Stat 443: Time Series and Forecasting

Assignment 3: Time Series Models

Caden Hewlett

March 17, 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.

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

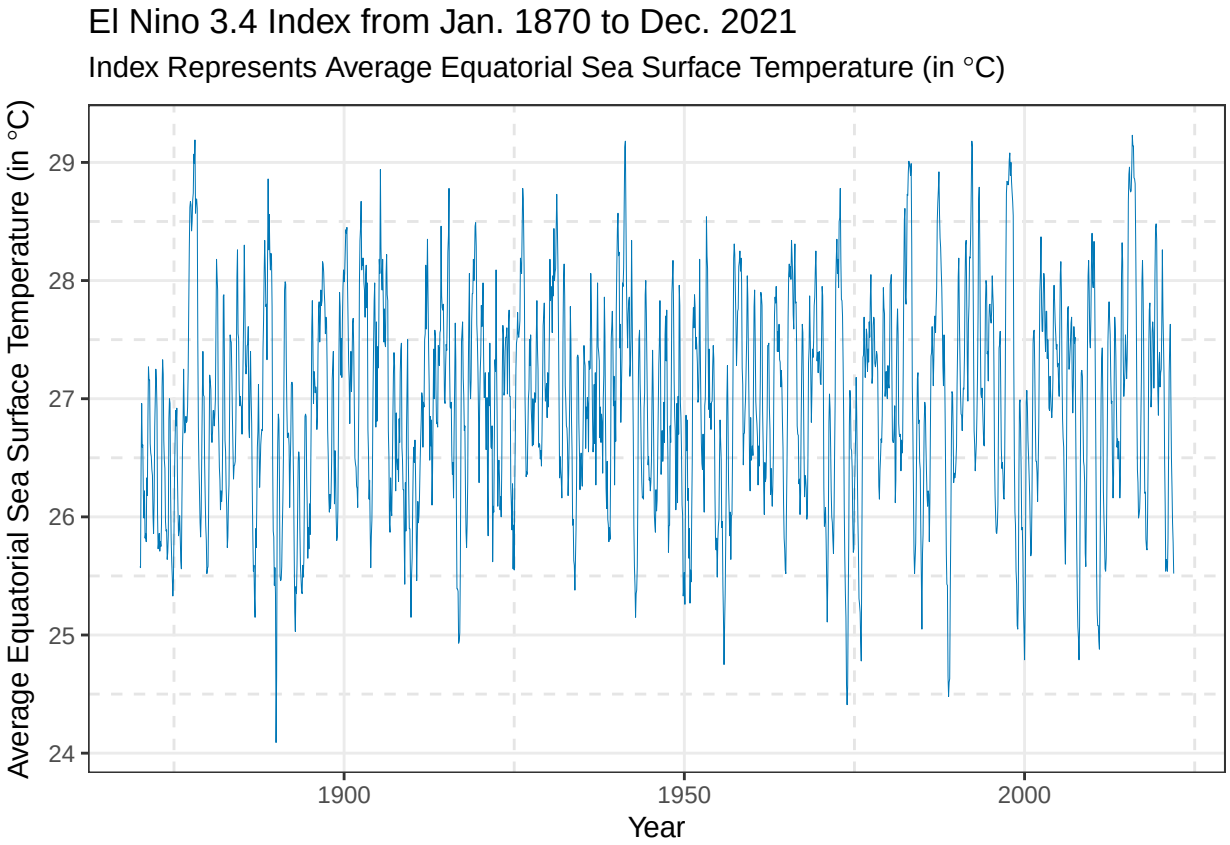
```
# split data into train and test
nino_train = window(nino_ts, start = c(1870, 1), end = c(2021, 12))
nino_test = window(nino_ts, start = c(2022, 1), end = c(2023, 11))
# verify split
stopifnot(all.equal(
          length(nino_test)+length(nino_train), length(na.remove(nino_ts))
      )
)
```

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)
```

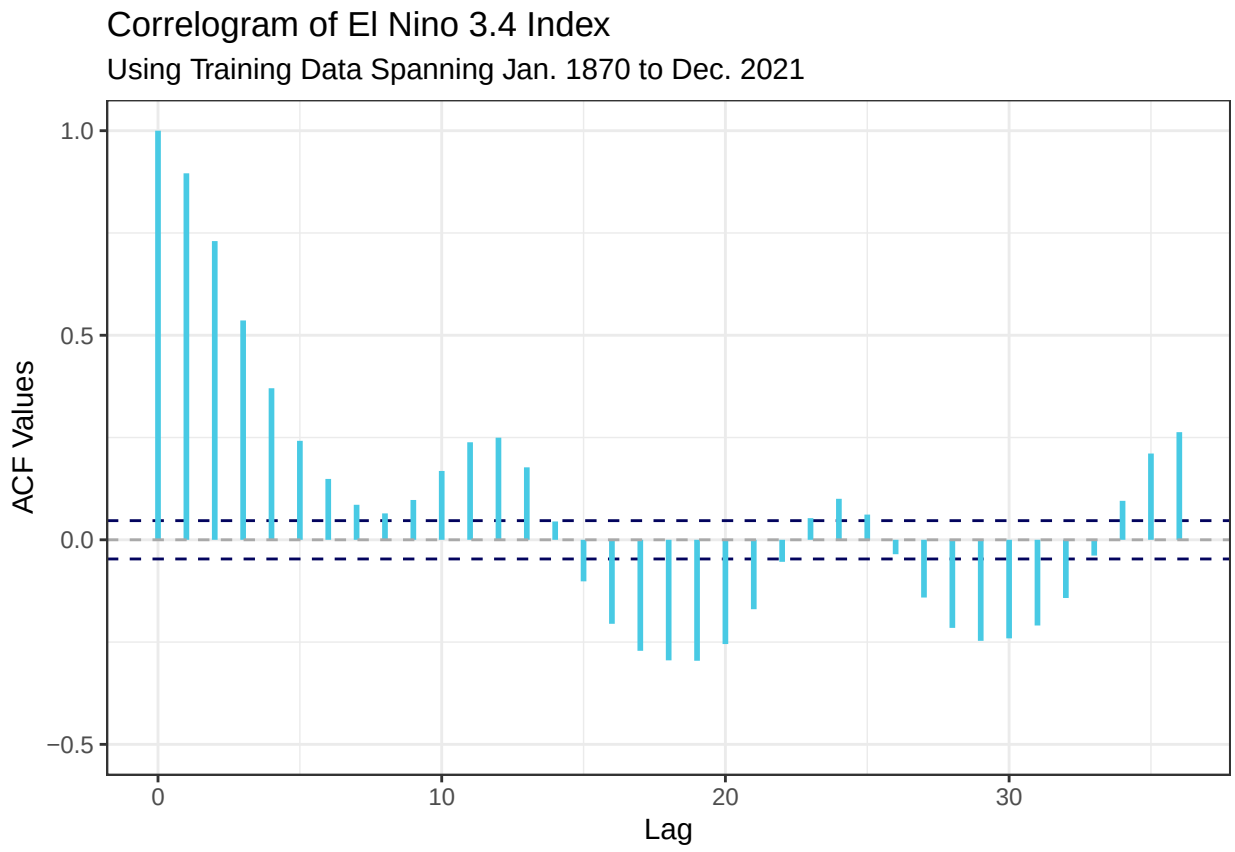


Autocorrelation Function

```
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)
```

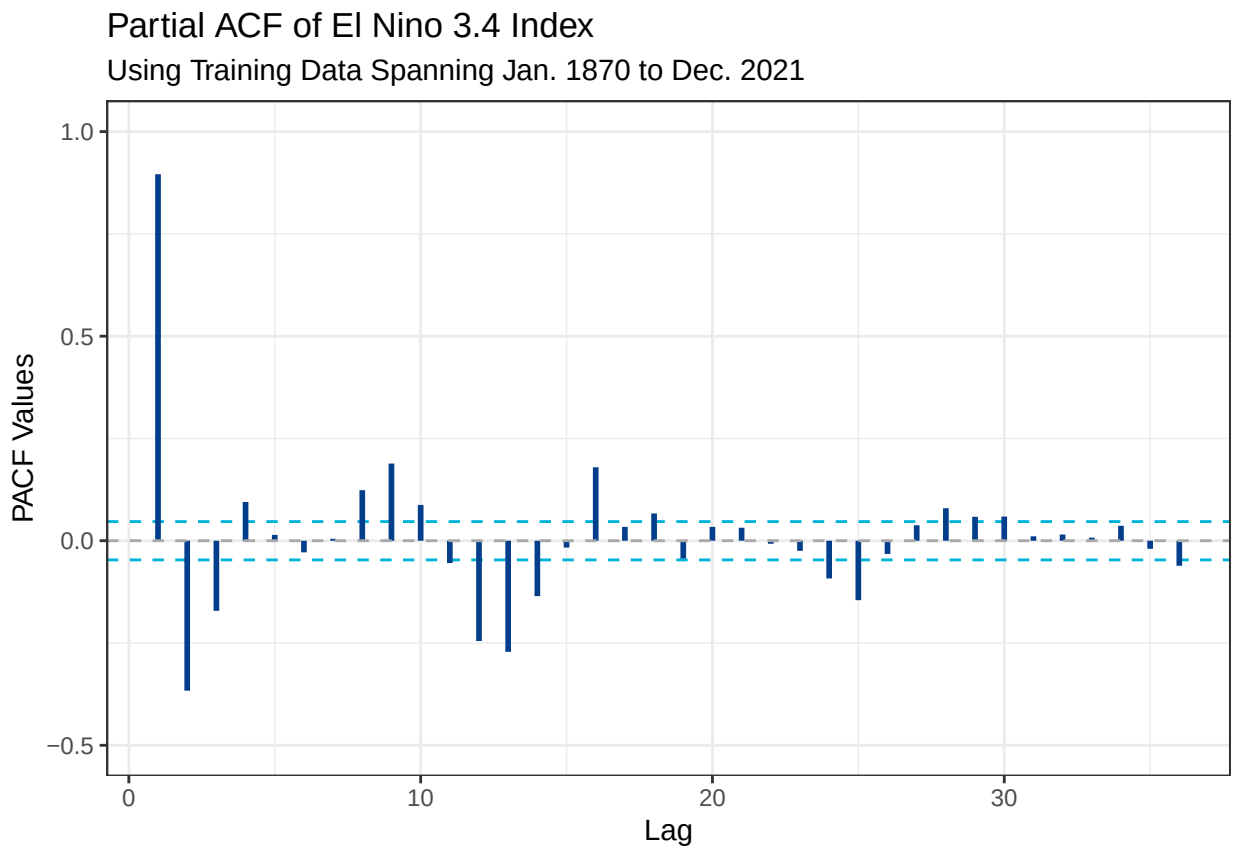


Partial Autocorrelation Function

```
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)
```

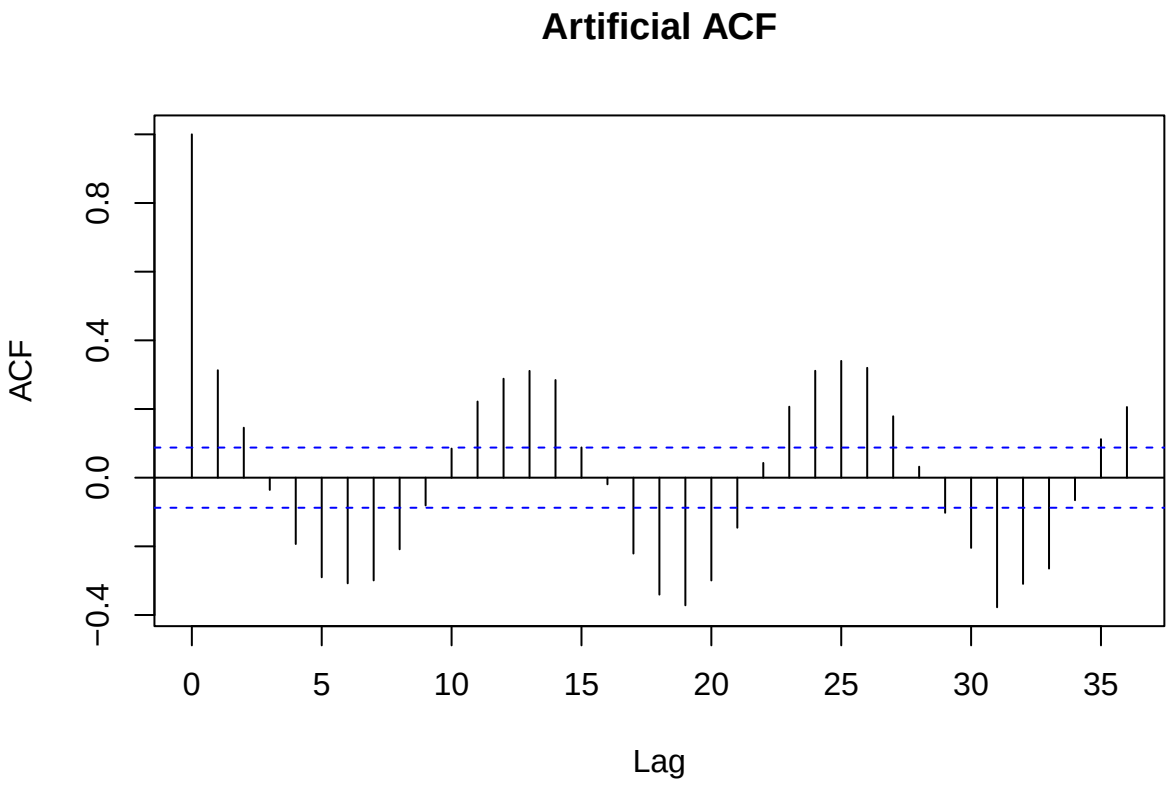


Comments

From the ACF, we see that the series exhibits a clear trend. We know that this trend is present due to the sinusoidal component of the ACF.

In fact, to replicate this pattern exactly. 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:

```
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 ACF from earlier. As such, it is very likely that there exists some seasonal term s_t. Further, 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 changing peak heights between seasons) which may indicate an additive seasonality rather than multiplicative.

Further, 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 a weak stationarity - that the mean is constant. We defined this formally in Assignment 1 previously, and was given as follows:

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

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.)

Hence, for our particular series, consider the following implication, letting s_t be the seasonal component:

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

Where the conclusion of the implication is due to the negation of Property One. In short, due to the presence of a trend, the training time series is not 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")
loess_decomp = data.frame(nino_stl$time.series)
nino_filtered =  nino_train - loess_decomp$seasonal # - loess_decomp$trend
acf_filtered = acf(nino_filtered, lag.max = 30, plot = F)$acf
```

```
pacf_filtered = pacf(nino_filtered, lag.max = 30, plot = F)$acf
```

```
# plot(nino_stl)
```

ideas:

```
# fit1 = arima(nino_filtered, order = c(6, 0, 0)) # -278.9
# fit2 = arima(nino_filtered, order = c(17, 0, 0)) # -278.9
# arima(nino_filtered, order = c(0, 0, 12))
# ?arima
```

```
# fit2$loglik
```

```
auto.arima(nino_train)
```

```
## Series: nino_train
## ARIMA(1,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2
##       0.9275  -0.6989  -0.3877
## s.e.  0.0088   0.0217   0.0216
##
## sigma^2 = 0.1155:  log likelihood = -618.27
## AIC=1244.53   AICc=1244.55   BIC=1266.54
```

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 $t \in \mathbb{Z}$ 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.

```
hw_no_trend = HoltWinters(nino_train, beta = 0, 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
0.961	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.

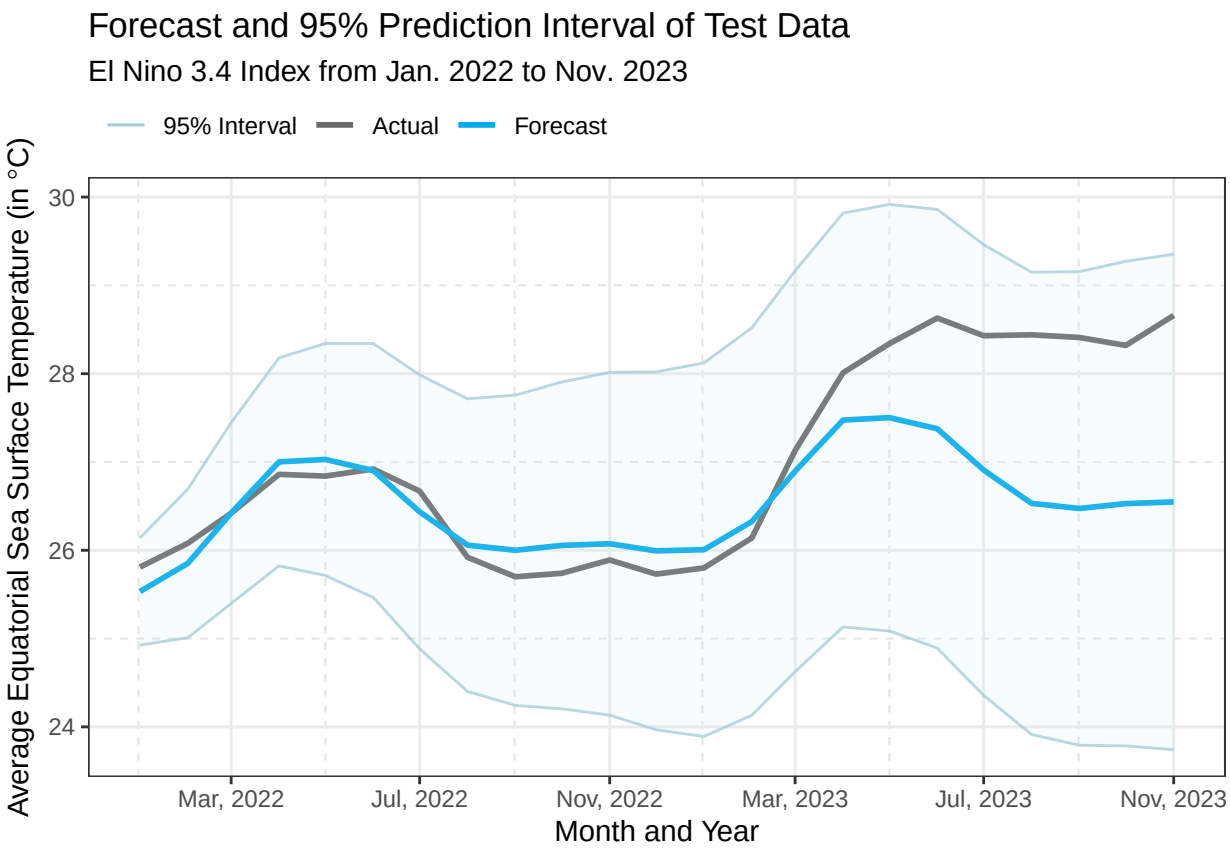
Part c.2.

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

```
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
  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$fit)))
# calculate ms prediction error
ms_pre_hw = mean( (hw_predictions$fit - nino_test)^2 )
ms_pre_hw
```

```
## [1] 0.8965602
```

Part c.4.

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