Stat 443: Time Series and Forecasting

Assignment 3: Time Series Models

Caden Hewlett

March 19, 2024

Question 1: El Niño Forecasting

The file `NINO34.csv` contains the monthly El Niño 3.4 index from 1870 to 2023. The El Niño 3.4 index represents the average equatorial sea surface temperature (in degrees Celsius) from around the international dateline to the coast of South America.

Part a

Perform exploratory data analysis.

Part a.1

Import the data into R and create a time-series object for the El Niño 3.4 index.

```r
# import data
# setwd('C:/Users/caden/OneDrive/Desktop/STAT_443/STAT443')
df <- read.csv("NINO34.csv")

# raw data is in a bad format, need to pivot it without years
dflong <- df %>%
  pivot_longer(cols = -Year, names_to = "Month", values_to = "Value")
# re-translate into dates with month abbreviations
dflong <- dflong %>%
  mutate(Date = make_date(Year, match(Month, month.abb), 1))
# then, format into a time series!
nino_ts <- ts(dflong$Value,
              start = c(1870, 1), frequency = 12)

# time series length = data frame length = 154 years * 12 months per year
stopifnot(all.equal(length(nino_ts), nrow(df)*12, 154*12))
```

Break the time series object into a training and test set. You can use the function `window()` on a ts object to split the data. Let the training set be from January 1870 to December 2021, and let the test set start in January 2022 and end in November 2023.

```r
# split data into train and test
nino_train = window(nino_ts, start = c(1870, 1), end = c(2021, 12))
nino_test = window(nino_ts, start = c(2022, 1), end = c(2023, 11))
# verify split
stopifnot(all.equal(
          length(nino_test)+length(nino_train), length(na.remove(nino_ts))
```

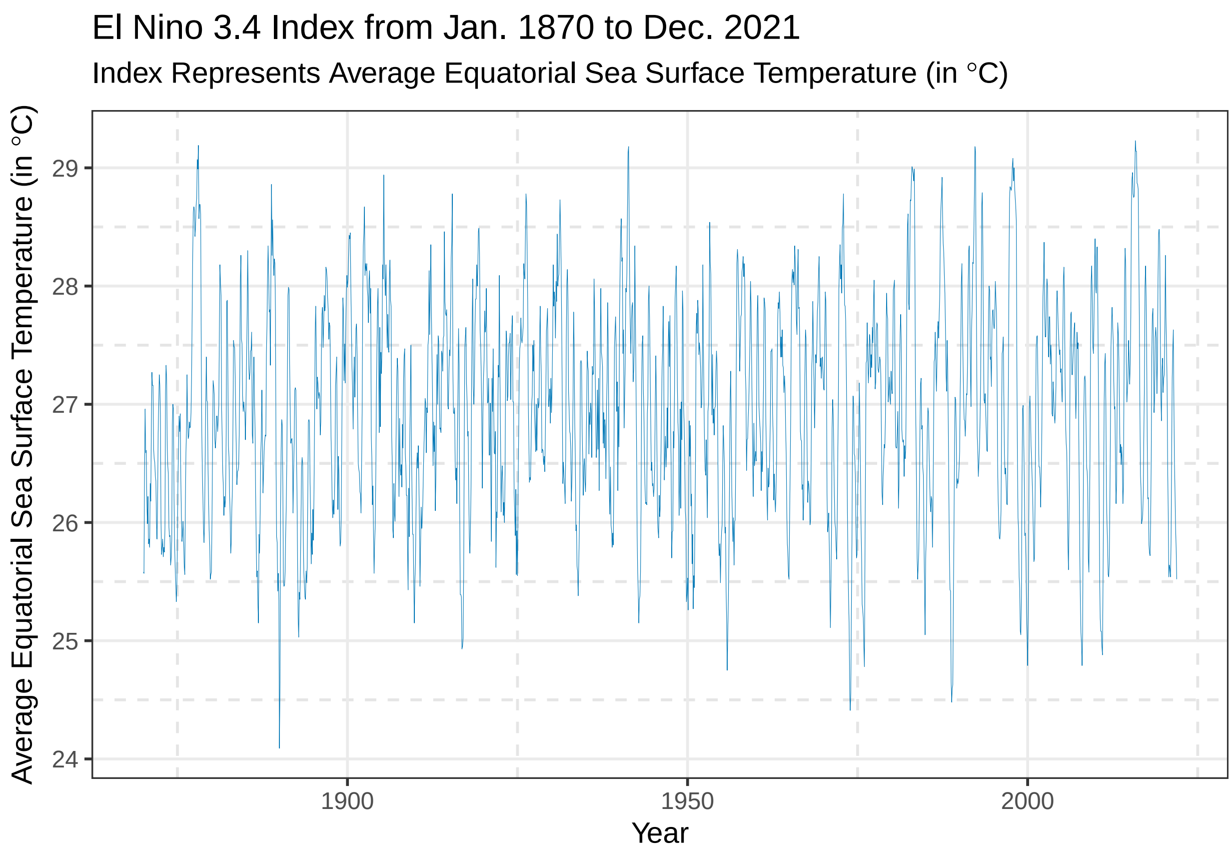
```
    )
)
```

Part a.2

Plot the training data as well as its acf and pacf.

Training Data

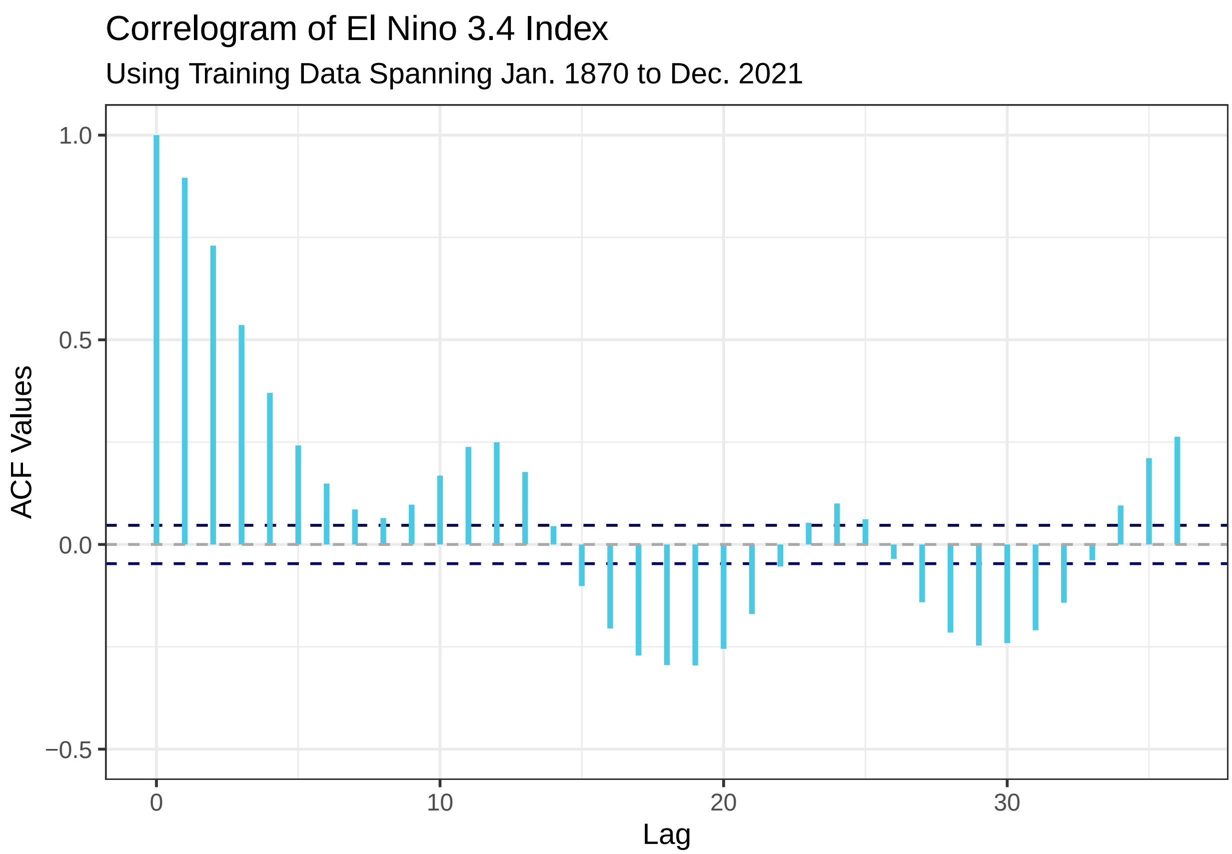
```
p1data = fortify.zoo(nino_train)
p1 <- ggplot(p1data, aes(x = Index, y = nino_train)) +
  geom_line(color = "#0077b6", linewidth = 0.1) +
  labs(
    title = "El Nino 3.4 Index from Jan. 1870 to Dec. 2021",
    subtitle = expression(
      paste("Index Represents Average Equatorial Sea Surface Temperature (in ",
          degree, "C)")),
    y = expression(
      paste("Average Equatorial Sea Surface Temperature (in ", degree, "C)")),
    x = "Year"
  ) + theme_bw() +
  theme(panel.grid.minor = element_line(
    color = "grey90",
    linetype = "dashed",
    linewidth = 0.5
  ))
print(p1)
```



Autocorrelation Function

```r
p2data = data.frame(
  h = 0:36,
  rh = acf(nino_train, plot = FALSE, lag.max = 36)$acf
)
n = length(nino_train)
p2 <- ggplot(p2data, aes(x = h, y = rh)) +
  geom_hline(yintercept = 2/sqrt(n),
             linetype = "dashed",
             col = "#03045e") +
  geom_hline(yintercept = -2/sqrt(n),
             linetype = "dashed",
             col = "#03045e") +
  ylim(-0.5, 1) +
  geom_segment(aes(xend = h, yend = 0),
               color = "#48cae4",
               linewidth = 1) +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "darkgray") +
  labs(x = "Lag",  y = "ACF Values",
       title = "Correlogram of El Nino 3.4 Index",
       subtitle = "Using Training Data Spanning Jan. 1870 to Dec. 2021") +

  theme_bw()
print(p2)
```

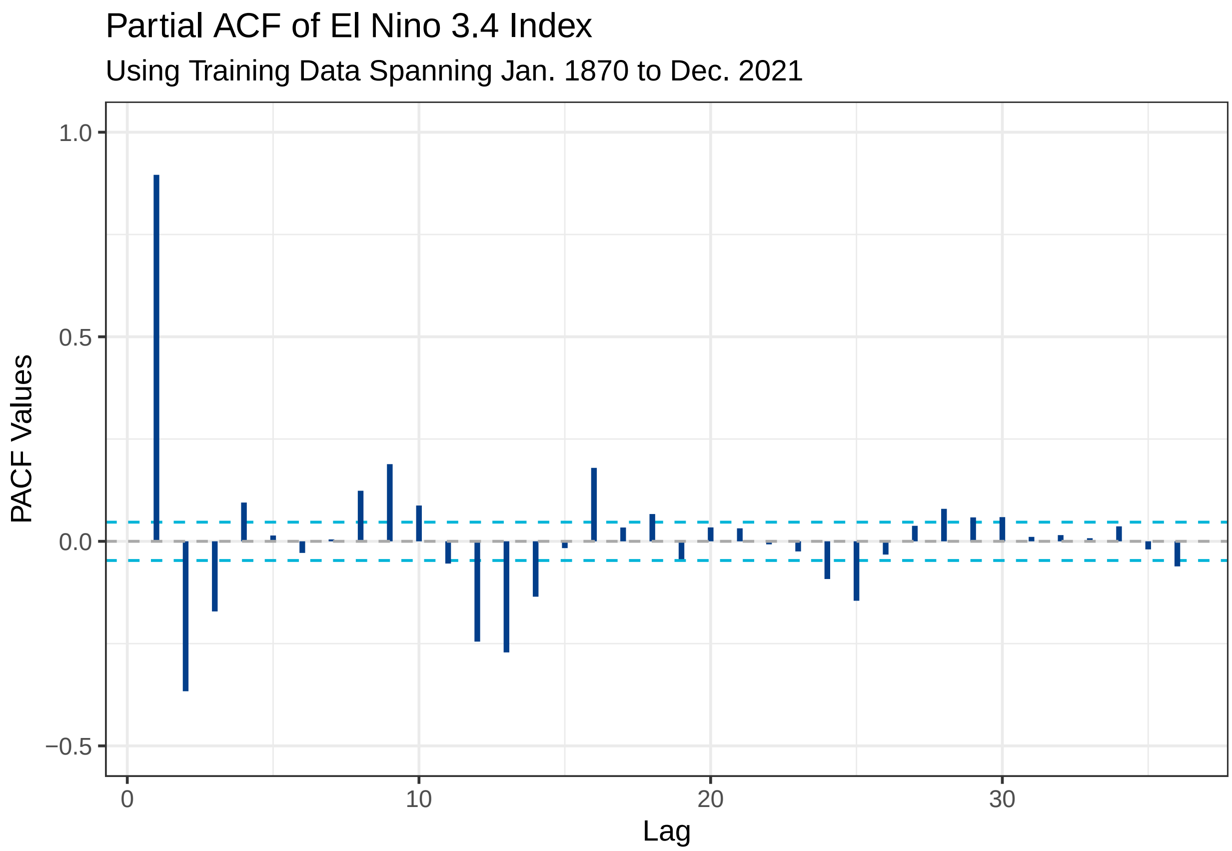


Partial Autocorrelation Function

```
p3data = data.frame(
  h = 1:36,
  rhh = pacf(nino_train, plot = FALSE, lag.max = 36)$acf
)

p3 <- ggplot(p3data, aes(x = h, y = rhh)) +
  geom_hline(yintercept = 2/sqrt(n),
             linetype = "dashed",
             col = "#00b4d8") +
  geom_hline(yintercept = -2/sqrt(n),
             linetype = "dashed",
             col = "#00b4d8") +
  ylim(-0.5, 1) +
  geom_segment(aes(xend = h, yend = 0),
               color = "#023e8a",
               linewidth = 1) +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "darkgray") +
  labs(x = "Lag",  y = "PACF Values",
       title = "Partial ACF of El Nino 3.4 Index",
       subtitle = "Using Training Data Spanning Jan. 1870 to Dec. 2021") +

  theme_bw()
print(p3)
```

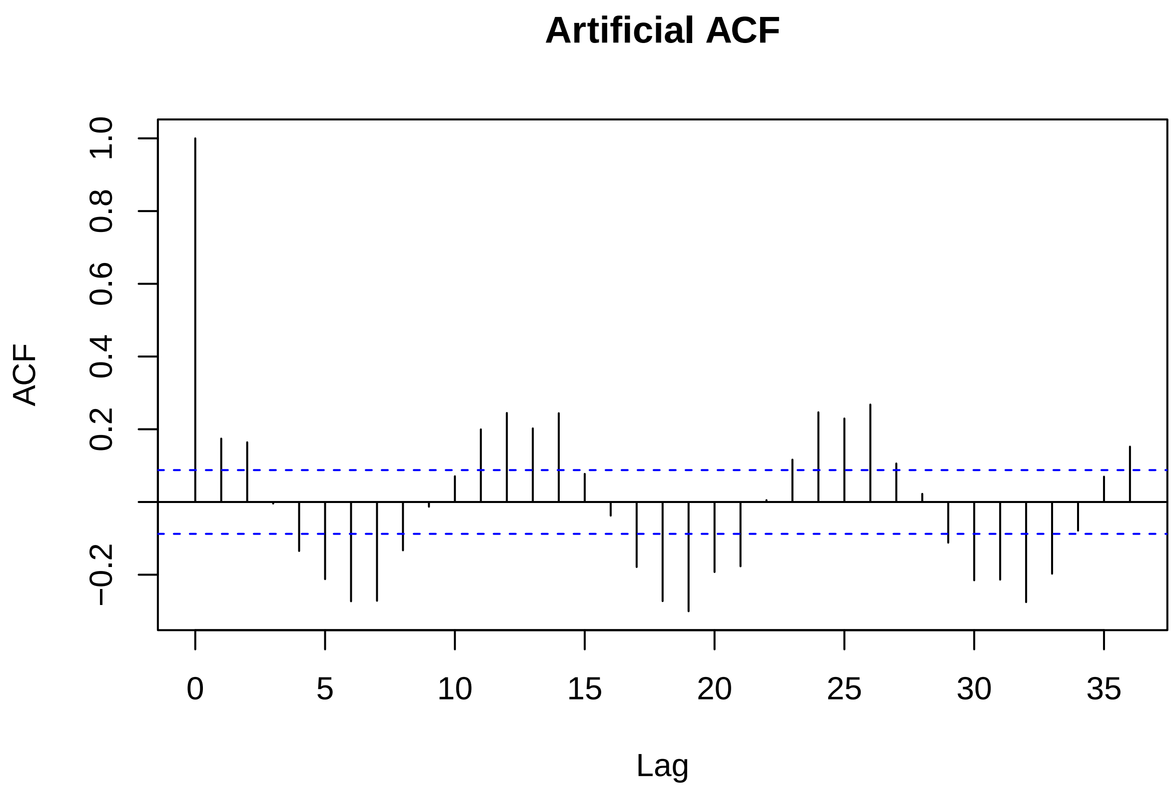


Comments

From the ACF, it is very likely that the original series exhibits a trend. This is due to the fact that there is a clear sinusoidal component of the ACF that is not dampening at a high rate (as we could potentially expect in models with AR components with $\alpha_i < 0$ for $i \in [1, p]$).

In fact, we can replicate the pattern in the sample ACF quite closely by simply using a sine function alongside random noise. Let $\{Z_t\}_{t=1}^{500} \overset{\text{iid}}{\sim} N(0,1)$ and let $X_t = Z_t + \sin(t/2)$, for $t \in \{1, 2, \ldots, 500\} \subset \mathbb{N}$. A plot of this artificial additive seasonal model is below:

```
set.seed(443); Z = rnorm(500); t = seq(1:500)
acf(Z + sin(t/2), main = "Artificial ACF", lag.max = 36)
```



As you can see, this artificial ACF closely matches our observed sample ACF from earlier. As such, it is very likely that there exists some seasonal term s_t in the time series of sea surface temperature data. We would intuitively anticipate a seasonal component to these data, as it is very likely that sea temperatures vary over the course of the year due to seasons as measurements are coming from the same area (the international dateline to the coast of South America.) Moreover, from a more informal observational perspective, a plot of the original series data doesn't seem to support a non-constant seasonal amplitude - as in there's no changing peak heights over time - which may indicate an additive seasonality rather than multiplicative.

In addition, neither the plot of the data nor the ACF seem to indicate a significant trend in the data. If there were a trend component m_t, we would see some consistent change over time in the overall direction of the series beyond simple seasonality. It doesn't seem like there is anything like that in this series; **however**, a purely visual analysis isn't comprehensive so this doesn't mean that a trend component doesn't exist.

Finally, to determine whether or not the series is stationary, we recall the definition of a weakly stationary stochastic process. Specifically, we will consider the first property of weak stationarity - that the mean is constant. We defined this formally in Assignment 1 previously, and was given similar to the following:

$$\text{Weak Stationarity Property One: } \exists \mu \in \mathbb{R} \text{ s.t. } \forall t \in \mathbb{Z},\ \mathbb{E}(X_t) = \mu$$

The presence of *either* a seasonal component or a trend causes a contradiction, as the expected value of the series becomes some function of t (and hence cannot be a constant.)

Thus, for our particular series, we have the following implication, letting s_t be the seasonal component, $c \in \mathbb{R}$ be some real constant and $f(t)$ be a non-constant real function (as in, it changes with t.) Then,

$$\exists s_t \implies \forall t\ \mathbb{E}(X_t) = f(t) + c \implies \nexists \mu \in \mathbb{R} \text{ s.t. } \forall t\ \mathbb{E}(X_t) = \mu \ \therefore\ X_t \text{ is not stationary. } \square$$

In short, due to the presence of a trend, the time series for the training data is not stationary. It should be noted that not much can be concluded with results from the PACF, as the series is non-stationary.

Part b

Forecast sea surface temperature for 2022 and 2023 using the Box-Jenkins method and the data from 1870-2021.

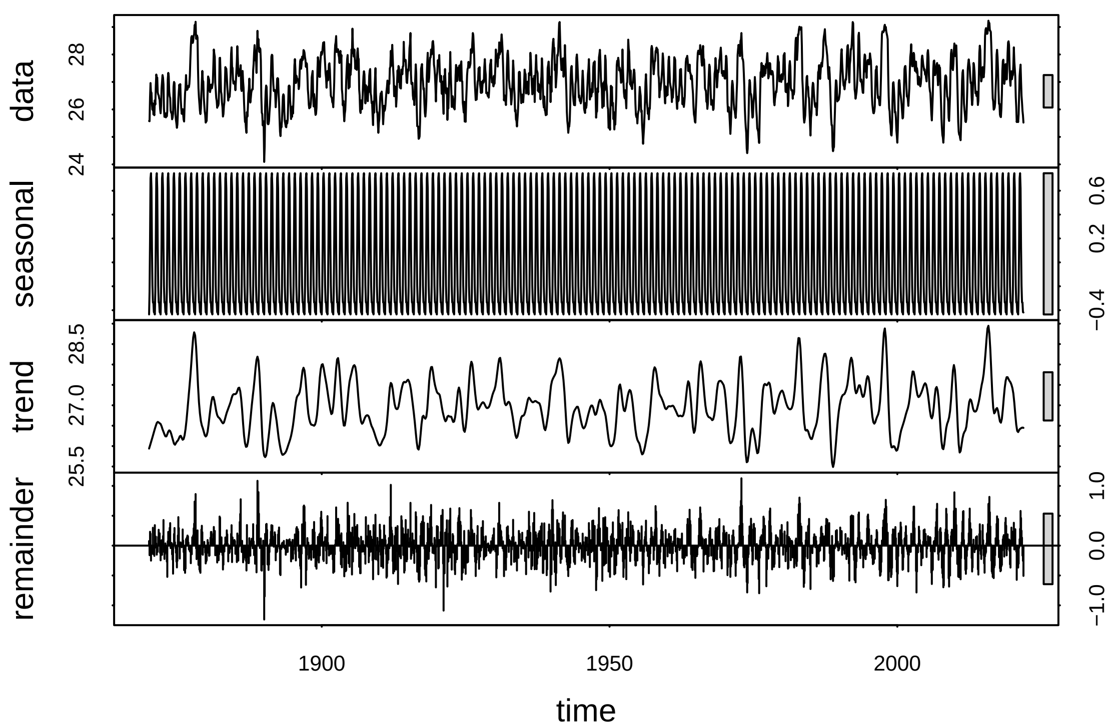
Part b.1.

Remove any seasonal variation and trend from the training data, if there is any, using the `stl` function in R.

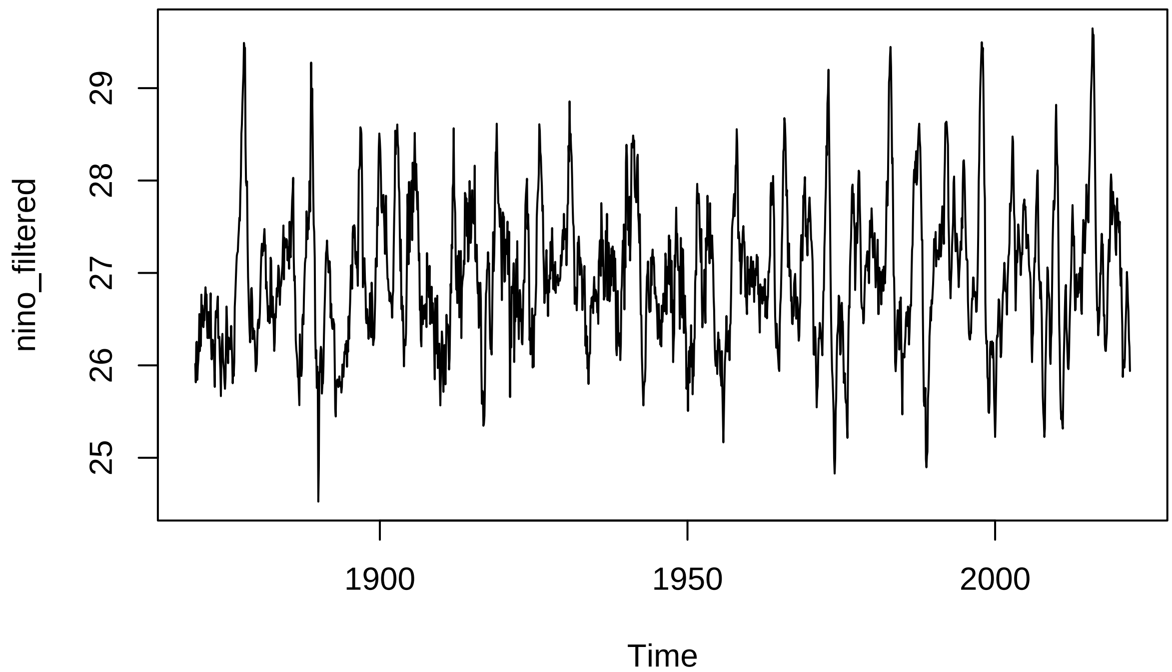
Plot the filtered data set, as well as its acf and pacf.

In experimenting with only de-seasonalizing and only de-trending the data, or both, the best performance (i.e. stationary/near stationary in the ACF) came from removing both season and trend.

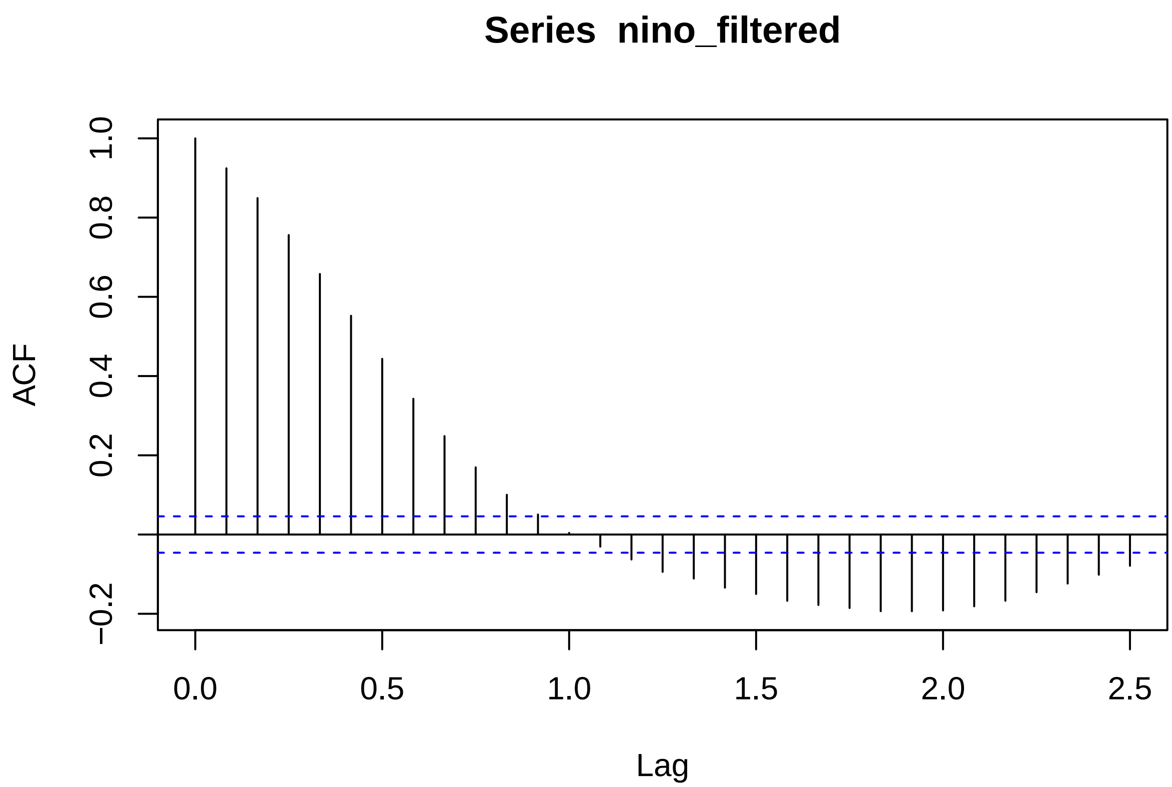
```
nino_stl = stl(nino_train, s.window = "periodic")
plot(nino_stl)
```



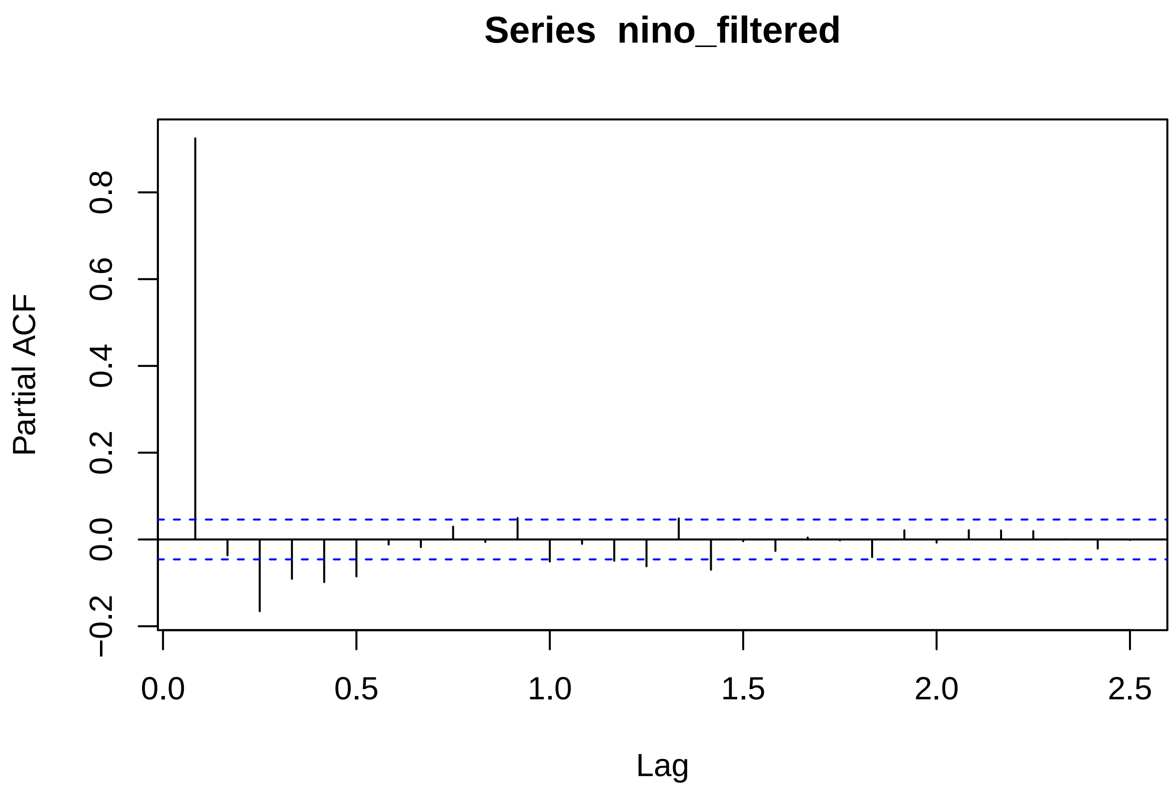

```
loess_decomp = data.frame(nino_stl$time.series)
nino_filtered =  nino_train - loess_decomp$seasonal # - loess_decomp$trend
plot(nino_filtered)
```




```
acf_filtered = acf(nino_filtered, lag.max = 30, plot = T)$acf
```




```
# ar(1) vs. ar(6)
pacf_filtered = pacf(nino_filtered, lag.max = 30, plot = T)$acf
```




```r
# TODO: AR(1) and AR(6) are the two candidate models
```

ideas:

```r
fit1 = arima(nino_filtered, order = c(6, 0, 0)) # -278.9
fit2 = arima(nino_filtered, order = c(1, 0, 0)) # -278.9

# ?arima
```

```r
fit2$loglik
```

```
## [1] -340.9082
```

```r
fit1$loglik
```

```
## [1] -290.0224
```

Part c

Forecast sea surface temperature for 2022 and 2023 using the Holt-Winters method and the data from 1870-2021.

Part c.1.

Use the HoltWinters function in R to fit an appropriate model to the training data.

As in our original series (and with the Box-Jenkins method) we did not observe a significant trend, we will construct a Holt-Winters model with $\beta = 0$.

We will use an additive seasonal effect, which inherits the `Frequency` of the time series as the period. In this case, then, $p = 12$.

We then have an additive seasonal effect I_t for t given by the following:

$$I_t = \gamma(x_t - L_t) + (1 - \gamma)I_{t-p}$$

Where L_t is the level component that defines Holt-Withers smoothing/forecasting methods. Since $\beta = 0$, it is constructed without a T component, and is given by:

$$L_t = \alpha(x_t - I_{t-p}) + (1 - \alpha)L_{t-1}$$

For $\alpha, \gamma \in [0, 1]$. Further, for $\ell \in \mathbb{Z}$, the ℓ-step ahead forecast at time t is given by the following.

$$\hat{x}_t(\ell) = L_t + I_{t-p+\ell}$$

We report the coefficients from `HoltWinters` in the table below.

```r
hw_no_trend = HoltWinters(nino_train, beta = FALSE, seasonal = "additive") # no trend
hw_results_coefs = (data.frame(
  Alpha = round(hw_no_trend$alpha, 3),
  Beta = paste(0, ".000", sep = ""),
  Gamma = paste(hw_no_trend$gamma, ".000", sep = "")
))
rownames(hw_results_coefs) = NULL
kable(hw_results_coefs, caption = "Holt-Winters Coefficients")
```

Table 1: Holt-Winters Coefficients

Alpha	Beta	Gamma
0.96	0.000	1.000

Our fitted $\hat{\gamma} = 1$, implying that the model places maximum smoothing weight on the current state of the season (i.e. the $(1 - \gamma)I_{t-p}$ component has zero weight according to the fitting process.) Further, our fitted $\hat{\alpha} \approx 0.96$ - this means that the model is placing a lot of significance on the current level (and period) and much less on levels in the past when fitting.

Part c.2.

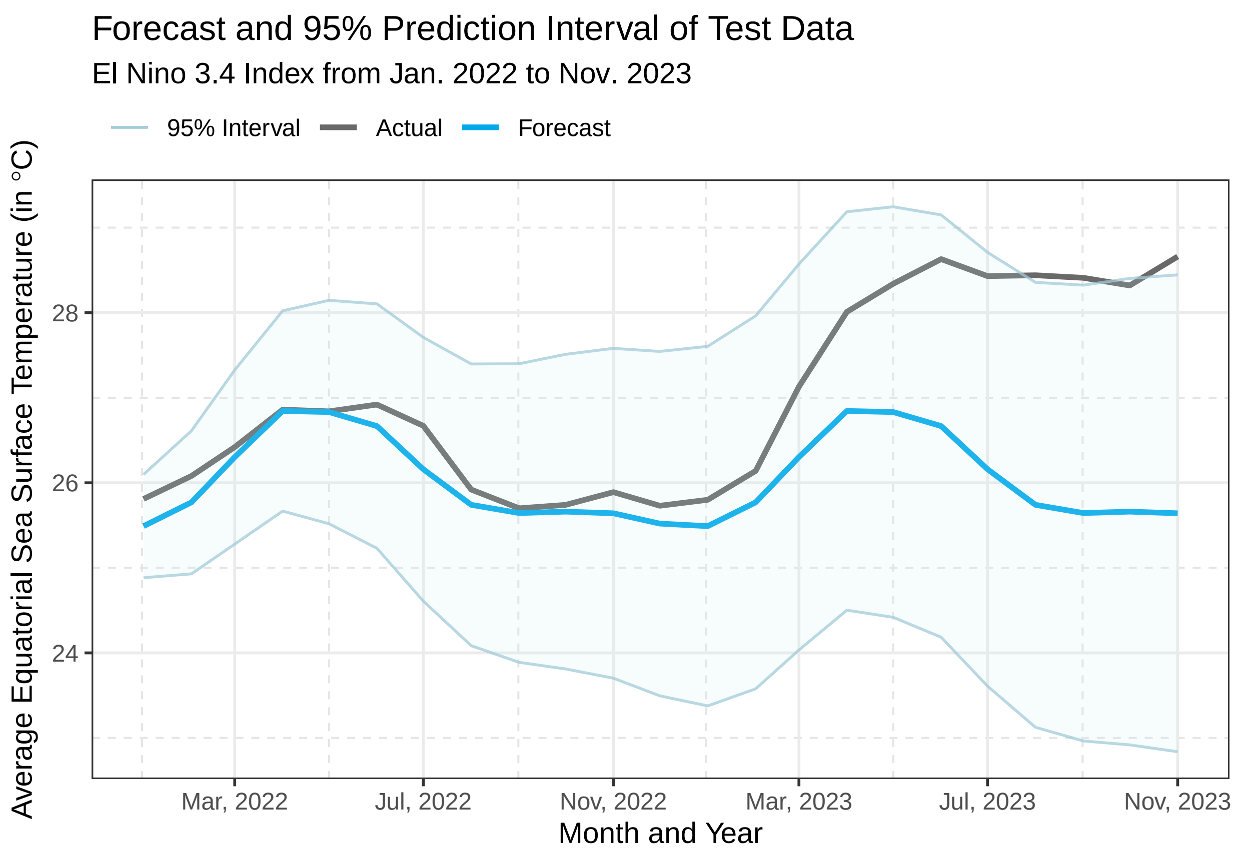
Use this model to predict sea surface temperature from Jan 2022 through Nov 2023.

```r
hw_predictions = data.frame(
  predict(hw_no_trend, n.ahead = 23, prediction.interval = TRUE, level = 0.95))

# prepare the hw forecast data
p4data = data.frame(
  Time = as.Date(time(nino_test)),
  Observed = as.numeric(nino_test),
  Forecast = as.numeric(hw_predictions$fit),
  Lower = as.numeric(hw_predictions$lwr),
  Upper = as.numeric(hw_predictions$upr)
)
```

```r
# make fancy as heck plot
p4 = ggplot(data = p4data, aes(x = Time)) +
  # lines for obsv, pred and interval bounds
```

```r
  geom_line(aes(y = Observed, color = "Actual"),  lwd = 1) +
  geom_line(aes(y = Forecast, color = "Forecast"),lwd = 1) +
  geom_line(aes(y = Lower, color = "95% Interval"), alpha = 0.75) +
  geom_line(aes(y = Upper, color = "95% Interval"), alpha = 0.75) +
  # light blue fill area between prediction intervals
  geom_ribbon(aes(ymin = Lower, ymax = Upper), fill = "#caf0f8", alpha = 0.15) +
  # legend and colour assignment
  scale_color_manual(name = "",
    values = c(
    "Actual" = "#696969",
    "Forecast" = "#02a9ea",
    "95% Interval" = "#9fc8d6"
  )) +
  # customize x-axis for nice dates
  scale_x_date(date_breaks = "4 month",  date_labels = "%b, %Y") +
  # add titles with units
  labs(
    title = "Forecast and 95% Prediction Interval of Test Data",
    subtitle = "El Nino 3.4 Index from Jan. 2022 to Nov. 2023",
    x = "Month and Year",
    y = expression(
      paste("Average Equatorial Sea Surface Temperature (in ", degree, "C)")
    )
  ) +
  theme_bw() +
  theme(panel.grid.minor =  element_line(
    color = "grey90",
    linewidth  = 0.35,
    linetype = "dashed"
  ), legend.position = "top", legend.justification = "left",
  legend.margin = margin(0,0,0,0))
print(p4)
```



Part c.3.

Calculate the mean squared prediction error.

The mean squared prediction error for our $N_\ell = 23$ total ℓ step ahead forecast is given by:

$$\text{MSE}_{\text{pred}} = \frac{1}{N} \sum_{\text{Holdout}} \Big(\text{Truth} - \text{Forecast}\Big)^2 = \frac{1}{N_\ell} \sum_{\ell=1}^{N_\ell} \Big(x_{t+\ell} - \hat{x}_t(\ell)\Big)^2$$

```
N = length(nino_test)
# verify equality in set cardinalities
stopifnot(all.equal(N, length(hw_predictions$fit)))
# calculate ms prediction error
ms_pre_hw = sum ( (nino_test - hw_predictions$fit )^2 ) /23
ms_pre_hw
```

```
## [1] 1.972297
```

The observed Mean Square prediction error for the Holt-Withers forecast is approximately 1.972.

Part c.4.

How does it compare to the Box-Jenkins models above?

```
# TODO: Comparison code goes here...
```

Question 2 : Hours Worked Forecasting

In this question we will predict the time series of monthly average values of the usual hours worked across all industries in Canada for the period from January 1987 until December 2023, which was explored in Assignment 1, using the file `usual_hours_worked_ca.csv`.

We'll use the Box-Jenkins method and Holt-Winters method.

Part 1: Data Preparation

Read in the data and create a time-series object for the mean monthly working hours:

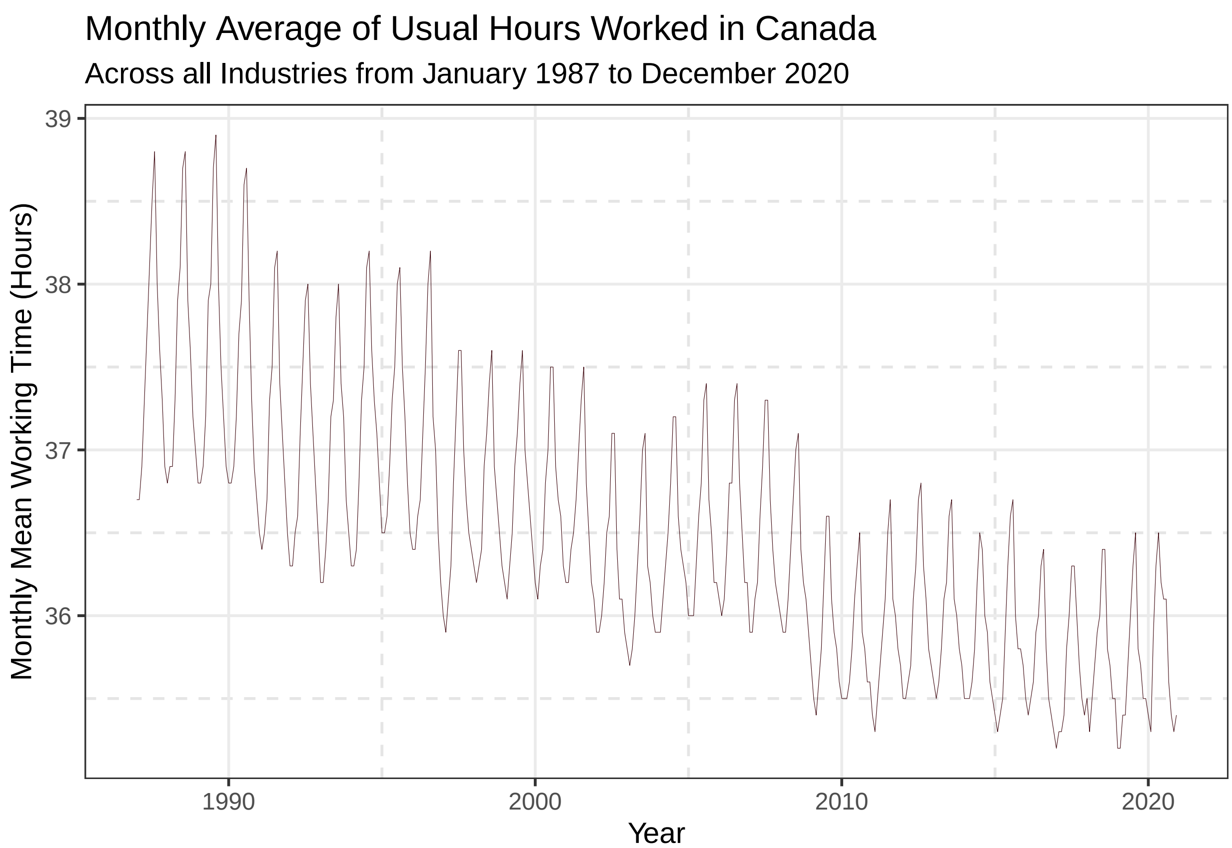
```
hDF = read.csv("usual_hours_worked_ca.csv")
hours_series = ts(hDF$Hours, start = c(1987, 1), frequency = 12)
```

Separate the data into train and test. The training dataset should include all observations up to and including December 2020; The test dataset should include all observations from January 2021 to December 2023

```
hours_train = window(hours_series, end = c(2020, 12))
hours_test = window(hours_series, start = c(2021, 1), end = c(2023, 12))
# verify split is valid
stopifnot(all.equal(length(hours_test)+length(hours_train),
                    length(na.remove(hours_series))))
```

Plot the training data.

```
p5data = fortify.zoo(hours_train)
n = length(hours_train)
p5 <- ggplot(p5data, aes(x = Index, y = hours_train)) +
  geom_line(color = "#410B13", linewidth = 0.1) +
  labs(
    title = "Monthly Average of Usual Hours Worked in Canada",
    subtitle = "Across all Industries from January 1987 to December 2020",
    y = "Monthly Mean Working Time (Hours)",
    x = "Year"
  ) + theme_bw() +
  theme(panel.grid.minor = element_line(
    color = "grey90",
    linetype = "dashed",
    linewidth = 0.5
  ))
print(p5)
```



Part 2: Box-Jenkins Method

In this part, we select and fit a SARIMA$(p, d, q) \times (P, D, Q)_s$ model and make forecasts using the fitted model.

Part A: Differencing

Difference the training set time series at lag 1. Let $\{X_t\}_{t\in\mathbb{Z}}$ be our original series. We define below $Y_t = \nabla^1 X_t = X_t - X_{t-1}$

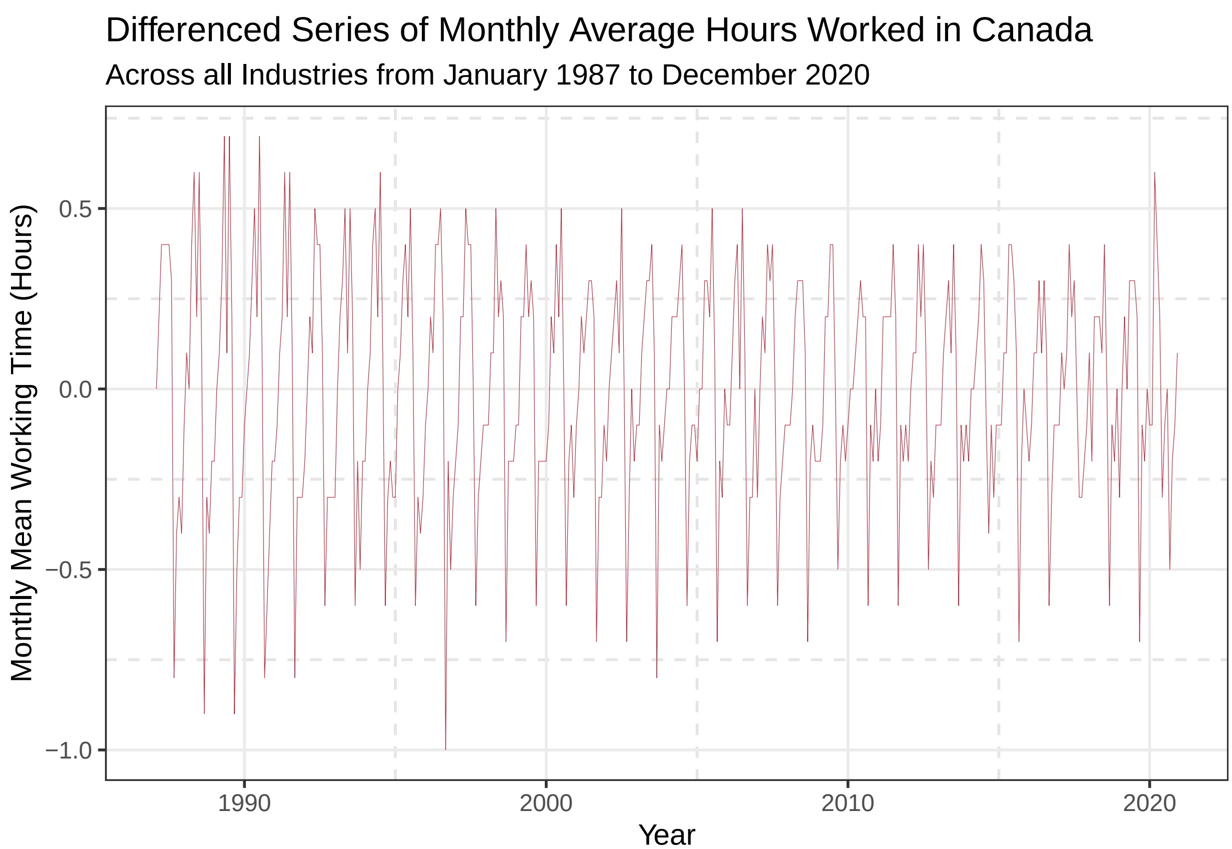
```
hours_train_diff = diff(hours_train, lag = 1, differences = 1)
```

Plot the new time series and its correlogram.

New Series

```
p6data = fortify.zoo(hours_train_diff)
p6 <- ggplot(p6data, aes(x = Index, y = hours_train_diff)) +
  geom_line(color = "#BA1F33", linewidth = 0.1) +
  labs(
    title = "Differenced Series of Monthly Average Hours Worked in Canada",
    subtitle = "Across all Industries from January 1987 to December 2020",
    y = "Monthly Mean Working Time (Hours)",
    x = "Year"
  ) + theme_bw() +
  theme(panel.grid.minor = element_line(
    color = "grey90",
    linetype = "dashed",
```

```
    linewidth = 0.5
  ))
print(p6)
```

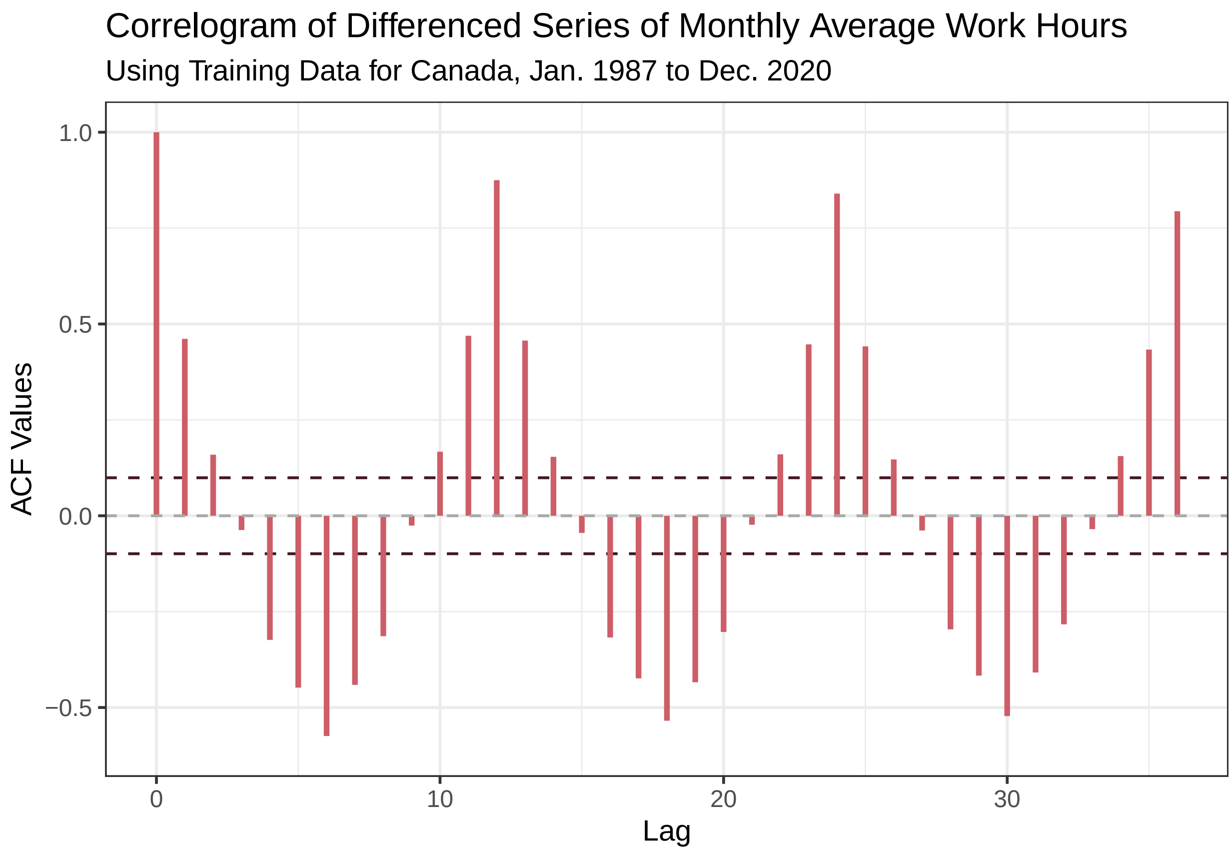


Correlogram

```
p7data = data.frame(
  h = 0:36,
  rh = acf(hours_train_diff, plot = FALSE, lag.max = 36)$acf
)
p7 <- ggplot(p7data, aes(x = h, y = rh)) +
  geom_hline(yintercept = 2/sqrt(n),
             linetype = "dashed",
             col = "#421820") +
  geom_hline(yintercept = -2/sqrt(n),
             linetype = "dashed",
             col = "#421820") +
  ylim(-0.6, 1) +
  geom_segment(aes(xend = h, yend = 0),
               color = "#CD5D67",
               linewidth = 1) +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "darkgray") +
  labs(x = "Lag",  y = "ACF Values",
       title = "Correlogram of Differenced Series of Monthly Average Work Hours",
```

```
    subtitle = "Using Training Data for Canada, Jan. 1987 to Dec. 2020") +

  theme_bw()
print(p7)
```



Comment on what you observe.

While the single-iteration ($d = 1$) of sequential differencing at lag 1 appears to have helped to mitigate the trend in the original training data, as seen in the plot of the differenced training time series, there still appears to be a notable seasonal component in the data. We know that this is likely the case due to the sinusoidal component in the correlogram, which is not really decreasing as a function of lag as we would like to see. This periodicity is indicative of a lingering seasonal component that we still need to difference out or remove via other methods.

Part B: Seasonal Differencing

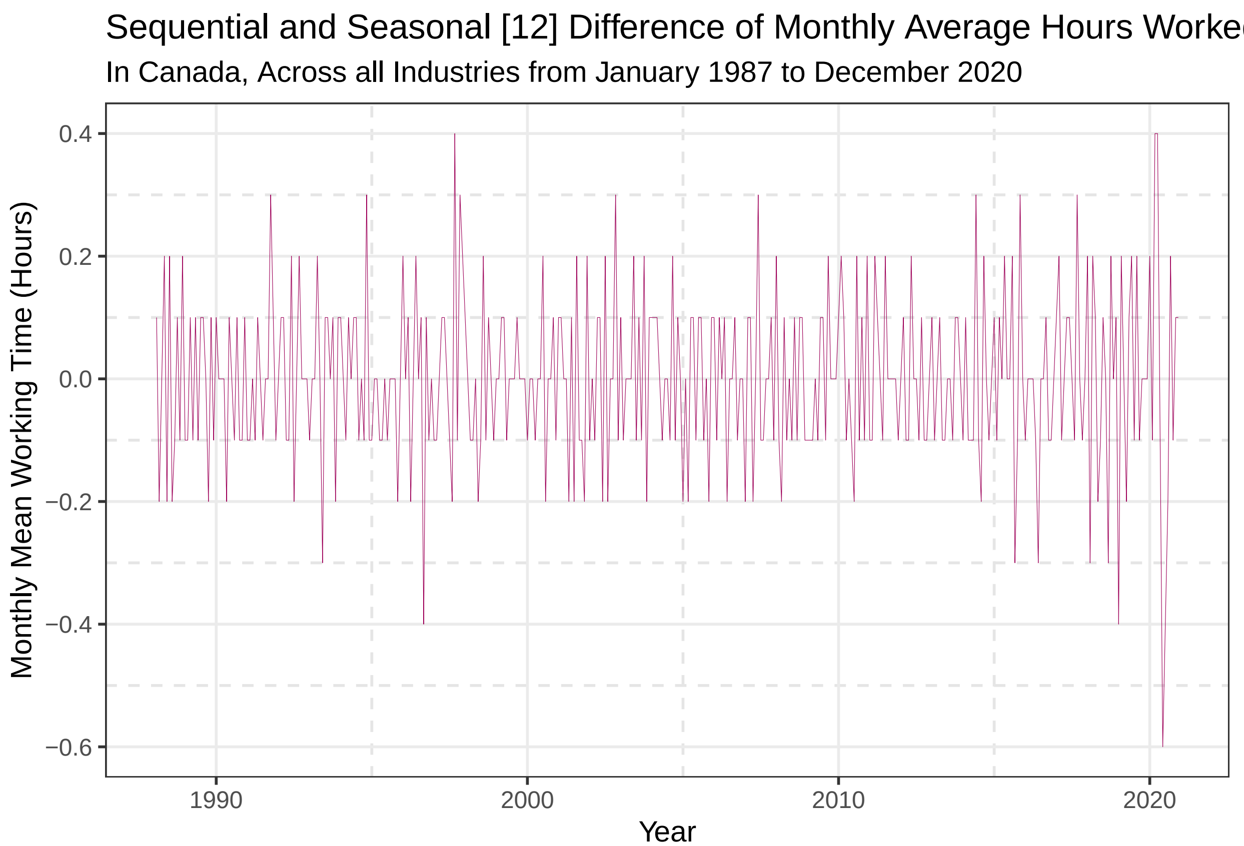
Apply seasonal differencing to remove seasonal variation. Since we have monthly data, $s = 12$ seems sensible.

```
hours_train_diff_s = diff(hours_train_diff, lag = 12, differences = 1)
```

Plot the resulting differenced time series along with its sample acf and pacf. pure AR only for PACF Series

```
p8data = fortify.zoo(hours_train_diff_s)
p8 <- ggplot(p8data, aes(x = Index, y = hours_train_diff_s)) +
  geom_line(color = "#9e0059", linewidth = 0.1) +
  labs(
    title = "Sequential and Seasonal [12] Difference of Monthly Average Hours Worked",
    subtitle = "In Canada, Across all Industries from January 1987 to December 2020",
```

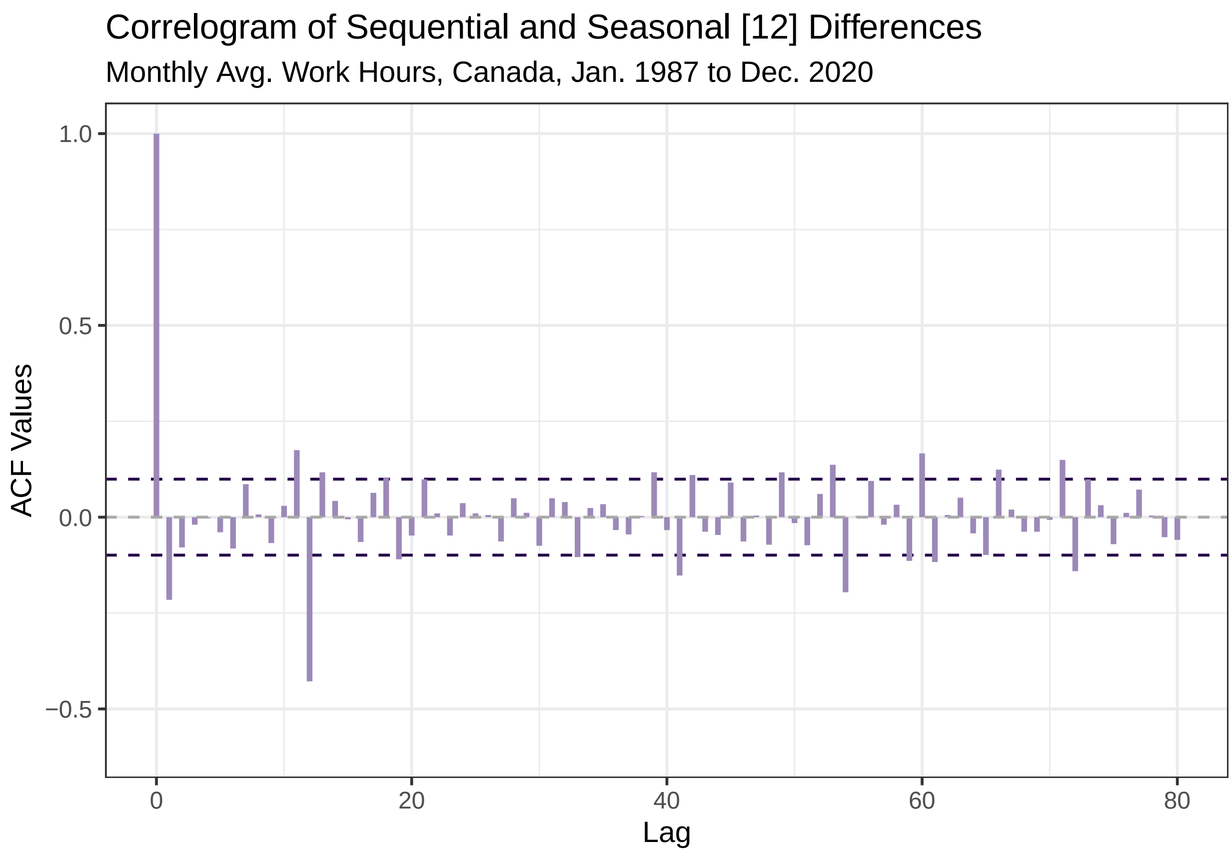
```
    y = "Monthly Mean Working Time (Hours)",
    x = "Year"
  ) + theme_bw() +
  theme(panel.grid.minor = element_line(
    color = "grey90",
    linetype = "dashed",
    linewidth = 0.5
  ))
print(p8)
```



Sample ACF

```
p9data = data.frame(
  h = 0:80,
  rh = acf(hours_train_diff_s, plot = FALSE, lag.max = 80)$acf
)
p9 <- ggplot(p9data, aes(x = h, y = rh)) +
  geom_hline(yintercept = 2/sqrt(n),
             linetype = "dashed",
             col = "#240046") +
  geom_hline(yintercept = -2/sqrt(n),
             linetype = "dashed",
             col = "#240046") +
  ylim(-0.6, 1) +
  geom_segment(aes(xend = h, yend = 0),
               color = "#9c89b8",
```

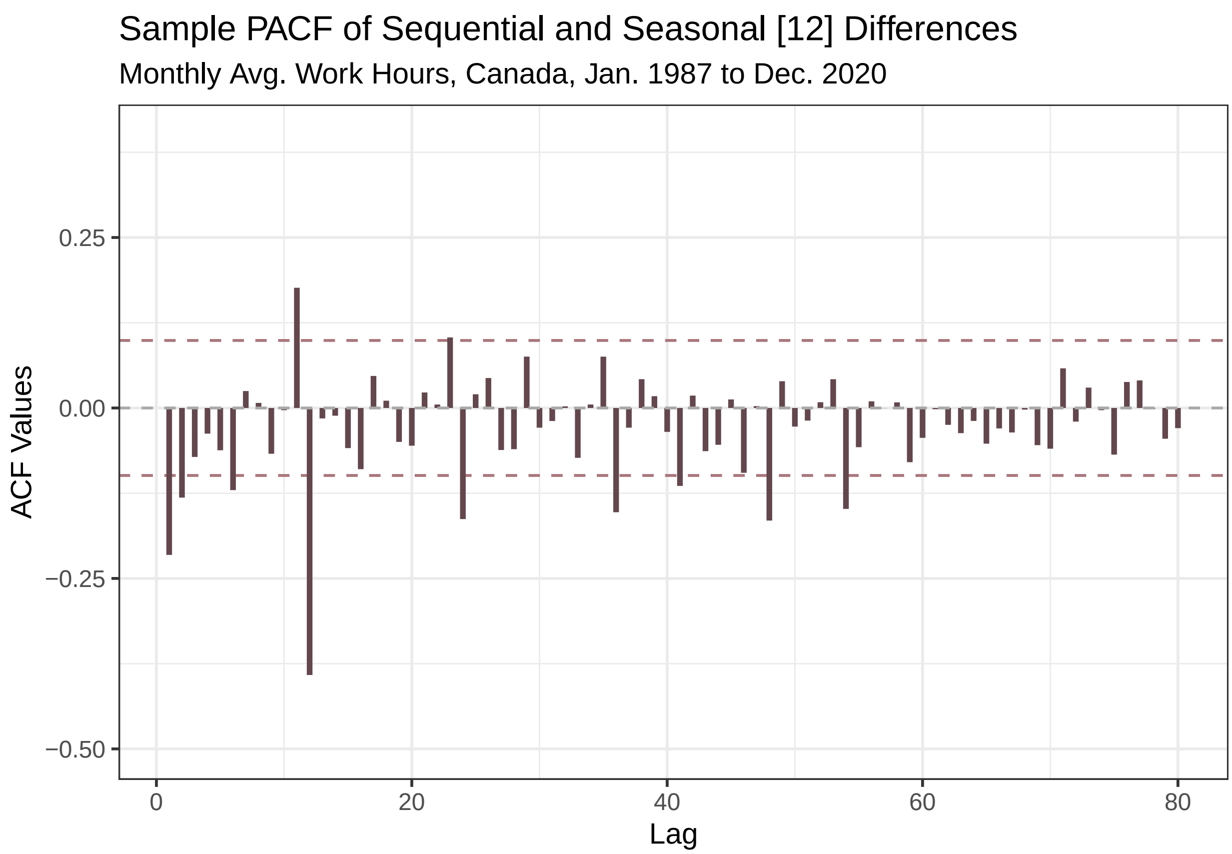
```
                linewidth = 1) +
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "darkgray") +
  labs(x = "Lag",  y = "ACF Values",
       title = "Correlogram of Sequential and Seasonal [12] Differences",
       subtitle = "Monthly Avg. Work Hours, Canada, Jan. 1987 to Dec. 2020") +
  theme_bw()
print(p9)
```



Sample PACF

```
p10data = data.frame(
  h = 1:80,
  rh = pacf(hours_train_diff_s, plot = FALSE, lag.max = 80)$acf
)
p10 <- ggplot(p10data, aes(x = h, y = rh)) +
  geom_hline(yintercept = 2/sqrt(n),
             linetype = "dashed",
             col = "#aa767c") +
  geom_hline(yintercept = -2/sqrt(n),
             linetype = "dashed",
             col = "#aa767c") +
  ylim(-0.5, 0.4) +
  geom_segment(aes(xend = h, yend = 0),
                color = "#63474d",
                linewidth = 1) +
```

```r
  geom_hline(yintercept = 0,
             linetype = "dashed",
             color = "darkgray") +
  labs(x = "Lag",  y = "ACF Values",
       title = "Sample PACF of Sequential and Seasonal [12] Differences",
       subtitle = "Monthly Avg. Work Hours, Canada, Jan. 1987 to Dec. 2020") +
  theme_bw()
print(p10)
```



Comment on what you observe:

The seasonal differencing paired with the sequential differencing seems to have done a good job in making the series stationary. We see from the plotted data as well as the correlogram that there isn't evidence of a significant season or trend. Further, the visual complexity of both the ACF and the PACF hints at the fact that there may be a blended model (i.e. a mixture of AR and MA) underlying this process. We will discuss in the next sections exaclty what the parameters of the model equation are in this case.

Part C: Difference Parameters.

Based on the results of Part II (a) and (b), specify the values of d, D, and s.

From Part II (a), we applied one sequential difference using `diff`, so we know $d = 1$. Then, in Part II (b), we applied one seasonal difference using `lag = 12`, hence $D = 1$ with $s = 12$.

PACF ONLY FOR PURE AR

TAKE SOME MA MODEL

Part D: SARIMA Parameters

Based on the plots in Part II (b), suggest possible values of justifying your choices.

NOTE: It is very difficult to accurately determine these coefficients precisely, but I will justify my decisions to the best of my ability.

Firstly, let's discuss the presence of an MA component. We henceforth assume stationarity.

In the ACF, we see a fairly large r_h value at $h = 12$, that significantly cuts off for $h > 12$. Unlike a pure MA plot where we would see an expect a very sharp cutoff, in a blended model there will still be other components at work after the MA component.

All of the lags after $h = 12$ for $h \in [0, 36] \subset \mathbb{Z}$ are noteably smaller, as the Sample ACF shows.

This would imply that either we have a sequential MA with $q = 12$ or a seasonal MA with $Q = 1$. By the prinicple of parsimony, we prefer the latter. Also, this conclusion would imply that it is unlikely that there is a p component.

To explain, we recall the right-hand side of a SARIMA model, assuming $Q = 1$.

$$\begin{aligned}
\text{RHS } &= \Theta(B^s)\theta(B)Z_t \\
\text{RHS } &= (1 + \tilde{\beta}B^{12})\theta(B)Z_t \\
\text{RHS } &= \theta(B)Z_t + \theta(B)\tilde{\beta}Z_{t-12}
\end{aligned}$$

This would help explain the decently large "spike" at lag 12 (also perhaps suggesting that $\tilde{\beta} < 0$.)

However, is there a sequential component $q > 0$? By the above, under the assumption that $Q = 1$, if $\theta(B)$ was of an order greater than zero, there would be additional components in the right-hand side that we'd expect to see reflected in the ACF.

The most likely visually is that $q = 1$. An explanation of why is facilitated by the equation below:

$$\begin{aligned}
\text{RHS } &= (1 + \beta B)Z_t + (1 + \beta B)\tilde{\beta}Z_{t-12} \\
\text{RHS } &= Z_t + \beta Z_{t-1} + \tilde{\beta}Z_{t-12} + \beta\tilde{\beta}Z_{t-13}
\end{aligned}$$

In other words, from our understanding of MA models, we would see large in magnitude r_1 and (slightly less large) r_{13}. In the sample ACF, with a spike at lag 1 less than $-2/\sqrt{n}$ and a shorter spike at lag 13, slightly above $2/\sqrt{n}$, however, it is difficult to discern whether or not these are caused by the AR components of the model.

I am more inclined to favour the AR components causing these observed r_h values than additional MA components due to the fact that there is a large autocorrelation at lag 11, which isn't described by this sort of MA formulation. Notably, the fact that visually $|r_{11}| \approx |r_1|$ lends informal credence to the hypothesis that it is AR components rather than MA causing the r_1 and r_{13} observations. If there were a $q > 0$, I would expect it to have a notable magnitude outside of the general AR pattern. So, while it's theoretically plausible that $q > 0$, by this reasoning we settle onto $q = 0$ in favour of a more robust AR components.

Now let's consider whether or not there are AR components to the series. Visually, the slowly-decaying oscillating pattern is indicative of one or more AR components. Further, since the lag doesn't cut off to white noise immediately after the MA components (we see it sometimes creep above the $\pm 2/\sqrt{n}$) this tells us there is very likely an AR component of some variety.

To figure out exactly what order these components are, we utilize the PACF, recalling the assumption of stationarity.

However, we first must recall the left-hand side of the SARIMA equation, so we have a better understanding of the PACF Values

$$\begin{aligned}
\text{LHS} &= \Psi(B^s)\psi(B)W_t \\
\text{LHS} &= \Psi(B^s)\psi(B)(\nabla_1^{12}\nabla^1 X_t) \\
\text{LHS} &= \Psi(B^s)\psi(B)(X_t - X_{t-12} - X_{t-1} + X_{t-13})
\end{aligned}$$

So, it's a bit more difficult to directly examine the PACF for the largest value of $\hat{\alpha}_{kk}$ such that $\forall i > k, |\hat{\alpha}_{ii}| < 2\sqrt{n}$ as we normally would, since it isn't direct to discern which coefficient in the AR decomposition it actually aligns to in the overall model (i.e. in the full seasonal/sequential differenced with MA.)

Then we find a logical combination of p and P, we can informally examine the PACF for what seems "sensible." What immediately jumps out to me is the presence of a large $\hat{\alpha}_{kk}$ at $k = 12$. This is indicative of a seasonal AR component, i.e. $P \geq 1$. Further, there is an immediate jump at $\hat{\alpha}_{11}$ as well as other "spikes" across the PACF at $\hat{\alpha}_{11,11}, \hat{\alpha}_{23,23}$, etc. Since these values $\{1, 11, 23, \dots\}$ are not evenly divisiple by 12, we know that there must exist one or more sequential AR components, i.e. $p \geq 1$. For the sake of parsimony, my initial suggestion would be to choose $P = p = 1$ as a starting point.

Due to the complexity inherent in this particular mixed models, it is difficult to visually determine values of p, P, q, and Q, however, from the arguments above, $\{p, P, q, Q\} = \{0, 1, 1, 1\}$ seems plausible.

```
# n = length(hours_train_diff_s)
# # (p,d,q): the AR order, the degree of differencing, and the MA order.
# p9data$rh[p9data$h]
# all_lags = acf(hours_train_diff_s, plot = FALSE, lag.max = n)$acf
# twelves = (all_lags[seq(from = 0, to = n, by = 1) %% 11 == 0])
#
# plot(0:(length(twelves)-1), twelves, type="n",
#      xlab="Index", ylab="Value", main="Lags at Step 12")
# segments(x0=0:(length(twelves)-1), y0=0, x1=0:(length(twelves)-1), y1=twelves)
# abline(h = 0)
```

Part E: Iterative AIC

Now use the Akaike's Information Criterion (AIC) to select the model based on the training dataset in Part I. Fix the values of p and P as your suggestions in Part II (d),

Consider $q \in [0, 5]$ and $Q \in [0, 5]$.

Select the values of q and Q according to the AIC values.

Fit the model you choose and print the values of the estimated parameters along with the AIC value for the model.

First, we declare our fixed p, P, s, d and D values, then iterate through the different 25 models that can be developed and report the AIC for each.

```
p = 0; P = 0; s = 12; d = 1; D = 1
# declare empty matrix
results = matrix(0, nrow = 6, ncol = 6)
# with legible names
rownames(results) = c("q=0", "q=1", "q=2","q=3","q=4","q=5")
colnames(results) = c("Q=0", "Q=1", "Q=2","Q=3","Q=4","Q=5")
# iterate through all candidates
for (q in 0:5){
  for (Q in 0:5){
    results[q+1, Q+1] = arima(hours_train,
                          order = c(p, d, q),
                          seasonal = c(P, D, Q))$aic
```

```
  }
}
```

The AIC results are summarized in the table below.

```
kable(round(results, 3), caption = "AIC Values for Varying (q, Q)")
```

Table 2: AIC Values for Varying (q, Q)

	Q=0	Q=1	Q=2	Q=3	Q=4	Q=5
q=0	-464.461	-613.121	-611.440	-609.462	-612.427	-612.305
q=1	-487.115	-616.650	-614.907	-612.958	-616.029	-616.225
q=2	-489.810	-618.001	-616.272	-614.388	-617.448	-618.428
q=3	-488.932	-622.962	-621.387	-619.411	-623.477	-623.785
q=4	-487.747	-624.882	-623.135	-621.160	-625.432	-625.629
q=5	-488.378	-623.821	-621.895	-619.917	-624.506	-624.633

According to this table, the best model corresponds to:

```
which(results == min(results), arr.ind = TRUE)
```

```
##     row col
## q=4   5   6
```

```
# arima(hours_train, order = c(1, 1, 0), seasonal = c(1, 1, 0))$aic
# arima(hours_train, order = c(1, 1, 0), seasonal = c(1, 1, 1))$aic
min(results)
```

```
## [1] -625.6292
```

Which corresponds to $q = 4$ and $Q = 5$. It has an AIC of -627.61

Hence, we will use a model of the form:

$$\text{SARIMA}(\underset{p}{0}, \underset{d}{1}, \underset{q}{4}) \times (\underset{P}{0}, \underset{D}{1}, \underset{Q}{5})_{\underset{s}{12}}$$

The full model is given below:

```
best_fit = arima(hours_train, order = c(0, 1, 1), seasonal = c(1, 1, 5))
#?auto.arima(hours_train)
```

```
# Series: hours_train
# ARIMA(1,0,0)(0,1,4)[12] with drift
# auto.arima(hours_train, trace = F, ic = "aic",
#             start.p = 1, start.P = 1, stepwise = F,
#             max.Q = 5, max.q = 5, max.d = 1, max.D = 1)

# predict(best_fit, prediction.interval = T, level = 0.95)
```

```
# best_fit$coef
auto.arima(hours_train, D = 1, d = 1, stepwise = F,
           ic = "aic")
```

```
## Series: hours_train
## ARIMA(1,1,1)(1,1,2)[12]
##
## Coefficients:
```

```
##          ar1      ma1     sar1     sma1     sma2
##       0.7207  -0.8819  -0.4004  -0.3467  -0.2801
## s.e.  0.1046   0.0766   0.7864   0.7811   0.5714
##
## sigma^2 = 0.01155:  log likelihood = 318.86
## AIC=-625.72   AICc=-625.5   BIC=-601.84
```

```
# auto.arima(hours_train,
#             d = 1, D = 1,
#             max.p = 5, max.P = 5,
#             max.q = 5, max.Q = 5,
#             start.P = 3, start.p = 5,
#             start.q = 0, start.Q = 0,
#             stepwise = FALSE, approximation = TRUE)
```