Bayesian Time Series Analysis of US Unemployment Rates

STAT 447C Final Project - Project Report | Due Date: 21st April 2024

Rohan Joseph (67089839)

Project GitHub URL: https://github.com/RohanUBC/STAT-447C-Final-Project

Introduction

The forecasting of United States (US) unemployment rates presents a complex and significant challenge for US policy makers. Accurate predictions of these rates enables the US government and businesses to make informed economic decisions, adapt to economic shifts, and implement strategies to foster stability and growth in the US economy. Furthermore, the unpredictability of labor markets, influenced by factors ranging from global economic conditions to domestic policy changes, necessitates the use of sophisticated methods not only provide a framework for forecasting, but also allow for the quantification of uncertainty in those forecasts. Understanding and predicting unemployment rates is not only central to economic analysis but also to the well-being of the US economy.

The focus of this project is to model and forecast unemployment rates in the US, utilizing a real-world dataset that has documented the monthly unemployment rate (from 01/1948 to 02/2024) in the US. A further aim is to quantify the uncertainty of the forecast. This is done to ensure that the forecasts are not only precise, but are also accompanied by reliable uncertainty measures. This approach aims to assess potential risks effectively, thereby facilitating more informed and prudent economic decision making. The dataset that is being used is from the US Federal Reserve on monthly US unemployment rates.

URL: https://fred.stlouisfed.org/series/UNRATE

A Bayesian time series analysis approach was adopted, utilizing a Bayesian Structural Time Series (BSTS) model to capture the dynamics of the US labor market based on the provided historical data. This involves uncovering and modeling the historical trends and seasonal patterns within the data, in order to provide probabilistic forecasts of future unemployment rates. A BSTS model was used due to its flexibility in modeling complex time series data with trend and seasonality components, while providing a framework for forecasting future rates. Please see Appendix C to see a general overview on BSTS models.

This project will face challenges in applying Bayesian time series analysis to economic data, particularly in modeling the seasonal patterns and long-term trends inherent in unemployment rates. The primary challenge lies in identifying the specific type of model, as well its specific components. This would require diving into the inherent structure of the data, while also validating the model's performance by analyzing error metrics and posterior predictive checks.

Literature Review

Bayesian forecasting techniques have increasingly been recognized for their efficacy in economic predictions. In his 2013 paper, "Bayesian Model Averaging and exchange rate forecasts", Jonathan H. Wright explored how Bayesian Model Averaging (BMA) could be used to effectively manage a model's uncertainty, a crucial factor in improving forecast accuracy across various economic indicators. Wright's research demonstrated that forecast models could maximize prediction reliability and minimize potential biases caused by model selection by using BMA to evaluate multiple models according to their historical accuracy (Wright).

In addition, Abdullah M. Almarashi and Khushnoor Khan, in their 2020 paper "Bayesian Structural Time Series", explored the application of Bayesian Structural Time Series (BSTS) models on stock data utilizing a combination of a Kalman filter and Monte Carlo Markov Chain (MCMC) simulations. Their research demonstrated that BSTS models could be efficiently used in diverse fields, including complex engineering processes, by incorporating external regression information and probabilistic elements into a forecasting model (Almarashi and Khan).

Wright, Almarashi, and Khan's work demonstrated that that BSTS models not only provides a viable alternative to classical forecasting models but also enhances the predictive capabilities through the integration of advanced statistical techniques.

However, both of these research focused primarily on general economic forecasting rather than on unemployment rates specifically. This project aims to fill this niche, specifically focusing on adapting Bayesian forecasting to closely analyse and predict the nuanced patterns in US unemployment rates. Furthermore, there is a clear need for any model to effectively quantify and communicate forecast uncertainty. By providing a quantification of forecast uncertainty, this project aims to ensure that the probabilistic forecasts are both reliable and comprehensible for economic stakeholders.

Data Analysis

Framework of Model

As established in Appendix B, a general BSTS model consists of two sets of equations: the observational (response variable) and the state-space equations (how the parameters change over time). In order to set these, first a decomposition of a time series of the data would need to be done, including generating an autocorrelation (ACF) plot and a partial autocorrelation (PACF) plot of the time series. This would also help in deciding want components of a time series should be included in the model (for e.g., trend, seasonality, auto-regressive process, etc.). All of this will be done on the training set (see Appendix B).

NOTE: All the data up until 02/2023 will be used as the training set, and the remaining data (from 03/2023 to 02/2024) as the testing set (see Appendix B - Section 1). The test set is composed of one year of monthly US unemployment rates.

Key Definitions:

Decomposition of a Time Series: This involves separating a time series into its constituent components, such as trend, seasonality, and noise, to better understand its underlying patterns and behaviors.

Autocorrelation: This is the correlation between a time series and its own lagged values, revealing how past observations relate to present or future values within the same series.

Autocorrelation Function (ACF) Plot: This is a visual representation of the correlation between a time series and its lagged values at different lags, which helps to identify patterns of autocorrelation and inform the selection of appropriate models for time series analysis.

Partial Autocorrelation Function (ACF) Plot: This shows the correlation between observations at different time lags, accounting for the influence of shorter lags, helping identify the direct effects of specific lags on the current observation in a time series.

Summary of the Decomposition of the Time Series

Additive Decomposition (Refer to Figure 3 in Appendix C) - The additive decomposition revealed a seasonal component with constant amplitude, aligning with the typical assumptions of an additive model where seasonal effects do not scale with the level of the time series. The random (residual) component is fairly homoscedastic, indicating the potential effectiveness of the additive model in encompassing most fluctuations within the time series.

Multiplicative Decomposition (Refer to Figure 4 in Appendix C) - The multiplicative decomposition displays the seasonal component's amplitude slightly changing in proportion to the trend, which is characteristic of multiplicative models. This suggests that as the time series level changes, the seasonal effect scales accordingly.

While both decompositions indicates elements of their respective counterparts: the additive model shows some periods where amplitude changes slightly, while the multiplicative model demonstrates periods where amplitude remains constant. However, the additive model consistently presents a clearer pattern in unemployment rates across economic cycles with a constant seasonal amplitude. Considering that the seasonal component is a crucial part of the BSTS model's structure, and since the decomposition doesn't provide any definite insights on whether an additive or multiplicative model is a better fit for the data, the plan is to test both model types.

Trend Component (Refer to Section 5 in Appendix C - After decomposing the trend component and constructing a general linear regression of the trend, it was observed that the intercept and slope values are statistically significant at the 95% confidence interval as they both have p-values of $< 2 \times 10^{-16}$. However, given the the low R-squared value of 0.1133, this may suggest that the trend component may not be a reliable indicator for making future predictions. To address this, we will testing models with a local-linear trend, semi-local-linear trend, and even no trend, in order to select the best trend fit of the model.

AR(p) Process (Refer to Sections 6 and 7, Figures 5 and 6 in Appendix C) - After looking at the ACF and PACF plots for the training data, it was observed that there is significant autocorrelation at multiple lags, so much so that it would not make sense to simply isolate a few. However, the `bsts` library allows the data to set the lag values through its `AutoAr()` function, which can

be added to the state-space for any models being constructed. Doing this, acknowledges the broader range of autocorrelation patterns revealed by the ACF plot. Hence, all models will have an auto AR process.

Model Selection Strategy

Based on the decomposition of the time series, 6 unique combinations of model type (additive/multiplicative) and the presence of a trend component (none/semi-local linear/local linear) would be tested, before selecting the best 2 out 6 to further evaluate using posterior predictive checks. A hybrid model selection strategy that combines frequentest and Bayesian validation methods will be used, ensuring a robust framework for forecasting unemployment rates, as well as a comprehensive measures of the uncertainty in any forecasts. Both a seasonal and auto-regressive components will be added to all models.

Models being Tested: "additive with no trend", "additive with semi-local linear trend", "additive with local linear trend", "multiplicative with no trend", "multiplicative with semi-local linear trend", "multiplicative with local linear trend".

Initial Model Selection

In the initial phase of models selection, each model was evaluated based on a series of metrics, after which a composite score was calculated in order to facilitate an objective comparison. The criteria used included a blend of statistical information and error metrics. Please see Appendix D - Section 1 to see code used to build the different models.

Criteria Used for Evaluation

AIC (Akaike Information Criterion): The AIC measures the quality of a model by assessing the trade-off between the goodness of fit and the complexity of the model, penalizing excessive complexity to avoid overfitting.

BIC (Bayesian Information Criterion): The BIC also evaluates model quality, but with a stricter penalty for complexity, often favoring simpler models, especially as the sample size increases.

MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), and MAPE (Mean Absolute Percentage Error): These error metrics provide insights into the average error magnitude, error variance, and error proportionality respectively, contributing to a comprehensive understanding of model accuracy.

Results of the Initial Model Selection

The initial selection process ranked the six models based on the computed scores, Please see Appendix D - Section 2 to see how the model's and their associated statistics were built and compiled, and Appendix D - Section 3 to see the code used for the decision matrix and the table of composite scores.

The model with an additive model type and a local linear trend component scored the highest. This model achieved the best balance between complexity and fit with the lowest MAE, MAPE, and RMSE, suggesting a high accuracy and efficiency in forecasts. In addition, the additive model with a semi-local linear trend component had the second highest composite score, and was very close across all the metrics with the local-linear trend model. The other four models performed well as seen by their good performance across the error metrics, However, the multiplicative models were ranked lower primarily due to their higher model complexity (as indicated by their very high AIC and BIC scores). Furthermore, the additive model with no trend was close, in terms of its composite score, with its semi-local linear and local linear trend model counterparts.

Selected Models for Further Analysis

Additive Model with Local Linear Trend: This model offers the best balance between complexity and accuracy, making it the primary candidate for further validation using Bayesian posterior predictive checks.

Additive Model with Semi-Local Linear Trend: This model also provides reasonable balance between complexity and accuracy, while providing a valuable point of comparison to our initial model choice. It was also very close across all the metrics with the local-linear trend model.

Model Validation and Comparison

In order to analyse and compare the two selected models, the analysis would include by generating the plots of the posterior predictive distribution for each model. Visualizing the distributions serves to provide insights into the models' ability to capture the structure of the data and predict future values. Following this, a cross-validation study will be done to conduct a more

rigorous statistical evaluation of the models' predictive performance. The cross-validation metrics includes the MAE, MAPE, RMSE, and the Average Credible Intervals Width (Avg. CI Width).

Posterior Prediction Distribution and Credible Intervals Visual Analysis Results

The additive semi-local linear model displayed a more faithful following of the actual unemployment trend, as indicated by the overlap of the predicted and actual lines in the plot (see Figures 7 and 8 in Appendix E). Visually, its credible intervals were also consistently narrower than those of the Additive Local Linear Model, indicating a higher precision of forecasts.

Cross Validation Study Results

The results of the cross-validation study revealed that the additive semi-local linear trend model outperformed the additive local linear trend model across all metrics, as seen by its overall lower MAE, MAPE, and RMSE values, suggesting that its predictions were closer to the actual values and generally more accurate. Furthermore, it presented a smaller average credible intervals width, signifying a higher average precision of the model's forecasts. The plots of the posterior predictive distribution supported the results of the study, with the additive semi-local linear trend model showing a narrower credible interval and the predicted values tracking the actual data more closely than the additive local linear trend model.

Final Model

Based on the combined evidence from the posterior predictive distribution analysis and the cross-validation study, the additive semi-local linear trend model is selected as the best model for forecasting US Unemployment rates. This model not only offers the greatest accuracy but also provides more precise predictions, which is critical for reliable forecasting. Furthermore, the narrower credible intervals suggest that the forecasts made by the additive semi-local linear trend model carried less uncertainty, a valuable attribute for making informed decisions in uncertain environments.

Quantifying Uncertainty

In order to quantify the uncertainty of our forecasts, this was achieved this by analyzing a variety of error metrics and the calculation of credible intervals. They served as important pieces of evidence in both the initial model selection, and the model validation and comparison phase. By carefully considering both the expected error metrics and the credible range of the forecasts, a holistic view of the forecast uncertainty was able to be provided, allowing for more informed decision-making processes, while taking into account both the precision and the potential variability of future estimates.

BSTS Model for US Unemployment Rates

Consider the following mathematical representation of the additive semi-local linear trend model using the Bayesian framework.

Observation Model: The observed unemployment rate at time t, y_t, is modeled as:

$$y_t \sim \text{Normal}(\mu_t + s_t + X_t, \sigma_y^2), \qquad \text{where } \sigma_y^2 \sim \text{Inverse-Gamma}(2, 1)$$

where μ_t represents the trend component, s_t denotes the seasonal component, X_t denotes the autoregressive process, and σ_y^2 is the observation variance.

Semi-Local Level (Trend) Component:

The semi-local component μ_t and its slope δ_t evolves over time as:

$$\mu_t \sim \text{Normal}(\mu_{t-1} + \delta_t, \sigma_\mu^2), \qquad \text{where } \sigma_\mu^2 \sim \text{Inverse-Gamma}(2, 1)$$

$$\delta_t \sim \text{Normal}(D + \rho(\delta_{t-1} - D), \sigma_\delta^2), \qquad \text{where } D = 0.0013537,\ \rho \sim \text{Uniform}(0, 1),\ \text{and } \sigma_\delta^2 \sim \text{Inverse-Gamma}(2, 1)$$

where σ_μ^2 is the variance of the semi-local component, D is the long-term slope (see Appendix C - Section 5), ρ is the autoregressive behavior of the trend, δ_{t-1} is the trend slope at the previous time point, and σ_δ^2 is the variance of the trend slope.

Seasonal Component:

The seasonal component s_t evolves over time as:

$$s_t \sim \text{Normal}\left(-\frac{1}{S-1}\sum_{i=1}^{S-1} s_{t-i}, \sigma_s^2\right), \qquad \text{where } s_1, s_2, ..., s_{12} \sim \text{Normal}(0, \sigma_s^2),\ \text{and } \sigma_s^2 \sim \text{Inverse-Gamma}(2, 1)$$

where S is the seasonal period (in this case $S = 12$) for monthly, s_{t-i} is the seasonal component as the previous time point, σ_s^2 is the variance of the seasonal component, and $s_1, s_2, ..., s_{12}$ are the initial seasonal component values (associated with every month of a year). The factor $\frac{1}{S-1}$ is used to ensure that the seasonal effects average out to zero over one full cycle.

Autoregressive Component:

The autoregressive process X_t evolves over time as:

$$X_t = \sum_{i=1}^{p} \phi_i \cdot X_{t-i} + \epsilon_t, \qquad \text{where } \epsilon_t \sim \text{Normal}(0, \sigma_X^2), \text{ and } \sigma_X^2 \sim \text{Inverse-Gamma}(2, 1)$$

where ϕ_i are the coefficients for the lags $i = 1, 2, ..., p$ and ϵ_t denotes the white noise of the AR(p) term. The order p of the autoregressive process is not pre-specified but is determined by the `AutoAr()` function, allowing the data itself to identify the most appropriate lags.

Justification of Hyperparameters and Hyperpriors

This model uses weakly informative priors and data-driven specifications to robustly forecast US Unemployment rates, while providing a quantification of its forecast uncertainty. Furthermore, by adopting a flexible approach to the autoregressive order and the slope's persistence, the model remains adaptable to various economic scenarios.

The Inverse-Gamma distribution is utilized as a hyperprior for the variance components in the model because it is the conjugate prior for the variance of a normal distribution, which simplifies the Bayesian updating process. Furthermore, this choice helps in ensuring that the variance estimates remain positive, allowing for flexibility in modeling the inherent variability of the data. Using weakly informative priors in this context allows the historical data to predominantly influence the posterior distributions, thereby maximizing the use of historical trends and variability in the data, while still providing reasonable constraints to ensure that the model is stable and to prevent overfitting.

Discussion

By applying Bayesian forecasting techniques, through the implementation and evaluation of BSTS models, has led to several insightful findings. The additive semi-local linear trend model emerged as the superior model after an extensive cross-validation study and comparison against other model variants. This model demonstrated the lowest mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE), indicating a high accuracy in forecasting unemployment rates. Additionally, it featured narrower credible intervals in its predictions, reflecting a greater precision and reduced uncertainty in its forecast outputs.

An key limitation to note is the significant lags that were identified in the ACF plots of the time series of the data, as in the context of time series analysis it indicated than an Auto-Regressive Integrated Moving Average (ARIMA) model could have been a better fit for the data. ARIMA models are specifically designed to handle data with clear autocorrelation patterns (as is the case here), potentially offering more precise forecasts than a BSTS model. This emphasizes the need for exploring a broader range of models in future research to fully leverage the information contained in the data's autocorrelative structure.

Another limitation of the this study is the reliance on historical data to set weakly informative priors. While beneficial in incorporate past trends, it may limit the model's ability to adapt to new economic conditions not seen in the historical data. Secondly, the model's performance is heavily reliant on the cyclic seasonality of past trend, which may not fully capture sudden economic shifts due to unprecedented events or policy changes. Furthermore, the choice to implement Bayesian model, while offering robust probabilistic insights, could also lead to increased model complexity and computational requirements that could be avoided with a more direct forecasting approach (like an ARIMA implementation).

There are several avenues for future research to consider, that may potentially enhance the findings form this study. For example, integrating other labor market metrics such as labor force participation rates, historical inflation rates, etc., could provide a more comprehensive view of the economic landscape influencing US unemployment rates. Additionally, testing the model against alternative economic scenarios, such as those involving significant policy shifts or global economic crises, could further validate its robustness and adaptability. By broadening the scope of the historical indicators used in the model, future research could improve the robustness and accuracy of forecasting unemployment rates, making them more reflective of the dynamic nature of economic fluctuations.

In conclusion, the additive semi-local linear trend model emerged as a reliable tool for predicting US unemployment rates, by offering precise forecasts along with a set of quantified uncertainties. While acknowledging its limitations, the model's capability in managing complex historical trends makes it a valuable asset for economic analysts and policy-makers who need reliable predictions to make informed decisions.

References

Almarashi, Abdullah M., and Khushnoor Khan. "Bayesian structural time series." Nanoscience and Nanotechnology Letters, vol. 12, no. 1, 1 Jan. 2020, pp. 54–61, doi.org/10.1166/nnl.2020.3083.

Brodersen, Kay. Fitting Bayesian Structural Time Series with the Bsts R Package, Blogger, 12 July 2017, unofficialgoogledatascience.com/2017/07/fitting-bayesian-structural-time-series.html

"Bayesian Structural Time Series." *SAP HANA Predictive Analysis Library (PAL)*, help.sap.com/docs/SAP_HANA_PLATFORM/2cfbc5cf2bc14f028cfbe2a2bba60a50/b9972576368640da9831d73a9d749c3b.html. Accessed 8 Apr. 2024.

DeWitt, Michael. "Bayesian Time Series Analysis with Bsts." Insufficient Statistics, 5 July 2018, michaeldewittjr.com/programming/2018-07-05-bayesian-time-series-analysis-with-bsts/.

Radtke, Tim. "Minimize Regret." *Minimize Regret - Rediscovering Bayesian Structural Time Series*, minimizeregret.com/post/2020/06/07/rediscovering-bayesian-structural-time-series/. Accessed 9 Apr. 2024.

Wright, Jonathan H. "Bayesian model averaging and exchange rate forecasts." Journal of Econometrics, vol. 146, no. 2, Oct. 2008, pp. 329–341, doi.org/10.1016/j.jeconom.2008.08.012.

Yabe, Taka. "Pystan - Causal Inference Using Bayesian Structural Time Series." *Takahiro Yabe*, 21 Feb. 2021, takayabe.net/post/pystan-bsts. Accessed 9 Apr. 2024.

Appendix

Appendix A - Preliminary Data Analysis

Reading in the Dataset and Data Cleaning

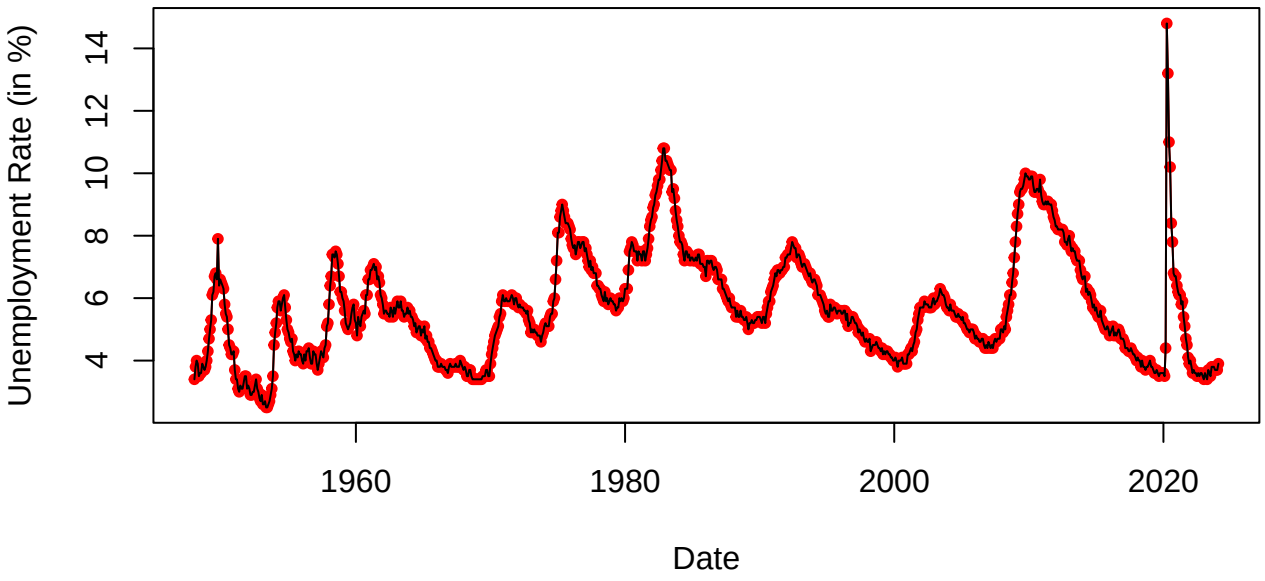
```
dat <- read_csv("UNRATE.csv")
dat$DATE <- as.Date(dat$DATE)
head(dat)
```

```
## # A tibble: 6 x 2
##   DATE       UNRATE
##   <date>      <dbl>
## 1 1948-01-01    3.4
## 2 1948-02-01    3.8
## 3 1948-03-01    4
## 4 1948-04-01    3.9
## 5 1948-05-01    3.5
## 6 1948-06-01    3.6
```

Historical US Unemployment Rates from 01/1948 to 02/2024

Figure 1: Historical US Unemployment Rate Over Time

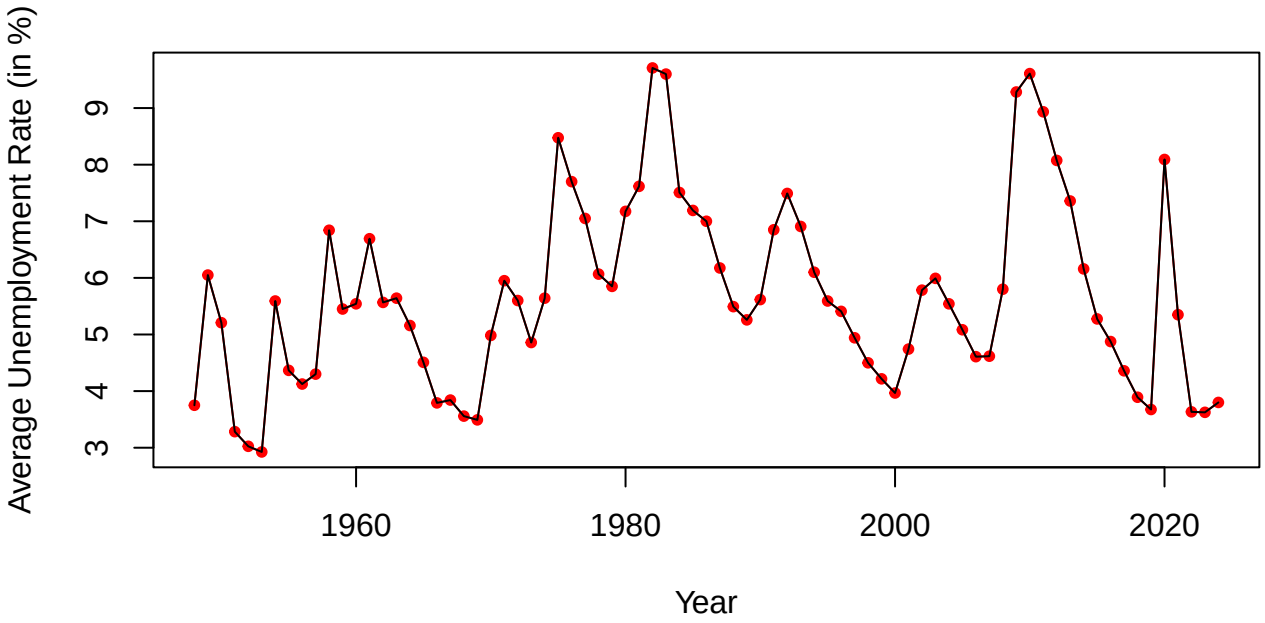
```
plot(dat$DATE, dat$UNRATE,
     type = "o", pch = 20,
     col = "red",
     xlab = "Date",
     ylab = "Unemployment Rate (in %)")
lines(dat$DATE, dat$UNRATE, col = "black")
```



Average Yearly US Unemployment Rates from 01/1948 to 02/2024

Figure 2: Average Yearly Unemployment Rate in the US

```
avg_yearly_unrate <- dat |>
  mutate(YEAR = format(DATE, "%Y")) |>
  group_by(YEAR) |>
  summarise(Average_Unemployment_Rate = mean(UNRATE, na.rm = TRUE))

avg_yearly_unrate$YEAR <- as.numeric(as.character(avg_yearly_unrate$YEAR))

plot(avg_yearly_unrate$YEAR, avg_yearly_unrate$Average_Unemployment_Rate,
     type = "o", pch = 20,
     col = "red",
     xlab = "Year",
     ylab = "Average Unemployment Rate (in %)")
lines(avg_yearly_unrate$YEAR, avg_yearly_unrate$Average_Unemployment_Rate, col = "black")
```



Appendix B - General Overview on the Bayesian Structural Time Series (BSTS) Model

The Bayesian Structural Time Series (BSTS) model is a statistical method that serves several applications such as feature selection, time series forecasting, and causal inference analysis. It operates on time series data to ascertain underlying patterns and forecast future data points.

BSTS models are composed of three primary elements:

1. *Kalman Filter:* A method for decomposing time series into components like trend and seasonality, allowing state variables to be modeled dynamically over time.
2. *Spike-and-Slab Prior:* A technique for feature selection that identifies which predictors in a regression are most informative.
3. *Bayesian Model Averaging (BMA):* A process where multiple models are averaged together to produce predictions or infer parameters, accounting for model uncertainty. In the BSTS framework, BMA is utilized extensively to generate samples from repeated Markov Chain Monte Carlo (MCMC) simulations into final model outputs, providing a comprehensive prediction that encompasses model variability.

A general BSTS model consists of twos set of equations,

1. *Observational equation:* Response variable as a function of predictors and/or latent variables.
2. *State-space equations:* How the parameters evolve over the time.

BSTS models are usually implemented using the `bsts` library in R, but can also be implemented using `rstan`. Stan's advanced MCMC algorithms enhance the flexibility and scalability of BSTS models, making it a powerful tool for time series analysis. However, in this study, the `bsts` library was used as it was less computationally intensive than a Stan implementation.

Appendix C - Time Series Analysis

Section 1 - Setting up the Training and Testing Sets

We will be using all the data up until 02/2023 as the training set, and have the remaining data (from 03/2023 to 02/2024) as the testing set.

```
start_date <- as.Date("1948-01-01")
train_end_date <- as.Date("2023-02-01")
test_start_date <- as.Date("2023-03-01")
end_date <- as.Date("2024-02-01")

dat_train <- subset(dat, DATE <= train_end_date)
dat_test <- subset(dat, DATE >= test_start_date)

nrow(dat_train)
```

```
## [1] 902
```

```
nrow(dat_test)
```

```
## [1] 12
```

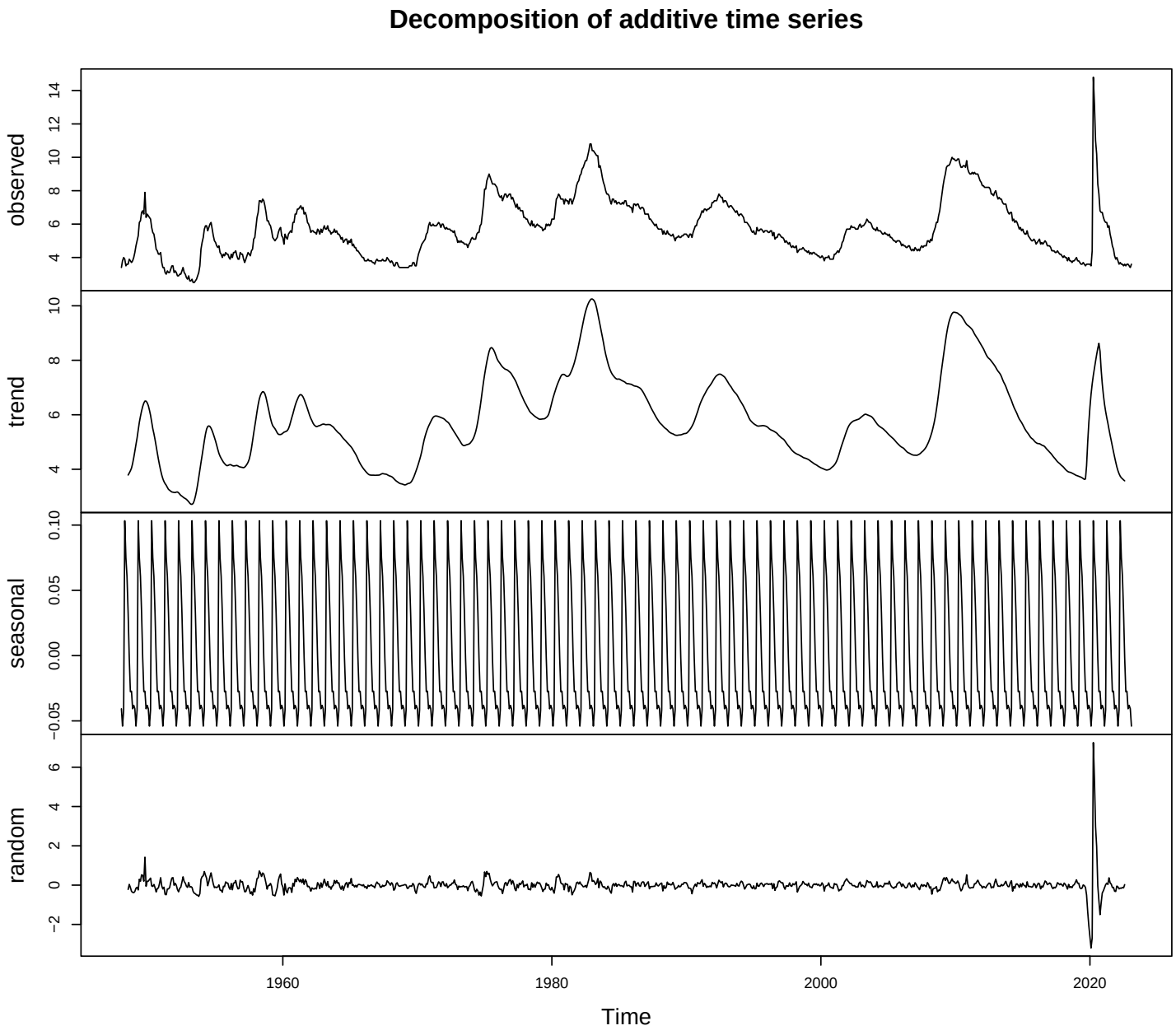
Section 2 - Creating a Time Series Object for the Training and Testing Sets

```
start_train <- c(1948, 1)
end_train <- c(2023, 2)
start_test <- c(2023, 3)
end_test <- c(2024, 2)

ts_unrate <- ts(dat, start = start_train, end = end_test, frequency = 12)

ts_unrate_train <-
  ts(dat_train$UNRATE, start = start_train, end = end_train, frequency = 12)

ts_unrate_test <-
  ts(dat_test$UNRATE, start = start_test, end = end_test, frequency = 12)
```

Section 3 - Additive Decomposition of the Time Series

Figure 3: Additive Decomposition of the Time Series of US Unemployment Rates

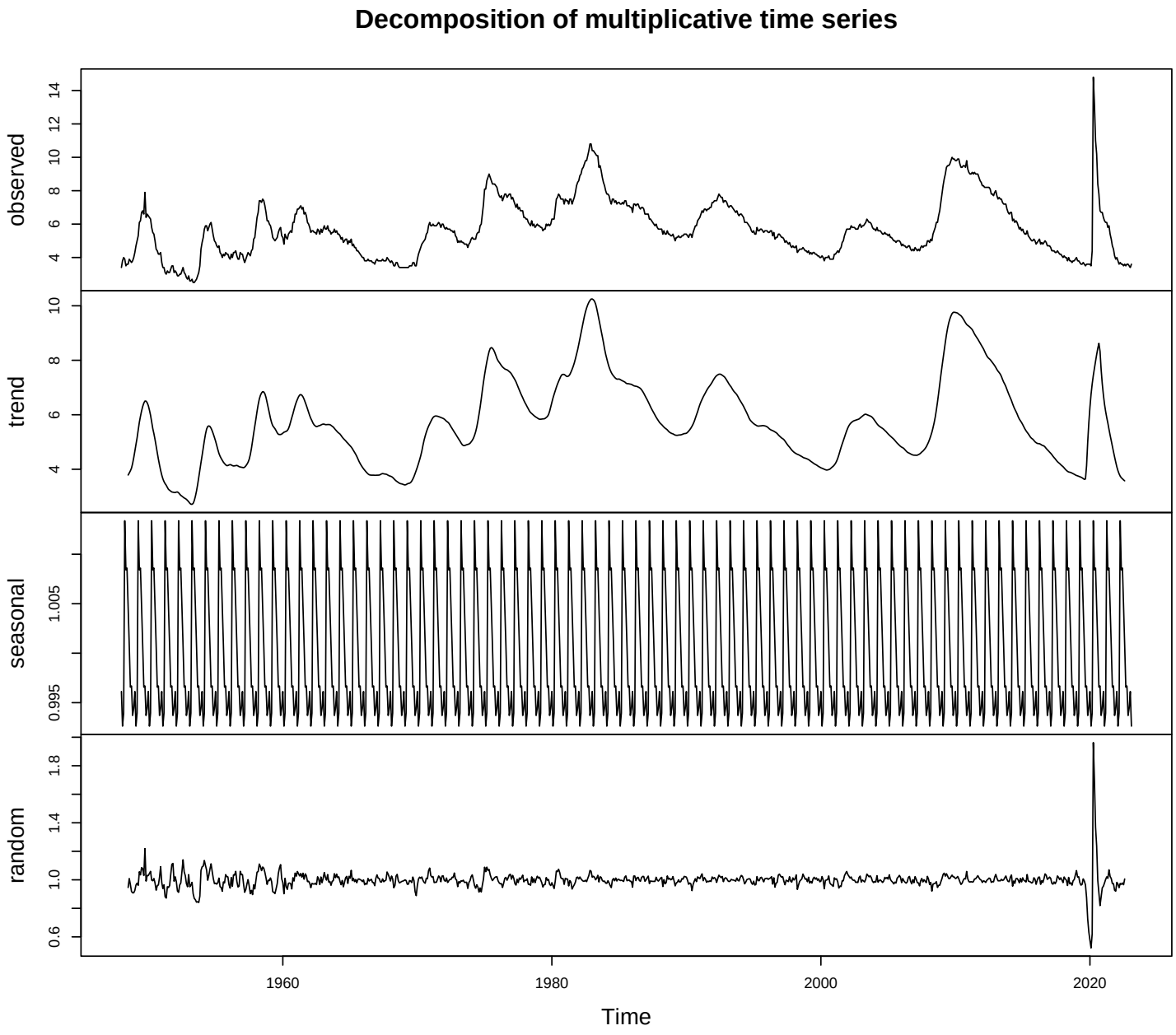
```
decompose_train_additive <- decompose(ts_unrate_train, type = "additive")
plot(decompose_train_additive)
```



Section 4 - Multiplicative Decomposition of the Time Series

Figure 4: Multiplicative Decomposition of the Time Series of US Unemployment Rates

```
decompose_train_multiplicative <- decompose(ts_unrate_train, type = "multiplicative")
plot(decompose_train_multiplicative)
```



Section 5 - Trend Component

```r
decompose_train <- stl(ts_unrate_train, s.window = "periodic")
trend <- decompose_train$time.series[, "trend"]
time <- c(1:length(trend))

lm_trend <- lm(trend ~ time)
summary(lm_trend)
```

```
##
## Call:
## lm(formula = trend ~ time)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8314 -1.2338 -0.1731  1.0934  4.5906
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.1160463  0.1039631  49.210  < 2e-16 ***
## time        0.0013537  0.0001995   6.787 2.08e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.56 on 900 degrees of freedom
## Multiple R-squared:  0.04869,    Adjusted R-squared:  0.04763
## F-statistic: 46.06 on 1 and 900 DF,  p-value: 2.079e-11
```

$$Trend = 5.1160463 + 0.0013537 \times Time$$

The intercept and slope values are statistically significant at the 95% confidence interval as they both have p-values of $< 2 \times 10^{-16}$. However, given the the low R-squared value of 0.04869, this may suggest that the trend component may not be a reliable indicator for making future predictions.

Section 6 - ACF and PACF Plots

Figure 5: Autocorrelation (ACF) Plot of a Time Series of US Unemployment Rates

```
acf_ts_train <- acf(ts_unrate_train)
```

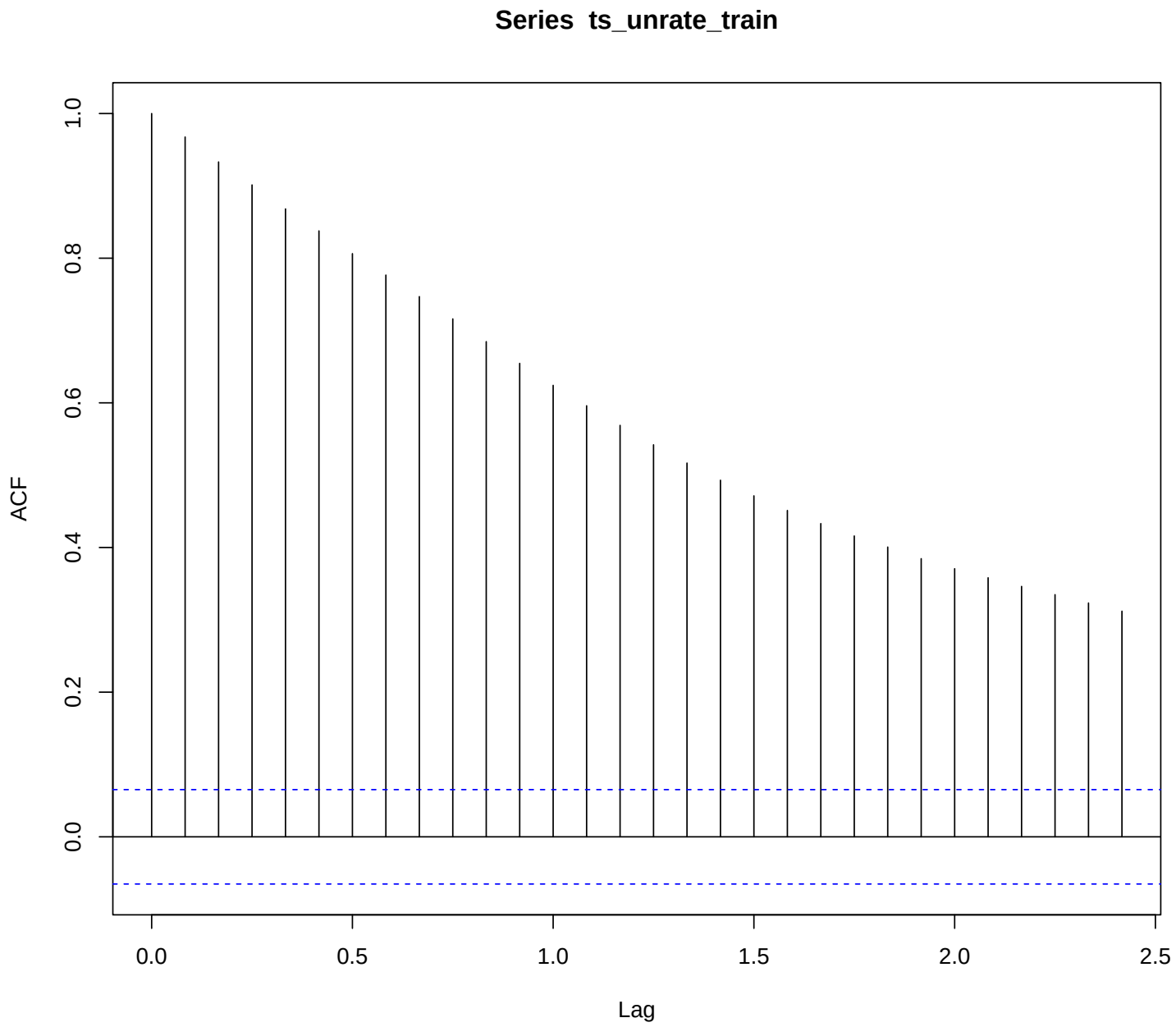
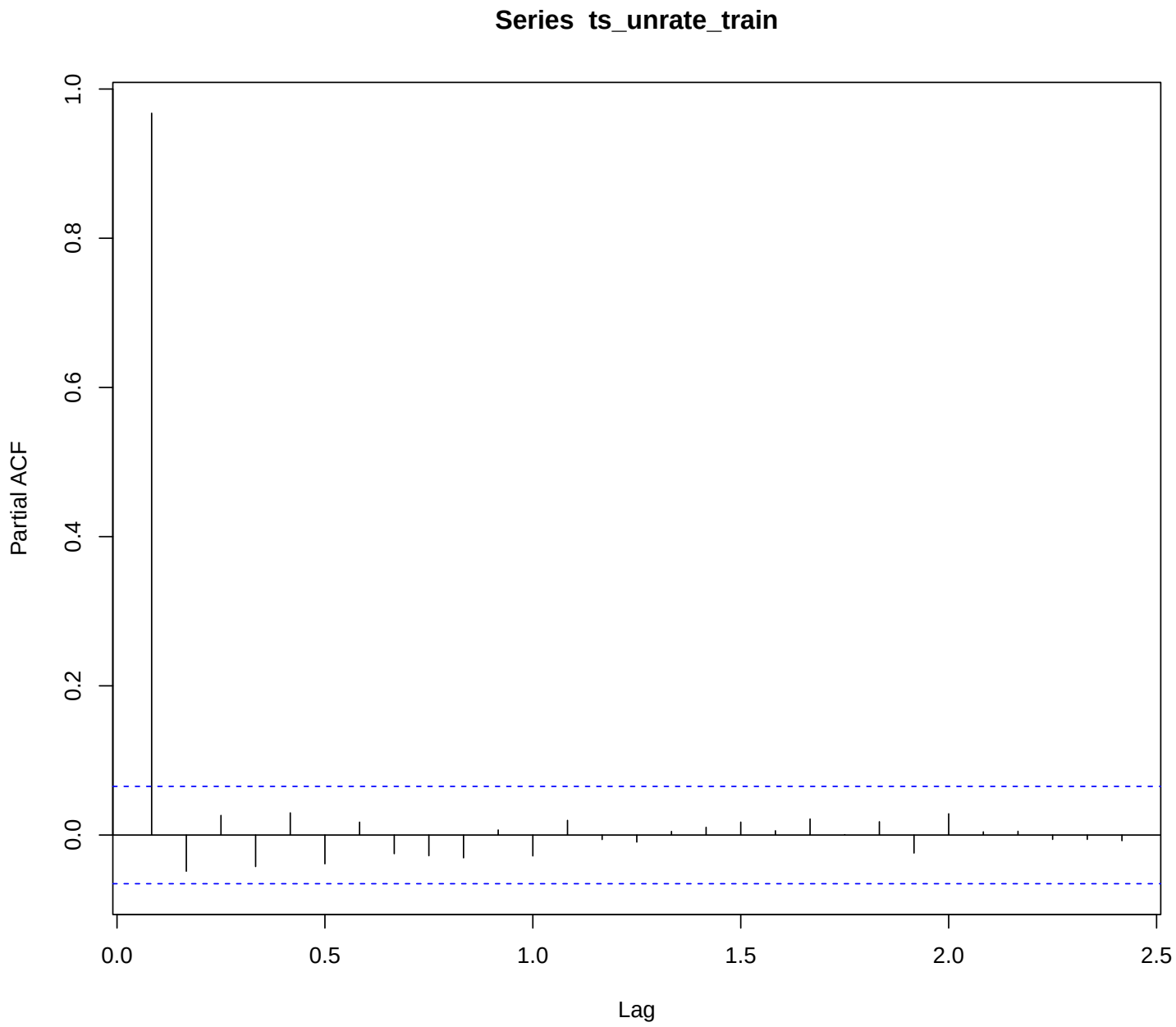


Figure 6: Partial Autocorrelation (PACF) Plot of a Time Series of US Unemployment Rates

```
pacf_ts_train <- pacf(ts_unrate_train)
```



Section 7: Significant ACF and PACF Values

```
n <- length(ts_unrate_train)
significance_level <- 1.96 / sqrt(n)

significant_lags_acf <- which(abs(acf_ts_train$acf[-1]) > significance_level) - 1
significant_lags_pacf <- which(abs(pacf_ts_train$acf[-1]) > significance_level) - 1

cat("Significant lags for ACF: \n", significant_lags_acf, "\n\n")
```

```
## Significant lags for ACF:
##  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

```
cat("Significant lags for PACF: \n", significant_lags_pacf, "\n")
```

```
## Significant lags for PACF:
##
```

Appendix D - Building Models and Initial Model Selection

Section 1 - Functions for Building BSTS Models and the Initial Model Selection

```
# BSTS Model Building Function
build_bsts_model <- function(trend, n_iter, ts_unrate_train, model_type) {
  if (model_type == "multiplicative") {
    ts_unrate_train <- log(ts_unrate_train)
  }

  state_space <- list()
  state_space <- AddSeasonal(state_space, y = ts_unrate_train, nseasons = 12)
  state_space <- AddAutoAr(state_space, y = ts_unrate_train)

  if (trend == "local_linear") {
    state_space <- AddLocalLinearTrend(state_space, y = ts_unrate_train)
  }

  if (trend == "semi_local_linear") {
    state_space <- AddSemilocalLinearTrend(state_space, y = ts_unrate_train)
  }

  bsts_model <- bsts(ts_unrate_train, state.specification = state_space,
                     niter = n_iter, ping = 0, seed = 123)

  return(bsts_model)
}

# Initial Model Selection
calculate_log_likelihood <- function(model, data) {
    predicted_means <- predict(model)$mean
    residuals <- data - predicted_means
    estimated_sigma <- sd(residuals)
    log_likelihood <- sum(dnorm(residuals, mean = 0, sd = estimated_sigma, log = TRUE))
    return(log_likelihood)
}

calculate_aic <- function(model, data, k) {
    n <- length(data)
    log_likelihood <- calculate_log_likelihood(model, data)
    aic <- 2 * k - 2 * log_likelihood
    return(aic)
}

calculate_bic <- function(model, data, k) {
    n <- length(data)
    log_likelihood <- calculate_log_likelihood(model, data)
    bic <- log(n) * k - 2 * log_likelihood
    return(bic)
}

calculate_mape <- function(actuals, pred_values) {
  epsilon <- 1e-10
  actuals[actuals == 0] <- epsilon
  pred_values[pred_values == 0] <- epsilon

  ape <- abs((actuals - pred_values) / actuals)
  mape <- mean(ape, na.rm = TRUE) * 100
  return(mape)
```

```
}

model_selection <- function(bsts_model, ts_unrate_test, model_type, trend) {
  k_seasonal <- 11                              # 12 months - 1
  k_trend <- switch(trend,                      # Corrected variable name here
                    "none" = 0,
                    "local_linear" = 2,         # Level + Slope
                    "semi_local_linear" = 2)    # Assuming standard 2 parameters
  k <- k_seasonal + k_trend

  burn <- SuggestBurn(0.1, bsts_model)
  pred <- predict(bsts_model, horizon = length(ts_unrate_test),
                  burn = burn, quantiles = c(0.025, 0.975))

  if (model_type == "multiplicative") {
    pred_values <- exp(as.numeric(pred$mean))
  } else {
    pred_values <- as.numeric(pred$mean)
  }

  actuals <- as.numeric(ts_unrate_test)

  # Calculate metrics
  mae <- mean(abs(actuals - pred_values))
  rmse <- sqrt(mean((actuals - pred_values)^2))
  mape <- calculate_mape(actuals, pred_values)

  # Extract AIC and BIC from the model
  aic <- calculate_aic(bsts_model, ts_unrate_test, k)
  bic <- calculate_bic(bsts_model, ts_unrate_test, k)

  return(tibble("MAE" = mae,"RMSE" = rmse,"MAPE" = mape,
                "AIC" = aic, "BIC" = bic))
}
```

Section 2 - Code to Loop over the Initial 6 Models and Generate Statistics

```
# Set seed for reproducibility
set.seed(123)

# Initialize the results tibble with proper column names
results <- tibble(
  "Model Type" = character(),
  "Include Trend" = character(),
  "MAE" = double(),
  "MAPE" = double(),
  "RMSE" = double(),
  "AIC" = double(),
  "BIC" = double()
)

model_types <- c("additive", "multiplicative")
trend_types <- c("none", "local_linear", "semi_local_linear")
n_iter <- 1000

model_dict <- list()

for (model_type in model_types) {
  for (trend in trend_types) {
    cat(sprintf("Running model: %s with trend: %s\n", model_type, trend))
    bsts_model <- build_bsts_model(trend, n_iter, ts_unrate_train, model_type)
    metrics <- model_selection(bsts_model, ts_unrate_test, model_type, trend)

    key <- paste(model_type, trend, sep = "-")
    model_dict[[key]] <- list(bsts_model = bsts_model, metrics = metrics)

    new_row <- tibble("Model Type" = model_type,
                      "Include Trend" = trend) %>%
      bind_cols(metrics)
    results <- bind_rows(results, new_row)
  }
}
```

```
## Running model: additive with trend: none
## Running model: additive with trend: local_linear
## Running model: additive with trend: semi_local_linear
## Running model: multiplicative with trend: none
## Running model: multiplicative with trend: local_linear
## Running model: multiplicative with trend: semi_local_linear
```

```
results
```

```
## # A tibble: 6 x 7
##   `Model Type`   `Include Trend`     MAE  MAPE  RMSE    AIC    BIC
##   <chr>          <chr>             <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 additive       none              0.205  5.57 0.246   29.1   34.5
## 2 additive       local_linear      0.160  4.36 0.188   25.1   25.9
## 3 additive       semi_local_linear 0.150  4.21 0.206   23.5   30.7
## 4 multiplicative none              0.169  4.58 0.194 3153.  3160.
## 5 multiplicative local_linear      0.208  5.60 0.240 3162.  3170.
## 6 multiplicative semi_local_linear 0.108  2.98 0.140 3130.  3145.
```

Section 3 - Decision Matrix

```
# Assigning weights to each metric
# Note: Negative weights are used for metrics where lower values are better
# (AIC, BIC, MAE, RMSE, MAPE)
weights <- c("AIC" = -1, "BIC" = -1, "MAE" = -1, "RMSE" = -1, "MAPE" = -1)

# Calculate a comprehensive score for each model configuration considering all metrics
# Multiplying each metric by its corresponding weight and summing the results
# to compute a total score
results <- results %>%
  mutate(score = (AIC * weights["AIC"] +
                  BIC * weights["BIC"] +
                  MAE * weights["MAE"] +
                  RMSE * weights["RMSE"] +
                  MAPE * weights["MAPE"]))

# Sort the models by the scores from the decision matrix
model_selection_df <- results %>% arrange(desc(score))
print(model_selection_df)
```

```
## # A tibble: 6 x 8
##   `Model Type`   `Include Trend`     MAE  MAPE  RMSE    AIC    BIC   score
##   <chr>          <chr>             <dbl> <dbl> <dbl>  <dbl>  <dbl>   <dbl>
## 1 additive       local_linear      0.160  4.36 0.188   25.1   25.9   -55.7
## 2 additive       semi_local_linear 0.150  4.21 0.206   23.5   30.7   -58.7
## 3 additive       none              0.205  5.57 0.246   29.1   34.5   -69.6
## 4 multiplicative semi_local_linear 0.108  2.98 0.140 3130.  3145.  -6278.
## 5 multiplicative none              0.169  4.58 0.194 3153.  3160.  -6317.
## 6 multiplicative local_linear      0.208  5.60 0.240 3162.  3170.  -6337.
```

Appendix E - Posterior Predictive Distributions

Section 1 - Code to Generate a Plot of the Posterior Predictive Distributions

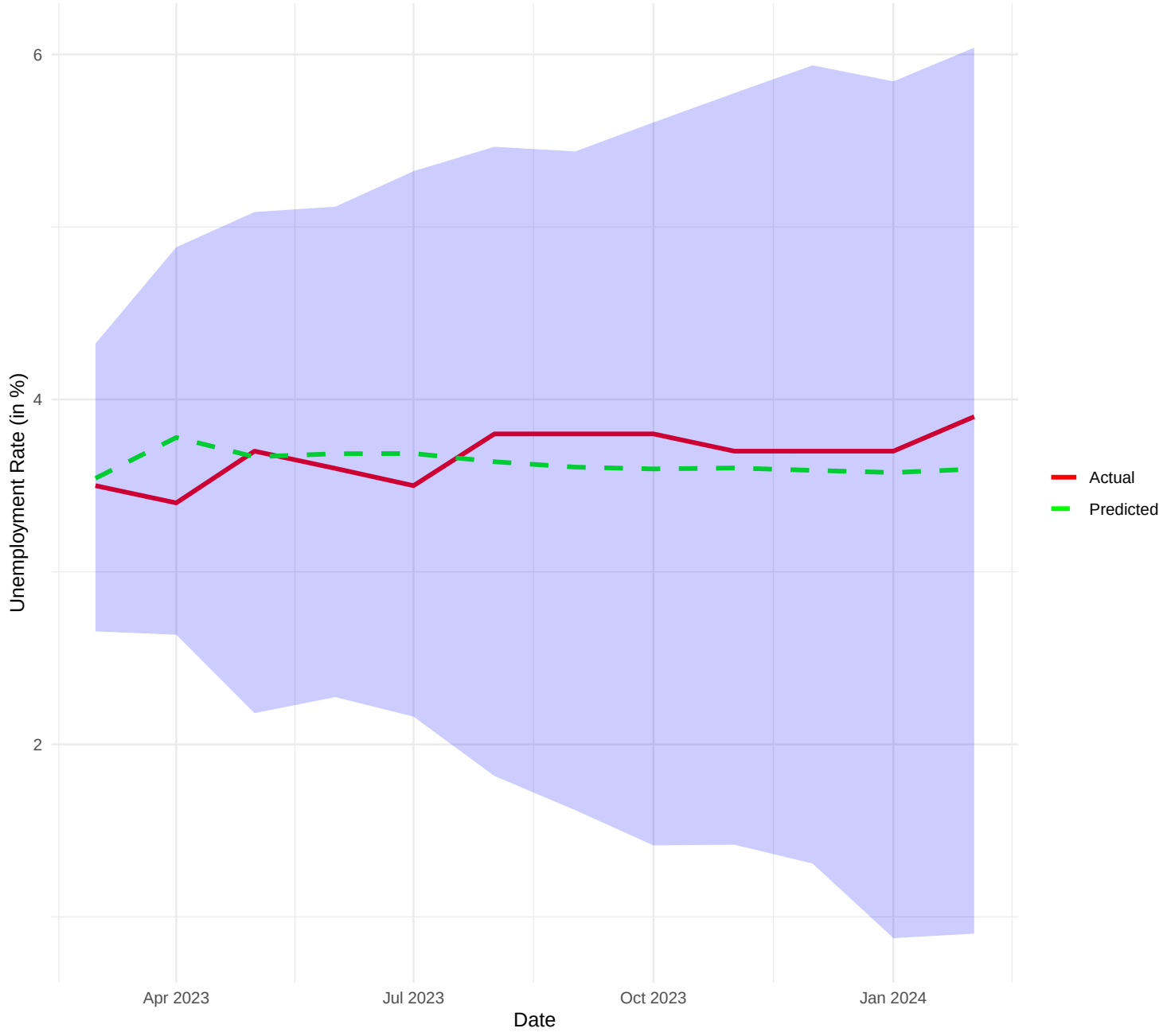
```
generate_plot_of_post_pred_dist <- function(bsts_model, ts_unrate_test) {
  burn <- SuggestBurn(0.1, bsts_model)
  predictions <- predict(bsts_model, horizon = length(ts_unrate_test),
                         burn = burn, quantiles = c(0.025, 0.975))

  lower_bound <- predictions$interval["2.5%",]
  upper_bound <- predictions$interval["97.5%",]

  credible_intervals <- data.frame(
    Time = seq(test_start_date, by = "month", length.out = length(ts_unrate_test)),
    Lower = lower_bound,
    Upper = upper_bound
  )

  credible_intervals_widths <- credible_intervals$Upper - credible_intervals$Lower
  avg_credible_intervals_widths <- mean(credible_intervals_widths)
  cat("Average Credible Intervals Width:", avg_credible_intervals_widths, "\n")

  test_dates <- seq(from = as.Date("2023-03-01"), by = "month",
                    length.out = length(ts_unrate_test))

  plot_data <- data.frame(
    Time = test_dates,
    Actual = as.numeric(ts_unrate_test),
    Predicted = predictions$mean,
    Lower = credible_intervals$Lower,
    Upper = credible_intervals$Upper
  )

  posterior_plot <-
    ggplot(plot_data, aes(x = Time)) +
    geom_line(aes(y = Actual, colour = "Actual"), size = 1.2) +
    geom_line(aes(y = Predicted, colour = "Predicted"),
              size = 1.2, linetype = "dashed") +
    geom_ribbon(aes(ymin = Lower, ymax = Upper), fill = "blue", alpha = 0.2) +
    scale_colour_manual(values = c("Actual" = "red", "Predicted" = "green")) +
    labs(x = "Date", y = "Unemployment Rate (in %)") +
    theme_minimal() +
    theme(legend.title = element_blank())

  return(posterior_plot)
}
```

Section 2 - Posterior Predictive Distribution for Additive Local Linear Model

Figure 7 - Posterior Predictive Distribution with Credible Intervals for the Additive Local Linear Model

```
additive_local_linear <- model_dict[["additive-local_linear"]]
additive_local_linear_model <- additive_local_linear$bsts_model

generate_plot_of_post_pred_dist(additive_local_linear_model, ts_unrate_test)
```

```
## Average Credible Intervals Width: 3.63104
```

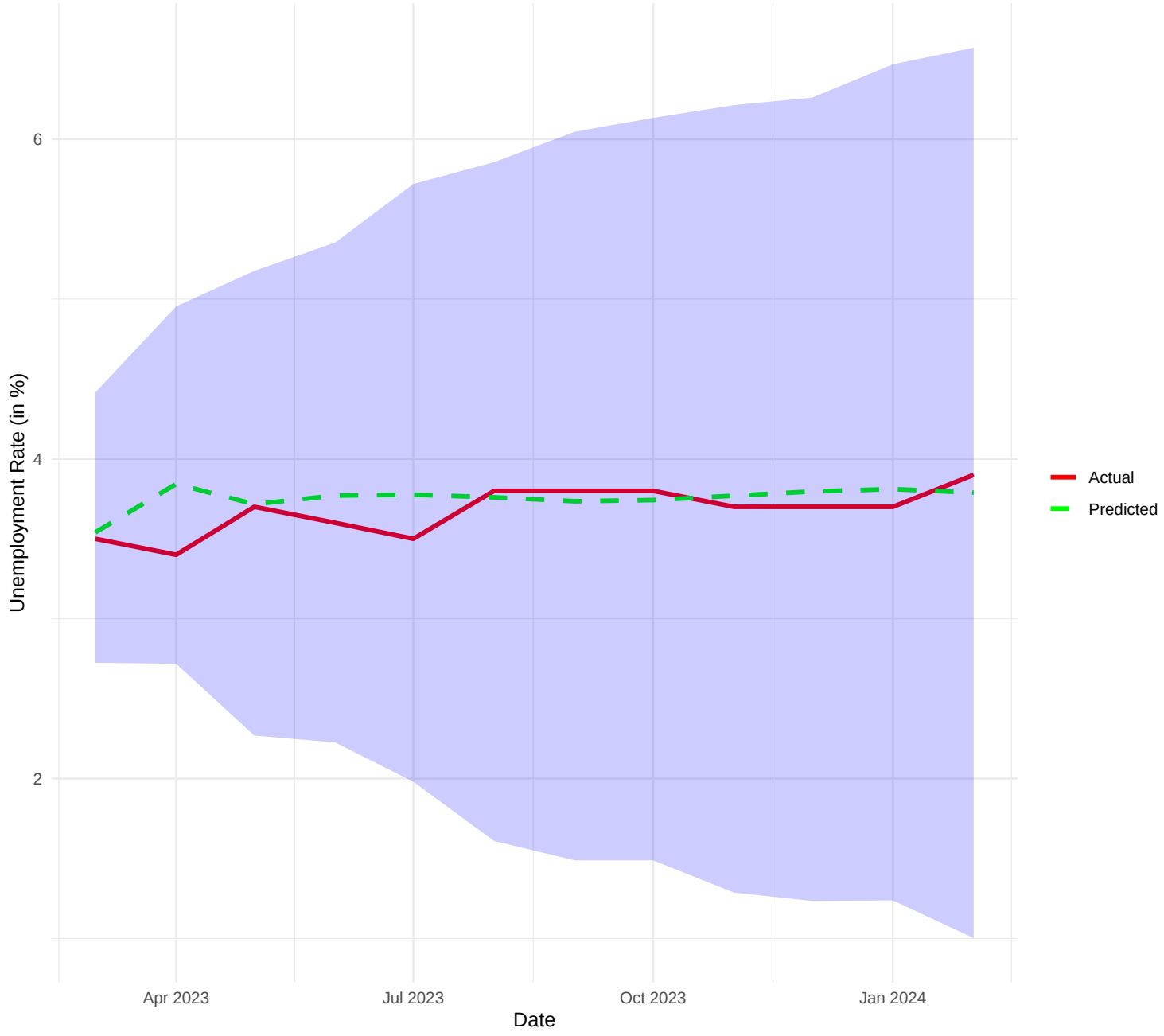


Section 3 - Posterior Predictive Distribution for Additive Semi-Local Linear Model

Figure 8 - Posterior Predictive Distribution with Credible Intervals for the Additive Semi-Local Linear Model

```
additive_semi_local_linear <- model_dict[["additive-semi_local_linear"]]
additive_semi_local_linear_model <- additive_semi_local_linear$bsts_model

generate_plot_of_post_pred_dist(additive_semi_local_linear_model, ts_unrate_test)
```

```
## Average Credible Intervals Width: 3.992042
```



Appendix F - Cross Validation

Section 1 - Function for General Cross-Validation of BSTS Model

```
cross_validate_bsts_model <- function(trend, ts_unrate_train, test_start_date) {
  # Set seed for reproducibility
  set.seed(123)

  n_iter <- 500
  forecast_horizon <- 12   # Forecast horizon (1 year)
  step_size <- 24          # Step size for the rolling window (2 years)

  error_metrics <- tibble(MAE = double(), MAPE = double(), RMSE = double(),
                          `Avg. Credible Intervals Width` = double())

  n_windows <- (length(ts_unrate_train) - forecast_horizon) / step_size

  # Perform rolling window cross-validation
  for (i in seq_len(n_windows)) {
    train_end <- i * step_size
    valid_start <- train_end + 1
    valid_end <- valid_start + forecast_horizon - 1

    train_set <- ts_unrate_train[1:train_end]
    valid_set <- ts_unrate_train[valid_start:valid_end]

    bsts_model <- build_bsts_model(trend, n_iter, train_set, "additive")

    # Forecast and calculate error metrics
    burn <- SuggestBurn(0.1, bsts_model)
    pred <- predict(bsts_model, horizon = length(valid_set), burn = burn)
    pred_values <- as.numeric(pred$mean)
    actuals <- as.numeric(valid_set)

    lower_bound <- pred$interval["2.5%",]
    upper_bound <- pred$interval["97.5%",]

    mae <- mean(abs(actuals - pred_values))
    rmse <- sqrt(mean((actuals - pred_values)^2))
    mape <- calculate_mape(actuals, pred_values)

    credible_intervals <- data.frame(
      Time = seq(test_start_date, by = "month", length.out = length(valid_set)),
      Lower = lower_bound,
      Upper = upper_bound
    )

    credible_intervals_widths <- credible_intervals$Upper - credible_intervals$Lower
    avg_credible_intervals_widths <- mean(credible_intervals_widths)

    new_row <- tibble(MAE = mae, RMSE = rmse, MAPE = mape,
                      `Avg. Credible Intervals Width` = avg_credible_intervals_widths)
    error_metrics <- bind_rows(error_metrics, new_row)
  }

  # Cross-validation Summary statistics
  summary_metrics <- error_metrics %>%
    summarise("Average MAE" = mean(`MAE`, na.rm = TRUE),
              "Average MAPE" = mean(MAPE, na.rm = TRUE),
```

```
              "Average RMSE" = mean(RMSE, na.rm = TRUE),
              "Avg. CI Width" =
                mean(`Avg. Credible Intervals Width`, na.rm = TRUE))

  return(summary_metrics)
}
```

Section 2 - Cross-Validation for the Additive Local Linear Model and Additive Semi-Local Linear Model

```
summary_metrics_local <-
  cross_validate_bsts_model("local_linear", ts_unrate_train, test_start_date)
summary_metrics_semi_local <-
  cross_validate_bsts_model("semi_local_linear", ts_unrate_train, test_start_date)

summary_metrics_local <- summary_metrics_local %>%
  mutate("Model Trend" = "local")
summary_metrics_semi_local <- summary_metrics_semi_local %>%
  mutate("Model Trend" = "semi-local")

combined_summary_metrics <-
  bind_rows(summary_metrics_local, summary_metrics_semi_local)

combined_summary_metrics
```

```
## # A tibble: 2 x 5
##   `Average MAE` `Average MAPE` `Average RMSE` `Avg. CI Width` `Model Trend`
##           <dbl>          <dbl>          <dbl>           <dbl> <chr>
## 1         0.639          10.6           0.764            4.32 local
## 2         0.593           9.77          0.696            3.61 semi-local
```