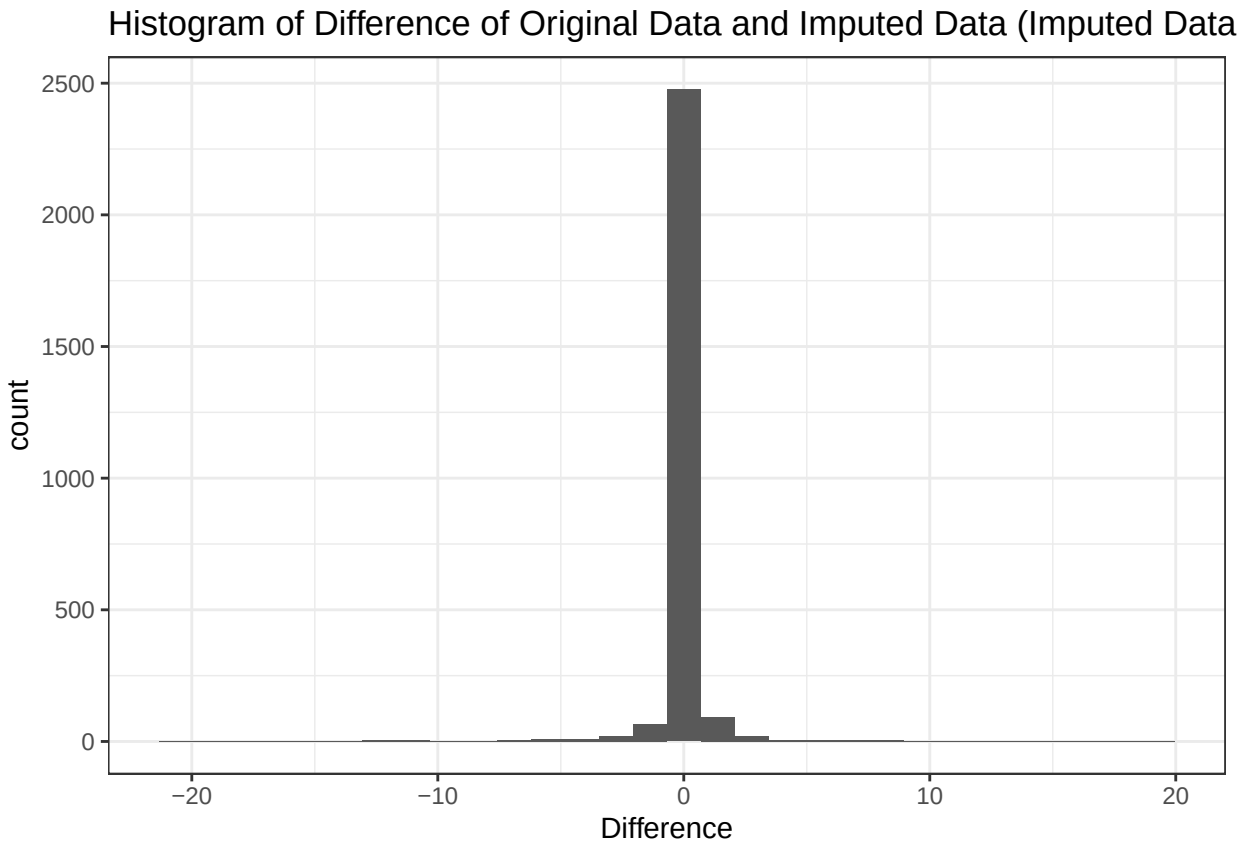
```
hist_gg(stocks, close_preamp - imp1, title = "Histogram of Difference of Original Data and
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



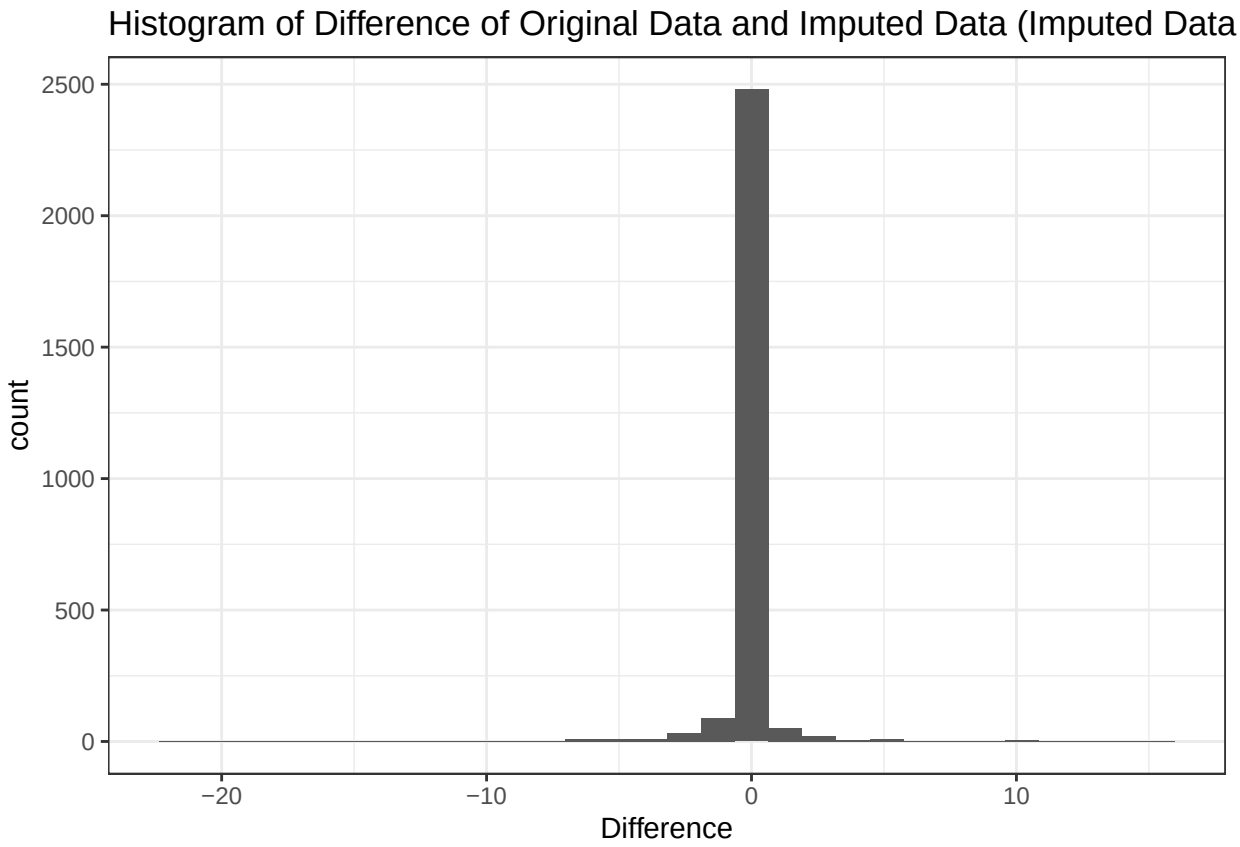

```
hist_gg(stocks, close_preamp - imp2, title = "Histogram of Difference of Original Data and
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



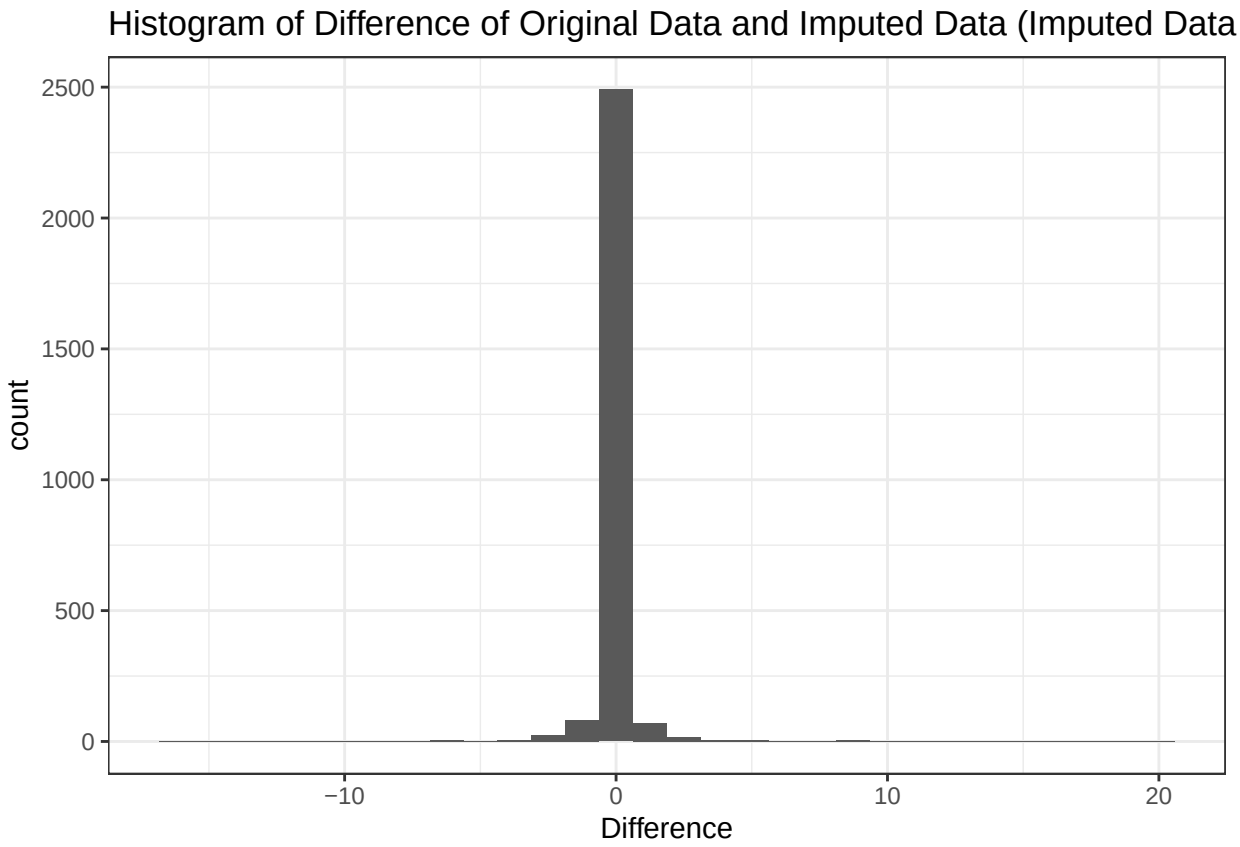

```
hist_gg(stocks, close_preamp - imp3, title = "Histogram of Difference of Original Data and
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



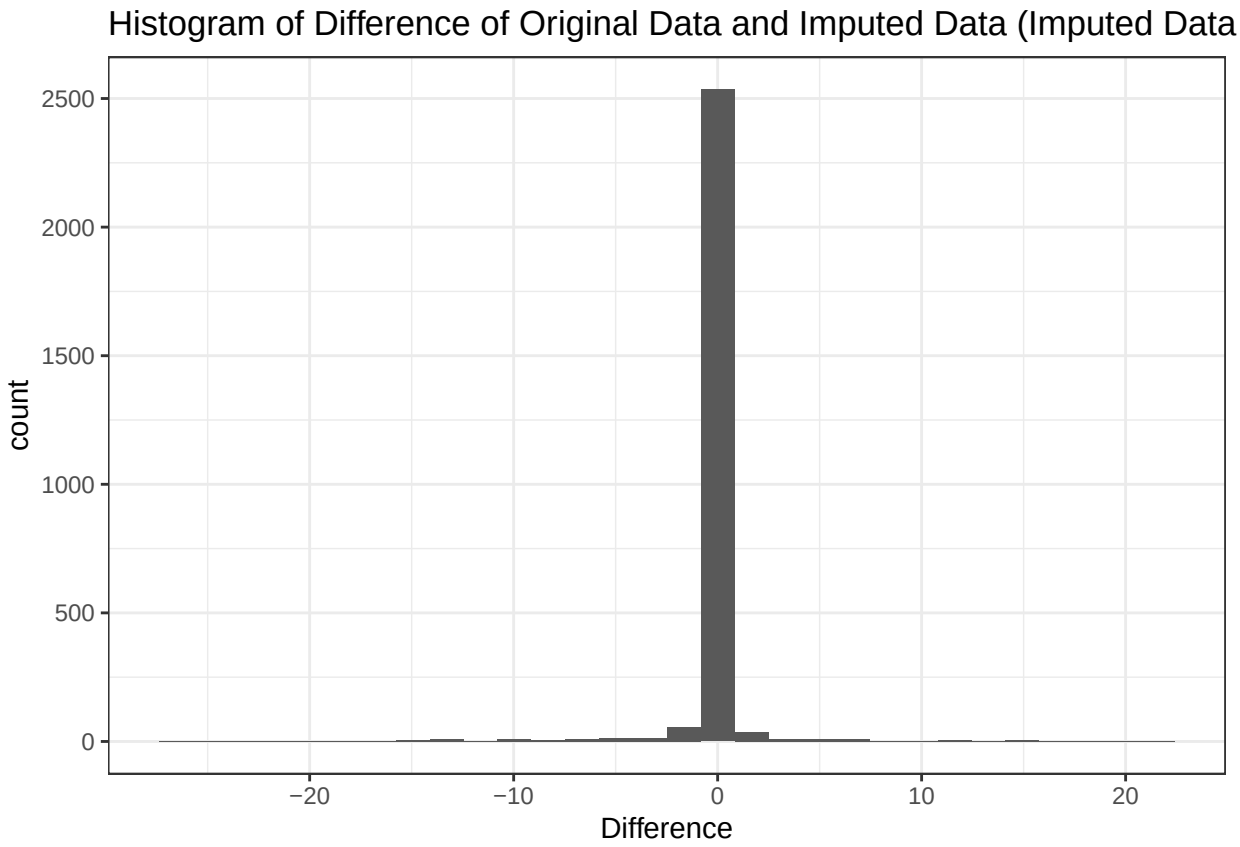

```
hist_gg(stocks, close_preamp - imp4, title = "Histogram of Difference of Original Data and
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



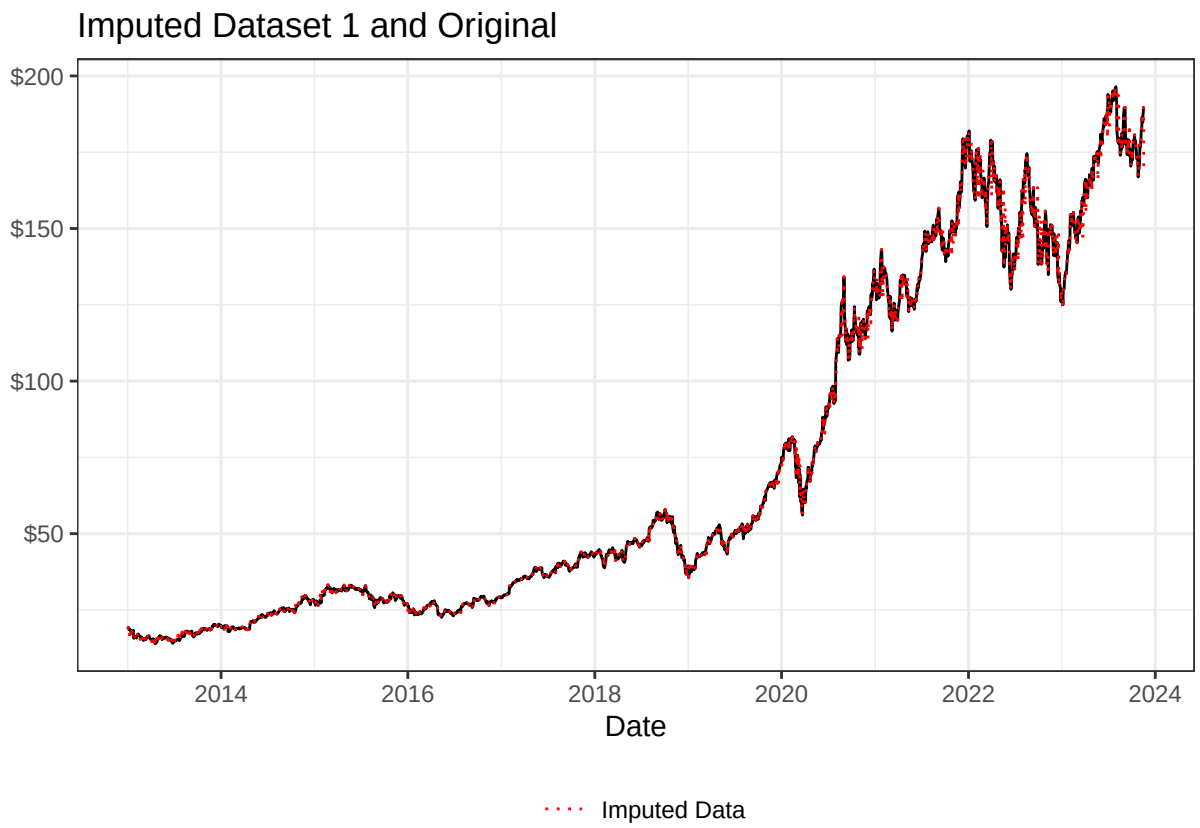

```
hist_gg(stocks, close_preamp - imp5, title = "Histogram of Difference of Original Data and
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

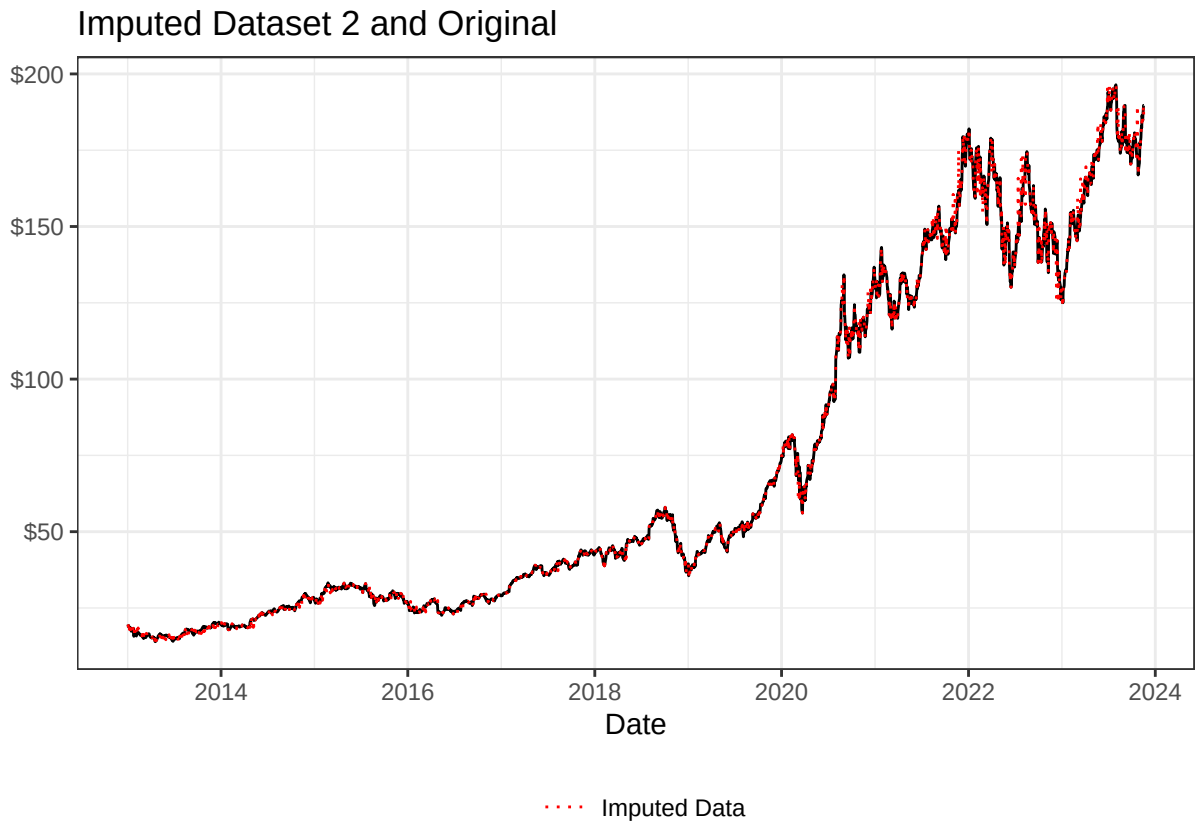


7.2.2 Graph of Imputed Data and Original Data

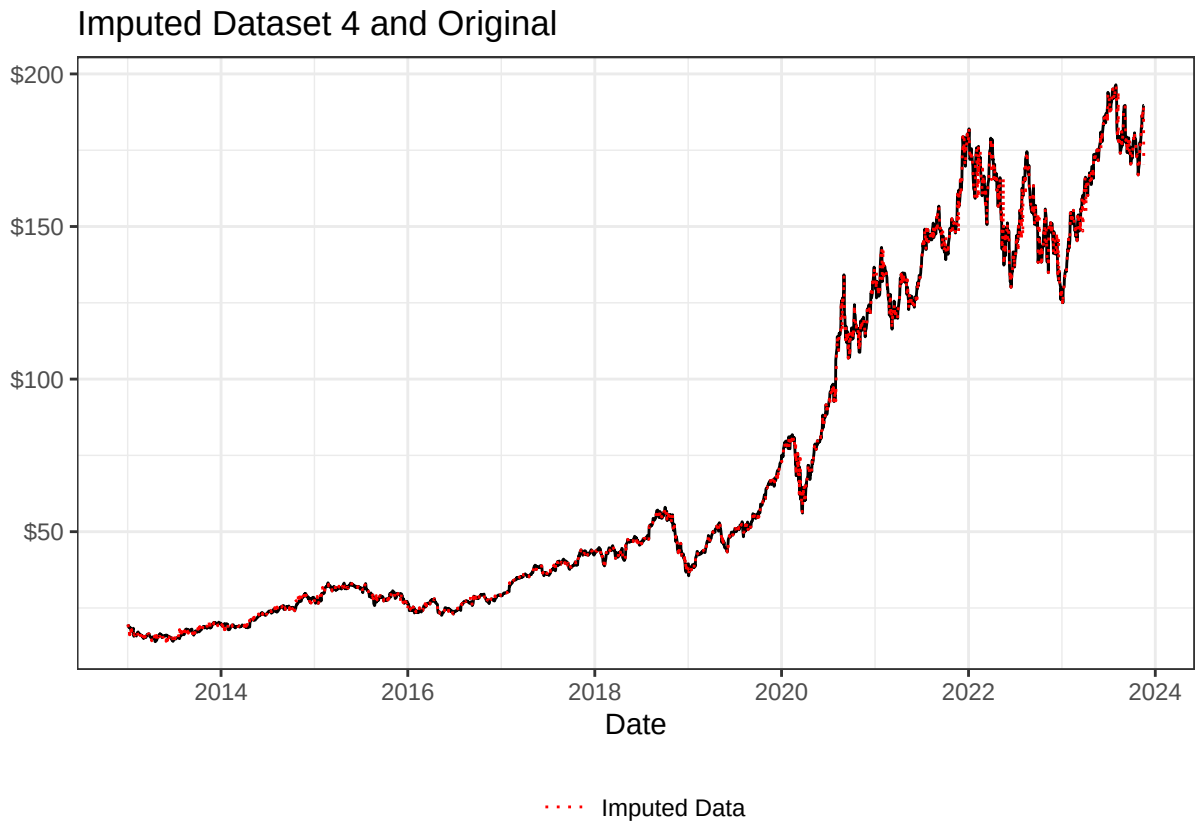
```
ts_line(stocks, "date", "close_preamp", title = 'Imputed Dataset 1 and Original') + geom_l
  # Adding a legend at the bottom with no title
  scale_color_manual(values = c("Original" = "black", "imputed" = "red"), labels = c("Impu
  theme(legend.position = "bottom", legend.title = element_blank())
```



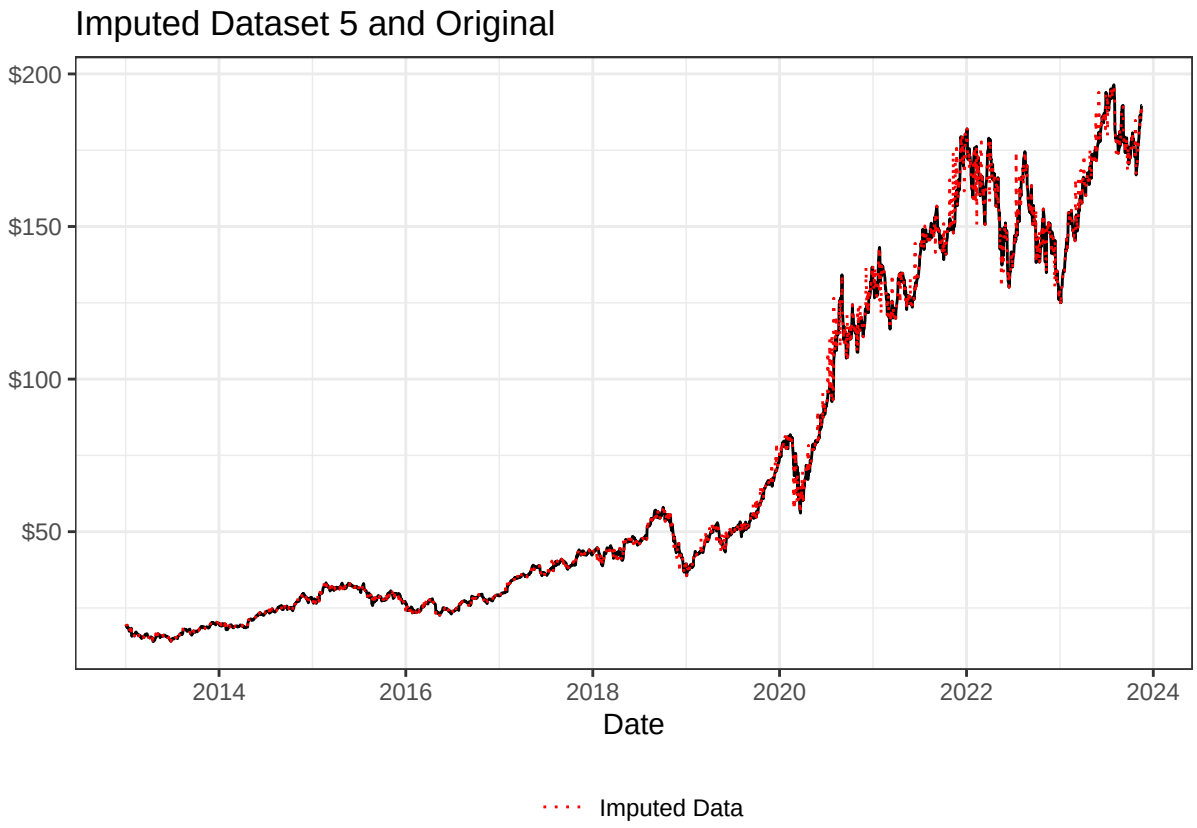

```
ts_line(stocks, "date", "close_preamp", title = 'Imputed Dataset 2 and Original') + geom_l
  # Adding a legend at the bottom with no title
  scale_color_manual(values = c("Original" = "black", "imputed" = "red"), labels = c("Impu
  theme(legend.position = "bottom", legend.title = element_blank())
```




```
ts_line(stocks, "date", "close_preamp", title = 'Imputed Dataset 4 and Original') + geom_l
  # Adding a legend at the bottom with no title
  scale_color_manual(values = c("Original" = "black", "imputed" = "red"), labels = c("Impu
  theme(legend.position = "bottom", legend.title = element_blank())
```




```
ts_line(stocks, "date", "close_preamp", title = 'Imputed Dataset 5 and Original') + geom_l
  # Adding a legend at the bottom with no title
  scale_color_manual(values = c("Original" = "black", "imputed" = "red"), labels = c("Impu
  theme(legend.position = "bottom", legend.title = element_blank())
```


Imputed Dataset 5 and Original
$200
$150
$100
$50
2014
2016
2018
2020
2022
2024
Date
Imputed Data