Statistical Methods for Discrete Response, Time Series, and Panel Data (W271): Lab 2

Nathan Martinez, Haile Bizunehe, Hannah George, and Meera Sharma

2022-11-01

Contents

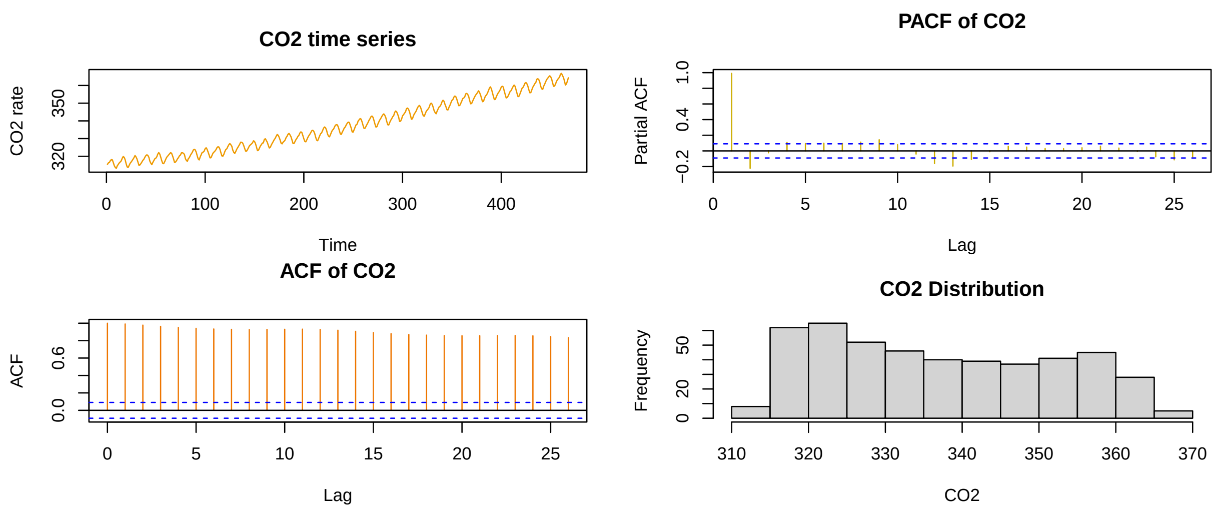
1 Report from the Point of View of 1997

1.1 (3 points) Task 0a: Introduction

Global temperatures are rising at a steady and alarming rate. This phenomenon is commonly referred to as climate change and it has been linked to anthropogenic emissions - greenhouse gas emissions associated with human activities like deforestation and the burning of fossil fuels. The climate change phenomenon has been well documented by notable researchers, including Charles David Keeling who is known for creating the Keeling Curve, a visualization that shows the change in the concentration of carbon dioxide in the Earth's atmosphere from 1959 to the present. We will use the same data that underpins the Keeling Curve to answer the following research questions: 1) How have carbon emissions increased over time? 2) What is the expected level of carbon emissions? To answer these research questions, we will apply time-series analysis to atmospheric C02 concentration data collected from Mauna Loa, Hawaii.

Our team will construct a time-series model that shows how carbon emissions have increased over the past 38 years. More importantly, we will predict how much we should expect carbon emissions to rise if nothing is done to reduce anthropogenic emissions. These findings will be important because increasing carbon emissions have the potential to threaten our way of life, resulting in higher temperatures, rising sea levels, and increased frequency of extreme weather events, among other threats. We are hoping these findings will be used to develop company strategies and actions that will reduce our carbon footprint for the betterment of our planet. We also hope these findings will be used to encourage people at our company to take measures that will reduce their carbon emission contributions.

1.2 (3 points) Task 1a: CO2 data



The top-left plot shows the timeseries over the period Jan 1959 through Jan 1997. A linear trend and annual seasonality are evident in the plot.

The top-right plot shows the ACF (autocorrelation function) of the series at lags of 1 month. The correlogram shows both a gradual decay, which is a sign of a existance of a trend, and annual seasonality, shown by the spikes occuring every 12 months.

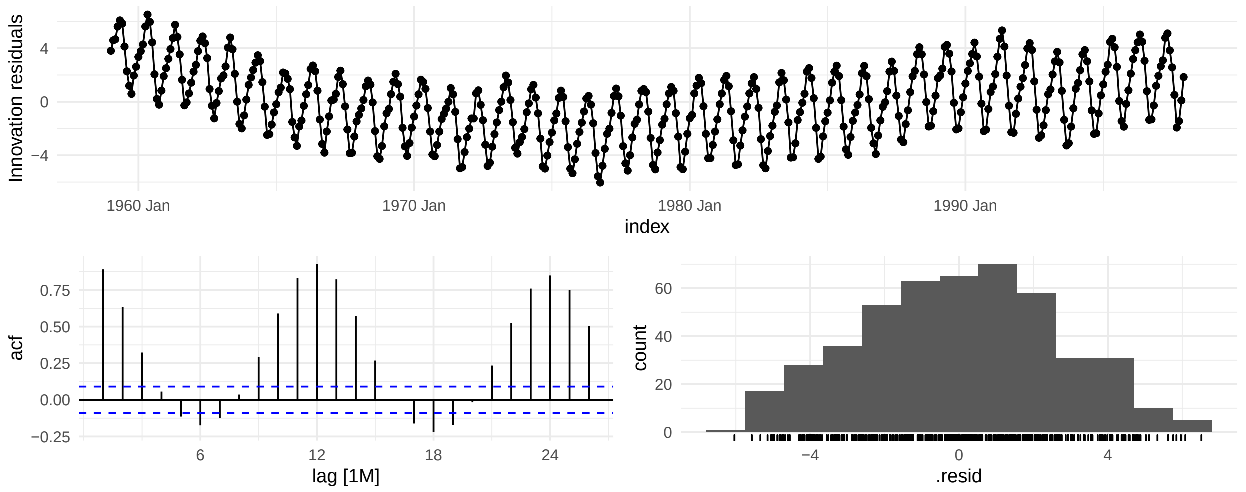
The bottom-left plot shows the PACF (partial autocorrelation function) of the series. The PACF also suggests some seasonality, due to the oscillating significant spikes at various lags. The PACF shows lags 1, 2, 12, and 13 to be significant. The high significance at different lag on the PACF might suggest that it is an ARMA process.

Lastly, the bottom-right plot shows the histogram of the data and helps us understand its spread. The distribution has close to a uniform distribution, which is not ideal for asympototic confidence interval estimation (requires normality in the distribution of residuals).

1.3 (3 points) Task 2a: Linear time trend model

```
as_tsibble(co2) -> co2_ts

co2_ts %>%
  model(TSLM(value ~ trend())) -> fit_co2_trend

## please uncomment for details. Commenting for now for brevity.
# report(fit_co2_trend)
```

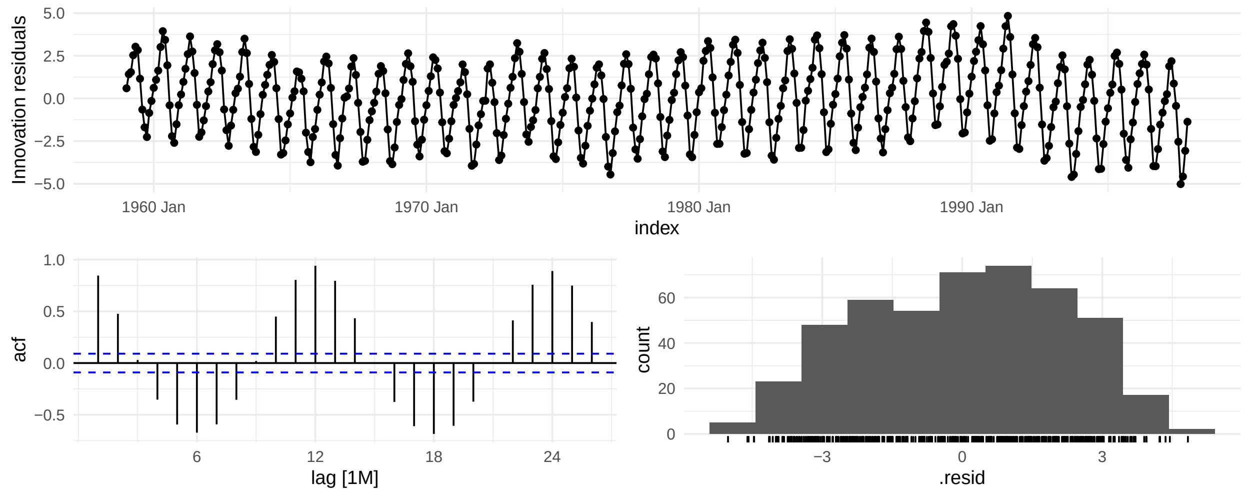
The linear time trend models estimates an increase of 0.11 in CO2 levels per month. The model residuals are shown below.



The timeseries in the top panel show the seasonality in the residuals, meaning we need to adjust the model for seasonal effects. Additionally, the U-shape of the residuals against the time index suggests non-linearity in the underlying process: a polynomial trend model may be need to capture this shape. The seasonality in the data is reflected in the ACF, which shows oscillating significant lags. The histogram of the residuals is normally distributed.

Note that we have not shown the Dickey-Fuller test here since the plot shows the series as being non-stationary. We have used the test later in the report when the plots are not very obvious in display stationarity in the series.

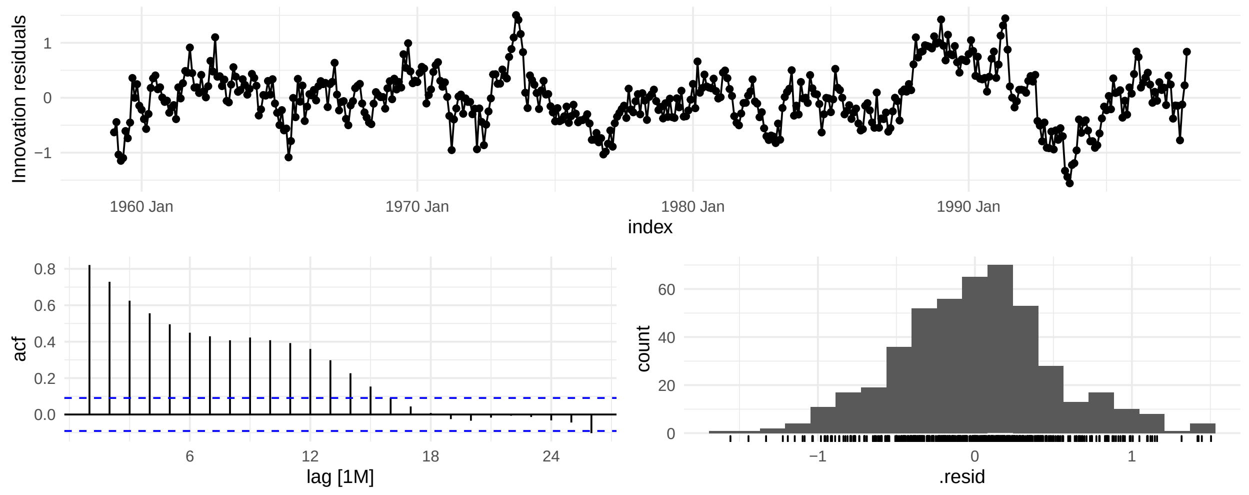
```
co2_ts %>%
  model(TSLM(value ~ trend()+ I(trend()^2))) -> fit_co2_quad_trend

## please uncomment for details. Commenting for now for brevity.
# report(fit_co2_quad_trend)
```



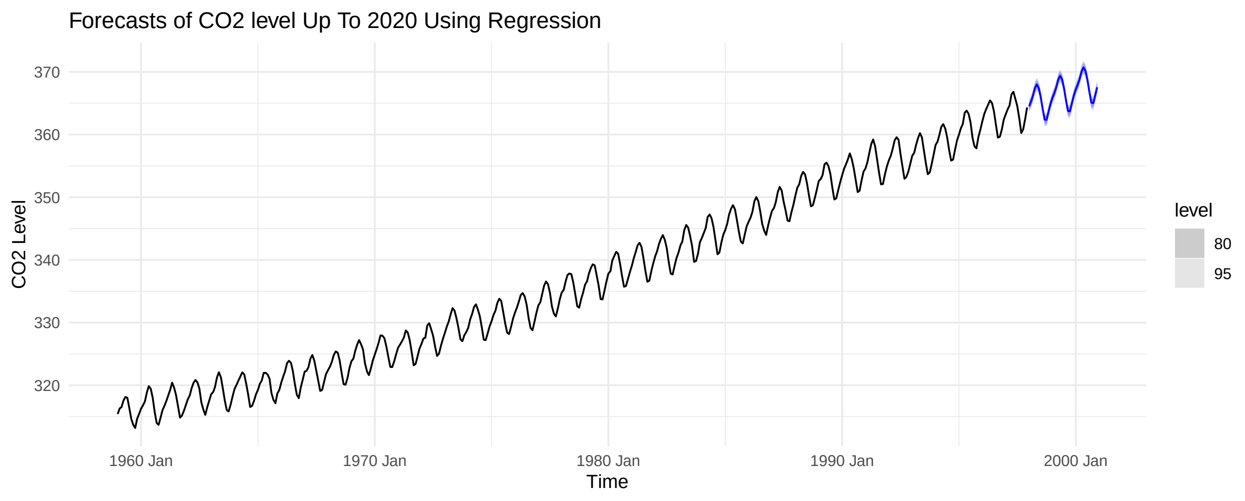
The quadratic model is able to capture some of the non-linearity in the data, reducing the exaggerated U-shape in the residuals in the top panel. The trend is not completely modeled and we will require a polynomial term to capture the trend. The corresponding ACF still shows seasonal effects in the data. The histogram of the residuals is normal.

The variance in the data is not increasing at higher values, meaning we do not need a log-transformation.

```
co2_ts %>%
  model(TSLM(value ~ trend() + I(trend()^2) + I(trend()^3) + season())) -> fit_co2_full_trend

## please uncomment for details. Commenting for now for brevity.
#report(fit_co2_full_trend)
```

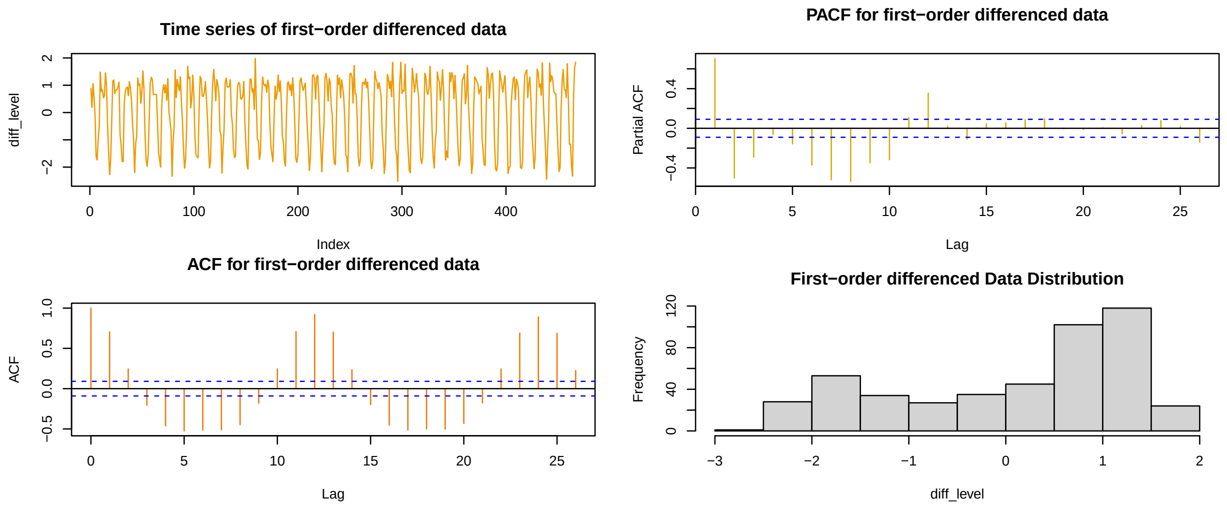


Using seasonality and polynomial time trend, we are able to remove the seasonality and non-linearity from the residuals: note the ACF does not show oscillating significant lags. However, we still have a case of heteroskedasticity in the residuals, shown by the non-constant variance in the top panel. The histogram shows normally distributed residuals.

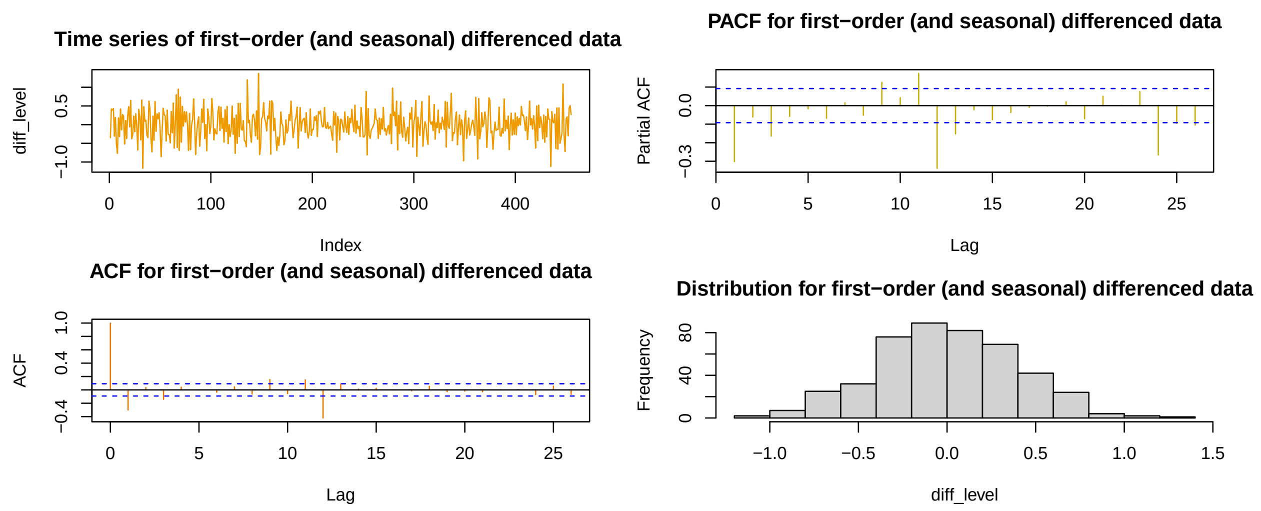


The seasonality and polynomial time trend model produces forecasts that extrapolate the linear trend of the series and include the seasonality of the data. However, as noted in the sections above, the residuals of this model do not represent white noise. Hence, a better model may be possible for this data.

1.4 (3 points) Task 3a: ARIMA times series model



'In Task 1a, we determined that the time series is not stationary. First-order differencing, performed above, removes the non-seasonal trend from the series. However, the plots above show seasonality: the ACF shows oscillating correlations every 12 months, implying seasonality. To create a stationary timeseries, we need to add in seasonal differencing:



```
tseries::adf.test(diff(diff(co2_ts$value), lag=12),
         alternative = "stationary", k = 3)
```

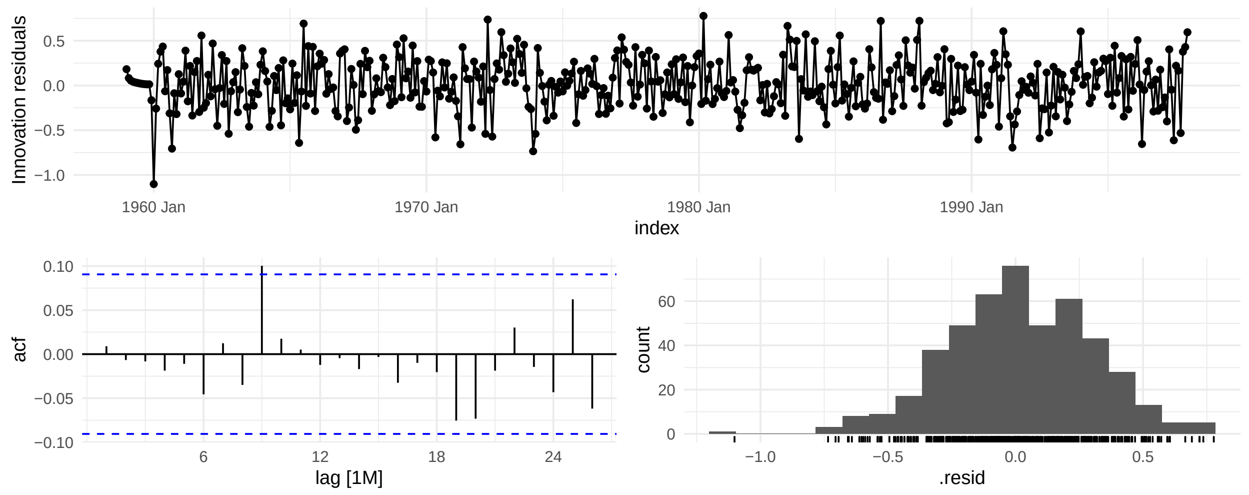
```
## Warning in tseries::adf.test(diff(diff(co2_ts$value), lag = 12), alternative =
## "stationary", : p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(co2_ts$value), lag = 12)
## Dickey-Fuller = -13.69, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

‘The Dickey-Fuller test, for the first-order differenced (seasonal as well as non-seasonal) series, rejects the null hypothesis (which states that the time series is not stationary). The time series plot (upper-left panel) appears to be stationary as well. Significant correlations at different lags on both ACF and PACF suggest that the process is ARMA. The differenced-data distribution is normal; this will help in constructing asymptotic confidence intervals. Next, we will search for a model that minimizes AICc after applying first-order seasonal and non-seasonal differencing.’

```
model.aicc <- co2_ts %>%
  model(ARIMA(value ~ 0 + pdq(0:10, 1, 0:10) + PDQ(0:10, 1, 0:10),
                              ic="aicc", stepwise=F, greedy=F))
model.aicc %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,3)(1,1,2)[12]
##
## Coefficients:
##           ma1      ma2      ma3     sar1     sma1     sma2
##       -0.3363  -0.0207  -0.1015  -0.4887  -0.3200  -0.4668
## s.e.   0.0476   0.0496   0.0468   0.4906   0.4755   0.4026
##
## sigma^2 estimated as 0.08531:  log likelihood=-82.7
## AIC=179.41   AICc=179.66   BIC=208.25
```



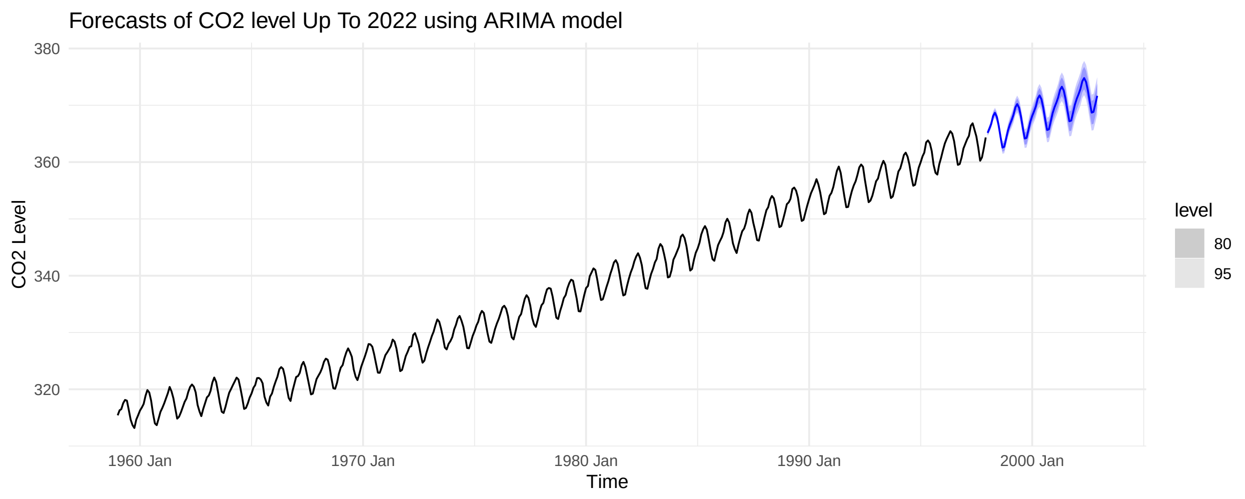
```
resid.ts <- model.aicc %>%
  augment() %>%
  dplyr::select(.resid) %>%
  as.ts()

Box.test(resid.ts, lag = 1, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.ts
## X-squared = 0.03798, df = 1, p-value = 0.8455
```

```
Box.test(resid.ts, lag = 10, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.ts
## X-squared = 6.9372, df = 10, p-value = 0.7314
```

'After performing first and seasonal differencing, we searched for the number of AR and MA terms for both seasonal and non-seasonal parts of the SARIMA model that minimizes AICc. We found a model with three non-seasonal MA terms, one seasonal AR term, and two seasonal MA terms that can better fit the data. We also examined the residuals, which show no systematic relationships and it has only one minor significant correlation. The Box-Ljung test also shows that the null hypothesis, which is that data are independently distributed, is not rejected. So we can say that the residuals are white noise.'



'The ARIMA does a reasonable job of predicting the trend and seasonality. Comparing it with the actual observed values will be helpful in evaluating if the model is useful. Compared to the linear regression forecast, the ARIMA model has a wider confidence interval.'

1.5 (3 points) Task 4a: Forecast atmospheric CO2 growth

```
fc_100_years <- fabletools::forecast(model.aicc, h=1236) %>%
  hilo() %>%
  unpack_hilo(c(`80%`, `95%`))

fc_100_years %>%
  filter(.mean >= 420 & .mean < 421) -> fc_420

fc_100_years %>%
  filter(.mean >= 500 & .mean < 501) -> fc_500

# 420 PPM levels for the first and last time
fc_420_first <- head(fc_420, n =1)
fc_420_last  <- tail(fc_420, n =1)

fc_420_first_time  <- fc_420_first$index
fc_420_first_lower <- round(fc_420_first$`95%_lower`, 2)
fc_420_first_upper <- round(fc_420_first$`95%_upper`, 2)
fc_420_first_mean  <- round(fc_420_first$.mean, 2)

fc_420_last_time  <- fc_420_last$index
fc_420_last_lower <- round(fc_420_last$`95%_lower`, 2)
fc_420_last_upper <- round(fc_420_last$`95%_upper`, 2)
fc_420_last_mean  <- round(fc_420_last$.mean, 2)

# 500 PPM levels for the first and last time
fc_500_first <- head(fc_500, n =1)
fc_500_last  <- tail(fc_500, n =1)

fc_500_first_time  <- fc_500_first$index
fc_500_first_lower <- round(fc_500_first$`95%_lower`, 2)
```

```r
fc_500_first_upper <- round(fc_500_first$`95%_upper`, 2)
fc_500_first_mean  <- round(fc_500_first$.mean, 2)

fc_500_last_time  <- fc_500_last$index
fc_500_last_lower <- round(fc_500_last$`95%_lower`, 2)
fc_500_last_upper <- round(fc_500_last$`95%_upper`, 2)
fc_500_last_mean  <- round(fc_500_last$.mean, 2)

# Atmospheric CO2 levels in the year 2100
fc_2100  <- tail(fc_100_years, n =1)

fc_2100_lower <- round(fc_2100$`95%_lower`, 2)
fc_2100_upper <- round(fc_2100$`95%_upper`, 2)
fc_2100_mean  <- round(fc_2100$.mean, 2)
```

'CO2 is expected to be at 420 ppm level for the first time on 2032 Apr with expected value of 420.08 and 396.7 - 443.46 95% confidence interval.'

'CO2 is expected to be at 420 ppm level for the last time on 2036 Oct with expected value of 420.79 and 393.46 - 448.13 95% confidence interval.'

'CO2 is expected to be at 500 ppm level for the first time on 2084 May with expected value of 500.12 and 420.24 - 580 95% confidence interval.'

'CO2 is expected to be at 500 ppm level for the last time on 2088 Oct with expected value of 500.26 and 414.64 - 585.89 95% confidence interval.'

'By the end of 2100 year, the CO2 is expected to be at 521.44 ppm level with 419.32 - 623.55 95% confidence interval.'

2 Report from the Point of View of the Present

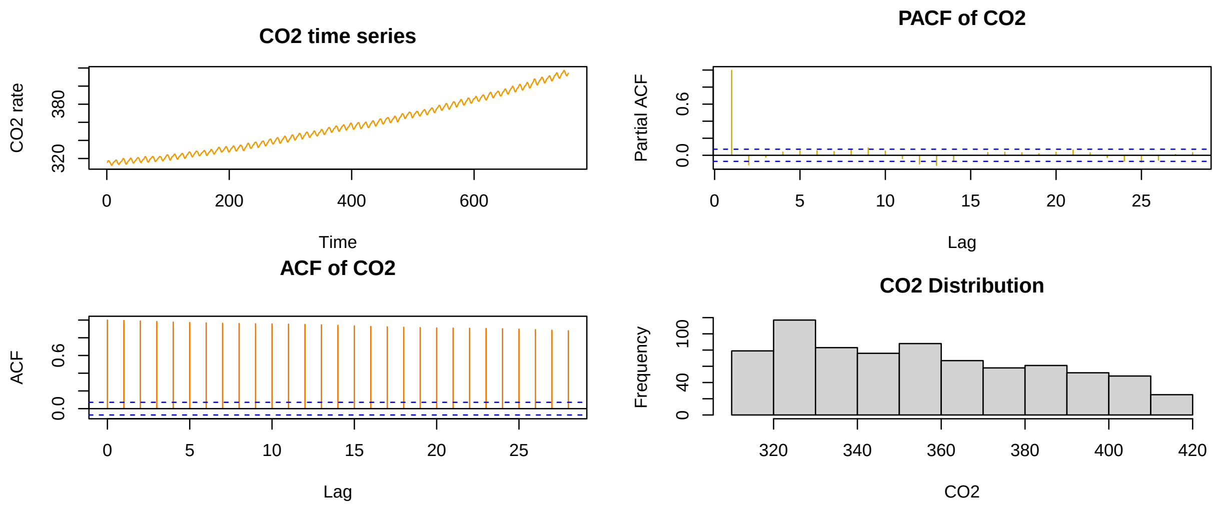
2.1 (1 point) Task 0b: Introduction

2.2 (3 points) Task 1b: Create a modern data pipeline for Mona Loa CO2 data.

Below is our code to read in the data from the appropriate URL and perform minor transformations in order to get the data into a proper time series object.

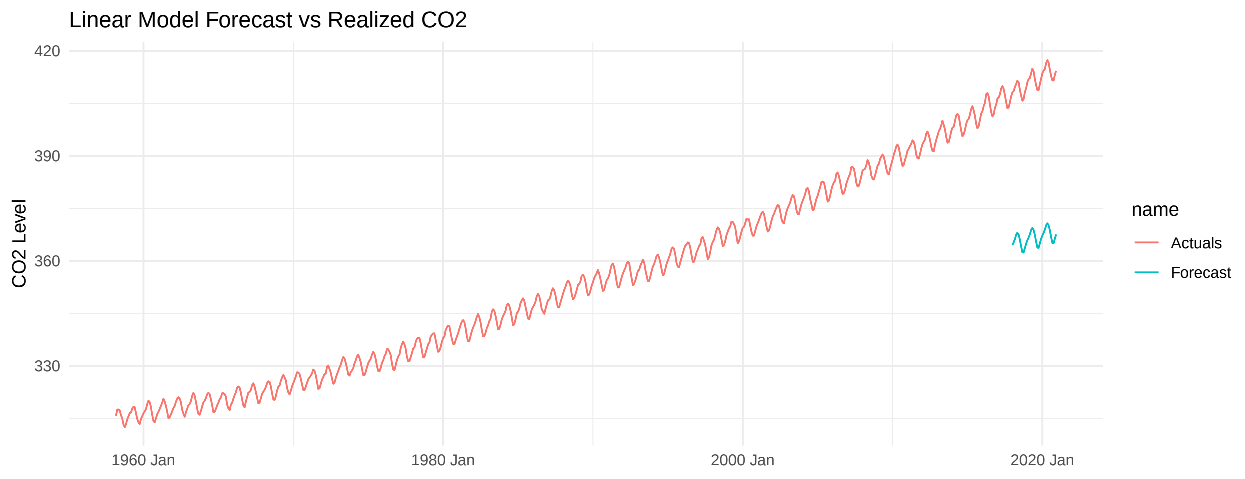
```r
read.csv(
  url("https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_mm_mlo.csv"),
  skip = 52) %>%
  mutate(time_index = make_datetime(year, month)) %>%
  mutate(time_index = yearmonth(time_index)) %>%
  filter(year < 2021) %>%
  as_tsibble(index = time_index) -> co2_present
```

Now we will start our EDA of the present data.



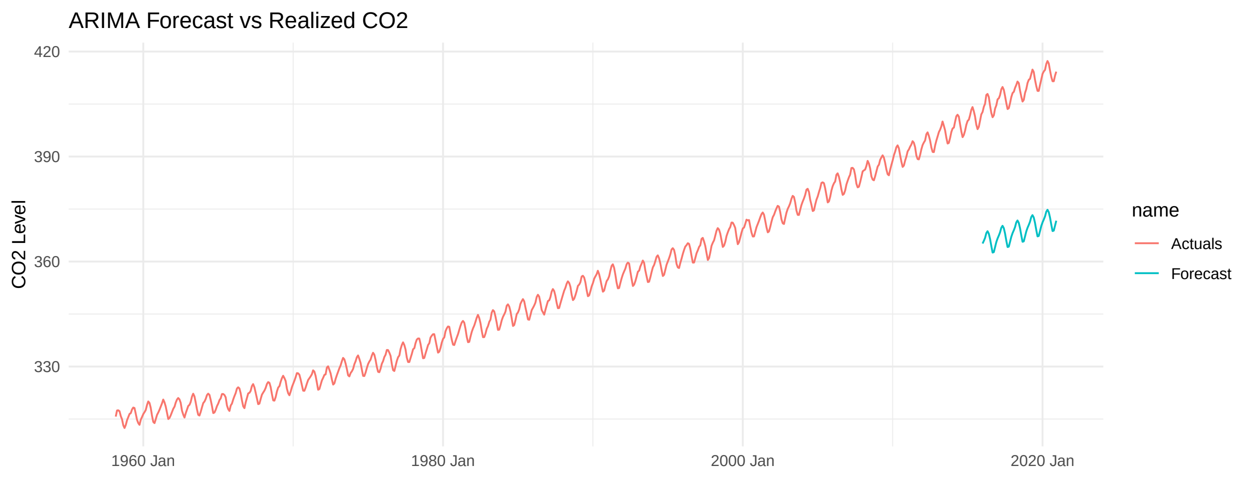
When looking at the present data, there is not a huge difference between how it looks now versus how it looked in 1997. The time series plot shows that the CO2 levels continued to grow - at perhaps a stepper level of growth than before. The most notable difference is the CO2 Distribution histogram. In 1997 the distribution appeared to be almost bimodal, whereas now the distribution looks more heavy-tailed, and extends to much higher levels than it did previously.

2.3 (1 point) Task 2b: Compare linear model forecasts against realized CO2



The linear model forecast did not correctly capture the trend of the realized CO2 levels. The forecast appears to predict a stabilization in the CO2 levels, whereas the actual CO2 level trend increased.

2.4 (1 point) Task 3b: Compare ARIMA models forecasts against realized CO2



The ARIMA forecast is much closer to the realized CO2 levels than the Linear Model forecast. The only difference is that the ARIMA model appears to have forecasted a linear trend, while the realized CO2 levels followed an almost exponential growth.

2.5 (3 points) Task 4b: Evaluate the performance of 1997 linear and ARIMA models

---

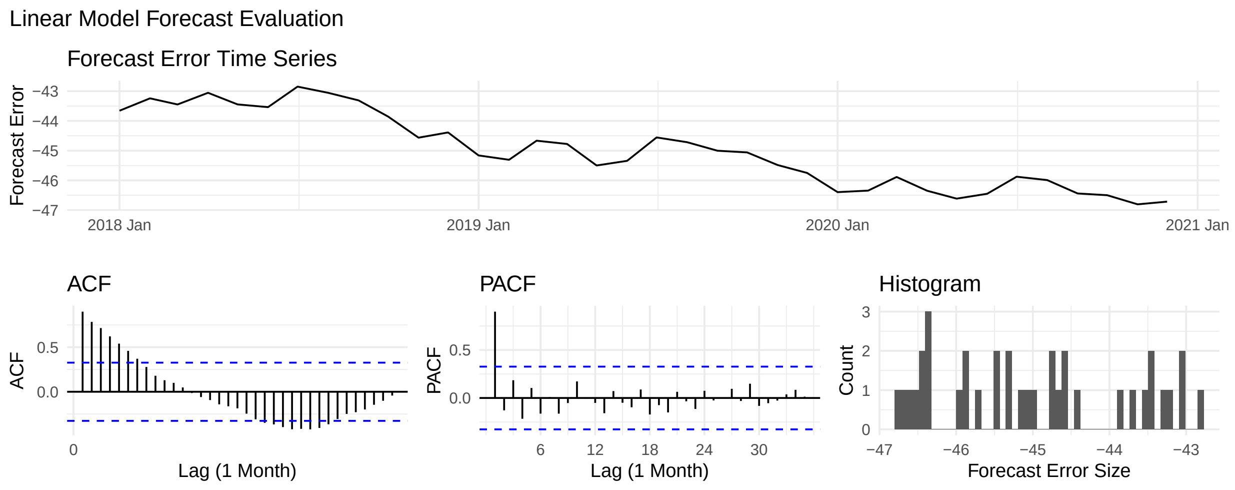
2.5.1 Evaluating 1997 Predictions

```r
# Max actual data from 1958 to 2020
max_actual_row <- co2_present[which.max(co2_present$average),]
max_actual_date <- max_actual_row$time_index
max_actual_value <- max_actual_row$average

# Predicted values and date over the max actual
fc_100_years %>%
  filter(.mean >= max_actual_value) -> fc_417
fc_417_first <- head(fc_417, n =1)
fc_417_first_value <- fc_417_first$.mean
fc_417_first_time  <- fc_417_first$index

# Precited values on actual maximum date
fc_100_years %>%
  filter(index == max_actual_date) -> max_date_pred_values
pred_date_average <- round(max_date_pred_values$.mean, 2)
pred_date_lower <- round(max_date_pred_values$`95%_lower`, 2)
pred_date_upper <- round(max_date_pred_values$`95%_upper`, 2)

# Difference between predicted and actual on max date
pred_act_diff <- max_actual_value - pred_date_average
```

We originally predicted that CO2 would cross the 420 ppm threshold for the first time on 2032 Apr with an expected value of 420.08 with a 95% confidence interval between 396.7 - 443.46. The maximum average monthly value to date is 417.31 on 2020 May. In our predictions, the nearest date with a value at or over 417.31 is 2030 May. Therefore, our predicted average CO2 values were approximately 10-years behind the actuals. When comparing our predicted values in 2020 May to the actual data, we observe an average predicted value of 402.31 with a 95% confidence interval between 388.36 - 416.27. Although our predicted average value was lower than the actual by value by 15 ppm, the actual value of 417.31 is between our 95% confidence interval 388.36 - 416.27.

2.5.2 Evaluating Linear Model Forecast

Now we will evaluate the accuracy of the Linear Model forecast created in Task 2a.



The time series plot of the forecast errors steadily increases in magnitude as the forecast lead time increases. Based on the negative values, we can tell that the linear model under forecasted. This is also clear in the 2b plot. The ACF plot shows significant autocorrelation of forecast errors for multiple time periods, demonstrating a lack of the forecast capturing specific elements of the realized CO2 levels time series. The histogram is long-tailed in the negative errors, again, demonstrating that the Linear Model under-forecasted.

Now, we will take a look at some of the forecast accuracy metrics.

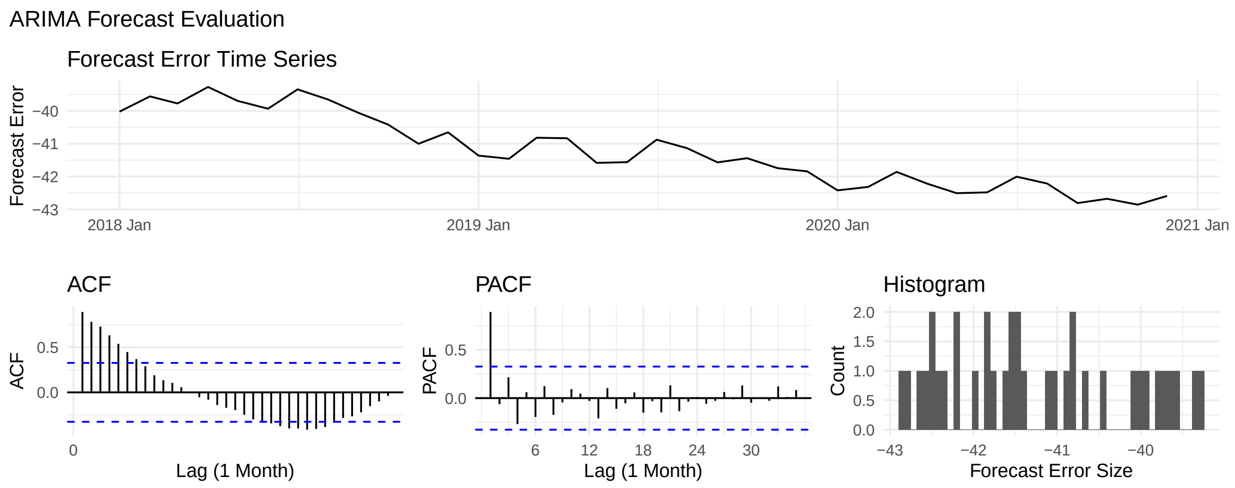
```r
train_data <- co2_present %>% filter(year < 1998)
test_data <- co2_present %>% filter(year >= 1998)

# Build the same model as before.
train_data %>%
  model(TSLM(average ~ trend() + I(trend()^2) + I(trend()^3) + season())) -> lm_present_fit

lm_present_forecast <- fabletools::forecast(lm_present_fit, h=276)
fabletools::accuracy(lm_present_forecast, test_data)
```

```
## # A tibble: 1 x 10
##   .model                   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 TSLM(average ~ trend() ~ Test   13.9  17.3  13.9  3.49  3.49   NaN   NaN 0.988
```

Above, we can see various accuracy metrics that the Linear Model forecast was evaluated on. In particular, the RMSE value is 17.26 - this value makes sense given the error histogram maxed out at around -30. The MAPE stands at around 349% which is not a great value for such an clean time series.

2.5.3 Evaluating ARIMA Forecast

Now we will evaluate the accuracy of the ARIMA forecast created in Task 3a.



The time series plot of the forecast errors steadily increases in magnitude as the forecast lead time increases, however, the forecast error values are not as large as they are from the Linear Model forecast. The ARIMA model also under-forecasted based on the forecast error time series and the histogram. The ACF plot shows significant autocorrelation of forecast errors for multiple time periods, demonstrating a lack of the forecast capturing specific elements of the realized CO2 levels time series.

Now, we will take a look at some of the forecast accuracy metrics.

```
# Build the same model as before.
train_data %>%
  model(ARIMA(average ~ 0 + pdq(0, 1, 1) + PDQ(1, 1, 2))) -> arima_present_fit

arima_present_forecast <- fabletools::forecast(arima_present_fit, h=276)
fabletools::accuracy(arima_present_forecast, test_data)
```

```
## # A tibble: 1 x 10
##   .model                   .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(average ~ 0 + pdq~ Test   6.01  7.38  6.02  1.51  1.51   NaN   NaN 0.985
```

The error metrics for the ARIMA model do appear to be much better than they were for the Linear Model. With the RMSE standing at 7.37 and the MAPE being 151%. There is still room for improvement in this model, but it is performing better than the linear model.

2.6 (4 points) Task 5b: Train best models on present data

Splitting both the SA and NSA time series into training and test sets.

```
# Test size
test.size <- 24 # Months
test.df <- tail(co2_present, test.size) %>%
  mutate(.mean = average) # Used later for ggplot

# Train for non-seasonal (ns)
train_ns <- head(co2_present, nrow(co2_present) -  test.size)

# Train for seasonal (s)
```

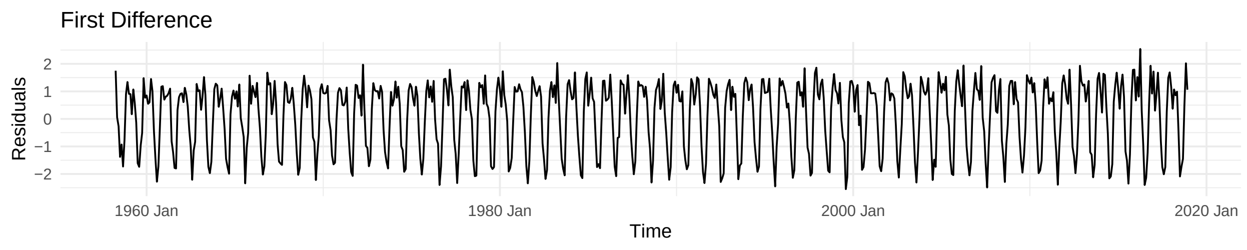
```
train_s <- head(co2_sa, nrow(co2_sa) -  test.size) %>%
  subset(select=-c(.model))
```

2.6.1 Training the ARIMA Model on the NSA Data

```
tseries::adf.test(train_ns$average, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  train_ns$average
## Dickey-Fuller = -0.72341, Lag order = 8, p-value = 0.9684
## alternative hypothesis: stationary
```

The ADF test statistic and p-value come out to be equal to -0.72 and 0.96 respectively. Since the p-value is greater than 0.05, we would fail to reject the null hypothesis that the time series is non-stationary. Thus, we will apply differencing in order to attempt to make the time series stationary.



Based on the above plot, it would appear that the first difference appears to be stationary. We will verify with an ADF test.

```
ns_diff_data <- train_ns %>%
  mutate(second_diff = difference(average, differences = 1)) %>%
  filter(!is.na(second_diff))

tseries::adf.test(ns_diff_data$second_diff, alternative = "stationary")
```

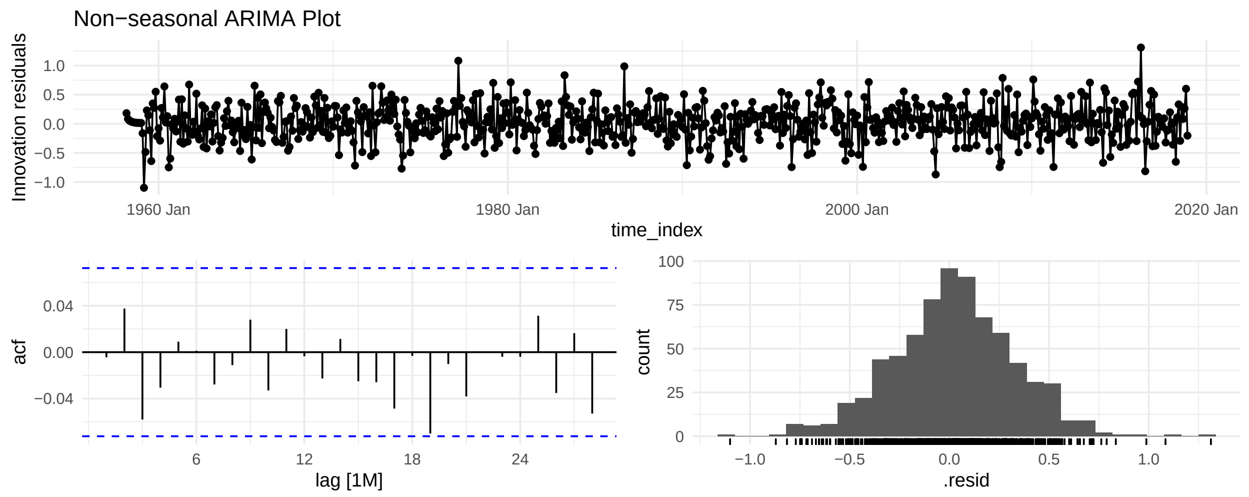
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ns_diff_data$second_diff
## Dickey-Fuller = -34.717, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

The ADF test statistic and p-value come out to be equal to -34.717 and 0.01, respectively. Since the p-value is less than 0.05, we would reject the null hypothesis that the time series is non-stationary. Because we know we should use first differencing, we will set these hyper parameters and then select a model with the lowest BIC.

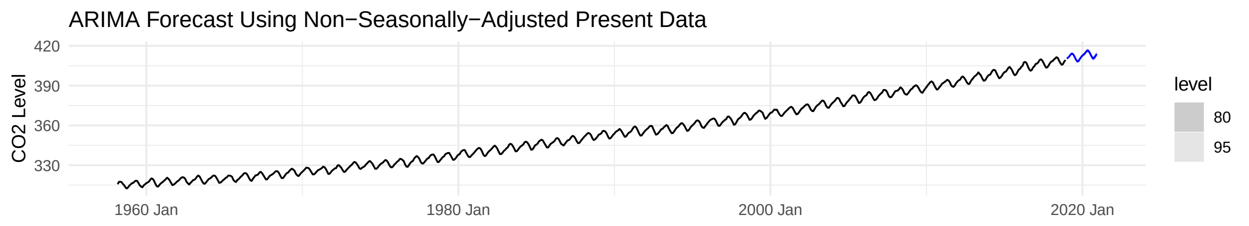
```
model.ns <- train_ns %>%
  model(ARIMA(average ~ 0 + pdq(0:10, 1, 0:10) + PDQ(0:10, 1, 0:10),
                         ic="bic", stepwise=F, greedy=F))
model.ns %>%
  report()
```

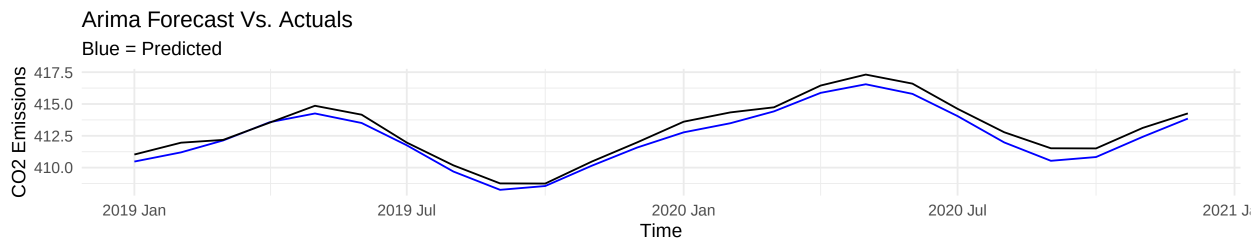
```
## Series: average
## Model: ARIMA(1,1,1)(2,1,1)[12]
##
```

```
## Coefficients:
##          ar1      ma1      sar1      sar2      sma1
##       0.1961  -0.5508   -0.0052   -0.0252   -0.8596
## s.e.  0.1019   0.0880    0.0443    0.0427    0.0244
##
## sigma^2 estimated as 0.0981:  log likelihood=-184.29
## AIC=380.57   AICc=380.69   BIC=408.02
```



The residual plots for the ARIMA model appear to have constant variance and a mean near zero. The histogram also appears to be normally distributed.





The forecast plots seem to track nicely with the actuals for the 24 month period.

2.6.2 Training the ARIMA Model on the SA Data

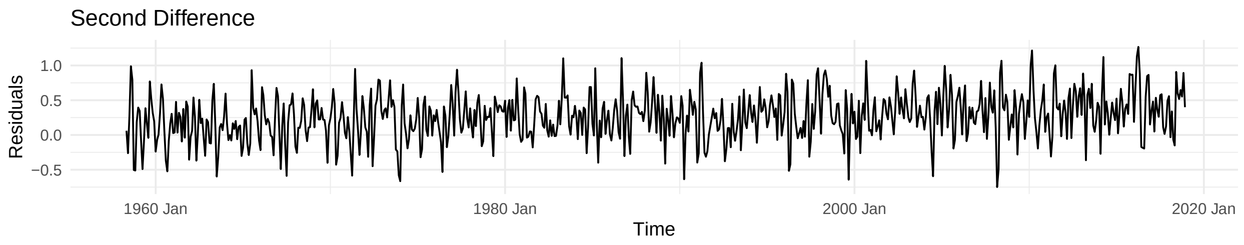
We will perform an ADF test to determine whether the seasonally-adjusted data is stationary.

```
tseries::adf.test(train_s$season_adjust, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  train_s$season_adjust
## Dickey-Fuller = -0.33025, Lag order = 8, p-value = 0.9895
## alternative hypothesis: stationary
```

The ADF test statistic and p-value come out to be equal to -0.33 and 0.9895 respectively. Since the p-value is greater than 0.05, we would fail to reject the null hypothesis that the time series is non-stationary. Thus, we will apply differencing in order to attempt to make the time series stationary.



Based on the above plot, it would appear that the first difference appears to be stationary. We will verify with an ADF test.

```
s_diff_data <- train_s %>%
  mutate(first_diff = difference(season_adjust, 2)) %>%
  filter(!is.na(first_diff))

tseries::adf.test(s_diff_data$first_diff, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  s_diff_data$first_diff
## Dickey-Fuller = -7.8626, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

The ADF test statistic and p-value come out to be equal to -7.86 and 0.01 respectively. Since the p-value is less than 0.05, we could reject the null hypothesis that the time series is non-stationary.

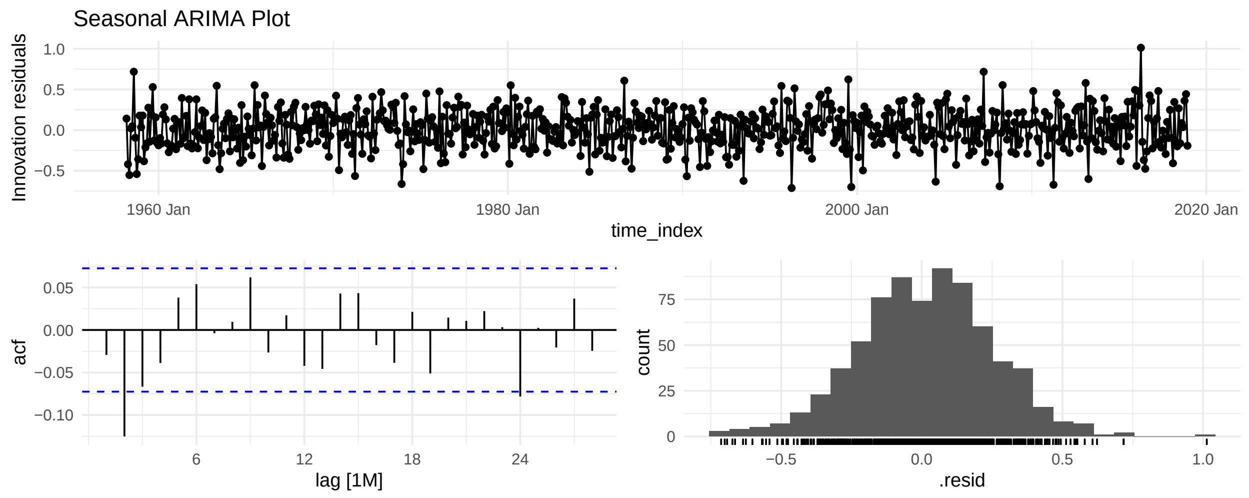
Due to these results, we decided to use a second difference and a seasonal difference of 0, and we let the ARIMA function choose the remaining hyper parameters.

```
model.s <- train_s %>%
  model(ARIMA(season_adjust ~ 0 + pdq(0:10, 2, 0:10) + PDQ(0:10, 0, 0:10),
                       ic="bic", stepwise=F, greedy=F))
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

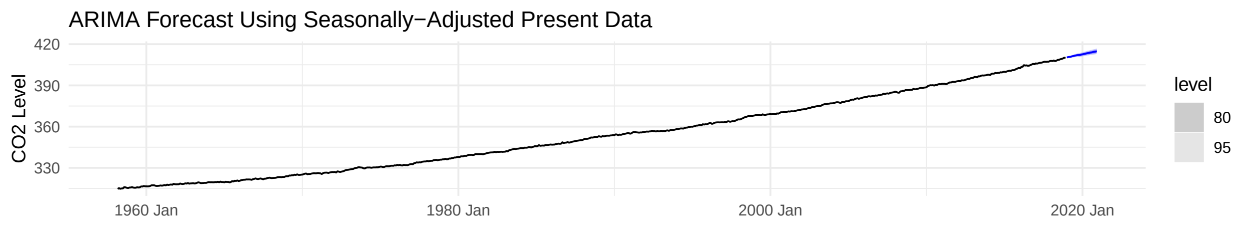
```
model.s %>%
  report()
```

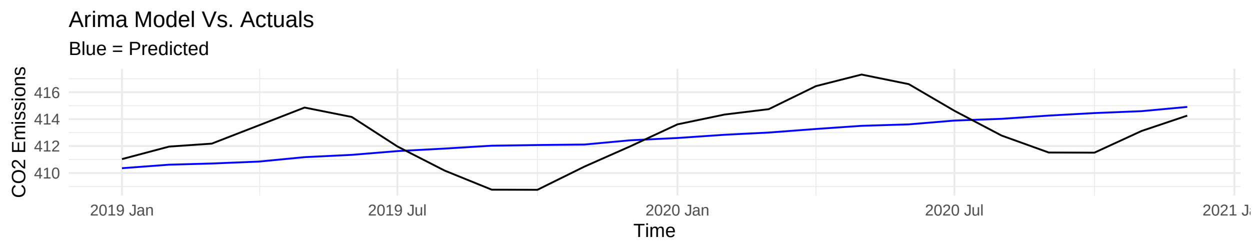
```
## Series: season_adjust
## Model: ARIMA(1,2,1)(4,0,0)[12]
##
## Coefficients:
##           ar1      ma1     sar1     sar2     sar3     sar4
##       -0.3249  -0.9694  -0.3854  -0.3802  -0.3001  -0.2129
## s.e.      NaN   0.0078   0.0037      NaN      NaN      NaN
##
## sigma^2 estimated as 0.05821:  log likelihood=1.5
## AIC=11   AICc=11.16   BIC=43.14
```



```
augment(model.s) %>%
  features(.innov, ljung_box, dof = 14, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model                                                      lb_stat lb_pv~1
##   <chr>                                                         <dbl>   <dbl>
## 1 "ARIMA(season_adjust ~ 0 + pdq(0:10, 2, 0:10) + PDQ(0:10, 0, ~    38.0 3.85e-5
## # ... with abbreviated variable name 1: lb_pvalue
```



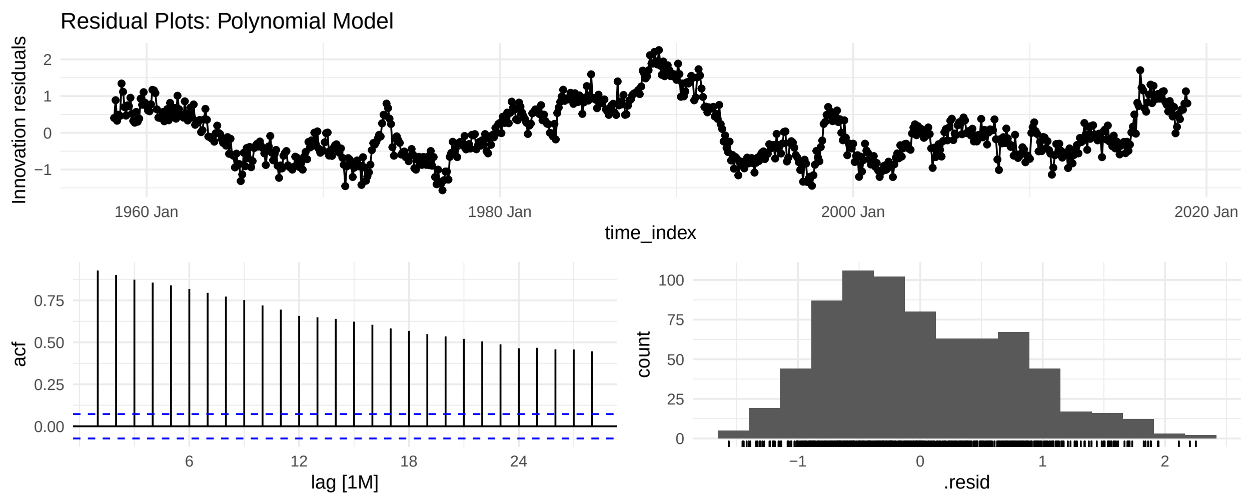


The ARIMA hyper parameters that were selected include the following: ARIMA(1,2,1)(4,0,0)[12]. Using these values, the residual plots appear to meet the condition of constant variance and a mean near zero. The test results also past the ljung_box statistical test. The ACF had some extreme values at the 2nd and 24th time lag, which seems a bit strange. The forecast estimates appear strange when juxtaposed next to the monthly data inclusive of the seasonal trends but that is because the underlying seasonality has been removed.

2.6.3 Training the Polynomial Time-Trend Model on the SA Data

```r
# Build model
train_s %>%
  model(TSLM(season_adjust ~ trend() + I(trend()^2) + I(trend()^3))) -> fit_poly

# Print model
report(fit_poly)
```

```
## Series: season_adjust
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5643 -0.5742 -0.1199  0.5283  2.2516
##
## Coefficients:
##                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  3.142e+02  1.103e-01 2849.568   <2e-16 ***
## trend()      6.772e-02  1.305e-03   51.875   <2e-16 ***
## I(trend()^2) 8.074e-05  4.148e-06   19.463   <2e-16 ***
## I(trend()^3) 6.554e-09  3.730e-09    1.757   0.0793 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.741 on 726 degrees of freedom
## Multiple R-squared: 0.9993,  Adjusted R-squared: 0.9993
## F-statistic: 3.375e+05 on 3 and 726 DF, p-value: < 2.22e-16
```



The residual diagnostic plots does not produce promising results. The residual time series does not have a constant variance or mean, and it is not clear that the mean is near zero. All values within the ACF exceed the threshold values and the histogram of the residuals appears to be skewed to the right.

As a result, we have decided to go with the `ARIMA(1,1,1)(2,1,1)[12]` model because it adjusts for the underlying seasonality, which we may need to adjust going forward, all of its residual values are are within the acceptable bands of the ACF, and it is easier to explain the model results to a non-technical audience.

2.7 (3 points) Task Part 6b: How bad could it get?

We will now retrain the selected model on all of the available data.

```
model.bic3 <- co2_present %>%
  model(ARIMA(average ~ 0 + pdq(1, 1, 1) + PDQ(2, 1, 1), ic="bic"))
```

Now we will get predictions from the model.

```
# Prediction from 2021 to 2121
fc2_100 <- fabletools::forecast(model.bic3, h=1224) %>%
  hilo() %>%
  unpack_hilo(c(`80%`, `95%`))

fc2_100 %>%
  filter(.mean >= 420 & .mean < 421) -> fc2_420

# 420 PPM levels for the first and last time
fc2_420_first <- head(fc2_420, n =1)
fc2_420_last  <- tail(fc2_420, n =1)

# Get values for 420 ppm predictions
fc2_420_first_time  <- fc2_420_first$time_index
fc2_420_first_lower <- round(fc2_420_first$`95%_lower`, 2)
fc2_420_first_upper <- round(fc2_420_first$`95%_upper`, 2)
fc2_420_first_mean  <- round(fc2_420_first$.mean, 2)

fc2_420_last_time  <- fc2_420_last$time_index
fc2_420_last_lower <- round(fc2_420_last$`95%_lower`, 2)
fc2_420_last_upper <- round(fc2_420_last$`95%_upper`, 2)
fc2_420_last_mean  <- round(fc2_420_last$.mean, 2)
```

Based on the findings from our model, we predict that CO2 will cross the 420 ppm threshold for the first time on 2023 Jan with an expected value of 420.42 with a 95% confidence interval between 418.48 - 422.36. Our model also predicts that the last time CO2 will be between the 420 and 421 ppm threshold will be on 2024 Oct with an expected value of 420.77 with a 95% confidence interval between 417.92 - 423.62.

```
fc2_100 %>%
  filter(.mean >= 500 & .mean < 501) -> fc2_500

# 420 PPM levels for the first and last time
fc2_500_first <- head(fc2_500, n =1)
fc2_500_last  <- tail(fc2_500, n =1)

# Get values for 420 ppm predictions
fc2_500_first_time  <- fc2_500_first$time_index
fc2_500_first_lower <- round(fc2_500_first$`95%_lower`, 2)
fc2_500_first_upper <- round(fc2_500_first$`95%_upper`, 2)
fc2_500_first_mean  <- round(fc2_500_first$.mean, 2)

fc2_500_last_time  <- fc2_420_last$time_index
fc2_500_last_lower <- round(fc2_500_last$`95%_lower`, 2)
fc2_500_last_upper <- round(fc2_500_last$`95%_upper`, 2)
fc2_500_last_mean  <- round(fc2_500_last$.mean, 2)
```

Based on the findings from our model, we predict that CO2 will cross the 500 ppm threshold for the first time on 2056 Apr with an expected value of 500.96 with a 95% confidence interval between 475.25 - 526.67. Our model also predicts that the last time CO2 will be between the 500 and 501 ppm threshold will be on 2024 Oct with an expected value of 500.68 with a 95% confidence interval between 472.63 - 528.72.

```
# Atmospheric CO2 levels in the year 2122
fc_2122  <- tail(fc2_100, n =1)

# Carbon values in 2122
fc_2122_lower <- round(fc_2122$`95%_lower`, 2)
fc_2122_upper <- round(fc_2122$`95%_upper`, 2)
fc_2122_mean  <- round(fc_2122$.mean, 2)
```

'By the end of 2122, our model predicts that CO2 will be 654.1 ppm with a 95% confidence interval between 546.98 - 761.22. Although we are not very confident in the point estimate for the average CO2 emissions at the end of 2022, we are fairly confident (95%!) that average CO2 emissions will be between our confidence interval of 546.98 - 761.22.