Stat 443 Assignment 1: Exploratory Data Analysis

Caden Hewlett

January 25, 2024

Task 1: Analyzing Usual Hours Worked in Canada

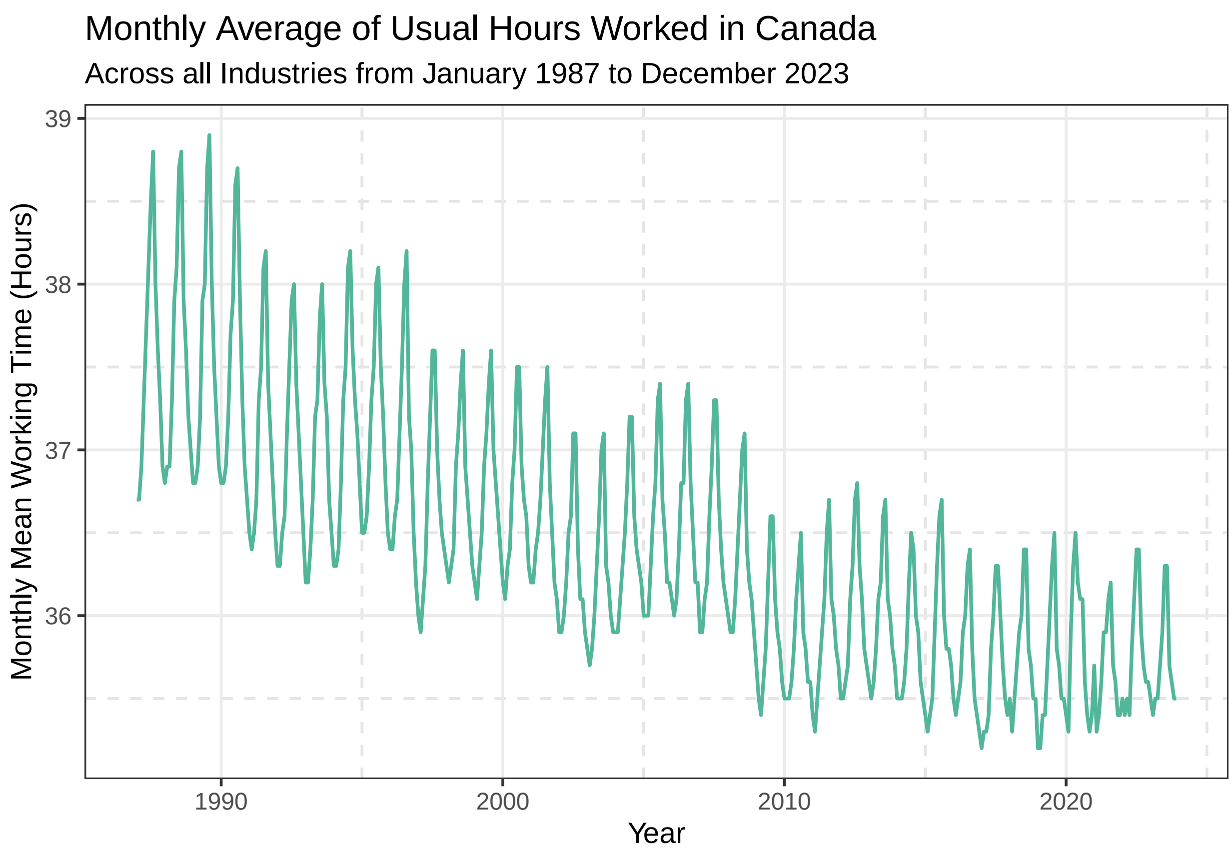
a) Part (a)

Read in the data and create a time-series object. Plot the series and comment on any features of the data that you observe. In particular address the following points:

```r
df <- read.csv("usual_hours_worked_ca.csv")

jobseries = ts(data = df$Hours, start = c(1987, 1),  frequency = 12)

p1data = fortify.zoo(jobseries)
p1 <- ggplot(p1data, aes(x = Index, y = jobseries)) +
  geom_line(color = "#52b69a", linewidth = 0.65) +
  labs(
    title = "Monthly Average of Usual Hours Worked in Canada",
    subtitle = "Across all Industries from January 1987 to December 2023",
    y = "Monthly Mean Working Time (Hours)",
    x = "Year"
  ) + theme_bw() +
  theme(panel.grid.minor = element_line(
    color = "grey90",
    linetype = "dashed",
    linewidth = 0.5
  ))
print(p1)
```



- Does the series have a trend?

Yes. It seems that there is a downward (negative) trend to the data, with the mean monthly average hours worked decreasing as a function of time. We would anticipate $m_t < 0$.

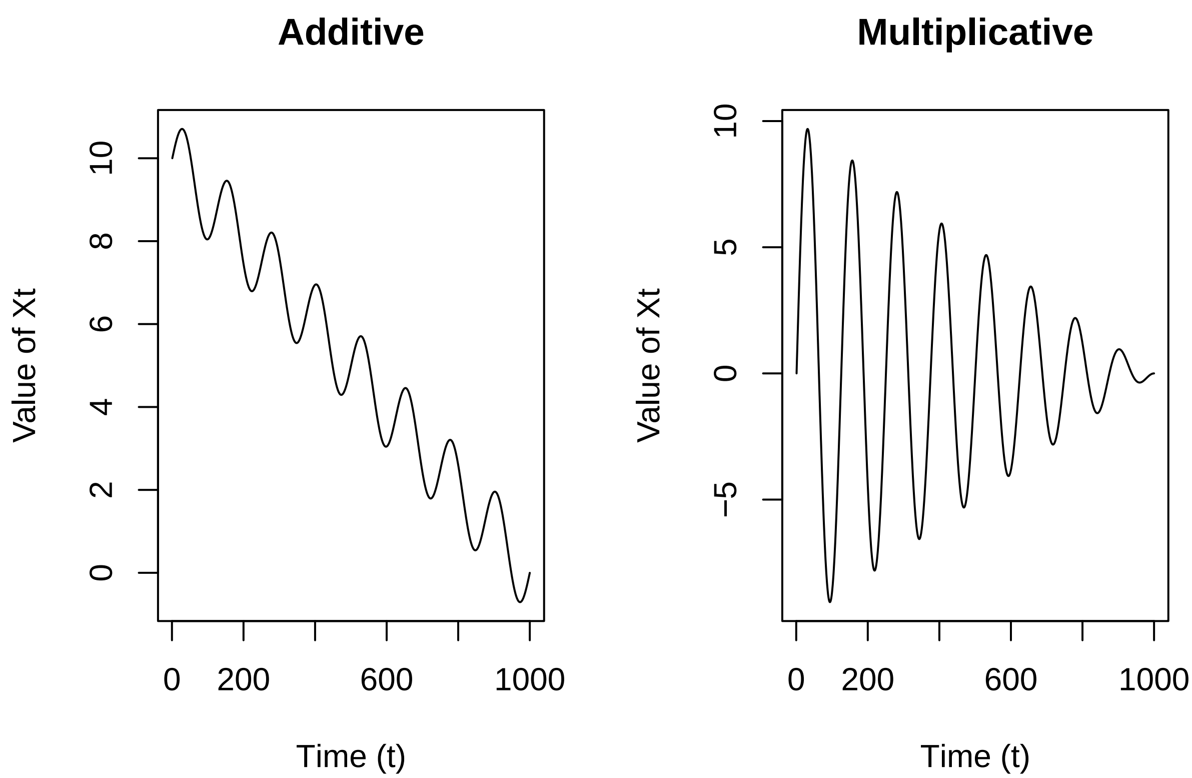
- Is there seasonal variation and if so would an additive or multiplicative model be suitable? Explain your reasoning.

Yes. There appears to be seasonal variation. Visually, we see this as as sinusoidal pattern to the time series. This seasonality could be caused by, for example, months during a given year.

Further, we would anticipate a multiplicative model, i.e. $\{X_t\} = m_t s_t Z_t$. Visually, we can notice this by the changing amplitude of the seasonal periods over time.

To give an example of this, consider the following noise-less toy examples of Additive ($X_t = m_t + s_t$) Model and Multiplicative Model ($X_t = m_t s_t$)

```
st = 2*seq(from = 0, to = 8*pi, length.out = 1000)
mt = seq(from = 10, to = 0, length.out = 1000)
par(mfrow = c(1,2))
plot( mt + sin( st ), type = 'l',
      ylab = "Value of Xt", xlab = "Time (t)",
      main = "Additive")
plot( mt*sin(st), type = 'l',
      ylab = "Value of Xt", xlab = "Time (t)",
      main = "Multiplicative")
```



We see in the additive model, that the seasonal "amplitude" (i.e. the height of each peak/trough) does not change as a function of t, whereas in the multiplicative model the amplitude is changing to due the product with m_t. This is a key delineation of additive vs. multiplicative models. Hence, from our observation of the time series of the data, it is safe to assume that it is likely that the true $\{X_t\}$ takes a multiplicative model.

- Is the series stationary? Justify referring to the definition of a weakly stationary stochastic process.

This series is non-stationary. We can confirm this by the first property of a weakly stationary stochastic process, that $\exists \mu \in \mathbb{R}$ s.t. $\forall t \in \mathbb{N} \cup \{0\}, \mathbb{E}(X_t) = \mu$. By the presence of both seasonality and trend, we know that there cannot exist a real constant μ such that for all discrete time the expected value of the stochastic process is constant μ. This is because, by definition, m_t and s_t are functions of time. Therefore, $\mathbb{E}(t) = f(t)$ for some real-valued function f, and, since $\mathbb{E}(X_t)$ is a function of time, it cannot concurrently be some real-valued constant μ.

In more formal terms, for m_t and s_t being the trend and seasonal components of X_t respectively:

$$\big((\exists m_t \in \mathbb{R}) \vee (\exists s_t \in \mathbb{R})\big) \implies \nexists \mu \in \mathbb{R} \text{ s.t. } \forall t \in \mathbb{N} \cup \{0\}, \mathbb{E}(X_t) = \mu$$

Thus, the existence of either s_t or m_t denies the existence of μ. So, we know by the first property of weakly stochastic processes that this time series is non-stationary.

b) Create training and test datasets.

Part 1: The training dataset should include all observations up to and including December 2021; this dataset will be used to fit ("train") the model. The test dataset should include all observations from January 2022 to December 2023; this dataset will be used to assess forecast accuracy. You can use the command `window()` on a ts object to split the data.

We'll start by splitting the data into train and test, then verifying our work. The verification process involves assuring that the sum of train and test is equal to the size of the series and also equal to the number of rows in the original data.

```
train <- window(jobseries,
           # starting at the beginning of 1987
           start = 1987,
           # ending at the end of 2021
           end = c(2021, 12),
           # monthly
           frequency = 12)

# get test data
test <- window(jobseries,
               start = c(2022, 1),
               end = c(2023, 12),
               frequency = 12)
# verify we've done things correctly
all.equal(length(test) + length(train),
          length(jobseries),
          nrow(df))
```

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

Part 2 Using a suitable decomposition model and the loess method (R function `stl()`) decompose the training series into trend, seasonal, and error components. Plot the resulting decomposition.

We concluded in the previous question that this is likely to be a multiplicative rather than additive model. Hence, the best way to extract each individual component is to take the (natural) logarithm of the multiplicative model before decomposing the series into components. In other words, we will let:

$$\log(X_t) = \log(m_t s_t Z_t) = \log(m_t) + \log(s_t) + \log(Z_t)$$

So that we can consider each component separately. We will use `s.window = "periodic"`.

```
to_additive = log(train)
loess = stl(to_additive, s.window = "periodic")

seasonal = loess$time.series[, 1]
trend = loess$time.series[, 2]
noise = loess$time.series[, 3]
```

Then, we can plot the three components of the decomposition.

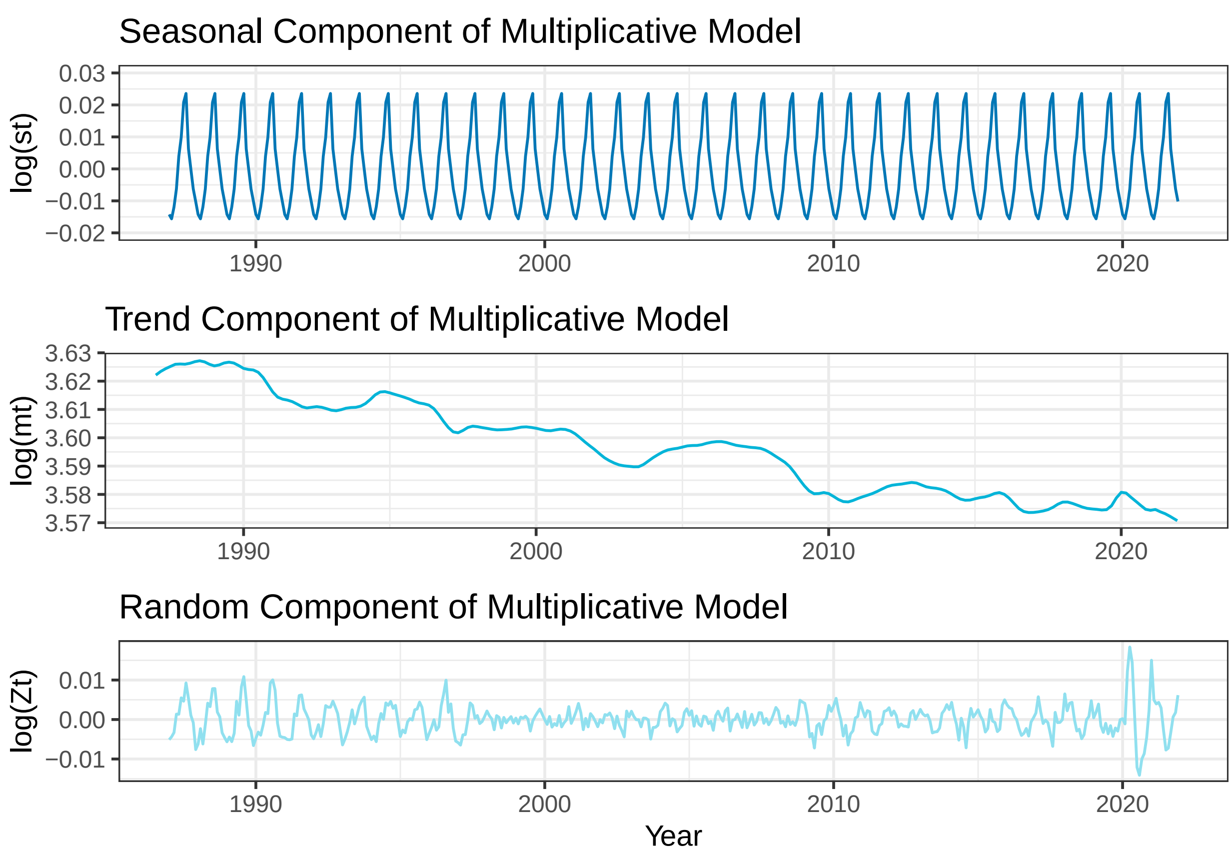
```
sp2 <- ggplot(data = fortify.zoo(seasonal), aes(x = Index, y = seasonal)) +
  geom_line(color = "#0077b6", linewidth = 0.5) +
  theme_bw() +
  labs(
    title = "Seasonal Component of Multiplicative Model",
    y = "log(st)",
    x = NULL
```

```r
  ) +
  ylim(-0.02, 0.03)

mp2 <- ggplot(data = fortify.zoo(trend), aes(x = Index, y = trend)) +
  geom_line(color = "#00b4d8", linewidth = 0.5) +
  theme_bw() +
  labs(
    title = "Trend Component of Multiplicative Model",
    y = "log(mt)",
    x = NULL
  )

zp2 <- ggplot(data = fortify.zoo(noise), aes(x = Index, y = noise)) +
  geom_line(color = "#90e0ef", linewidth = 0.5) +
  theme_bw() +
  labs(
    title = "Random Component of Multiplicative Model",
    y = "log(Zt)",
    x = "Year"
  )

grid.arrange(sp2, mp2, zp2, nrow = 3)
```



c) Fit a linear model to the trend component (you can use R function `lm()`).

- Write down the fitted model for the trend component.

Let $\hat{\beta}_0, \hat{\beta}_1 \in \mathbb{R}$ be the real-valued least squares estimator of the true intercept slope terms $\beta_0, \beta_1 \in \mathbb{R}$ respectively. Recalling that we are taking $\log(m_t)$, the fitted model is of the form.

$$\log(\hat{m}_t) = \hat{\beta}_0 + \hat{\beta}_1 t$$

The coefficients for `(Intercept)` ($\hat{\beta}_0$) and `Time` ($\hat{\beta}_1$) are found below:

```
# cast ts to data frame and rename
trend_df <- fortify.zoo(trend)
colnames(trend_df) = c("Time", "log_hours")
# fit model
trend_mod <- lm(log_hours~Time, data = trend_df)
# report coefficients
data.frame(value = trend_mod$coefficients)
```

```
##                    value
## (Intercept)  6.744901235
## Time        -0.001570879
```

- Does the linear model provide evidence of a trend at the 95% confidence level?

To see if the linear model provides evidence of a trend component at the 95% confidence level, we would test the following pair of hypotheses at $\alpha = 0.05$:

$$H_0 : \hat{\beta}_1 = 0 \quad \text{against} \quad H_A : \hat{\beta}_1 \neq 0$$

In this instance, we would use the following test statistic:

$$T_{\text{obs}} = \frac{\hat{\beta}_1}{\text{se}(\hat{\beta}_1)} \sim t_{n-(k+1)}$$

Which takes a Student's t-distribution on $n - (k + 1) = 418$ degrees of freedom.

```
trend_summary = summary(trend_mod)

# get the standard error and estimated coefficients
se = trend_summary$coefficients[2, "Std. Error"]
est = trend_summary$coefficients[2, "Estimate"]

# calculate observed t where hypothesized value is zero
tobs = (est - 0)/se

# report
print(paste("Our observed test statistic is approximately",
            round(tobs, 2)))
```

```
## [1] "Our observed test statistic is approximately -77.81"
```

Then, we would calculate the p-value for the two-tailed test by finding:

$$p = 2\ \min\big\{P(t_{n-(k+1)} > T_{\text{obs}}), P(t_{n-(k+1)} < T_{\text{obs}})\big\}$$

Which, in this case, is found as follows:

```r
n = nrow(trend_df); k = 1

2 * min( pt( tobs, df = n - (k + 1), lower.tail = TRUE ),
         pt( tobs, df = n - (k + 1), lower.tail = TRUE ))
```

```
## [1] 8.217246e-251
```

Since our observed p-value of approximately $8.217 \times 10^{-251} \approx 0$ is less than our declared $\alpha = 0.05$, we would reject H_0 at the 95% level. There is statistically significant evidence to suggest that the $\hat{\beta}_1$ coefficient, the slope of the $\log(\hat{m}_t)$ model, is nonzero. Therefore, we conclude that the linear model provides evidence of a trend at the 95% confidence level.

- Without doing any further analysis, would you use this trend component to make predictions? Justify your answer using the linear model results and the trend component plot.

From the trend component plot, it appears that the trend m_t is non-linear as a function of time. Therefore, I would anticipate a pattern in the residual plot for the linear model $\log(\hat{m}_t)$. Despite the fact that the linear model was significant (see: previous question), it is very likely that there exists a better model (polynomial, etc.) to describe the relationship between $\log(\hat{m}_t)$ and t. If I were to make predictions, I would first try some other OLS models of different styles and, using cross-validation or otherwise, pick a method better than the "purely linear" approach taken here. However, despite the fact that it may not be the case that a linear model best suits $\log(\hat{m}_t)$, for the sake of prediction it is better to use this trend component rather than nothing at all. In short, it's not the best model for $\log(\hat{m}_t)$, but it is better than nothing. I would prefer this trend component for predictions rather than ignoring trend altogether; however, it is probable that further analysis would find that a much better model exists for the trend component than this one.

d) Predict the monthly average values of the usual hours worked in Canada for the period from January 2022 to December 2023 using your seasonal decomposition model.

- Plot your predictions along with the actual observed values (on the same plot). Make sure to include a legend for your plot.

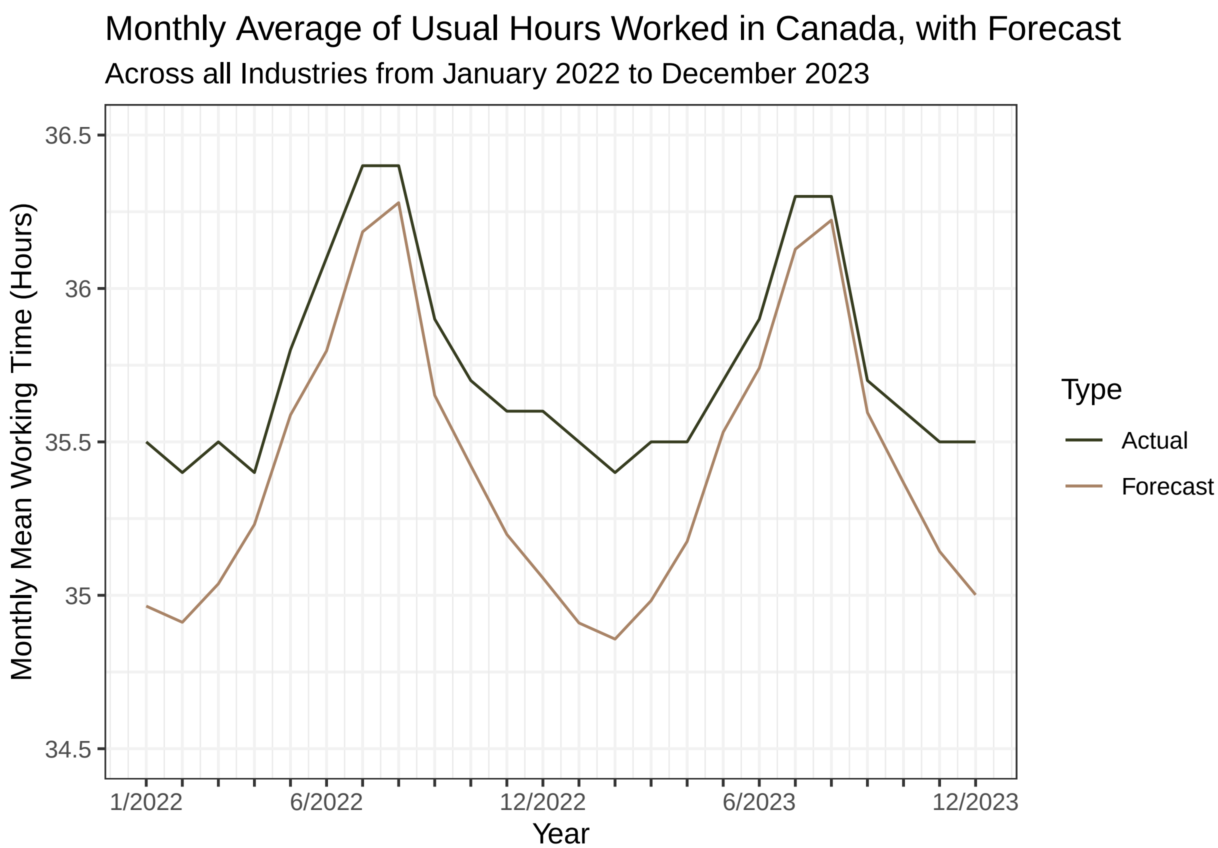
We'll use the linear model from the previous question.

```r
# extract model coefficients
beta_0 = trend_summary$coefficients[1, "Estimate"]
beta_1 = trend_summary$coefficients[2, "Estimate"]
# unique terms in the seasonal component give one period
period = unique(seasonal)
# use our linear model with time variable
mt_pred = beta_1*time(test) + beta_0
# we're predicting two periods into the future
st_pred = (rep(period, 2))
# then the predicted value is undoing the log transform
preds = exp(mt_pred + st_pred)
p3df <- data.frame(
```

```
          Time = as.numeric(time(test)),
          Actual = as.numeric(test),
          Predicted = preds)
# create custom-spaced month/date strings
date_strings <- unlist(lapply(2022:2023, function(year) {
  sapply(1:12, function(month) {
    if (month %% 6 == 0) paste(month, "/", year, sep = "")
    else ""
  })
}))
# specify additional parameters
date_strings[1] = "1/2022"
low_ylim = 34.5; up_ylim = 36.5
```

Now, we create a plot of the forecast vs. actual values.

```
p3 <- ggplot(p3df) +
  geom_line(aes(x = Time, y = Actual, color = "Actual")) +
  geom_line(aes(x = Time, y = Predicted, color = "Forecast")) +
  scale_color_manual(values = c("Actual" = "#373d20", "Forecast" = "#a98467"),
                     name = "Type", labels = c("Actual", "Forecast")) +
  labs(
    title = "Monthly Average of Usual Hours Worked in Canada, with Forecast",
    subtitle = "Across all Industries from January 2022 to December 2023",
    y = "Monthly Mean Working Time (Hours)",
    x = "Year"
  ) + theme_bw() +
  scale_x_continuous(breaks = p3df$Time, labels = date_strings) +
  scale_y_continuous(limits = c(low_ylim, up_ylim),
                     breaks = seq(low_ylim, up_ylim, by = 0.5),
                     labels = seq(low_ylim, up_ylim, by = 0.5)) +
  theme(panel.grid.major = element_line(
    color = "grey95",
    linetype = "solid",
    linewidth = 0.5
  ),
  panel.grid.minor.y = element_line(
    color = "grey95",
    linetype = "solid",
    linewidth = 0.5
  ))
print(p3)
```

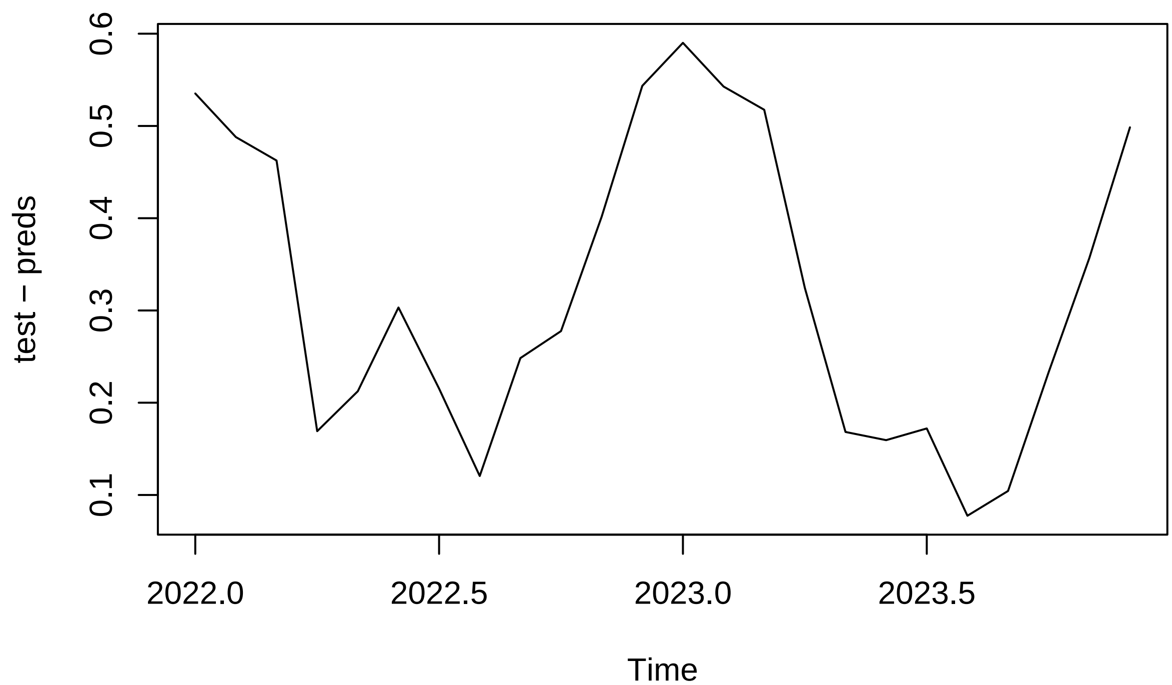


- Comment on the performance of your prediction method, explaining why or why not the method worked well for this data.

The prediction method seemed to perform decently well; specifically, in the capturing of the seasonal component of the time series. We see this fairly accurate performance of predicted seasons by the roughly-matching peak locations between the predicted values and the actual values in the test set.

However, there is certainly room for improvement. Consider the residual plot below:

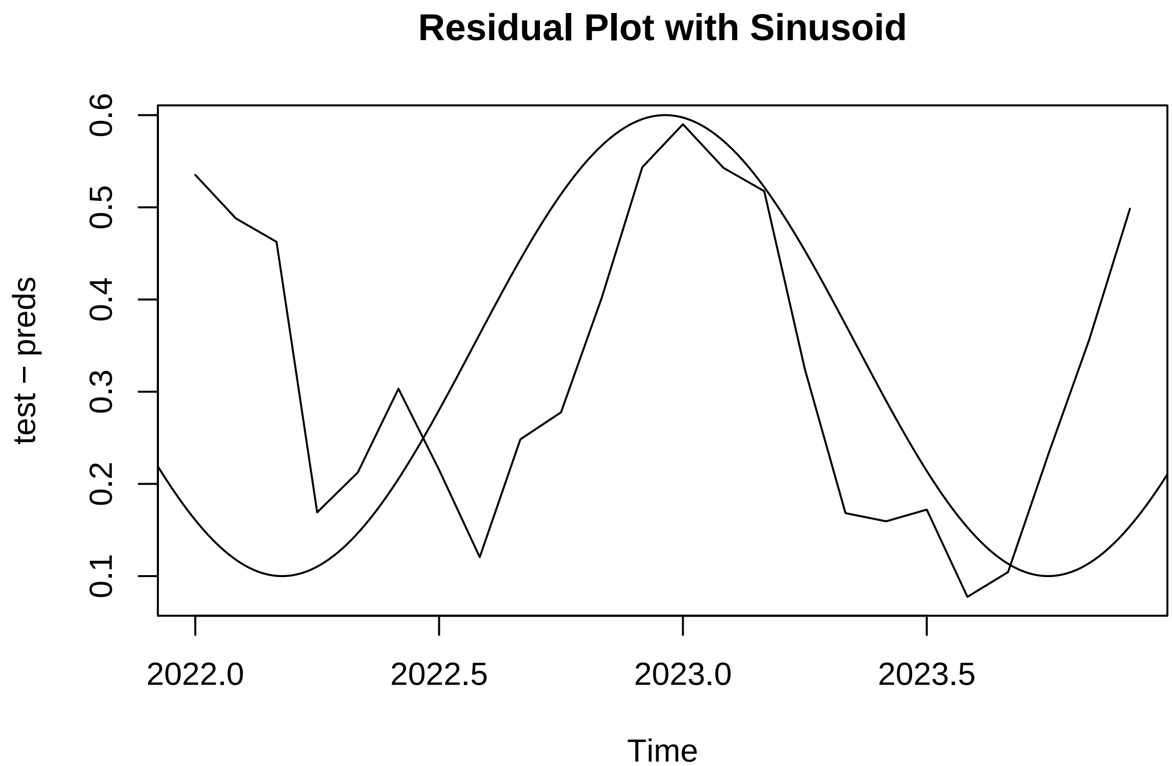
```
plot(test - preds)
```



The residual plot shows a clear pattern, and the fact that the residuals are wholly positive implies that our model is consistently under-estimating the truth. The pattern to the residual plot is quite interesting, and may suggest that part of the reason the predictions are off is that there is some additional pattern in the data that we are not capturing with our current model.

This can be exemplified in the plot below, where we are roughly able to fit a sinusoid to the residuals. (*NOTE*: this is purely to exemplify a pattern in the residuals, and is not calculated through OLS or otherwise.)

```
xtest = seq(from = 0, to = pi, length.out = 1000 )
plot(test - preds, main = "Residual Plot with Sinusoid")
lines(xtest  + 2021, 0.25*sin(4*xtest)+.35)
```



It's difficult to say exactly what this "additional component" could be; however, there could be some secondary seasonal component (i.e. multiple-seasonality) that we are not detecting, or, as was discussed earlier, perhaps a purely linear model does not suit the trend component well.

- How could the prediction method be improved?

```
went decently well, because I think that our methodology captured the seasonal component
quite well (see: peaks in the right location), however a better estimation of the trend
component could have been very beneficial
```

```
... attempt nonlinear functions of $\log(\hat{m}_t)$, use cross-validation to fit the
model,
```

```
... inspect the possibility of Complex /multiple seasonality
```

- As a statistician, what other information would you like to add to your forecasts in addition to the point forecasts you produced above?

```
... confidence bounds / prediction error
```

Task 2: Analyzing New York Temperature Data

a) Read the data into R and create an R object called dat for the data.

```r
# Code block for Task 2a
```

b) Create zoo objects for daily Max Temperature. Create monthly maxima time series. Plot the monthly maximum temperature series and comment on any features you observe.

```r
# Code block for Task 2b
```

c) Fit a suitable seasonal decomposition model to the monthly data using the moving average smoothing (R function decompose) and plot the estimates of the trend, seasonal, and error components.

```r
# Code block for Task 2c
```

d) Plot the correlogram for the deseasonalized series of monthly temperature maxima using the seasonal decomposition model you fit in part (c). Comment on the serial dependence of this series.

```r
# Code block for Task 2d
```

Task 3: Conducting a Simulation Study on the Autocorrelation Coefficient

i)

Simulate a time series of length $n = 2000$ from a white noise process with $Z_t \sim N(0, 1)$ (function `rnorm()`).

```r
# Code block for Task 3i
```

ii) Evaluate the sample autocorrelation coefficient.

At lag h for $h = 1$ and $h = 2$. Store these values.

```r
# Code block for Task 3ii
```

iii) Repeat steps (i) and (ii) m = 8000 times;

i.e. generate 8000 time series of length n and for each of them compute r1 and r2. You should now have two vectors of length m with estimates r1 and r2.

```r
# Code block for Task 3iii
```

To summarize the results of the simulation study:

- Compute the mean and variance of r1 and r2 values from your simulation study.
- In two separate figures plot the two histograms for the sample of r1 and r2 values from the simulation study (function hist()) add the smoothed version of the histogram (function density()) and the theoretical asymptotic normal density (function dnorm()). Make sure your plots are well-presented including a suitable title, axes labels, curves of different type or colour, and a legend.
- Comment whether there is an agreement between the empirical estimates of the bias, variance, and sampling density of the estimator of the autocorrelation at lag h and their theoretical approximation.

```
# Code block for summarizing the simulation study results
```