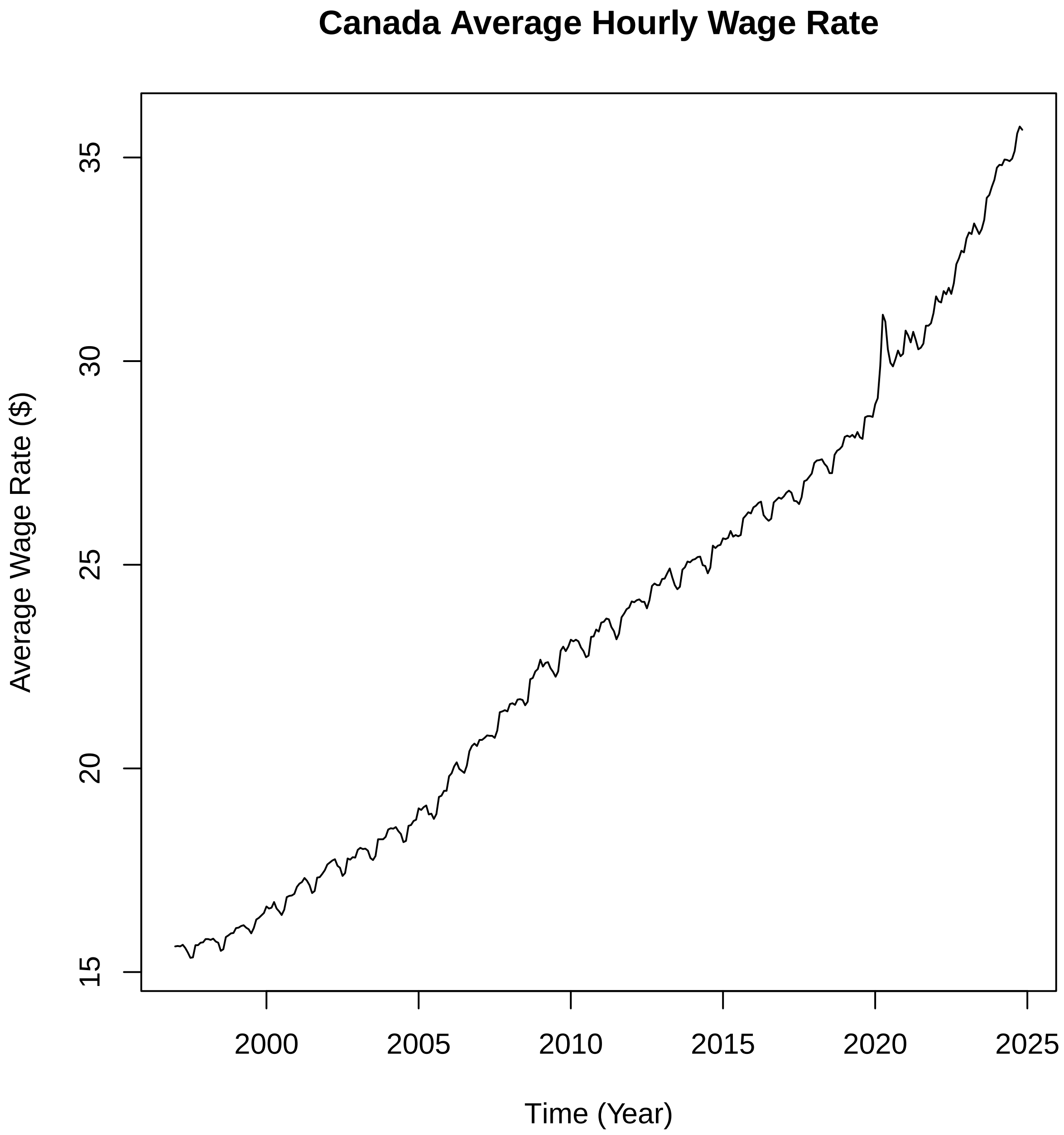
STAT 443: Homework 3

Aronn Grant Laurel (21232475)

14 March, 2025

Question 1 ARMA Model

```r
# this is where your R code goes

# Initialisation
data <- read.csv("employee_wages_total_industry.csv")

data.ts <- ts(data$employee_wages,
              start = c(1997, 1),
              frequency = 12)


# January 1997 to December 2018 as the training set for model fitting
data.ts.train <- window(data.ts,
                        start = c(1997, 1),
                        end = c(2018, 12),
                        frequency = 12)

# from January 2019 to December 2019 as the test set
data.ts.test <- window(data.ts,
                       start = c(2019, 1),
                       end = c(2019, 12),
                       frequency = 12)
plot(data.ts,
     main = "Canada Average Hourly Wage Rate",
     xlab = "Time (Year)",
     ylab = "Average Wage Rate ($)")
```

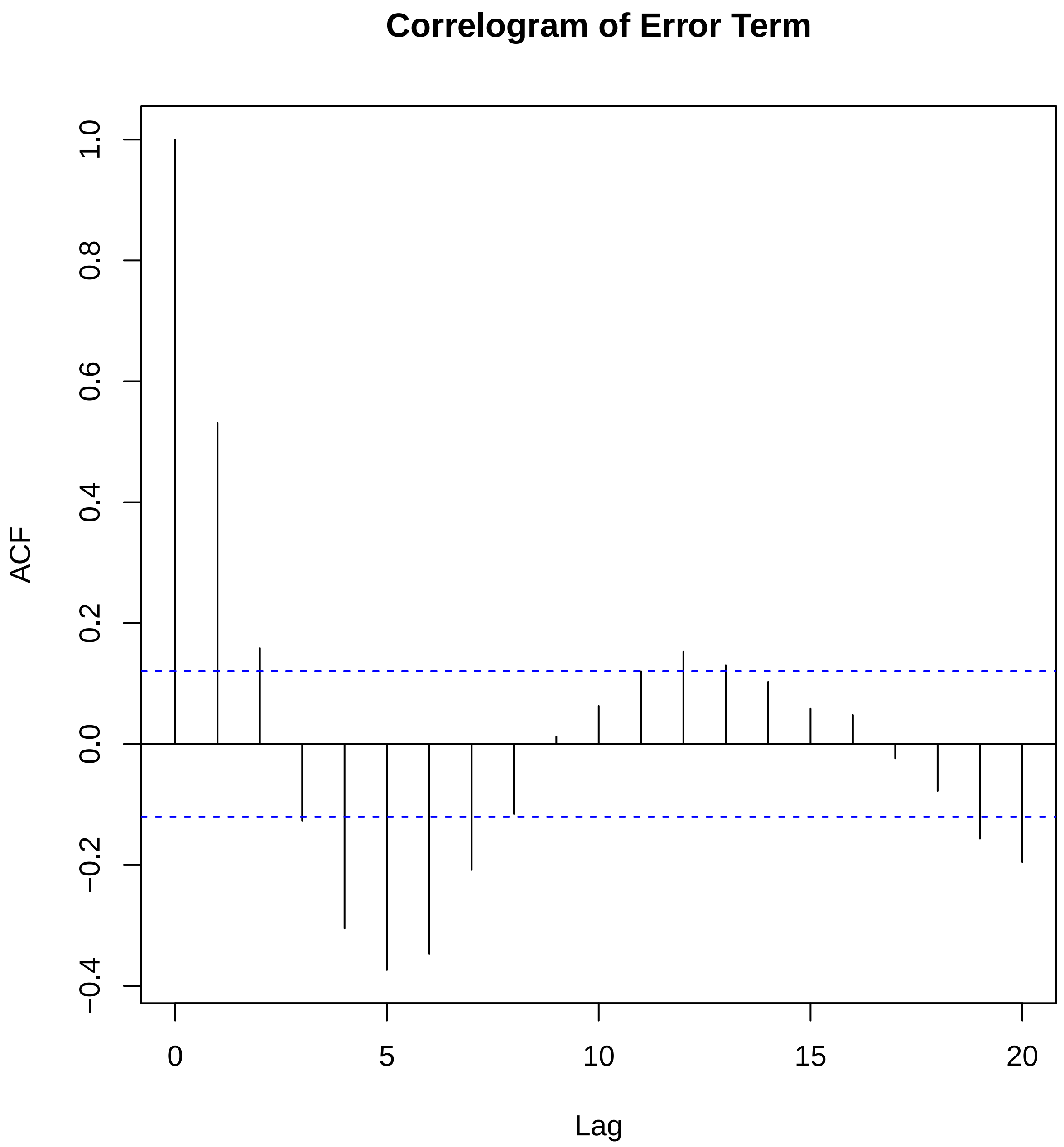


```
# LOESS Smoothing
# De-seasonalize Series
data.ts.ls <- stl(data.ts.train,
                  s.window = "periodic")

# From HW1
trend.ls <- data.ts.ls$time.series[,"trend"]
seas.ls <- data.ts.ls$time.series[,"seasonal"]
error.ls <- data.ts.ls$time.series[, "remainder"]

# ARMA Model fitting for Remainder Term
```

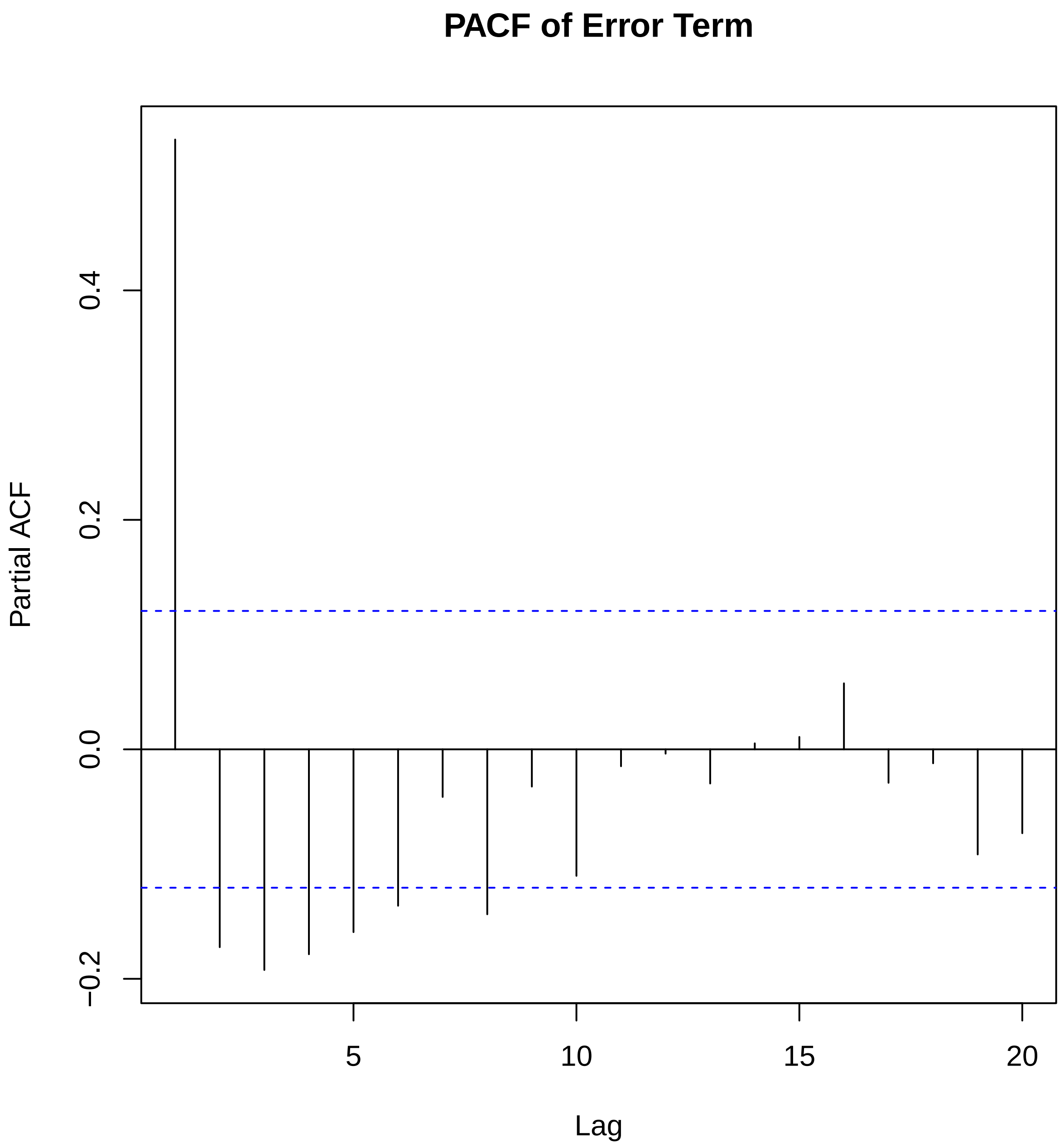
```
error.time.ls <- seq_along(error.ls)
error.ls.df <- data.frame("Remainder" = error.ls,
                          "Time" = error.time.ls)
```

From the time plot, we can observe clear upward trend meaning that our mean is not constant and therefore not stationary. We should apply an ARIMA series with d > 0 to transform the series into a stationary process, but in our question we will be fitting an ARMA model (d = 0).

```
# this is where your R code goes
error.ts <- ts(error.ls)

acf(error.ts,
    lag.max = 20,
    main = "Correlogram of Error Term")
```



```r
pacf(error.ts,
     lag.max = 20,
     main = "PACF of Error Term")
```



Our ACF plot shows a 'damped' like sine wave and our PACF plot displays a negative cut off at lag 6 and 8 which we will use (p) for our ARMA model (a Pure AR Process) and assess the better fitted model.

```
# this is where your R code goes
# Fitting our model
model <- arima(error.ts, order = c(0, 0, 8))
model
```

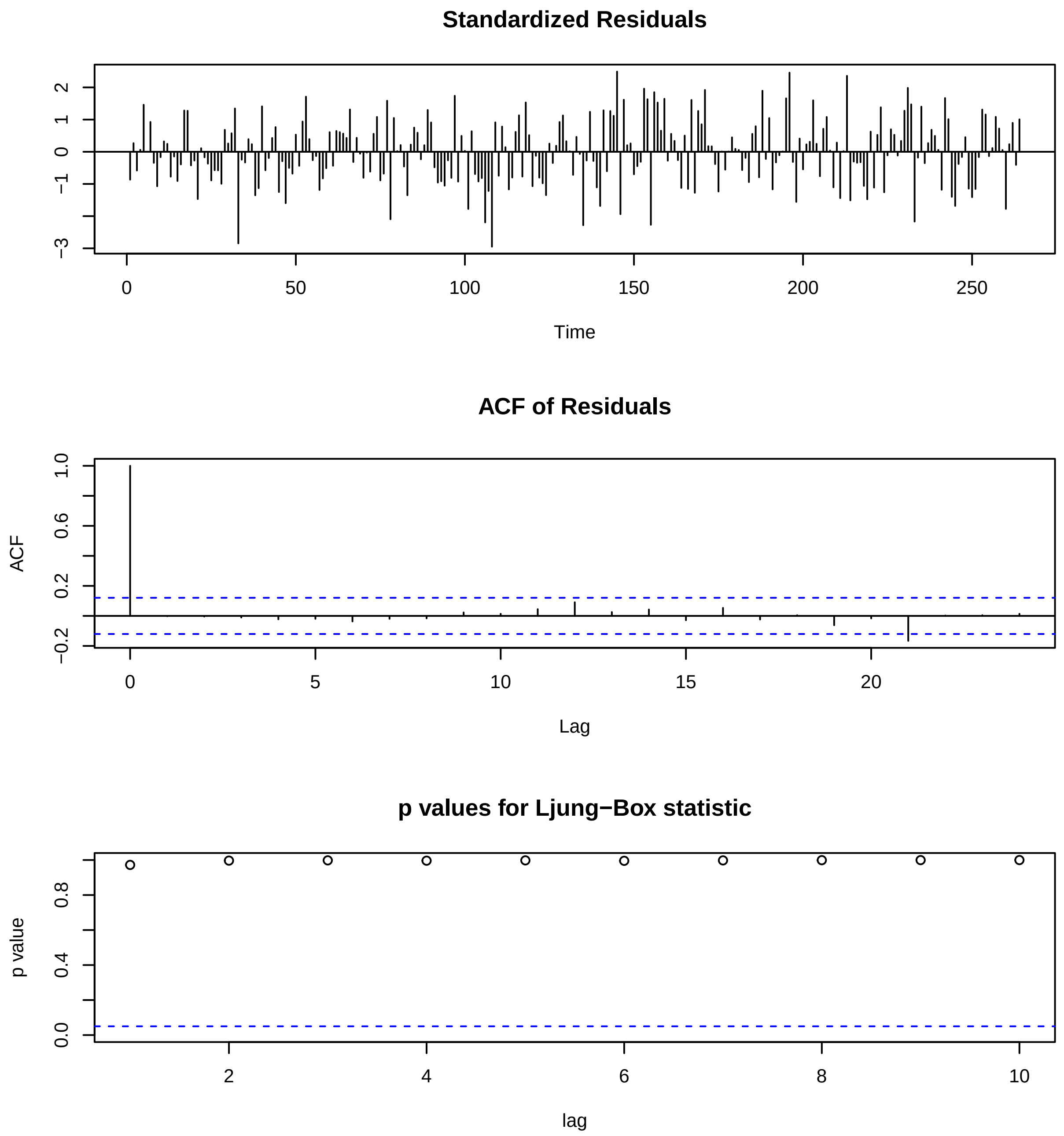
```
##
## Call:
## arima(x = error.ts, order = c(0, 0, 8))
##
```

```
## Coefficients:
##          ma1     ma2      ma3      ma4      ma5      ma6      ma7      ma8
##       0.4853  0.1137  -0.1369  -0.2723  -0.3352  -0.3242  -0.1494  -0.1021
## s.e.  0.0615  0.0686   0.0650   0.0637   0.0642   0.0652   0.0698   0.0648
##       intercept
##          -6e-04
## s.e.      9e-04
##
## sigma^2 estimated as 0.002105:  log likelihood = 438.26,  aic = -856.52
```

```
model_2 <- arima(error.ts, order = c(0, 0, 6))
model_2
```

```
##
## Call:
## arima(x = error.ts, order = c(0, 0, 6))
##
## Coefficients:
##          ma1     ma2      ma3      ma4      ma5      ma6  intercept
##       0.4625  0.0485  -0.2004  -0.3228  -0.3510  -0.2731    -0.0007
## s.e.  0.0606  0.0607   0.0623   0.0675   0.0612   0.0572     0.0011
##
## sigma^2 estimated as 0.002144:  log likelihood = 435.91,  aic = -855.81
```

```
tsdiag(model)
```



After comparing the models of ARMA(8, 0) and ARMA(6, 0), I have decided on ARMA(8, 0) because it produces the smallest AIC value between the two models. Furthermore, it's diagnostic plots shows an arbitrary standardized residual plot with no obvious pattern. The ACF of residuals displays insignificant lag values aside from lag 0 which ressembles a White Noise process along with non-significant p-values throughout our Ljung-Box Statistics plot. Hence, I believe our ARMA(8, 0) fitted model is the better fit for the remainder term.

The Fitted ARMA Model $X_t = -0.0006 + 0.4853X_{t-1} + 0.1137X_{t-2} - 0.1369X_{t-3} - 0.2723X_{t-4} - 0.3352X_{t-5} - 0.3242X_{t-6} - 0.1494X_{t-7} - 0.1021X_{t-8} + Z_t$

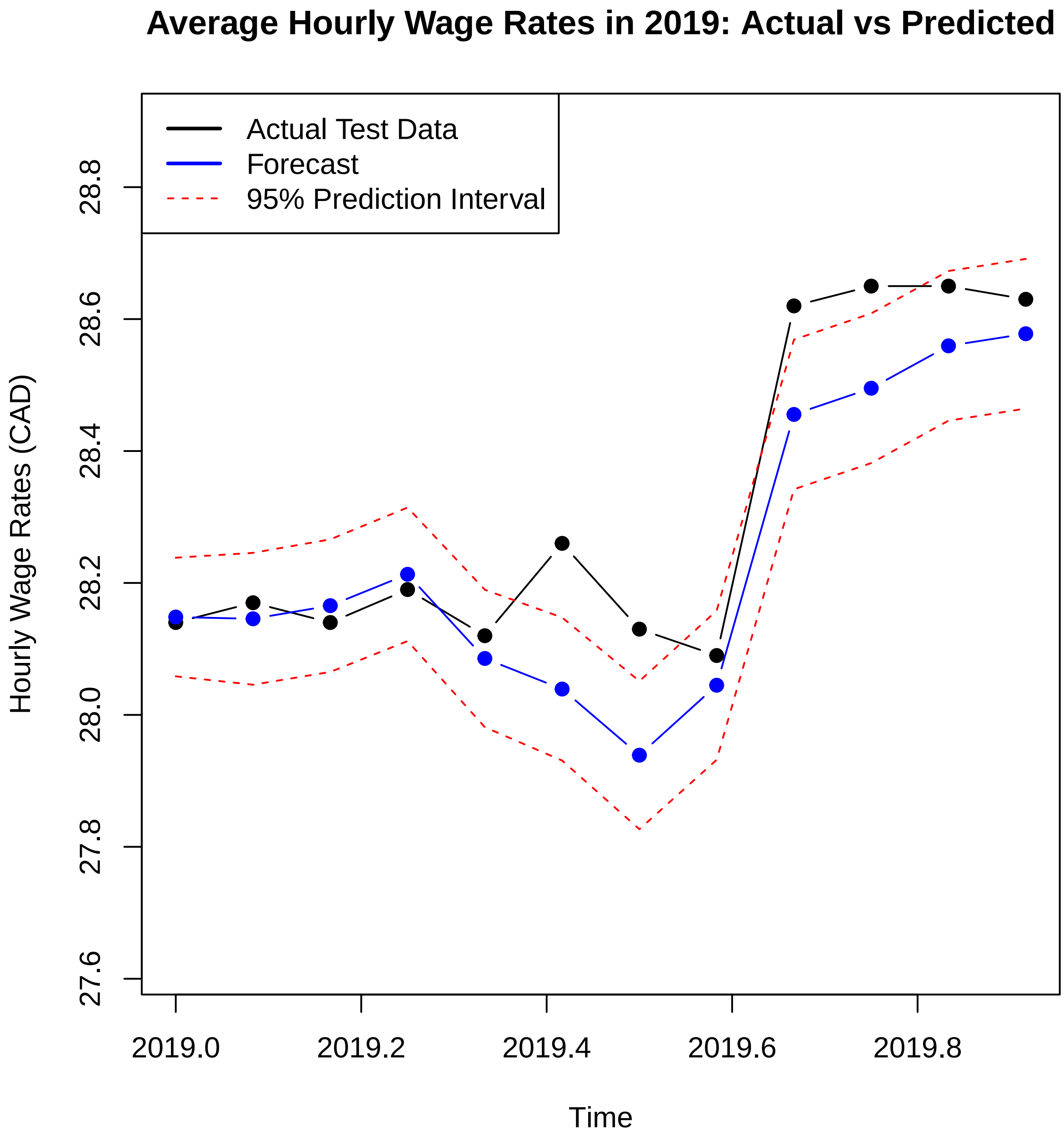
```r
# this is where your R code goes
# Remainder Component
remainder_forecast <- predict(model, n.ahead = 12)
remainder_lower <- remainder_forecast$pred - 1.96 * remainder_forecast$se
remainder_upper <- remainder_forecast$pred + 1.96 * remainder_forecast$se

# Trend Component
trend.time.ls <- seq_along(trend.ls)
trend.ls.df <- data.frame('trend'=trend.ls, 'time'=trend.time.ls)
trend_model.ls <- lm(trend ~ time, data=trend.ls.df)
trend_forecast_times <- max(trend.time.ls) + 1:12
trend_forecast <- predict(trend_model.ls, newdata=data.frame('time'=trend_forecast_times))

# Seasonal Component
seasonal_forecast <- seas.ls[1:12]

final_forecast <- trend_forecast + seasonal_forecast + remainder_forecast$pred

forecast_ts <- ts(final_forecast,
                  start=c(2019, 1),
                  frequency=12)

# Prediction Interval
lower_pi <- trend_forecast + seasonal_forecast + remainder_lower
upper_pi <- trend_forecast + seasonal_forecast + remainder_upper

lower_ts <- ts(lower_pi, start=c(2019, 1), frequency=12)
upper_ts <- ts(upper_pi, start=c(2019, 1), frequency=12)

# Prediction Table (From Lab 9)
forecast_table <- data.frame(Month = month.abb[1:12],
                             True_Value = data.ts.test,
                             Forecast = forecast_ts,
                             Lower_CI = lower_ts,
                             Upper_CI = upper_ts
                             )

kable(forecast_table,
      caption = "Monthly Forecasts of Average Hourly Wage Rates (2019)")
```

Table 1: Monthly Forecasts of Average Hourly Wage Rates (2019)

Month	True_Value	Forecast	Lower_CI	Upper_CI
Jan	28.14	28.14834	28.05843	28.23826
Feb	28.17	28.14567	28.04573	28.24562
Mar	28.14	28.16560	28.06513	28.26606
Apr	28.19	28.21311	28.11189	28.31432
May	28.12	28.08552	27.98139	28.18966
Jun	28.26	28.03917	27.93076	28.14758
Jul	28.13	27.93891	27.82665	28.05117
Aug	28.09	28.04496	27.93190	28.15802
Sep	28.62	28.45544	28.34200	28.56888

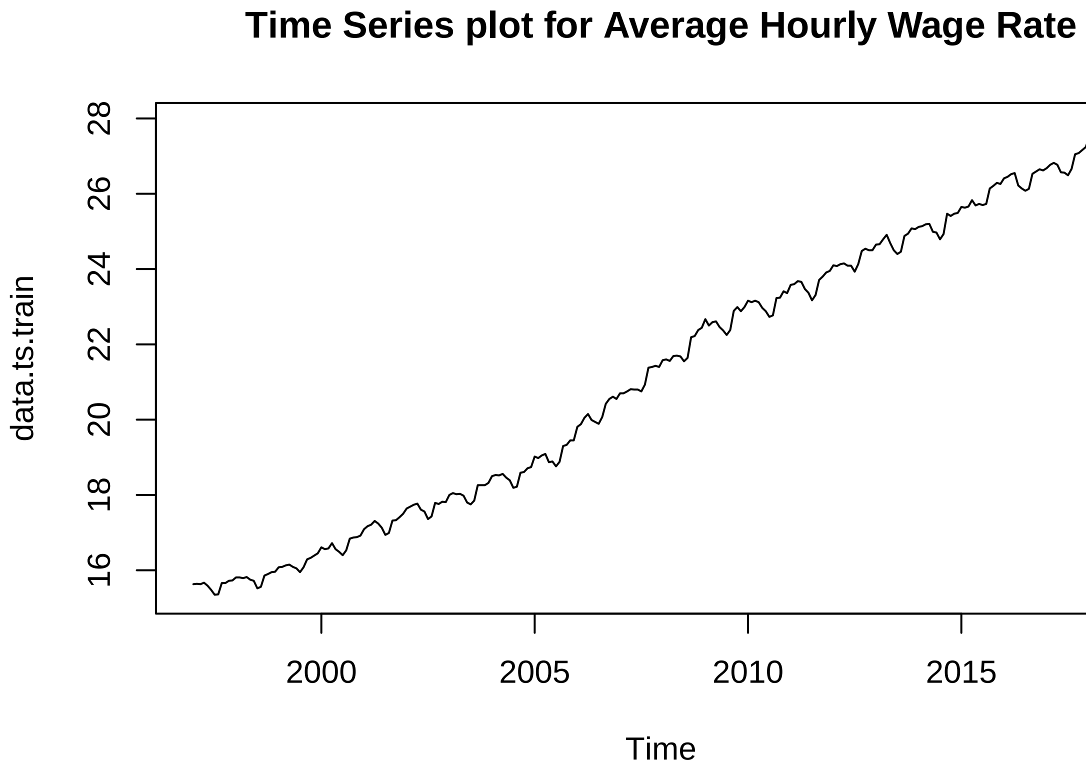
Month	True_Value	Forecast	Lower_CI	Upper_CI
Oct	28.65	28.49517	28.38173	28.60861
Nov	28.65	28.55944	28.44601	28.67288
Dec	28.63	28.57778	28.46434	28.69122

```r
# this is where your R code goes
# From Lab 8 Prediction plot
plot(data.ts.test,
    main = "Average Hourly Wage Rates in 2019: Actual vs Predicted",
    ylab = "Hourly Wage Rates (CAD)",
    ylim = c(min(lower_pi)-0.2, max(upper_pi)+0.2),
    col = "black",
    type = "b",
    pch = 19)

# Prediction
lines(forecast_ts,
      col = "blue",
      type = "b",
      pch = 19)

# Prediction interval
lines(lower_ts,
      type = "l",
      col = "red",
      lty = 2)

lines(upper_ts,
      type = "l",
      col = "red",
      lty = 2)

legend("topleft",
        legend=c("Actual Test Data", "Forecast", "95% Prediction Interval"),
        col=c("black", "blue", "red"),
        lty=c(1, 1, 2),
        lwd=c(2, 2, 1))
```



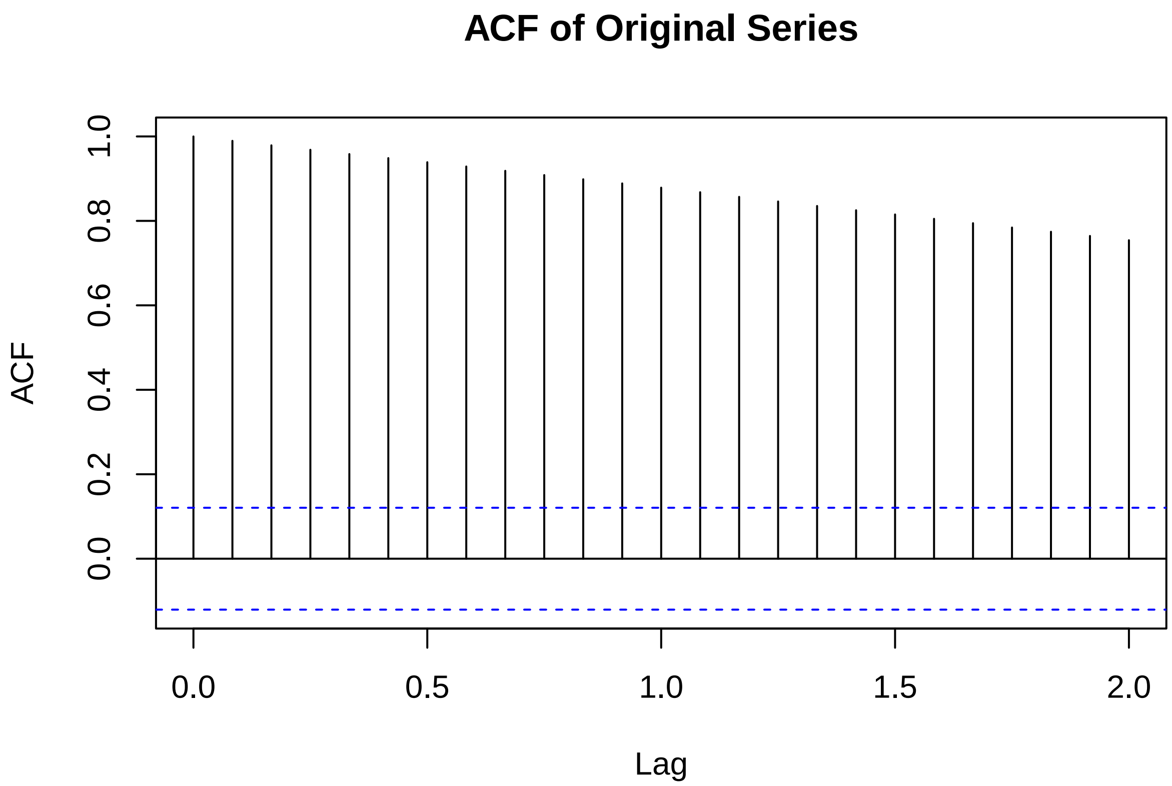
Question 2 Box-Jenkins

```
# this is where your R code goes
plot(data.ts.train, main = "Time Series plot for Average Hourly Wage Rate")
```



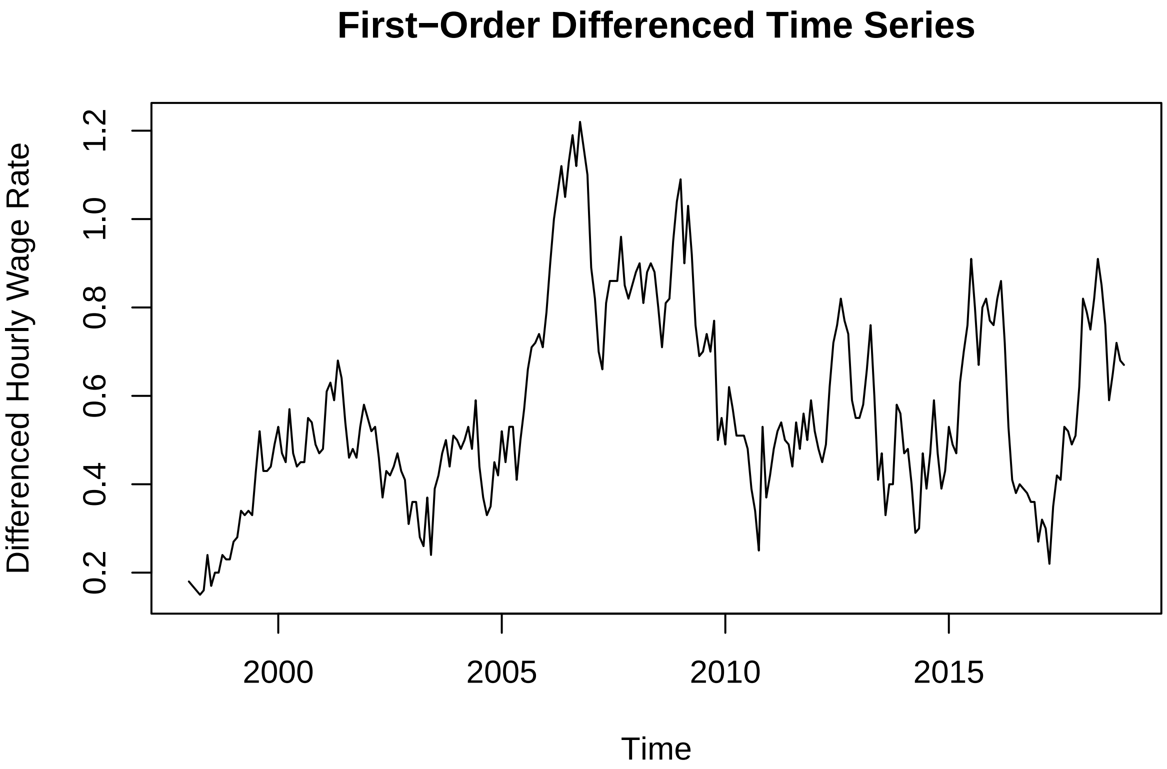
Based on our time series plot, we can see a strong positive slope / trend, hence the mean is changing over time and we may consider it as non-stationary. Hence, I will be removing its trend and seasonal components to make the series stationary.

```
# this is where your R code goes
acf(data.ts.train, main="ACF of Original Series")
```

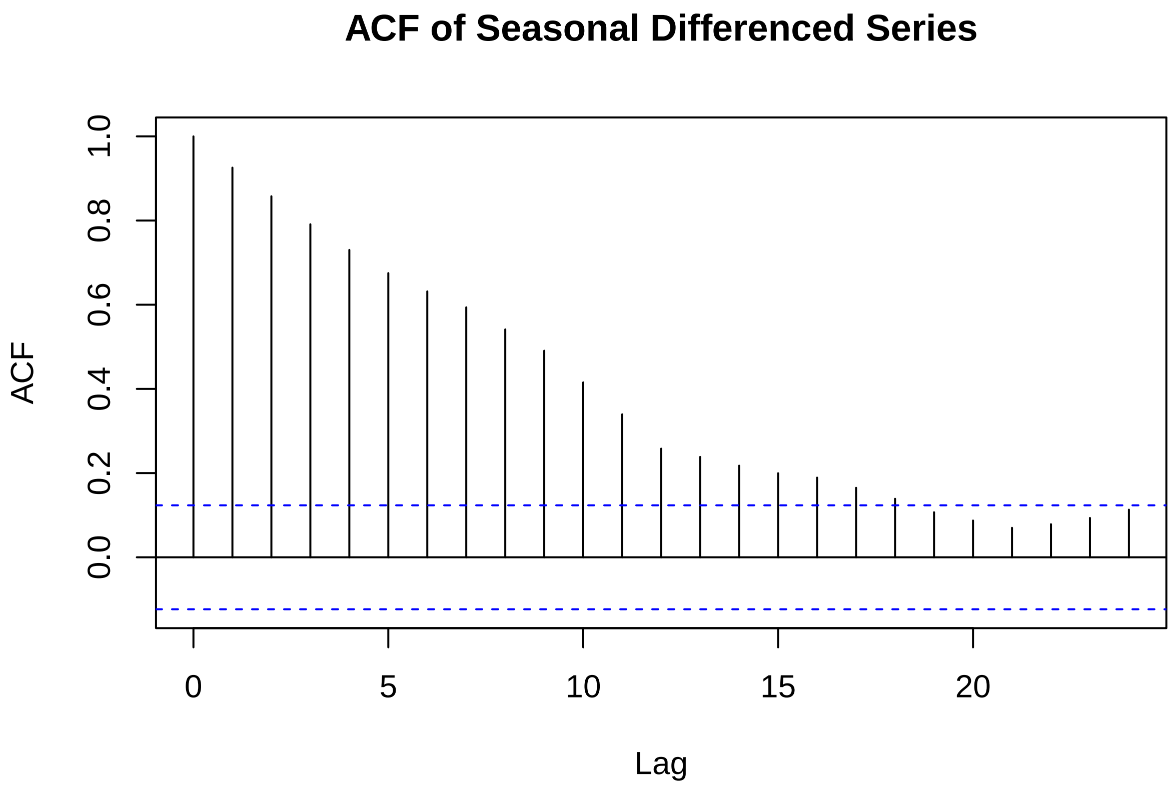


Now, we will be removing the linear trend by using the diff function with difference = 1 and remove its seasonality by differencing over 1 full season (12 months)

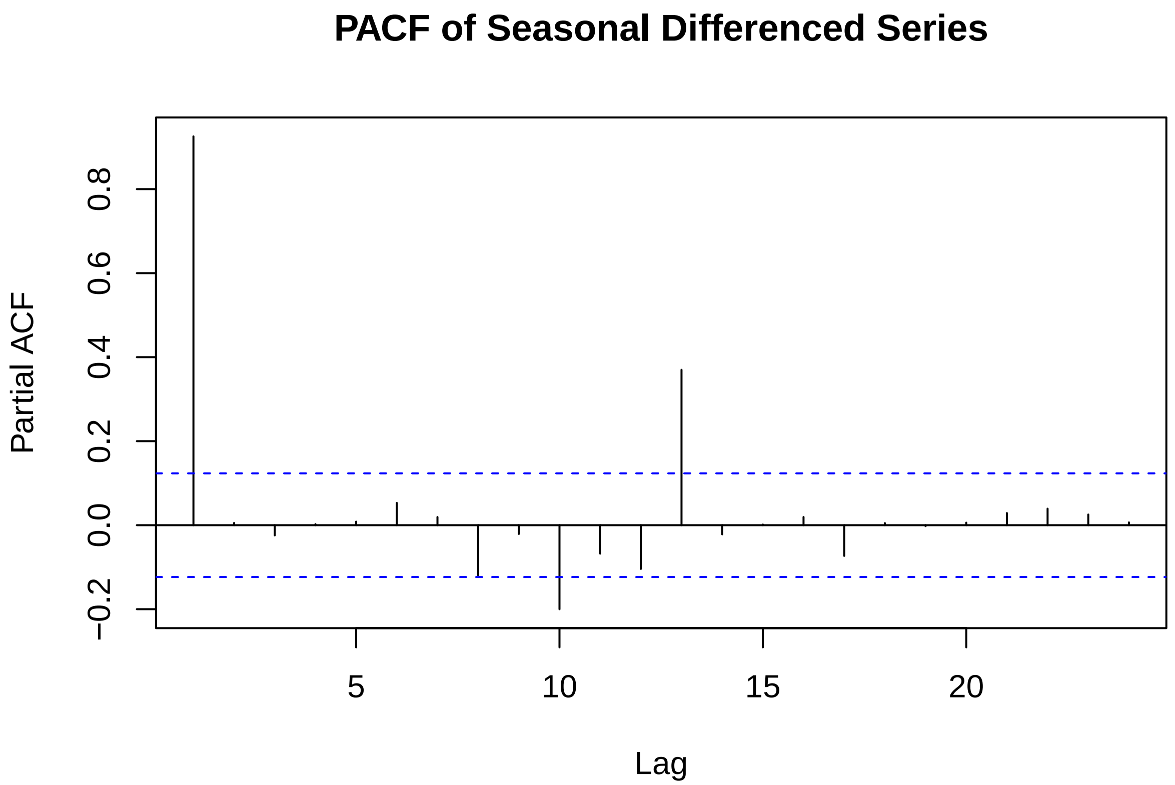
```
# FROM LAB 4 & Chapter 3
# Applying Seasonal Differencing
diff_1 <- diff(data.ts.train, lag = frequency(data.ts.train))

plot(diff_1,
     main="First-Order Differenced Time Series",
     xlab = "Time", ylab = "Differenced Hourly Wage Rate")
```



```r
acf(ts(diff_1), main = "ACF of Seasonal Differenced Series")
```

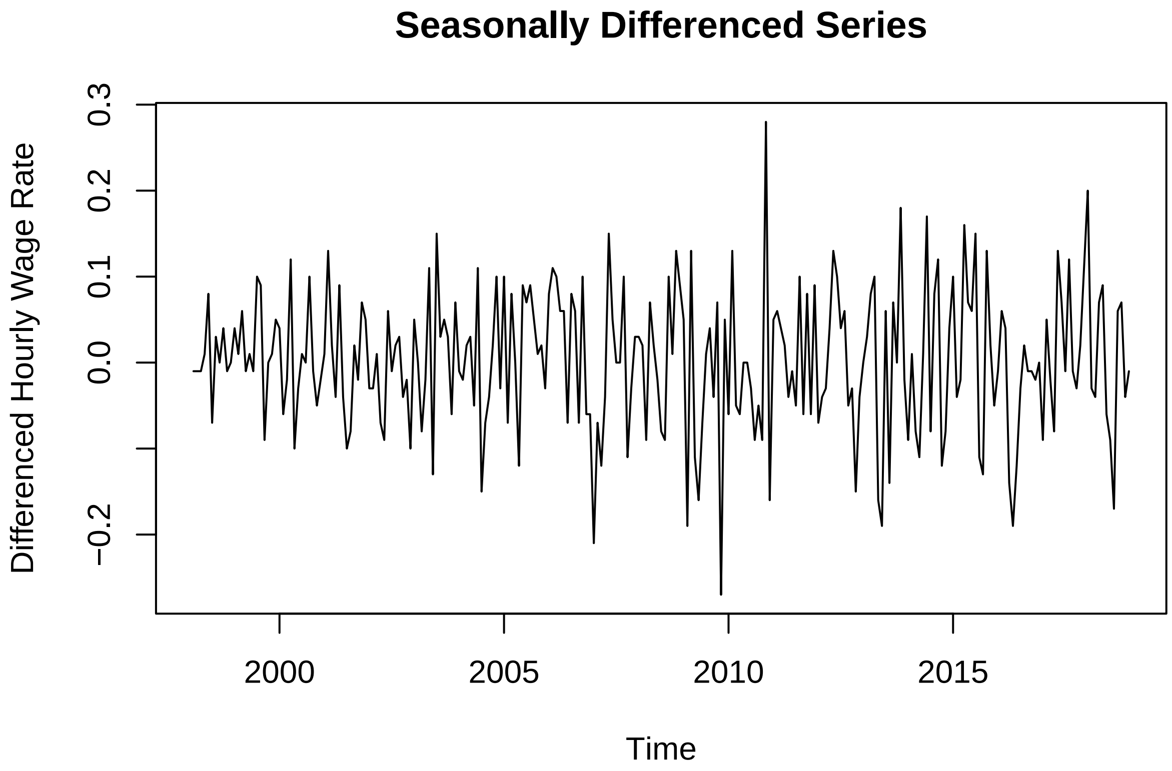


```
pacf(ts(diff_1), main = "PACF of Seasonal Differenced Series")
```

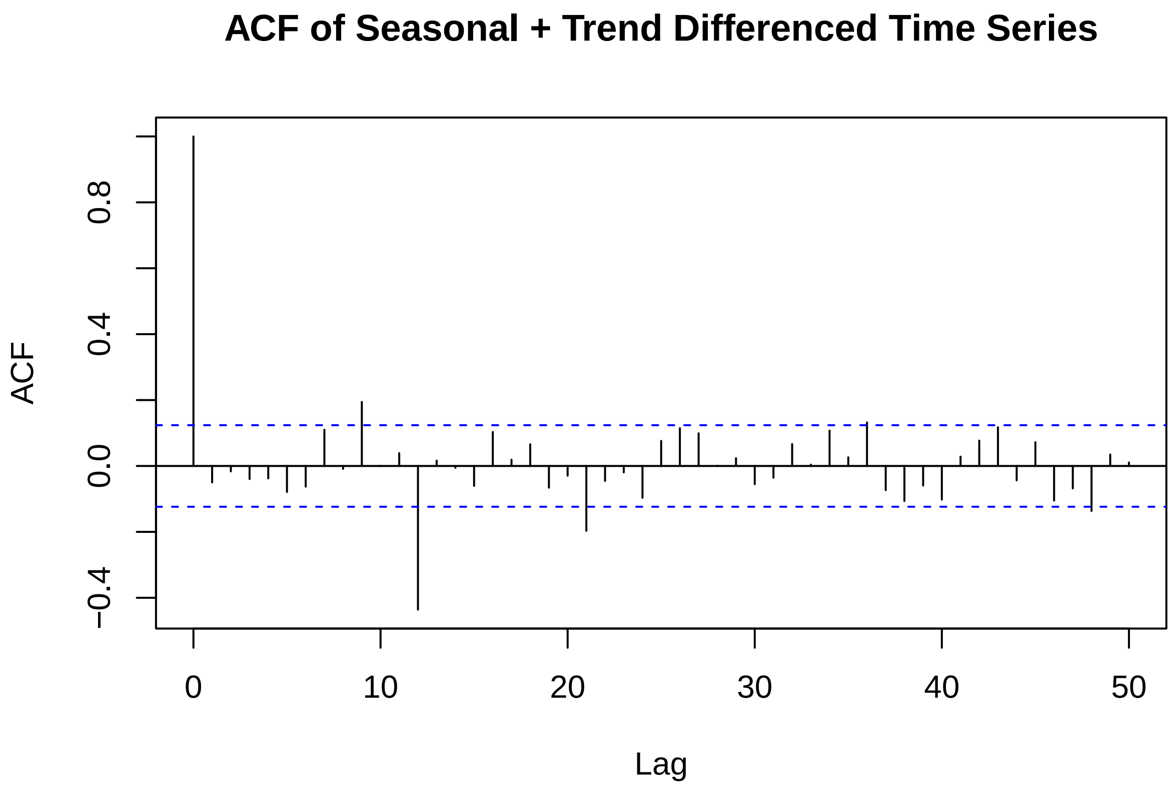


Applying seasonal differencing, we can observe a white noise resemblance in our pacf and acf plots. Hence, we can say that our p and q will be 0.

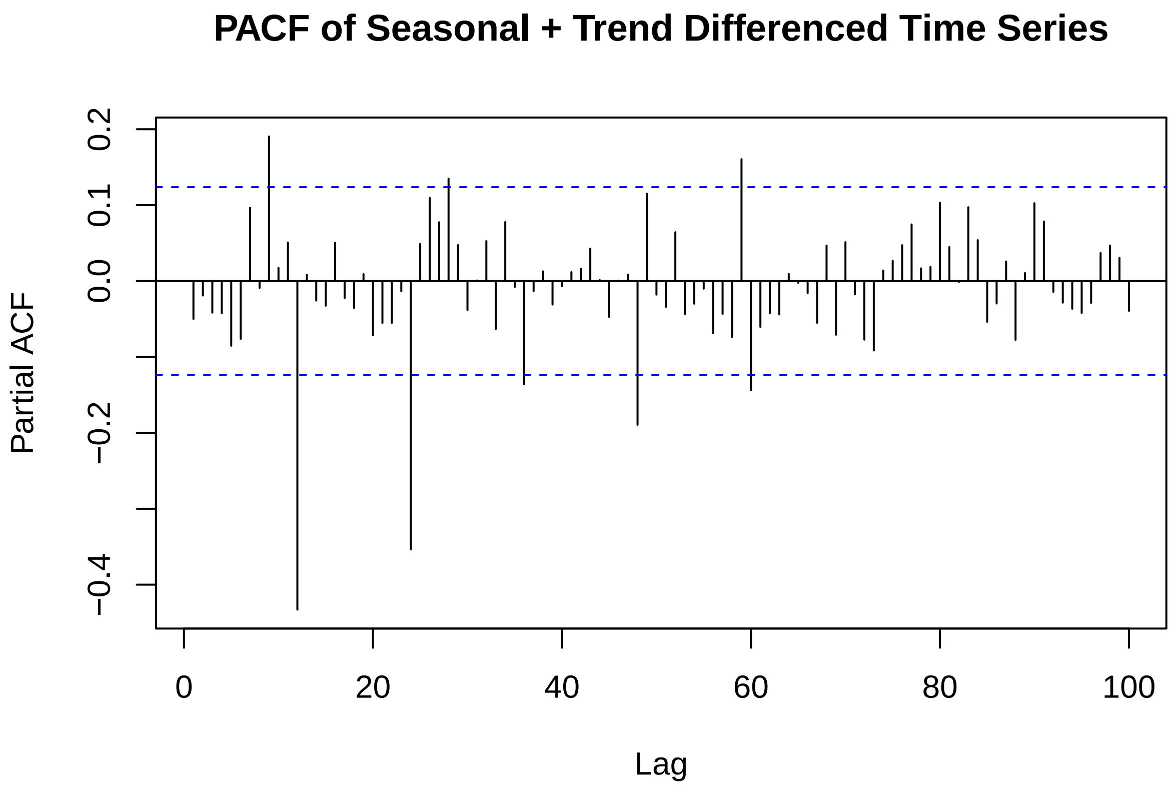
```
# FROM LAB 4 & Chapter 3

# Seasonality Differencing
seasonal_diff <- diff(diff_1, differences = 1)

plot(seasonal_diff,
     main = "Seasonally Differenced Series",
     xlab = "Time", ylab = "Differenced Hourly Wage Rate"
     )
```



```
acf(ts(seasonal_diff), lag.max = 50,
    main="ACF of Seasonal + Trend Differenced Time Series")
```



```
pacf(ts(seasonal_diff), lag.max = 100,
     main="PACF of Seasonal + Trend Differenced Time Series")
```



After our seasonal and trend differencing, we observe a cut off at 12 in our ACF plot while the PACF plot tails off like a 'damped' wave. Hence, we can determine that it is a pure MA(12) process. Since the values is a multiple of the seasonal period 12, so we should get Q = 12/12 = 1.

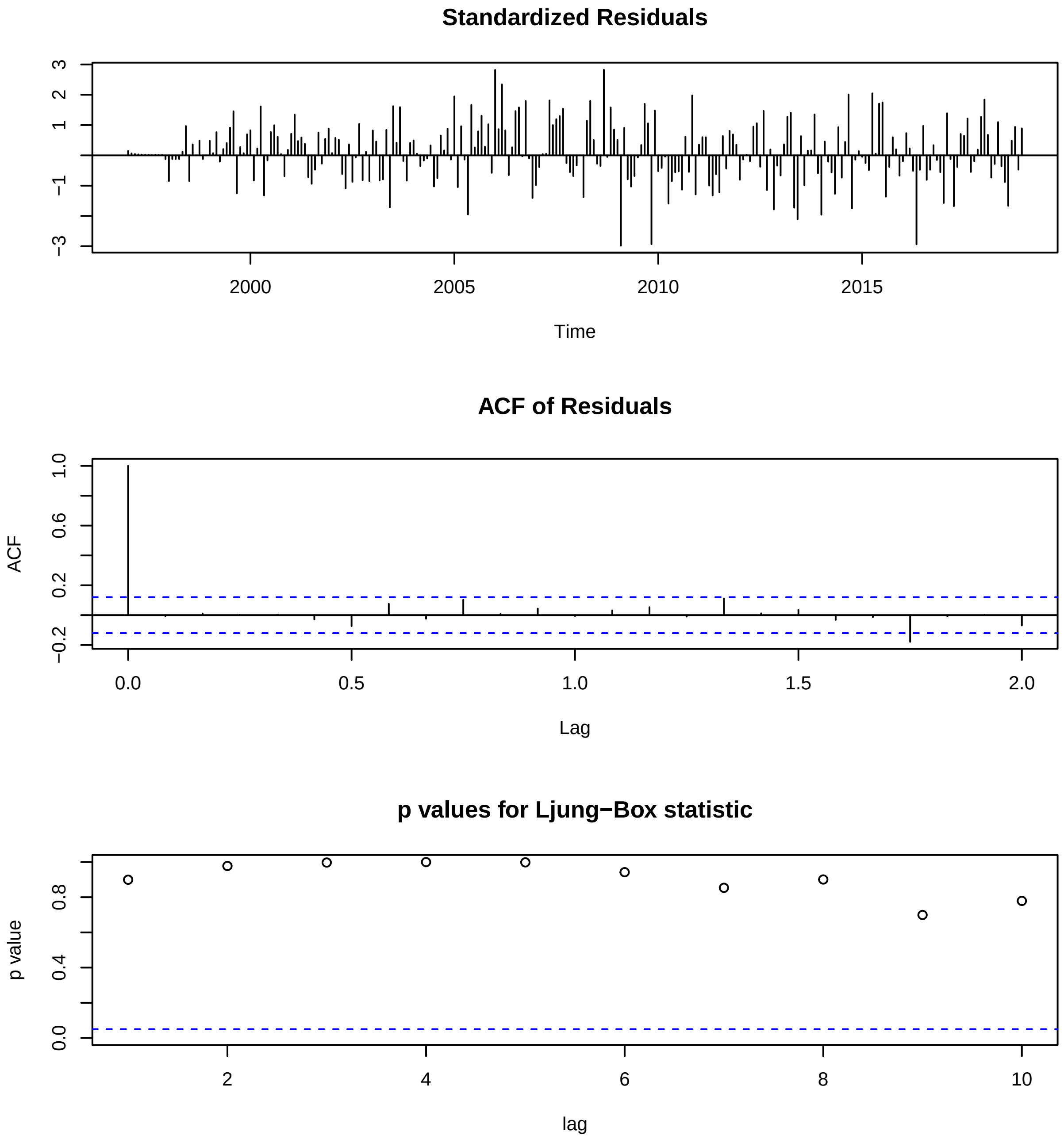
Therefore, our ARIMA model would be. ARIMA(0, 1, 0)(0, 1, 1)_12

For Sanity check, function auto.arima() is used to check if our determined model is the best ARIMA model based off it's AIC value.

```r
# this is where your R code goes
arima_selection <- auto.arima(data.ts.train, seasonal = TRUE)
summary(arima_selection)
```

```
## Series: data.ts.train
## 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
##
## Training set error measures:
##                        ME       RMSE        MAE        MPE      MAPE       MASE
## Training set 0.005726408 0.06206067 0.04806889 0.03164616 0.2255326 0.08427272
##                          ACF1
## Training set -0.007736081
```

```r
tsdiag(arima_selection)
```



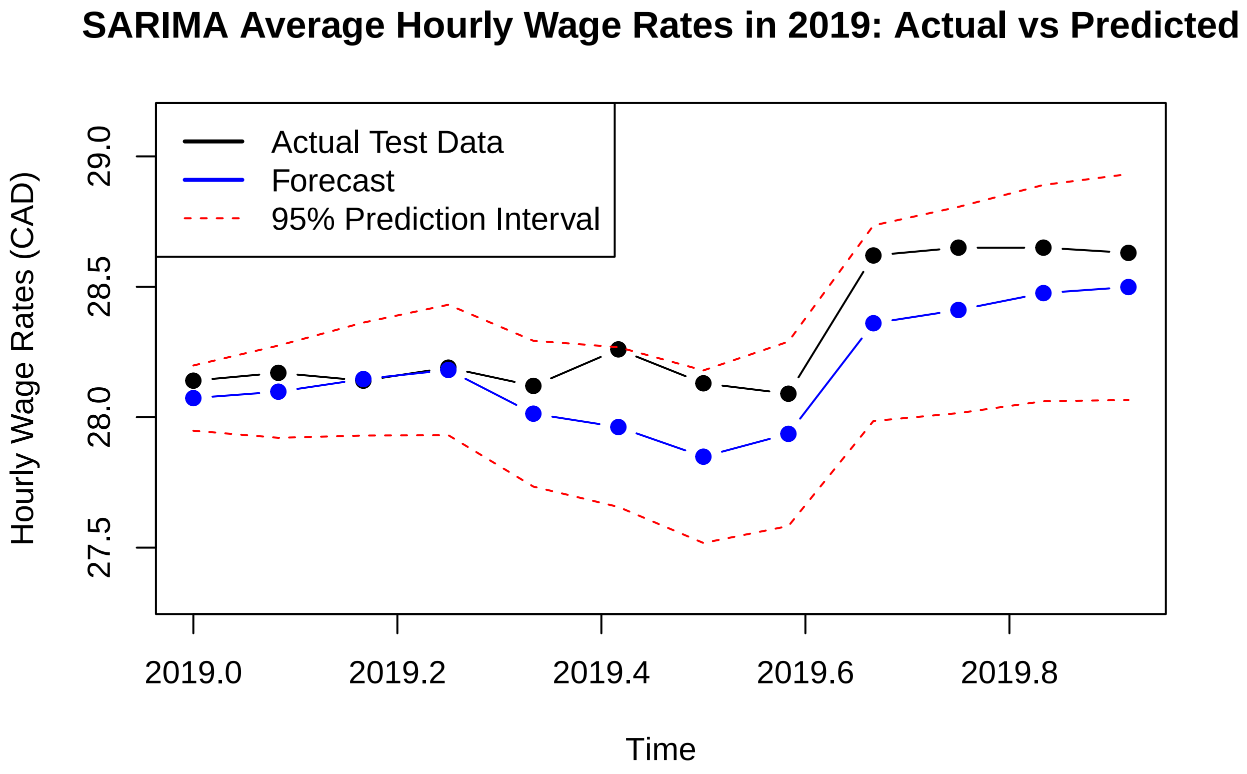
Standardized Residuals

ACF of Residuals

p values for Ljung−Box statistic

I believe that $ SARIMA(0,1,0)*(0,1,1)_{12} $ would fit the data best based on its small AIC value. Furthermore, the diagnostic plot suggests a good fit because the standardized residuals has no pattern, the acf of the residuals is a white noise process, and the p-values for Ljung-Box are insignificant.

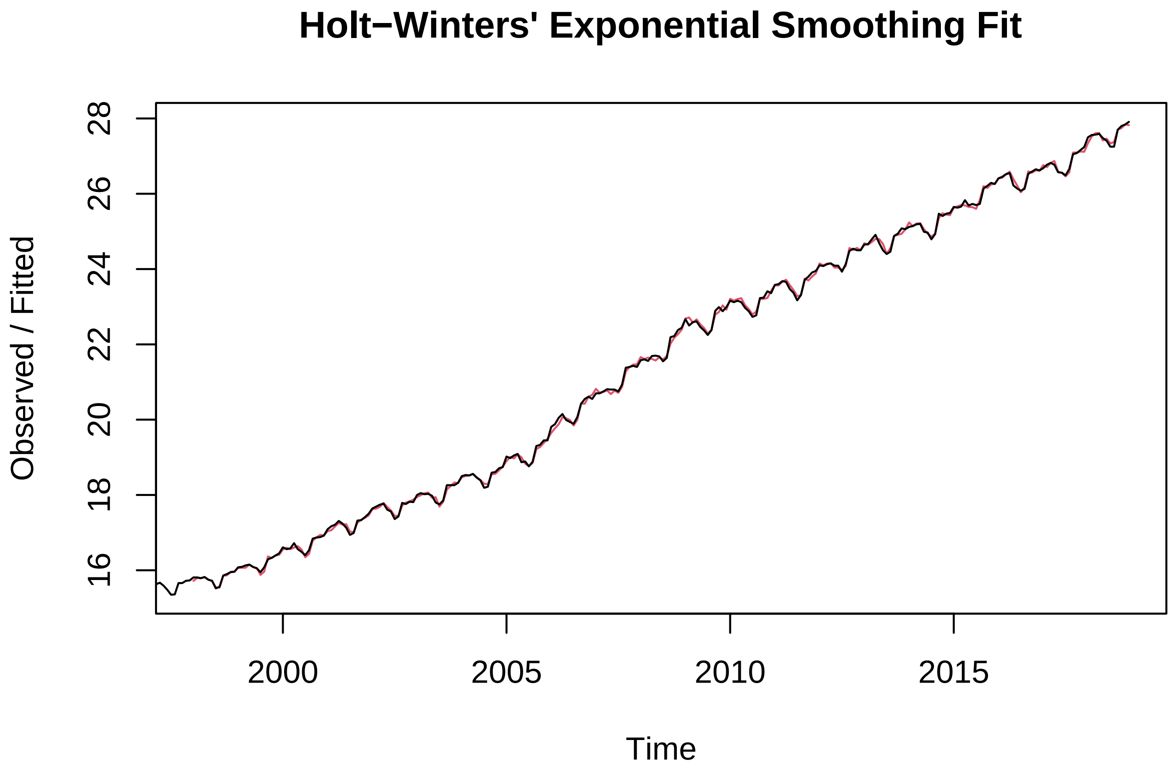
```r
# this is where your R code goes
# Forecasting
sarima_forecast <- predict(arima_selection, n.ahead=12)

sarima_pred <- sarima_forecast$pred
sarima_forecast_lower <- sarima_pred - 1.96 * sarima_forecast$se
sarima_forecast_upper <- sarima_pred + 1.96 * sarima_forecast$se

sarima_forecast_ts <- ts(sarima_pred, start=c(2019, 1), frequency=12)
sarima_lower_ts <- ts(sarima_forecast_lower, start=c(2019, 1), frequency=12)
sarima_upper_ts <- ts(sarima_forecast_upper, start=c(2019, 1), frequency=12)


# Prediction Table (From Lab 9)
forecast_table <- data.frame(Month = month.abb[1:12],
                             True_Value = data.ts.test,
                             Forecast = sarima_forecast_ts,
                             Lower_CI = sarima_lower_ts,
                             Upper_CI = sarima_upper_ts
                             )


# Graph

plot(data.ts.test,
     main = "SARIMA Average Hourly Wage Rates in 2019: Actual vs Predicted",
     ylab = "Hourly Wage Rates (CAD)",
     ylim = c(min(sarima_lower_ts) - 0.2, max(sarima_upper_ts) + 0.2),
     col = "black",
     type = "b",
     pch = 19)

lines(sarima_forecast_ts,
      col="blue",
      type = "b",
      pch = 19)

lines(sarima_lower_ts,
      col="red",
      lty = 2)

lines(sarima_upper_ts,
      col="red",
      lty = 2)

legend("topleft",
        legend=c("Actual Test Data", "Forecast", "95% Prediction Interval"),
        col=c("black", "blue", "red"),
        lty=c(1, 1, 2),
        lwd=c(2, 2, 1))
```



Question 3 Holt-Winters

Looking back into our time series plot, we can observe a roughly constant seasonal variation throughout the series. Hence, I will be using an additive model for Holt-Winters' Exponential Smoothing Fit.

```r
# From Lab 8
hw_model <- HoltWinters(data.ts.train, seasonal = "additive")

plot(hw_model, main="Holt-Winters' Exponential Smoothing Fit")
```

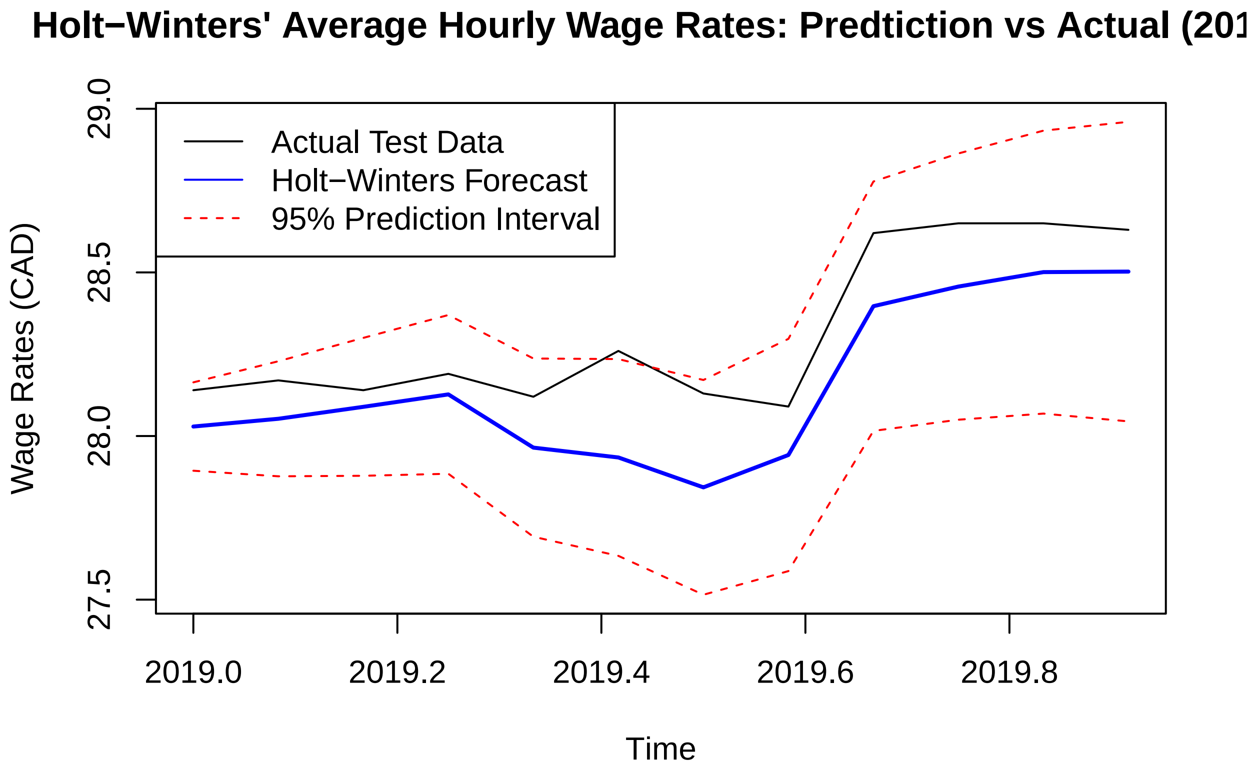


```
hw_predictions <- predict(hw_model,
                          n.ahead = 12,
                          prediction.interval = TRUE)

hw_fit <- hw_predictions[, "fit"]
hw_lower <- hw_predictions[, "lwr"]
hw_upper <- hw_predictions[, "upr"]

hw_forecast_ts <- ts(hw_fit, start = c(2019, 1), frequency = 12)

hw_lower_ts <- ts(hw_lower, start = c(2019, 1), frequency = 12)
hw_upper_ts <- ts(hw_upper, start = c(2019, 1), frequency = 12)

# Prediction Table (From Lab 9)
hw_forecast_table <- data.frame(Month = month.abb[1:12],
                             True_Value = data.ts.test,
                             Forecast = hw_forecast_ts,
                             Lower_CI = hw_lower_ts,
                             Upper_CI = hw_upper_ts
                             )

kable(forecast_table,
      caption = "Monthly Forecasts of Average Hourly Wage Rates (2019)")
```

Table 2: Monthly Forecasts of Average Hourly Wage Rates (2019)

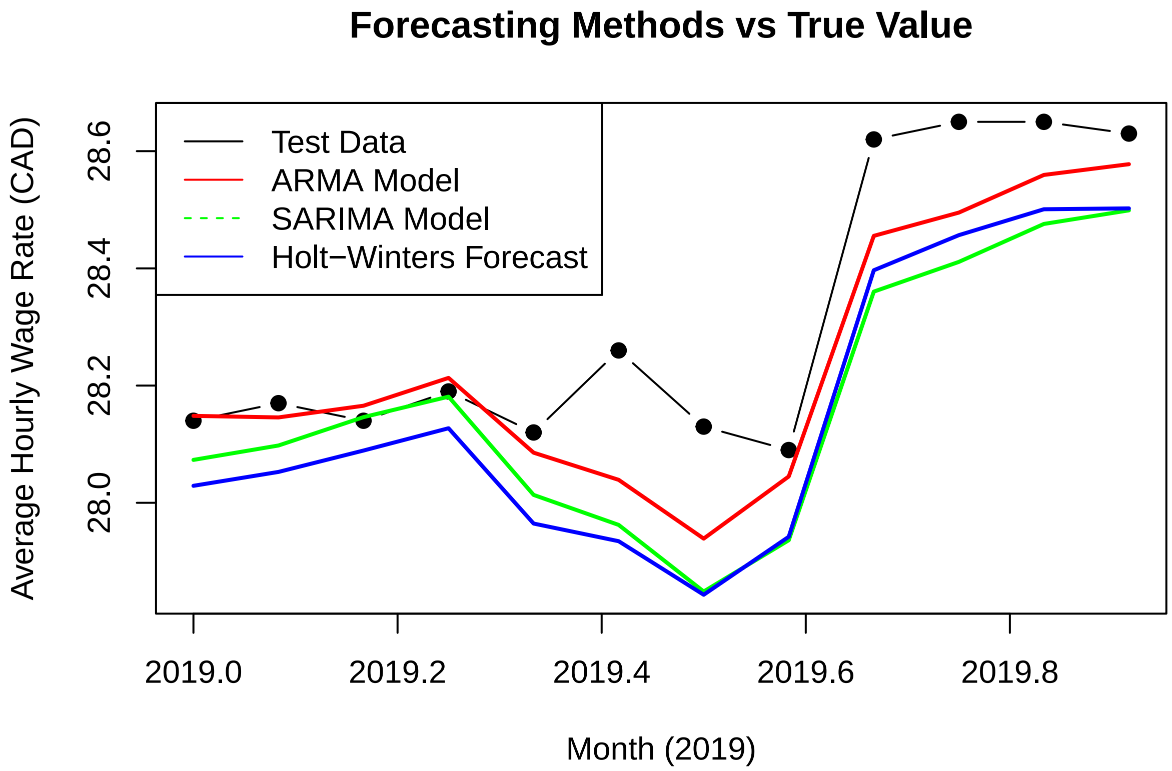
Month	True_Value	Forecast	Lower_CI	Upper_CI
Jan	28.14	28.07322	27.94821	28.19822
Feb	28.17	28.09777	27.92098	28.27455
Mar	28.14	28.14620	27.92969	28.36271
Apr	28.19	28.18110	27.93110	28.43111
May	28.12	28.01346	27.73394	28.29297
Jun	28.26	27.96230	27.65610	28.26849
Jul	28.13	27.84863	27.51790	28.17935
Aug	28.09	27.93615	27.58258	28.28971
Sep	28.62	28.36033	27.98532	28.73533
Oct	28.65	28.41103	28.01574	28.80632
Nov	28.65	28.47580	28.06121	28.89038
Dec	28.63	28.49898	28.06596	28.93201

```r
plot(data.ts.test,
     main="Holt-Winters' Average Hourly Wage Rates: Predtiction vs Actual (2019)",
     ylim=range(c(data.ts.test, hw_lower, hw_upper)),
     ylab="Wage Rates (CAD)",
     xlab="Time",
     col="black")

lines(hw_forecast_ts, col="blue", lwd=2)
lines(hw_lower_ts, col="red", lty=2)
lines(hw_upper_ts, col="red", lty=2)

legend("topleft",
       legend=c("Actual Test Data", "Holt-Winters Forecast", "95% Prediction Interval"),
       col=c("black", "blue", "red"),
       lty=c(1, 1, 2))
```



Question 4 Model Comparison

```
# GRAPH
plot(data.ts.test,
     col = "black",
     pch = 19,
     type = "b",
     xlab="Month (2019)",
     ylab="Average Hourly Wage Rate (CAD)",
     main="Forecasting Methods vs True Value",
     ylim=range(c(data.ts.test, forecast_ts, sarima_forecast_ts, hw_forecast_ts))
     )

lines(forecast_ts, col="red", lwd=2)
lines(sarima_forecast_ts, col="green", lwd=2)
lines(hw_forecast_ts, col="blue", lwd=2)

legend("topleft",
       legend=c("Test Data", "ARMA Model", "SARIMA Model", "Holt-Winters Forecast"),
       col=c("black", "red", "green", "blue"),
       lty=c(1, 1, 2))
```



```r
# Table
arma_mspe <- mean((data.ts.test - forecast_ts)^2)
sarima_mspe <- mean((data.ts.test - sarima_forecast_ts)^2)
hw_mspe <- mean((data.ts.test - hw_forecast_ts)^2)

table_compare <- data.frame(Model = c("ARMA", "SARIMA", "Holt-Winters"),
                            MSPE = c(arma_mspe, sarima_mspe, hw_mspe))
kable(table_compare,
      caption = "Forecast Models Comparison w/ Mean Squared Prediction Error")
```

Table 3: Forecast Models Comparison w/ Mean Squared Prediction Error

Model	MSPE
ARMA	0.0126940
SARIMA	0.0320554
Holt-Winters	0.0327192

Based on our gathered forecasts, the true values for are mostly within our prediction intervals but we get ARMA (seasonal decomposition) as the best model amongst them all because it has the smallest MSPE.

Pros & Cons Method 1) We get better Flexibility by isolating the remainder term because we have an ARMA model that is specifically tailored to the stochastic part of our training data, however, we need to assume stationarity in order to use ARMA, hence, the remainder term must be stationary which may not always work for other data sets.

Method 2) In contrast to Method 1), SARIMA models can handle non-stationary data by differencing, making them more flexible for non-stationary data sets. In spite of that, Method 1 & 2) could be subjective in terms of picking the 'best' (p, d, q, P, D, Q) and s.

Method 3) Using Holt-Winters' forecasting, we can extend the series with a trend and seasonal variation components and it is relatively shorter in terms of computation compared to Method 1 and 2. Similar to our ARMA model, it may not be flexible for non-linear series.