Forecasting Canadian Average Hourly Wages Using Time Series Models

Introduction

Multiple time series forecasting frameworks were applied to model and predict the monthly average hourly wages in Canada. Using data spanning from January 1997 to November 2024, the objective was to compare forecasting performance across decomposition-based modeling, SARIMA (Box-Jenkins), and Holt-Winters exponential smoothing approaches. To avoid structural distortions associated with the COVID-19 pandemic, only pre-pandemic observations were used for model estimation. Data from January 1997 to December 2018 were designated as the training set, while observations from January 2019 to December 2019 were reserved for out-of-sample forecast evaluation. Forecast accuracy was evaluated using the Mean Squared Prediction Error (MSPE).

Data Preparation

```r
#read the data
wages<-read.csv("D:/canadian-wage-forecasting-timeseries-analysis/employee_wages.csv")

#create time series object
wages_ts<-ts(wages$employee_wages, start = c(1997,1),frequency = 12)

#make sure that it is time series object
is.ts(wages_ts)
```

```
## [1] TRUE
```

STL Decomposition and ARMA Remainder Modeling

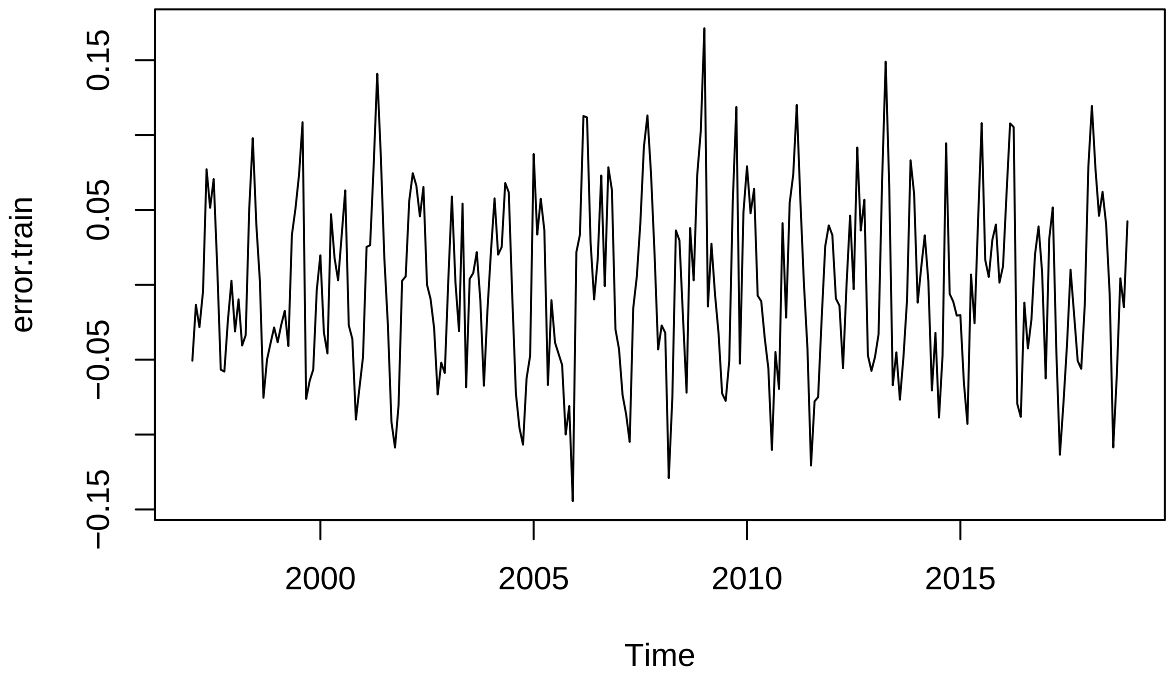
```r
#create training dataset
train.ts<-window(wages_ts, start = c(1997,1), end= c(2018,12))
#make sure that it is time series object
is.ts(train.ts)
```

```
## [1] TRUE
```

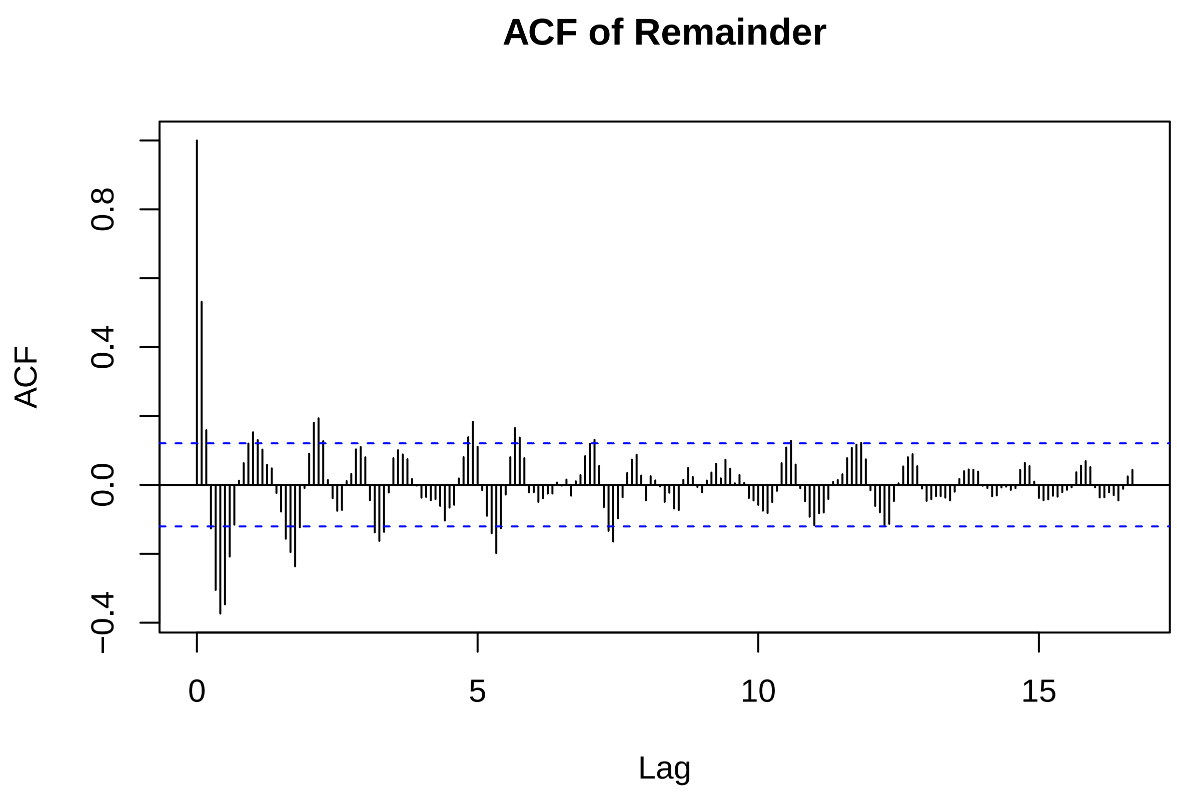
```r
#create testing dataset
test.ts<-window(wages_ts, start = c(2019,1), end=c(2019,12))
#make sure that it is time series object
is.ts(test.ts)
```

```
## [1] TRUE
```

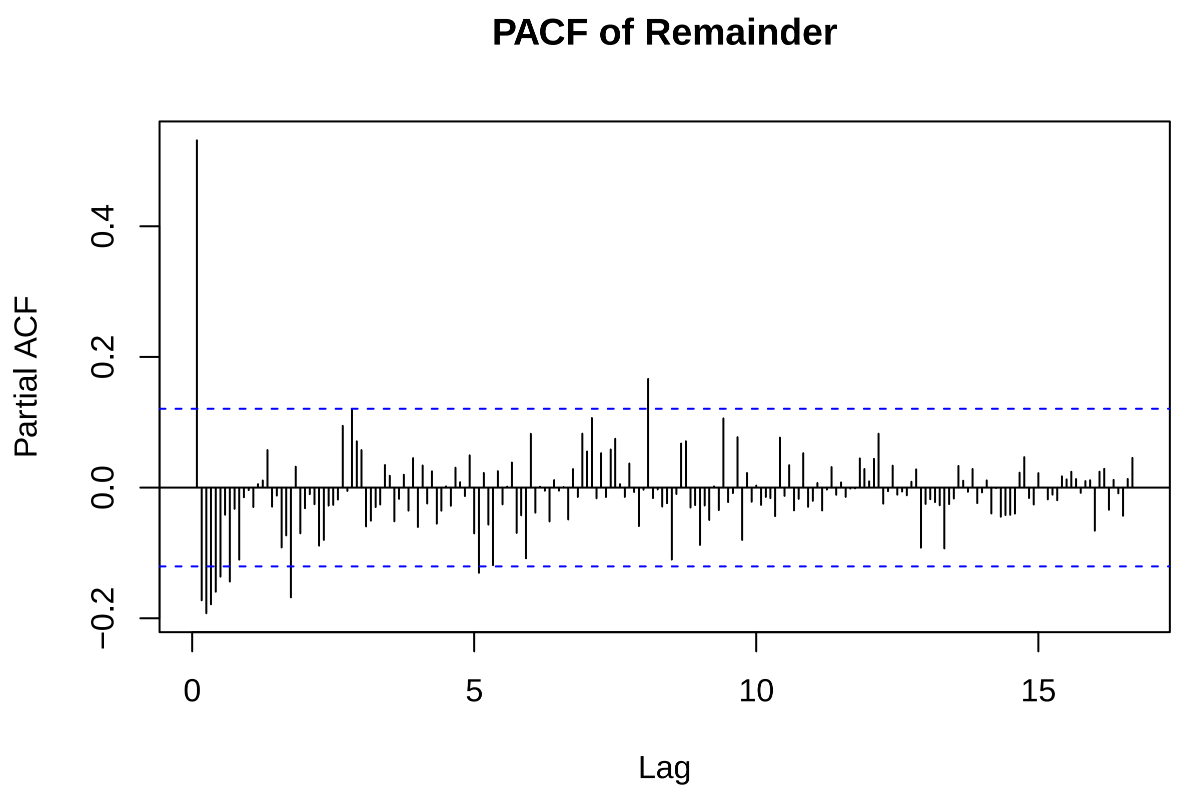
```r
#create seasonal decomposition
stl.decomp.train<-stl(train.ts, s.window = "periodic")

#create the remainder term
error.train<-stl.decomp.train$time.series[,"remainder"]

#create time plot for the remainder
plot(error.train) #look like it is follow white noise process
```



```r
#create sample acf for the remainder
acf(error.train, lag.max = 200, main = "ACF of Remainder")
```



```
#tails off like a damped sine waves

#create sample pacf for the remainder
pacf(error.train, lag.max = 200, main = "PACF of Remainder")
```



```
#tails off and no pattern
```

Based on the result above, the remainder seems to be stationary because there is no systematic change in mean (no trend) if we observed based on the time series plot. Then, if we look at the ACF plot, the ACF is like a "damped" since waves pattern and then tails off, where in the PACF we can see that it cuts-off at lag 8. Therefore, based on the observation of the ACF and PACF plot, we can conclude that the model is the AR(2) model. However, if we look at the AIC values as follows to confirm the model,

```
#make matrix to store aic values of range p and q [0,5]
out.aic <- matrix(nrow = 9, ncol = 9) #since matrix start from 1
for (p in 0:8) for (q in 0:8) #and here p,q range start from 0
{
  fm <- arima(error.train, order = c(p,0,q))
  out.aic[p+1,q+1] <- AIC (fm)
}
```

```
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
## Warning in arima(error.train, order = c(p, 0, q)): possible convergence
## problem: optim gave code = 1
```

```
round(out.aic,1)
```

```
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
##  [1,] -738.6 -815.8 -829.5 -829.0 -827.5 -837.6 -855.8 -856.1 -856.5
##  [2,] -824.3 -827.3 -828.3 -827.5 -852.5 -853.9 -857.1 -855.1 -854.6
##  [3,] -830.5 -864.4 -863.2 -861.2 -859.4 -857.8 -856.0 -853.4 -852.9
##  [4,] -838.2 -863.1 -861.2 -859.6 -857.7 -855.7 -855.3 -853.6 -850.9
##  [5,] -844.9 -861.2 -859.2 -861.6 -857.4 -857.7 -857.4 -855.4 -850.9
##  [6,] -850.4 -859.3 -857.7 -855.4 -853.6 -858.2 -857.7 -856.9 -858.7
##  [7,] -853.1 -851.1 -855.7 -858.6 -856.6 -855.7 -854.0 -851.8 -846.9
##  [8,] -851.5 -857.2 -856.7 -856.8 -854.7 -855.6 -853.6 -851.6 -848.7
##  [9,] -854.7 -854.1 -854.5 -856.0 -855.6 -852.8 -851.3 -850.2 -850.7
```

```
#return indices corresponding to the smallest aic value
which(out.aic == min(out.aic), arr.ind = T)
```

```
##      row col
## [1,]   3   2
```

```
#return indices corresponding to the smallest aic value
```

It showed that based on the AIC computed for each fitted ARMA (p,q) model, the model with the minimum AIC (-864.4) is ARMA (p,q) with $p = 2$ and $q = 1$. This confirmed by using auto.arima as follows,

```
#confirm the result using auto.arima
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

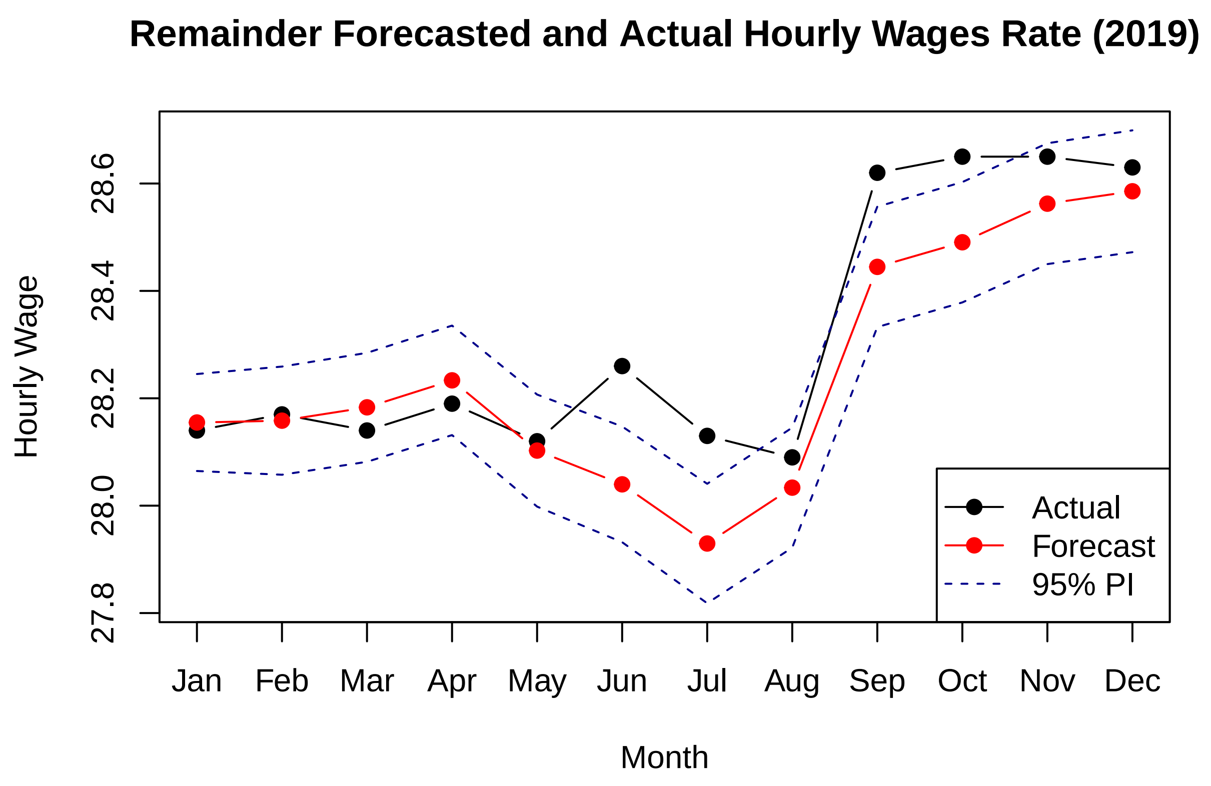
```
auto.arima(error.train, seasonal = FALSE) #assume stationary
```

```
## Series: error.train
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1
##       1.4165  -0.6218  -0.9135
## s.e.  0.0527   0.0480   0.0418
##
## sigma^2 = 0.002151:  log likelihood = 437.01
## AIC=-866.02   AICc=-865.87   BIC=-851.72
```

Then, if we look at the AIC for the AR(8), it also showed the AIC results of -854.7 where it is also similar with the AIC from the ARMA(2,1) model (-864.4)). Therefore, there are two possible models (AR(8) and ARMA(2,1)), and since for identify the model we need to look at ACF and PACF plot first, we consider to take AR(8) model as the fitted model . Then, we can use the fitted model to forecast monthly values of the average hourly wage rates as

```r
#fit the ar(8) model
error.fit <- arima(error.train, order = c(8,0,0))

#forecast the remainder from january 2019 to december 2019
error.forecast <- predict(error.fit, n.ahead = 12, prediction.interval = T,
                          level = 0.95, se.fit = T)

#extract the prediction interval
remainder.forecast <- error.forecast$pred
remainder.se <- error.forecast$se
lower.95 <- error.forecast$pred - 1.96 * error.forecast$se
upper.95 <- error.forecast$pred + 1.96 * error.forecast$se

#extract the trend and seasonal component
trend.train <- stl.decomp.train$time.series[, "trend"]
seasonal.train <- stl.decomp.train$time.series[, "seasonal"]

#extract the time indices from the training trend component
time.train <- time(trend.train)

#convert the last time value to numeric value
last_time <- as.numeric(tail(time.train, 1))
#retrieves last 1 value e.g. if data ends in december 2018,
#this represent time corresponding to december 2018

#generate forecast times for the next 12 months
forecast_times <- seq(from = last_time + 1/12, #move forward by one month
by = 1/12, #to increase by one month at a time since one month is 1/12 of a year
length.out = 12) #creates 12 equally spaced time points

#or manually by 265:276
#training set spans from january 1997 to december 2018,
#therefore thre are 22 years * 12 months = 264 observations
#the next month after training period (january 2019)
#become the 265th observations + 11 = december 2019 become the 276th observations

#fit the linear model on the trend component
trend.lm <- lm(trend.train ~ time.train)

#forecast the trend for 2019
trend.2019 <- predict(trend.lm, newdata = data.frame(time.train = forecast_times))
trend.2019 <- as.numeric(trend.2019)
```

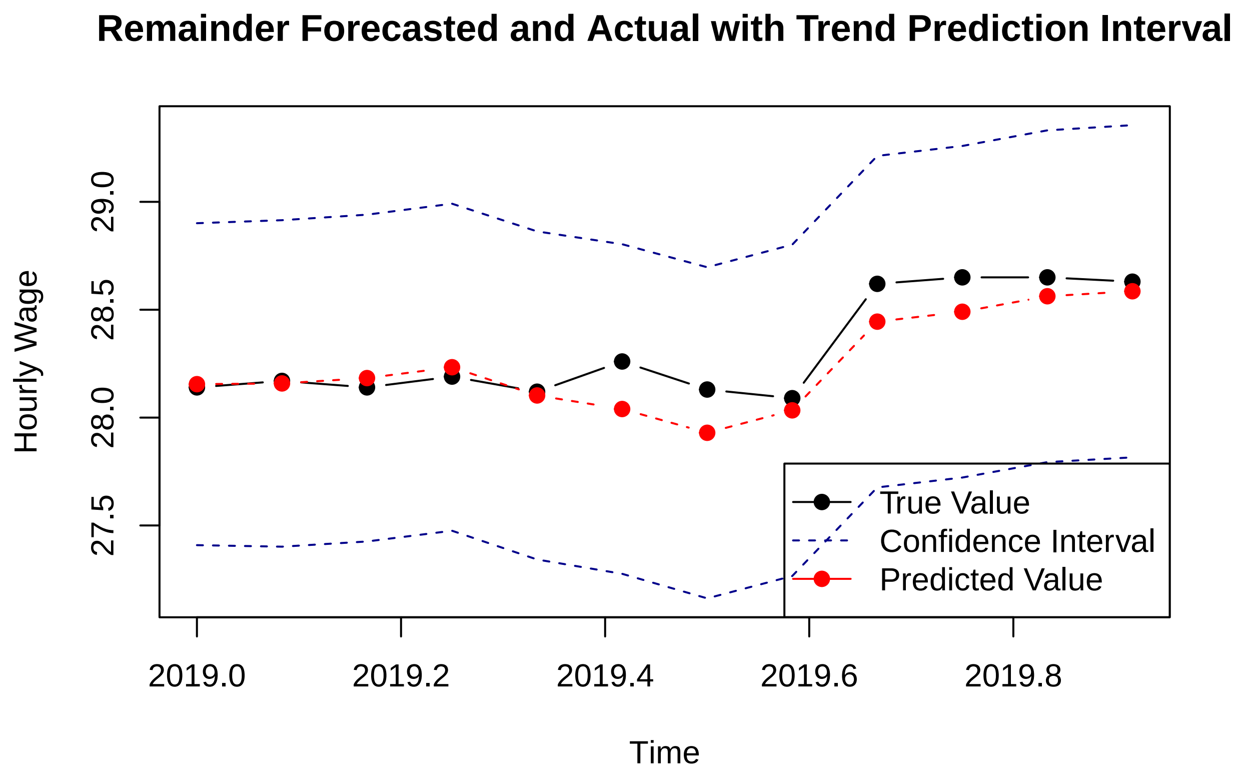
```r
#find the seasonal component for 2019
seasonal.2019 <- tail(seasonal.train, 12)
#retrieves last 12 values e.g. from january 2019 to december 2019
seasonal.2019 <- as.numeric(seasonal.2019)

#combine all components
forecast.2019 <- trend.2019 + seasonal.2019 + remainder.forecast

#combine all prediction intervals
lower.2019 <- trend.2019 + seasonal.2019 + lower.95
upper.2019 <- trend.2019 + seasonal.2019 + upper.95

#convert all to the time series for alignment
forecast.ts.stl <- ts(forecast.2019, start = c(2019, 1), frequency = 12)
lower.ts <- ts(lower.2019, start = c(2019, 1), frequency = 12)
upper.ts <- ts(upper.2019, start = c(2019, 1), frequency = 12)

#plot the time series
plot(test.ts, ylim = range(c(test.ts, lower.ts, upper.ts)),
     main = "Remainder Forecasted and Actual Hourly Wages Rate (2019)",
     ylab = "Hourly Wage",
     xlab = "Month", pch = 19, type = "b", xaxt = "n")
axis(1, at = time(test.ts), labels = month.abb)
lines(forecast.ts.stl, col = "red", pch = 19, type = "b")
lines(lower.ts, col = "darkblue", lty = 2)
lines(upper.ts, col = "darkblue", lty = 2)
legend("bottomright", legend = c("Actual", "Forecast", "95% PI"),
       col = c("black", "red", "darkblue"),
       lty = c(1, 1, 2), pch = c(19,19,NA))
```



However, we can include the prediction interval for the trend as well,

```
#predict the remainder component ar(8)
predict_remainder_test <- predict(error.fit, n.ahead = 12)

#predict the seasonal component
predict_seasonal_test <- rep(stl.decomp.train$time.series[, "seasonal"][1:12],
                             length.out = length(test.ts))

#predict the trend component
predict_trend_test <- predict(trend.lm,
                              newdata = data.frame(time.train = forecast_times),
                              interval = "prediction")

#combine trend, seasonal, and remainder predictions
employee_test_fit_ts <- predict_remainder_test$pred + predict_seasonal_test +
  predict_trend_test[, 1]

#combine lower and upper bounds with remainder
employee_test_lower_ts <- predict_remainder_test$pred + predict_seasonal_test +
  predict_trend_test[, 2] - 1.96 * predict_remainder_test$se
employee_test_upper_ts <- predict_remainder_test$pred + predict_seasonal_test +
  predict_trend_test[, 3] + 1.96 * predict_remainder_test$se

#convert all to time series
employee_test_fit_ts <- ts(employee_test_fit_ts, start = c(2019, 1), frequency = 12)
```

```r
employee_test_lower_ts <- ts(employee_test_lower_ts, start = c(2019, 1), frequency = 12)
employee_test_upper_ts <- ts(employee_test_upper_ts, start = c(2019, 1), frequency = 12)

#plot the actual vs predicted values
plot(test.ts,
     type = "b",
     ylab = "Hourly Wage",
     main = "Remainder Forecasted and Actual with Trend Prediction Interval",
     pch = 19,
     ylim = range(c(employee_test_upper_ts,
                    employee_test_fit_ts,
                    employee_test_lower_ts)))

#add the prediction interval and legend
lines(employee_test_lower_ts, col = "darkblue", lty = 2)
lines(employee_test_fit_ts, col = "red", lty = 2, type = "b", pch = 19)
lines(employee_test_upper_ts, col = "darkblue", lty = 2)
legend("bottomright",
       legend = c("True Value", "Confidence Interval", "Predicted Value"),
       lty = c(1, 2, 1),
       col = c("black", "darkblue", "red"),
       pch = c(19, NA, 19))
```

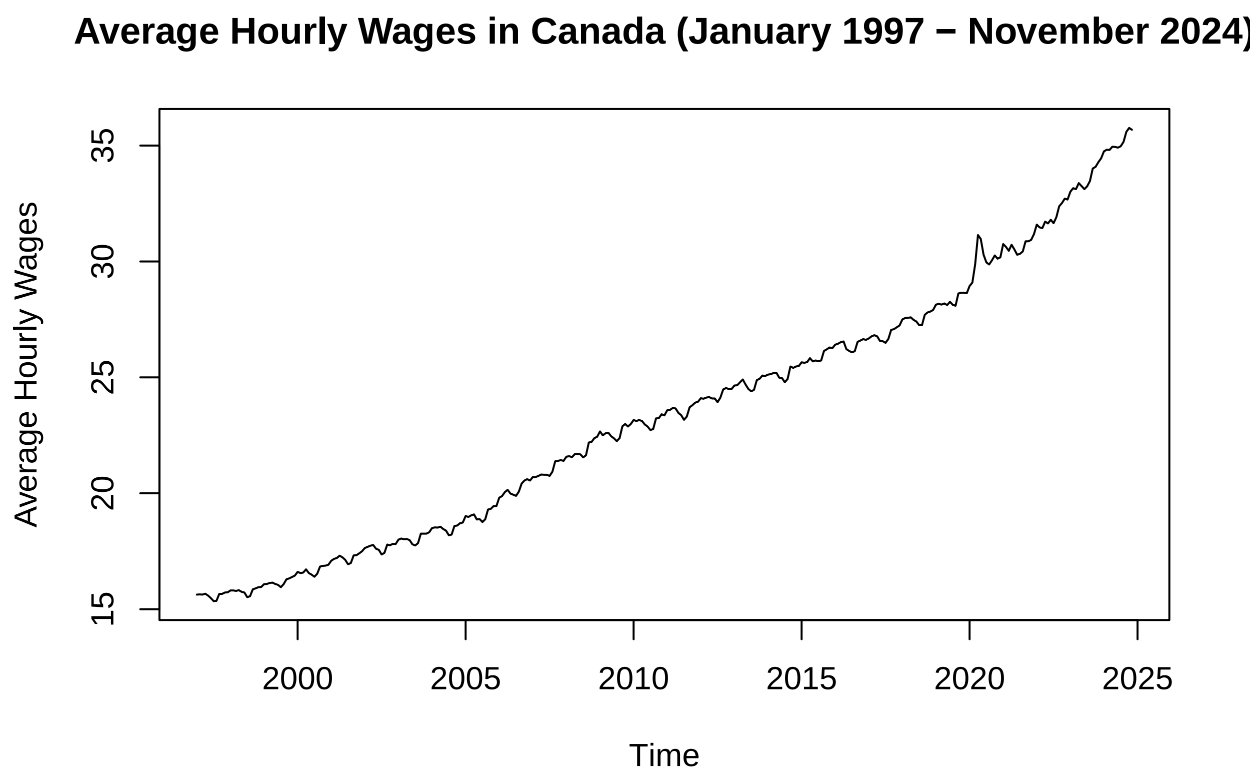


The first model based only on the ARMA model, fails to capture actual values in several months, likely due to excluding the trend component or external factors. The second interval, which includes both ARMA and trend model uncertainties, is wider but more accurate, successfully capturing the true values.

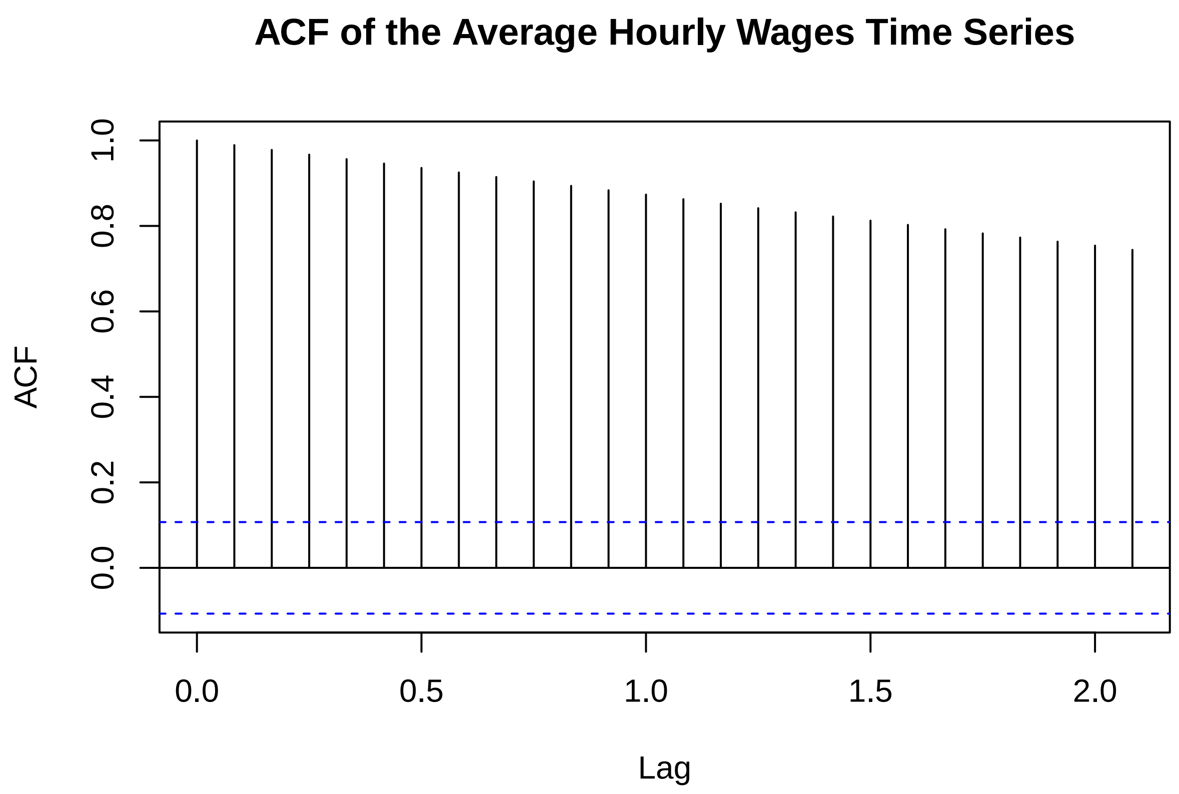
Box-Jenkins SARIMA Modeling

First, we can plot the series and its sample ACF and sample partial ACF as follows,

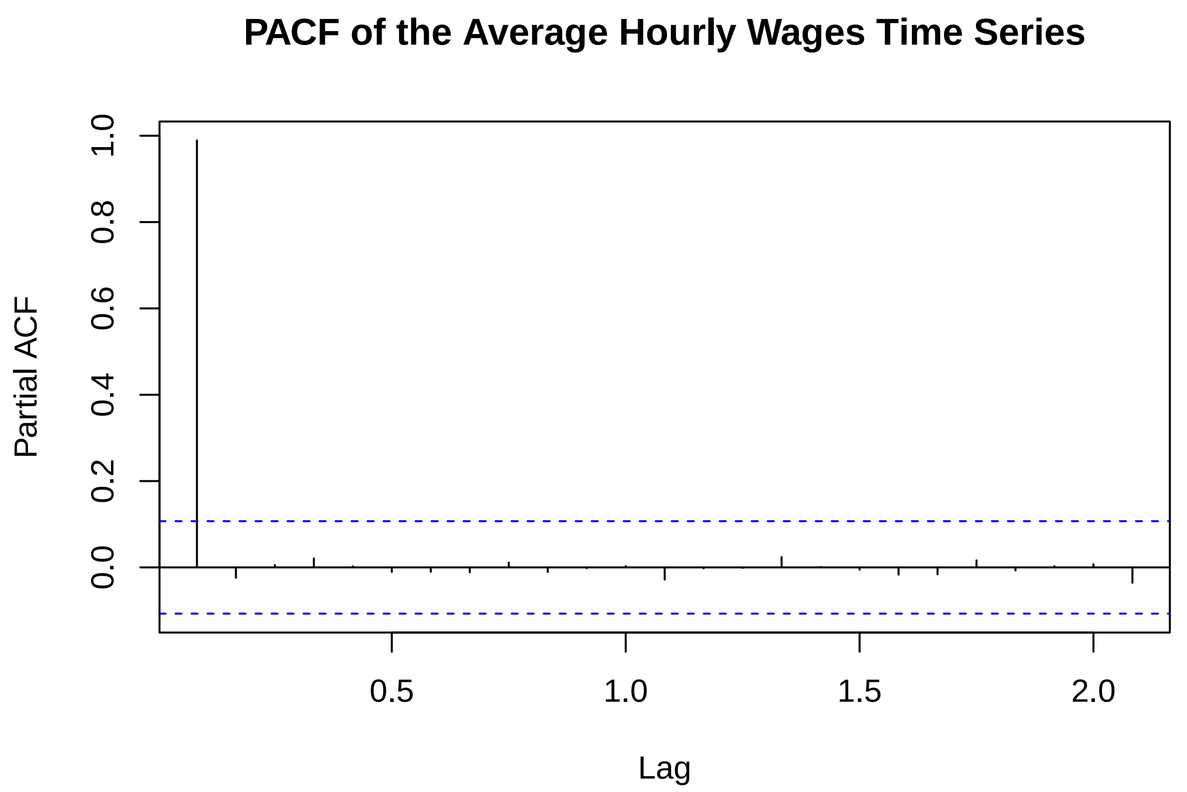
```r
#plot the time series
plot(wages_ts,
     main = "Average Hourly Wages in Canada (January 1997 - November 2024)",
     ylab = "Average Hourly Wages")
```



```r
#plot the sample acf
acf(wages_ts, main = "ACF of the Average Hourly Wages Time Series")
```

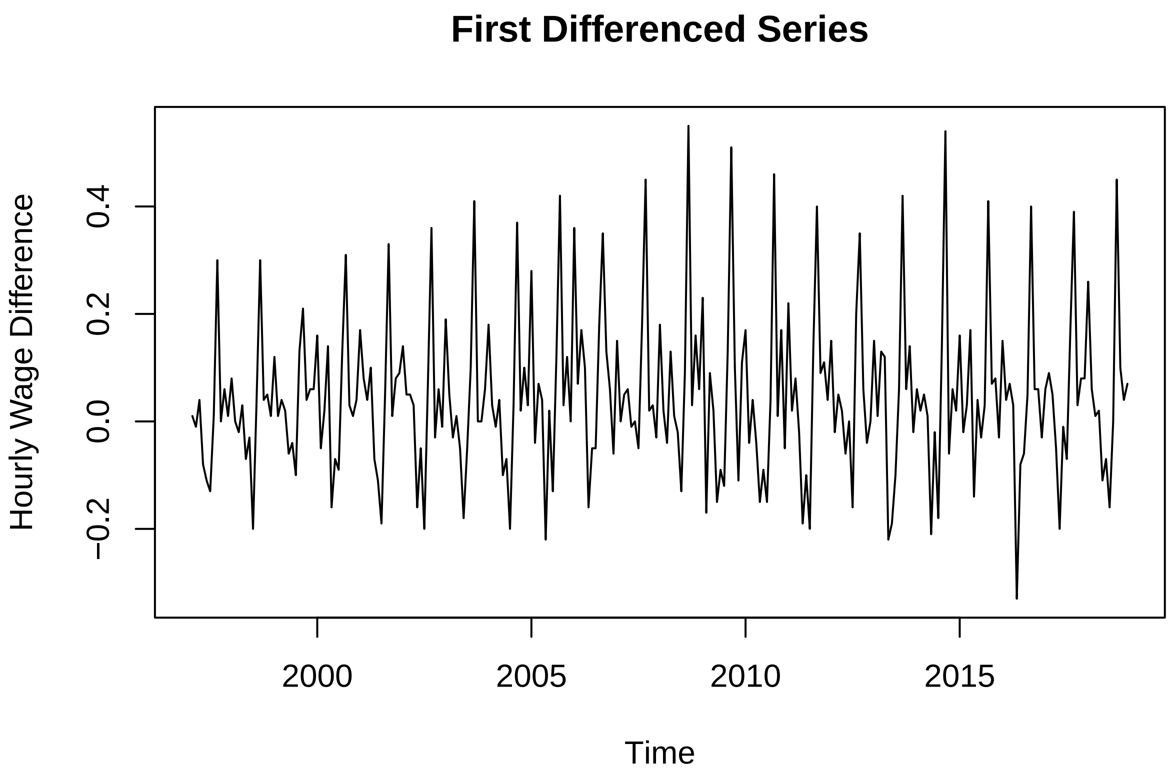


```r
#plot the sample pacf
pacf(wages_ts, main = "PACF of the Average Hourly Wages Time Series")
```

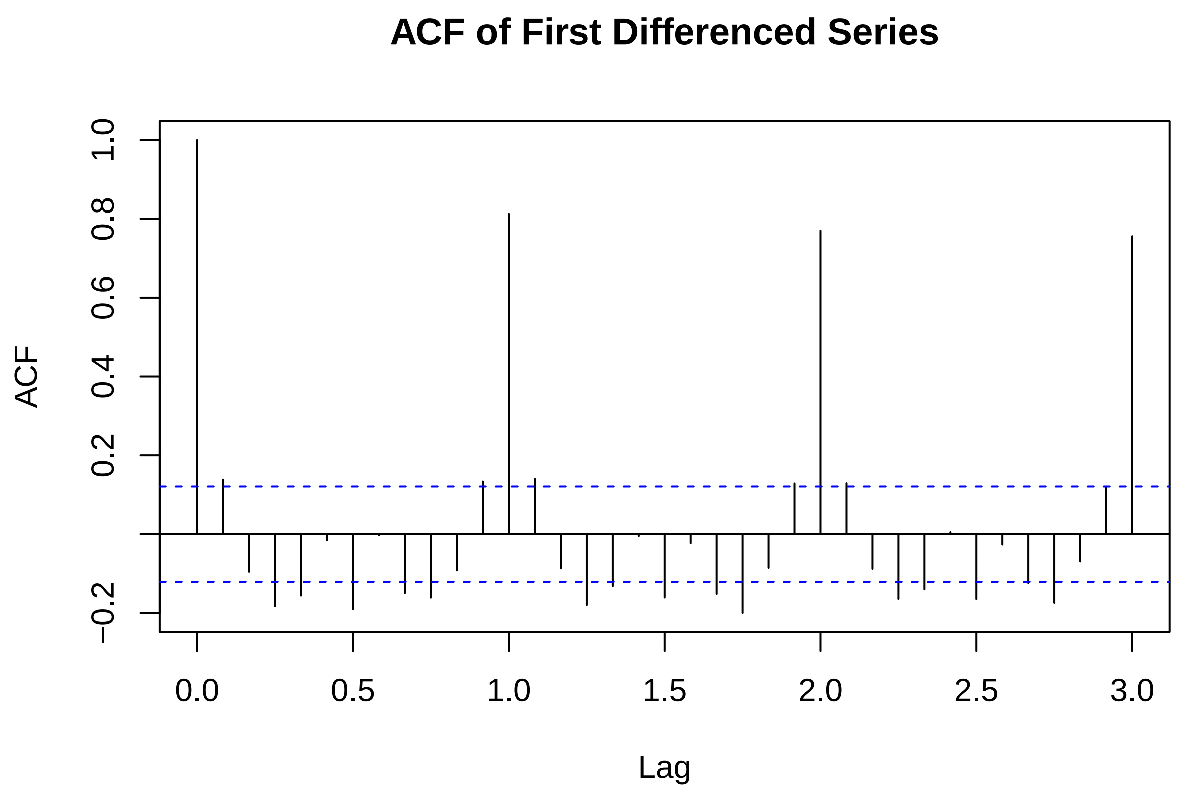


As we can see from the plot of the time series data of the average hourly wages in Canada, it was observed that the series has an increasing trend as the time increases. There is also seasonal variation observed from the plot, in which it is a additive seasonal effect since the size of the seasonal effect appears to constant from year to year as the mean increases. The series also seems to be not stationary since the mean is not constant i.e. there is an obvious upward trend observed from the time series plot, and the sample ACF of the series also decays slowly with the increase of the lags. Therefore, we can remove this trend by apply non-seasonal differencing (d) as follows,

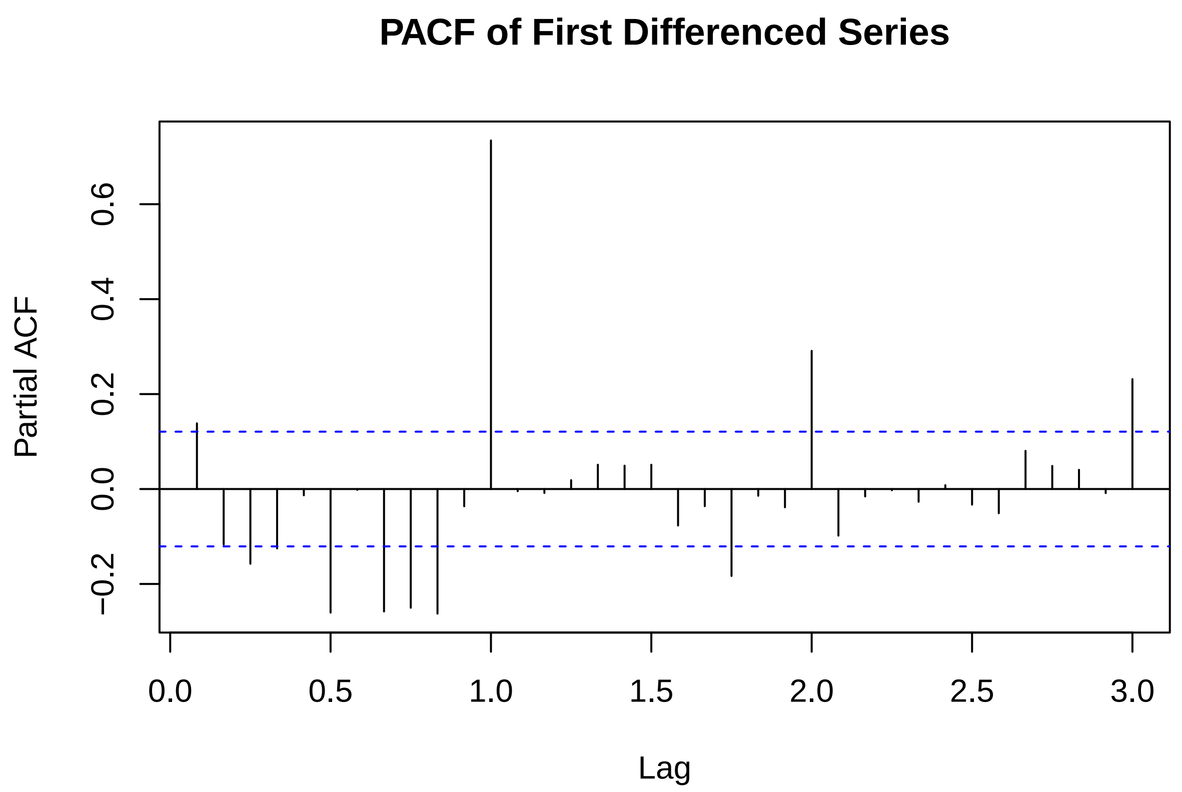
```r
#non-seasonal differencing
train_diff <- diff(train.ts, lag = 1)
plot(train_diff, main = "First Differenced Series", ylab = "Hourly Wage Difference")
```



```r
acf(train_diff, lag.max = 36, main = "ACF of First Differenced Series")
```

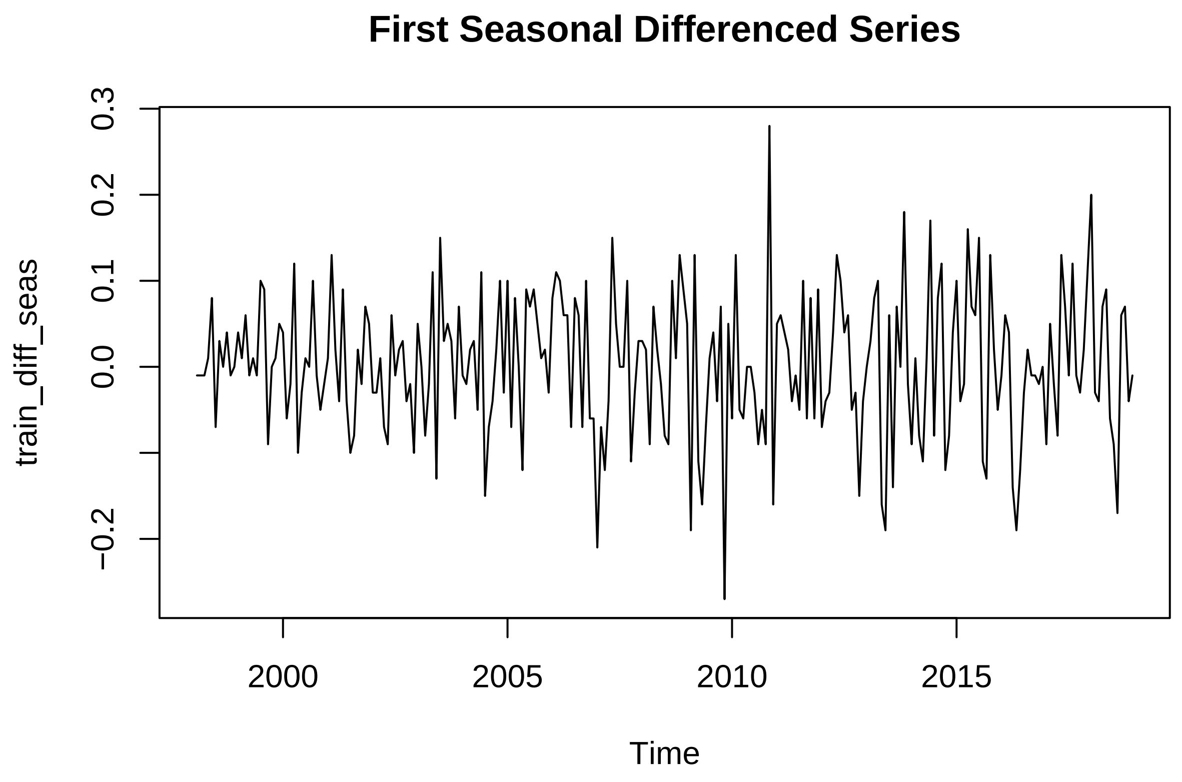


```r
pacf(train_diff, lag.max = 36, main = "PACF of First Differenced Series")
```

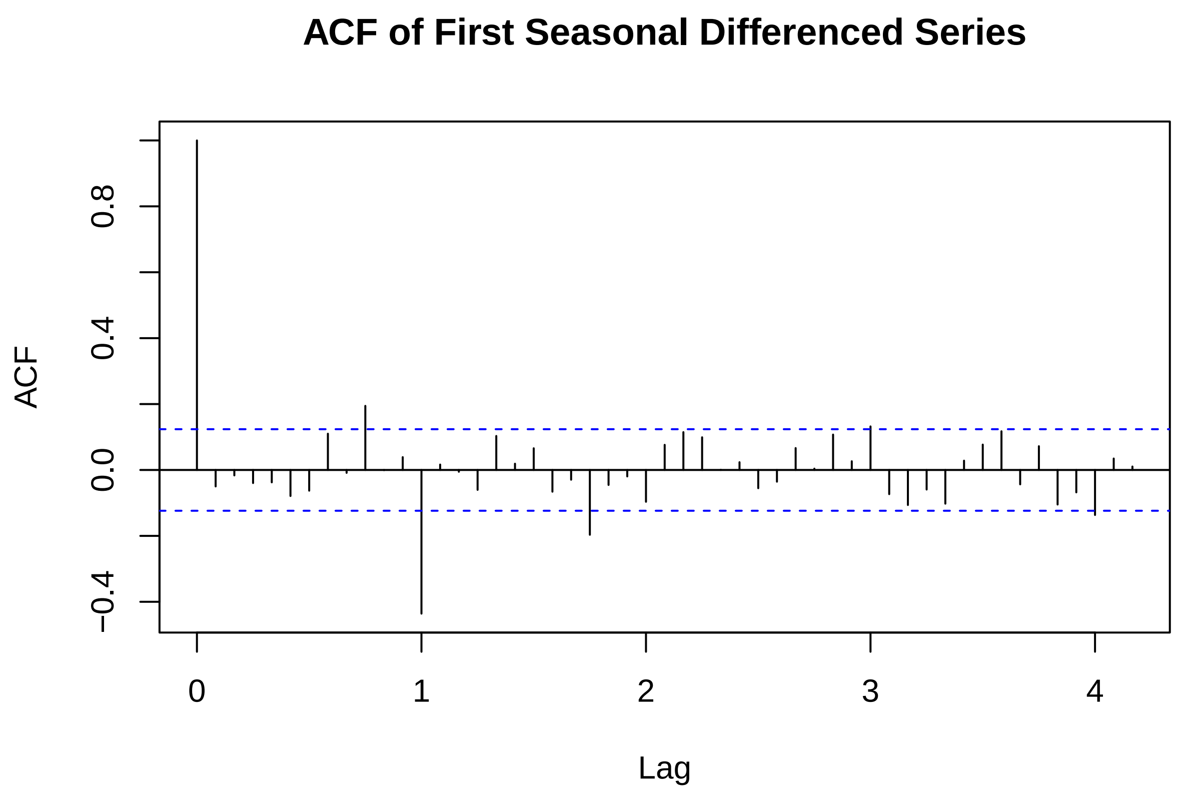


As we can see from the plot above, after first differencing (d=1), the series look more stationary since it seems that there is no trend observed in the time series plot. However, as we look at the ACF plot, there is observable seasonal spikes every 12 lag because of the monthly data (seasonal variation of period s = 12). Then, since the values for low lags are used to decide on values of p and q for pure AR and MA process, we can decide that p = 0 and q = 0 since the ACF already cuts-off at lag 0 and the PACF tails off with no pattern. Since there is seasonal pattern, we need to apply seasonal differencing (D) with difference at lag 12 as follows,

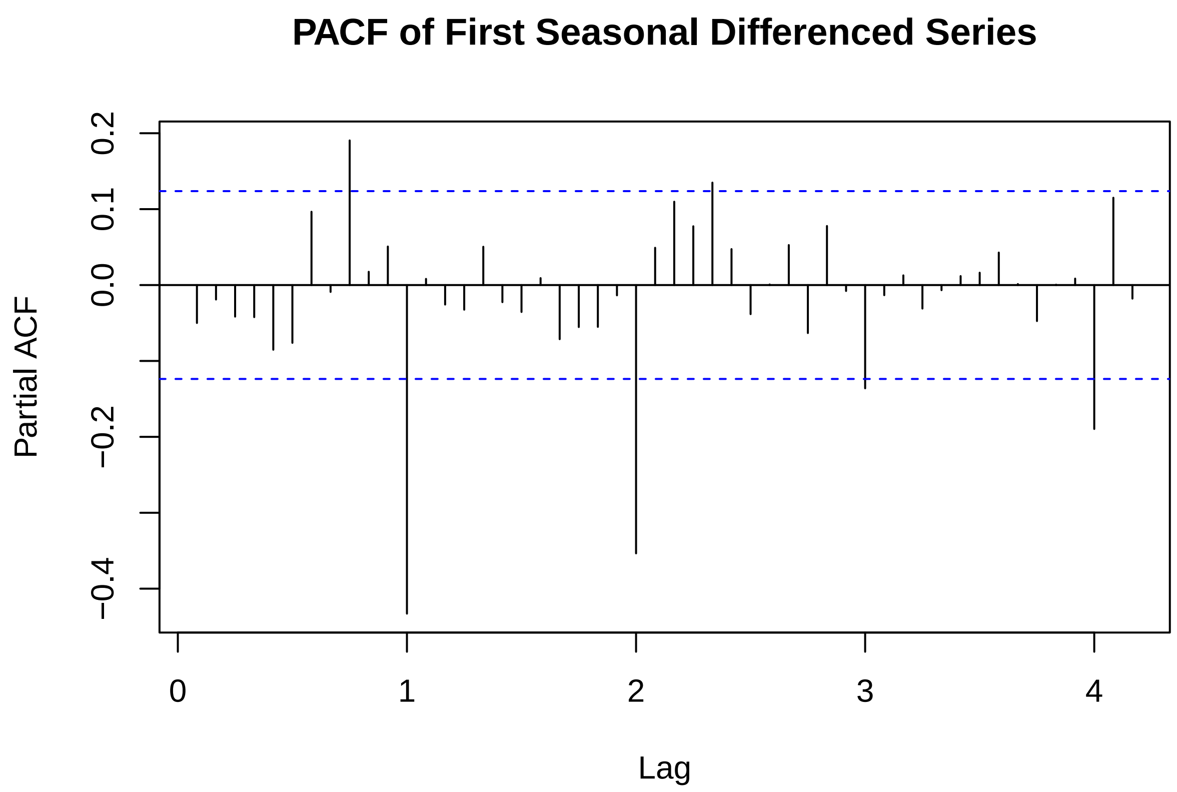
```r
#seasonal differencing
train_diff_seas <- diff(train_diff, lag = 12)
plot(train_diff_seas, main = "First Seasonal Differenced Series")
```



```
acf(train_diff_seas, lag.max = 50, main = "ACF of First Seasonal Differenced Series")
```



```
pacf(train_diff_seas, lag.max = 50, main = "PACF of First Seasonal Differenced Series")
```



Now, we can see that the seasonal pattern already removed with the seasonal differencing (D = 1) and s = 12. Then, since the values of the ACF cuts-off at lag 1 and the PACF is tails off with no specific pattern, we can decide that it follows seasonal MA component and the values of Q = 1 and P = 0. Alternatively, we can determine the order of the p and q by compute the AIC and taking the p,q, P, Q that corresponds to the lowest AIC values as follows,

```r
#fit models with different p, q, P, Q
models <- list()
aic_values <- c()
for (p in 0:2) {
  for (q in 0:2) {
    for (P in 0:1) {
      for (Q in 0:1) {
        model_id <- paste(p, q, P, Q, sep = ",")
        model <- tryCatch(
          arima(train.ts, order = c(p, 1, q),
                seasonal = list(order = c(P, 1, Q), period = 12)),
          error = function(e) NULL
        )
        if (!is.null(model)) {
          models[[model_id]] <- model
          aic_values[model_id] <- AIC(model)
        }
      }
    }
  }
}

# Find the best model
best_model <- names(which.min(aic_values))
best_model
```

```
## [1] "0,0,0,1"
```

or by confirming value using the auto.arima as follows,

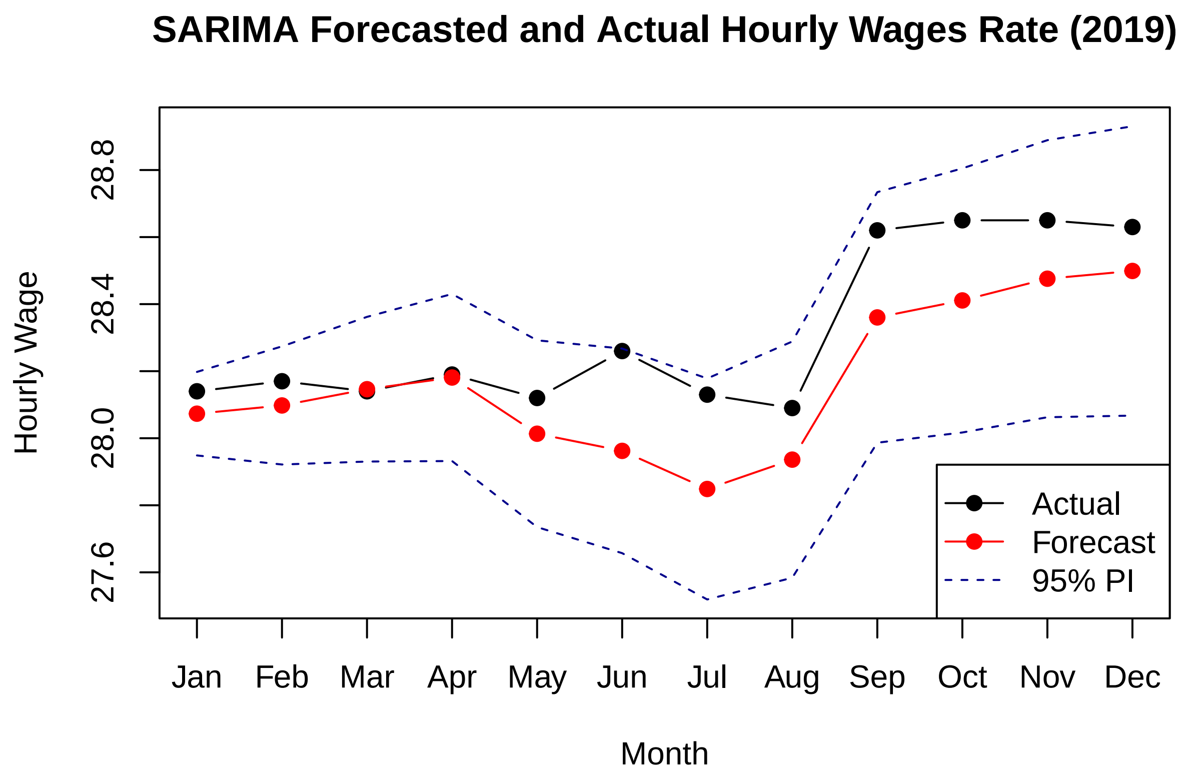
```
auto_model <- auto.arima(train.ts, seasonal = T)
auto_model
```

```
## Series: train.ts
## ARIMA(0,1,0)(0,1,1)[12]
##
## Coefficients:
##          sma1
##       -0.8247
## s.e.   0.0506
##
## sigma^2 = 0.004067:  log likelihood = 328.74
## AIC=-653.48   AICc=-653.44   BIC=-646.43
```

which all of them confirmed that we have p = 0, d = 1, q = 0, P = 0, D = 1, Q = 1, and s = 12 i.e. $SARIMA(0,1,0) \times (0,1,1)_{12}$. Then, we can forecast the monthly values of the average hourly wage rates using this model as follows,

```
#create the fitted model
sarima_fit <- arima(train.ts, order = c(0, 1, 0),
                    seasonal = list(order = c(0, 1, 1), period = 12))

#extract the prediction interval
sarima_forecast <- predict(sarima_fit, n.ahead = 12)
forecast.2019 <- as.numeric(sarima_forecast$pred)
lower.95 <- as.numeric(sarima_forecast$pred - 1.96 * sarima_forecast$se)
upper.95 <- as.numeric(sarima_forecast$pred + 1.96 * sarima_forecast$se)

#convert to time series
forecast.ts.sarima <- ts(forecast.2019, start = c(2019, 1), frequency = 12)
lower.ts <- ts(lower.95, start = c(2019, 1), frequency = 12)
upper.ts <- ts(upper.95, start = c(2019, 1), frequency = 12)

#plot the time series model
plot(test.ts, type = "b", pch = 19,
     ylim = range(c(test.ts, lower.ts, upper.ts)),
     main = "SARIMA Forecasted and Actual Hourly Wages Rate (2019)",
     ylab = "Hourly Wage",
     xlab = "Month", xaxt = "n")
axis(1, at = time(test.ts), labels = month.abb)
lines(forecast.ts.sarima, type = "b", pch = 19, col = "red")
lines(lower.ts, type = "l", lty = 2, col = "darkblue")
```

```r
lines(upper.ts, type = "l", lty = 2, col = "darkblue")
legend("bottomright", legend = c("Actual", "Forecast", "95% PI"),
       col = c("black", "red", "darkblue"), lty = c(1, 1, 2),
       pch = c(19, 19, NA))
```



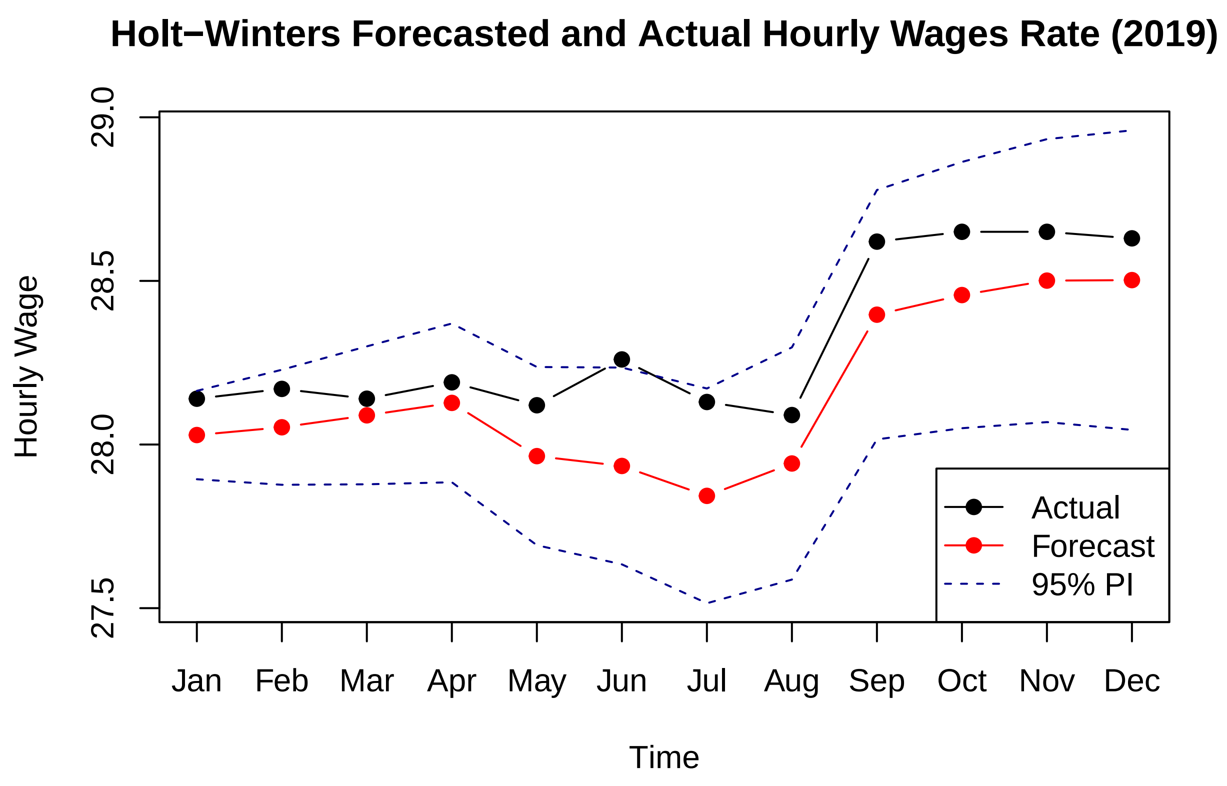
Holt-Winters Exponential Smoothing

We know previously from the time series plot shown in the previous section that the it was observed that the series has an increasing trend as the time increases, where the seasonal variation observed from the plot is an additive seasonal effect since the size of the seasonal effect appears to constant from year to year as the mean increases. Therefore, we fit the Holt-Winters with additive seasonality as follows,

```r
#fit holt-winters with additive seasonality
hw_fit_add <- HoltWinters(train.ts, seasonal = "additive")
HoltWinters(train.ts, seasonal = "additive")
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = train.ts, seasonal = "additive")
##
## Smoothing parameters:
##  alpha: 0.8055799
##  beta : 0.03454129
##  gamma: 1
##
## Coefficients:
##            [,1]
## a   27.81293239
## b    0.04936552
## s1   0.16668893
## s2   0.14102575
## s3   0.12816743
## s4   0.11682461
## s5  -0.09518329
## s6  -0.17463188
## s7  -0.31536320
## s8  -0.26576967
## s9   0.13952376
## s10  0.15010880
## s11  0.14490593
## s12  0.09706761
```

```
#extract the prediction interval
hw_forecast_add <- predict(hw_fit_add, n.ahead = 12, prediction.interval = T, level = 0.95)
forecast.2019 <- as.numeric(hw_forecast_add[, "fit"])
lower.95 <- as.numeric(hw_forecast_add[, "lwr"])
upper.95 <- as.numeric(hw_forecast_add[, "upr"])

#convert to time series
forecast.ts.hw <- ts(forecast.2019, start = c(2019, 1), frequency = 12)
lower.ts <- ts(lower.95, start = c(2019, 1), frequency = 12)
upper.ts <- ts(upper.95, start = c(2019, 1), frequency = 12)

#plot the time series model
plot(test.ts, type = "b", pch = 19,
     ylim = range(c(test.ts, lower.ts, upper.ts)),
     main = "Holt-Winters Forecasted and Actual Hourly Wages Rate (2019)",
     ylab = "Hourly Wage",
     xlab = "Time", xaxt = "n")
axis(1, at = time(test.ts), labels = month.abb)
lines(forecast.ts.hw, type = "b", pch = 19, col = "red")
lines(lower.ts, type = "l", lty = 2, col = "darkblue")
lines(upper.ts, type = "l", lty = 2, col = "darkblue")
legend("bottomright", legend = c("Actual", "Forecast", "95% PI"),
       col = c("black", "red", "darkblue"),
       lty = c(1, 1, 2), pch = c(19, 19, NA))
```



Forecast Accuracy Comparison using MSPE

The Mean Squared Prediction Error (MSPE) is calculated as,

$MSE = \frac{1}{n}\sum_{t=1}^{u}(y_t - \hat{y_t})^2$

where:

y_t = actual value

$\hat{y_t}$ = forecasted value

n = number of observation

Then, implementing these on all of the methods we get as the result as follows,

```
#find the mspe for the seasonal decomposition in which fit arma model for the remainder
mspe_stl <- mean((test.ts - forecast.ts.stl)^2)
mspe_stl
```

```
## [1] 0.01350391
```

```
#find the mspe for the box-jenkins forecasting
mspe_sarima <- mean((test.ts - forecast.ts.sarima)^2)
mspe_sarima
```

```
## [1] 0.03205542
```

```r
#find the mspe for holt-winters forecasting
mspe_hw <- mean((test.ts - forecast.ts.hw)^2)
mspe_hw
```

```
## [1] 0.03271917
```

Therefore, based on the Mean Squared Prediction Error (MSPE), the method that performs best in forecasting the values is the seasonal decomposition model that fitted in the ARMA model for the remainder term, which has the lowest value of the MSPE, followed by the model using Box-Jenkins forecasting and lastly the model from the Holt-Winters forecasting. The benefit of using the seasonal decomposition combined with the ARMA for the remainder is because it decompose the series into the trend, seasonal, and remainder, making it easier to understand and easier to interpret, but if the remainder still contains hidden seasonality or other structures not captured by decomposition, the ARMA model might be not that optimal. For the Box-Jenkins approach, the SARIMA model can directly incorporates seasonal AR and MA terms, as well as seasonal differencing in a single model. However, it involves quite complex model selection, and if the data's seasonal pattern changes over time or if there are nonlinearities, SARIMA can struggle unless extended with additional terms. Lastly, for the Holt-Winters method, it is more simple and easy to implement when the data have clear trend and seasonality, but typically only handles a single seasonal pattern and assumes either additive or multiplicative seasonality, therefore it is less flexible and if the seasonality or trend changes over time, or if the data have more complex patterns, Holt-Winters may not capture them as effectively.