W271 Group Lab 2

Salman Bashir, Jeffrey Day, Duncan Howard, & Erin Werner

November 8th, 2020

Contents

Part 1 **2**

Part 2 **9**

Part 3 **21**

Part 4 **26**

Part 5 **34**

Part 6 **38**

```
library(ggplot2)
library(dplyr)
library(car)
library(readr)
library(tseries)
library(forecast)
library(fable)
library(fpp2)
library(fpp3)
library(gridExtra)
library(grid)
library(tsibble)
library(tibble)
library(mvtnorm)
library(vars)
library(lubridate)
```

Part 1

Conduct a comprehensive Exploratory Data Analysis on the co2 series. This should include (without being limited to) a thorough investigation of the trend, seasonal and irregular elements. Trends both in levels and growth rates should be discussed. Consider expressing longer-run growth rates as annualized averages.

The co2 data set in R's datasets package (automatically loaded with base R) is a monthly time series of atmospheric carbon dioxide concentrations measured in ppm (parts per million) at the Mauna Loa Observatory from 1959 to 1997. The curve graphed by this data is known as the 'Keeling Curve'.

We can start by doing some basic exploration of the data set itself.

```
head(co2, 24)
```

```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39 314.65 313.68 313.18
## 1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01 315.74 314.00 313.68
##         Nov    Dec
## 1959 314.66 315.43
## 1960 314.84 316.03
```

```
class(co2)
```

```
## [1] "ts"
```

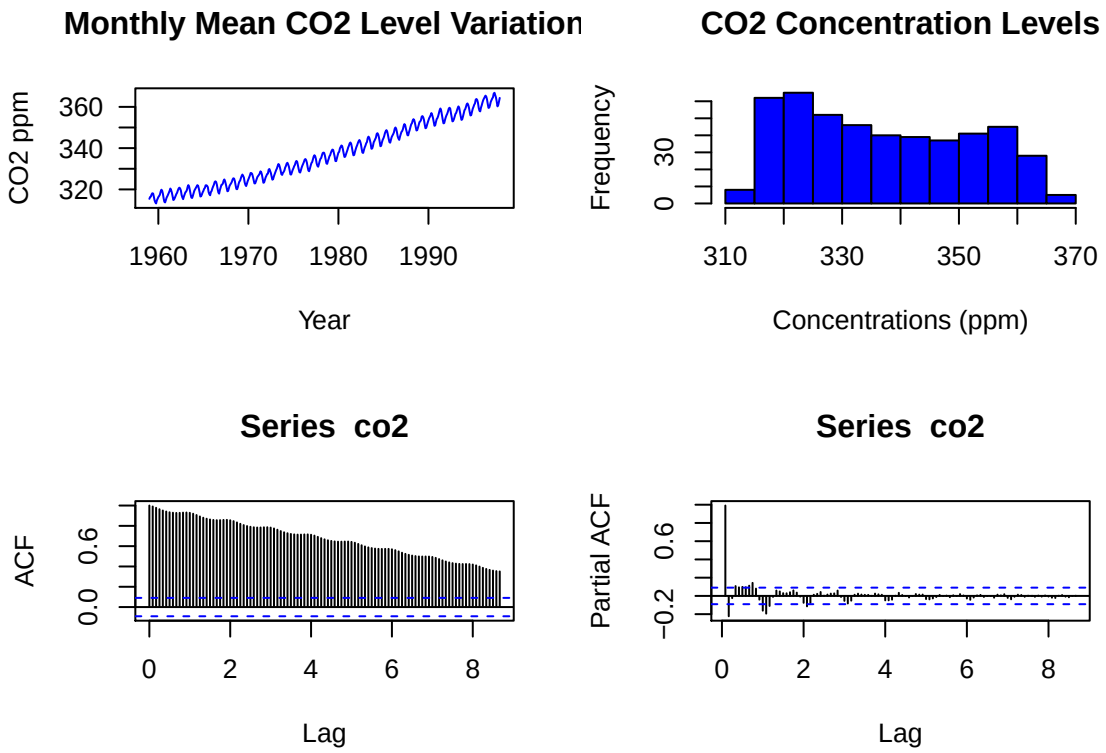
```
tsp(co2)
```

```
## [1] 1959.000 1997.917   12.000
```

Overall, we can see that co2 is a time-series that spans from 1959 to 1997, in which the CO2 level is report each month.

Now, we can take a more generalized look at the CO2 levels through visualizations. These include the time series, the histogram, the ACF, and the PACF. The histogram shows the distribution of overall values and the time series plots of those values with respect to time. The ACF describes how well the present value of the series is related with its past values. For instance, a time series can have components like trend, seasonality, cyclic and residual. The ACF considers all these components while finding correlations. The PACF, instead of finding correlations of present with lags like ACF, finds correlation of the residuals with the next lag value.

```
par(mfrow=c(2,2))
plot(co2,ylab=expression("CO2 ppm"),col='blue',las=1, xlab="Year",
     main="Monthly Mean CO2 Level Variation")
hist(co2,main="CO2 Concentration Levels", col="blue",xlab="Concentrations (ppm)")
acf(co2, lag.max = 104)
pacf(co2, lag.max = 104)
```



From the plots, we can see that:

- the time series of `co2` displays a very strong upward trend with strong seasonality. The fluctuation between each season appears to be quite consistent and pronounced.
- the histogram of `co2` does not necessarily have a normal distribution. Instead, the values appear to be somewhat bi-modal with peaks at each end of the distribution and a dip in between.
- the correlogram of ACF of `co2` has a gradual decline, with some seasonal effect. Yet, the ACF does display some persistence.
- the correlogram of PACF of `co2` has a sharp decline after the first lag. However, the PACF then gradually oscillates and declines.

Collectively, these plots demonstrate the non-stationary of the series. This is a result of trend and seasonality, yet there are no obvious irregular elements.

As there is apparent trend and seasonality in our series, we can take a greater look into those features. So, we can start by computing the seasonal and annual averages to give gross descriptions of the series.

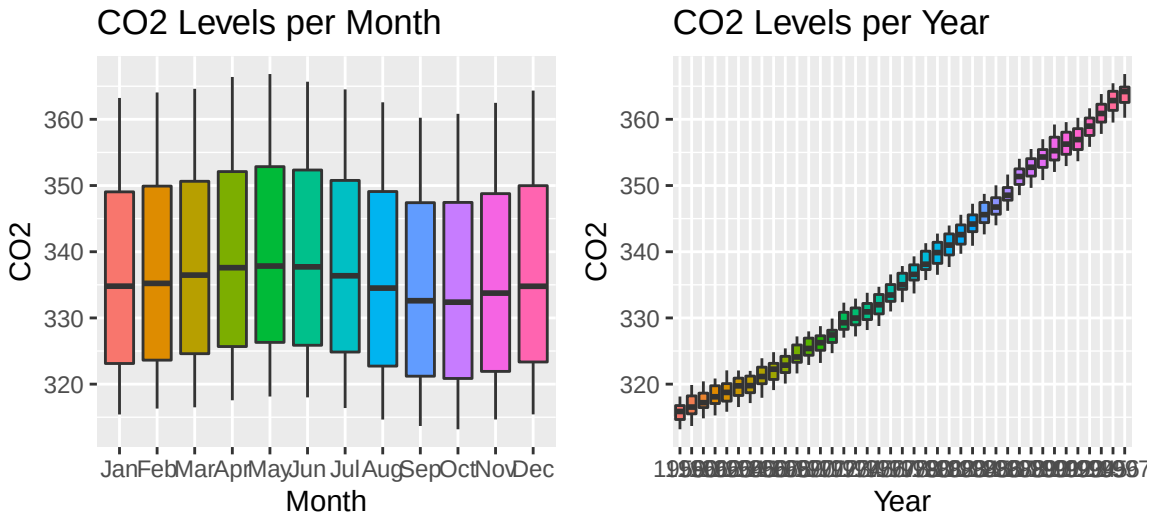
```
Month <- factor(rep(1:12, times = 39), labels = month.abb)
Year <- factor(rep(1959:1997, each = 12))
t1 <- head(aggregate(as.numeric(co2), by = list(Month = Month), mean),8)
t2 <- head(aggregate(as.numeric(co2), by = list(Year = Year), mean),8)
co2_df <- as.data.frame(co2 <- co2, Month = Month, Year = year)
grid.arrange(tableGrob(t1), tableGrob(t2), ncol = 2)
```

	Month	x
1	Jan	336.4308
2	Feb	337.2033
3	Mar	338.0546
4	Apr	339.2944
5	May	339.8821
6	Jun	339.3282
7	Jul	337.9164
8	Aug	335.9579

	Year	x
1	1959	315.8258
2	1960	316.7475
3	1961	317.4850
4	1962	318.2975
5	1963	318.8325
6	1964	319.4625
7	1965	319.8725
8	1966	321.2100

From the tables, we can see how the CO2 values gradually change overtime. We can get a better understanding of this through box-plots.

```
p1 <- ggplot(co2_df,aes(Month,co2))+theme(legend.position="none") +
  geom_boxplot(varwidth=T, aes(fill = Month)) + labs(x="Month",y="CO2") +
  labs(title="CO2 Levels per Month")
p2 <- ggplot(co2_df,aes(Year,co2))+theme(legend.position="none") +
  geom_boxplot(varwidth=T, aes(fill = Year)) + labs(x="Year",y="CO2") +
  labs(title="CO2 Levels per Year")
egg::ggarrange(p1, p2, nrow = 1)
```



From the box-plots, we can see that:

- with respect to the change by the month, there is some seasonality in the CO2 levels. There appears to be a fluctuation in levels as they increase from October to May then decrease from June to September.
- with respect to the change by the year, there is a very strong and consistent upward trend in the CO2 levels.

These results further indicate trend and seasonality in our series, making it non-stationary. However, the seasonality does not change too drastically over time, so there is no need to apply the Box-Cox method to our ARMIA model, as the Box-Cox transformation is designed to reduce non-normality of the errors in a model.

Therefore, we can statistically test for a unit root in our series. In the context of an ADF test, H_0 is the

presence of a unit root, with rejection (H_A) indicating that the series does not have a unit root.

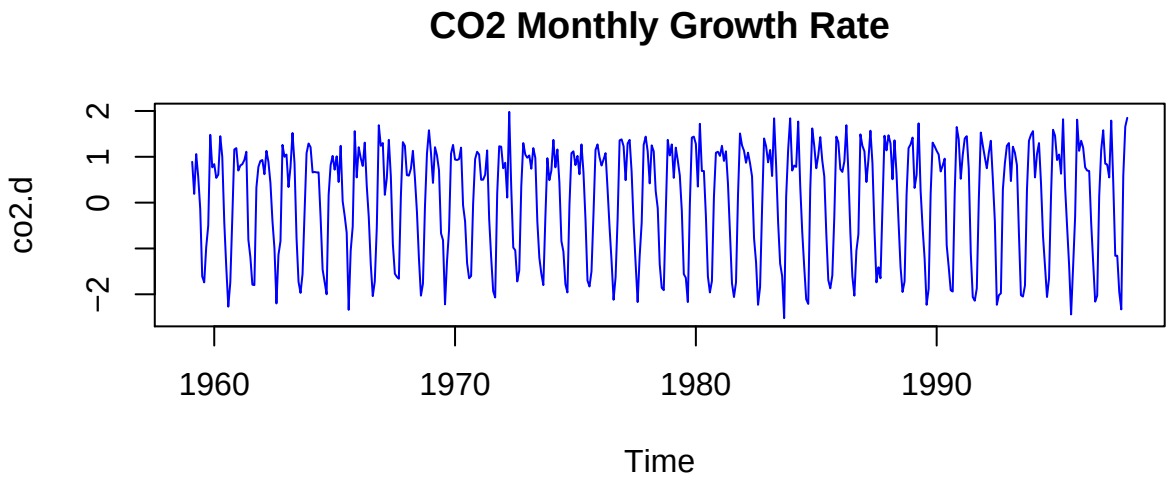
```
adf.test(co2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  co2
## Dickey-Fuller = -2.8299, Lag order = 7, p-value = 0.2269
## alternative hypothesis: stationary
```

The resulting p-value of the ADF test is 0.23, which is not less than $\alpha = 0.05$. So, we do not reject the null hypothesis that the series does have a unit root. This further implies that the series is not stationary.

Thus, we must consider taking the difference of our series, by a certain order, to achieve stationarity. So, to start, we can look at the first difference of our series. That way we can investigate the trend, seasonality, and irregular elements in the differenced series as well. However, we will now be examining the CO2 growth rates rather than the levels, which we had previously been looking at.

```
co2.d <- diff(co2)
plot(co2.d, main = "CO2 Monthly Growth Rate", col = "blue")
```



Although the seasonal averages and box-plots do not show the strong seasonal component exhibited in the raw data, we can see the strong seasonality very clearly in the first order difference of the series. Additionally, with first-differencing applied to the series, the trend is eliminated and the variance is largely subdued. Thus, the first order difference is stationary. This will be important for our model moving forward.

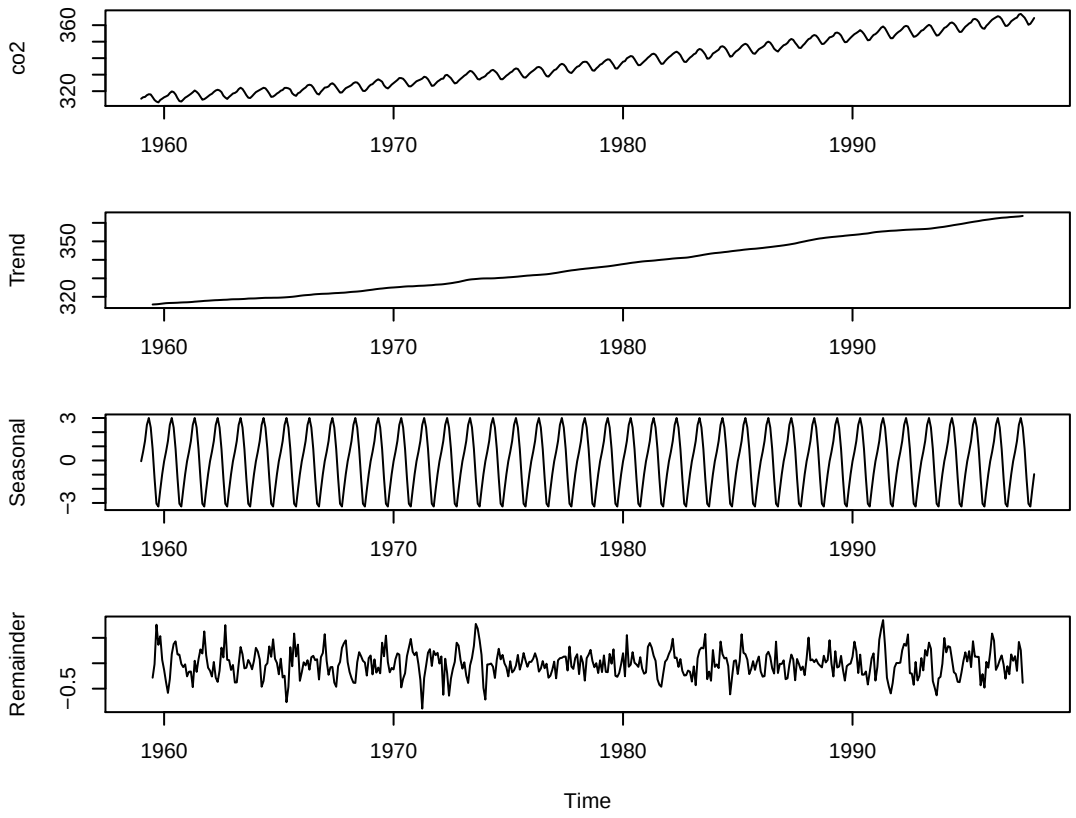
Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category. More generally, in order address the the trend, seasonal and irregular elements that are in the series, we can decompose the original time series into trend, seasonal and error components. Often this is done to help improve understanding of the time series, but it can also be used to improve forecast accuracy. Classical decomposition of time series is performed using the `decompose()` function.

The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. We can see this in our `co2` time series.

```
co2.deco <- decompose(co2, type = "additive")
```

We can then plot our results.

```
layout(matrix(1:4, ncol = 1))
op <- par(oma = c(2,0,0,1) + 0.1, mar = c(2,4,2,2) + 0.1, xpd = NA)
plot(co2, xlab = "")
plot(co2.deco$trend, xlab = "", ylab = "Trend")
plot(co2.deco$seasonal, xlab = "", ylab = "Seasonal")
plot(co2.deco$random, ylab = "Remainder")
```




```
par(op)
```

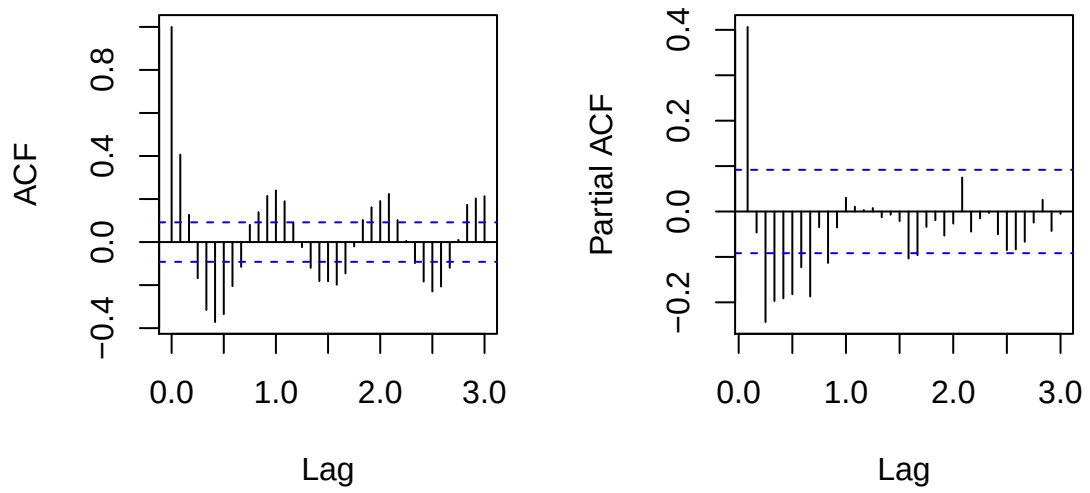
From the plots, we can see that:

- the overall CO2 levels have an upward trend with seasonality.
- the CO2 level trend is strong and upward.
- the CO2 level seasonality fluctuates consistently.
- the CO2 levels appear to have a random white noise element in the series as well. The variation in the remainder component is small compared to the variation in the data.

Therefore, we must consider the strong trend in building our model. Additionally, the seasonal and remainder components provide further evidence that we do not need to apply the Box-Cox method to our ARIMA model.

We can also look at the ACF and PACF of the random elements of the series.

```
layout(matrix(1:2, ncol = 2))
acf(co2.deco$random, na.action = na.omit, lag.max = 36, main = "")
pacf(co2.deco$random, na.action = na.omit, lag.max = 36, main = "")
```



From the correlograms, we can see that:

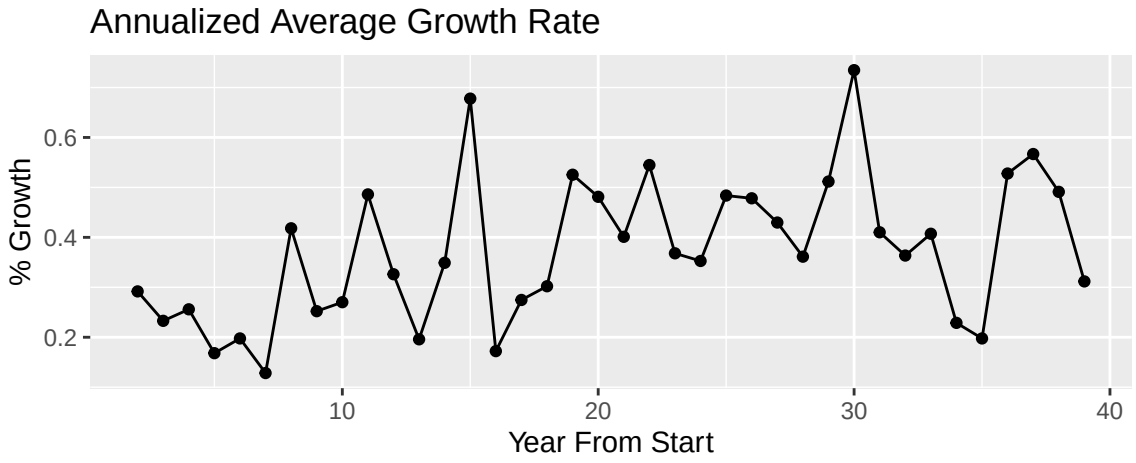
- the ACF gradually declines and oscillates around zero. This means we can expect a negative AR parameter for the random elements.
- the PACF also gradually declines and oscillates around zero. Yet, the PACF drastically cuts off after the first lag, so we know we should use a first order model.

As a result, the random element of the series appears to act like white noise. This is ideal for building our model.

Another approach to consider is to express longer-run growth rates as annualized averages. The average annual growth rate is the average increase in the value of an individual variable over the period of a year. It is calculated by taking the arithmetic mean of a series of growth rates. The average annual growth rate is helpful in determining long-term trends.

```
my_yr <- aggregate(as.numeric(co2), by = list(Year = Year), mean)
yr_avg <- c()
for(i in seq(2,nrow(my_yr))){
  temp <- (my_yr$x[i]/my_yr$x[i-1]) - 1
  yr_avg <- c(yr_avg, temp*100)
}
ann_avg <- data.frame(Year = my_yr$Year[2:39], Growth = yr_avg)
```

```
ggplot(ann_avg, aes(x=as.numeric(Year),y=Growth)) + geom_point() + geom_line() +
  labs(title="Annualized Average Growth Rate", x = "Year From Start",y="% Growth")
```



From the plot of annualized averages, we can see that there is not a large enough difference in average growth rates from year to year (less than 1% each year) in order to consider it as a good expression of longer-run growth rates. Furthermore, we would loose a lot of information and resolution in the data if we simplify our model to an annualized average.

Therefore, we will continue our analysis with the original `co2` data set. We will need to consider both the trend and seasonality that are apparent in the series when choosing our model.

Part 2

Fit a linear time trend model to the co2 series, and examine the characteristics of the residuals. Compare this to a higher-order polynomial time trend model. Discuss whether a logarithmic transformation of the data would be appropriate. Fit a polynomial time trend model that incorporates seasonal dummy variables, and use this model to generate forecasts to the year 2020.

To start, we will create a data frame with a few new variables to hold information about sampling month, time since start of sampling, etc. These variables will be used to describe trends and seasonal components in the regression models themselves.

```
carbon <- data.frame(CO2=as.numeric(co2),Month=rep(month.abb,39),
                     Year=rep(1959:1997,each=12),Time=seq_len(length(co2)),
                     Month_Num=rep(seq(1:12),39))
carbon <- within(carbon,Date<-as.Date(paste("01",Month,Year),format="%d %b %Y"))
carbon <- within(carbon,MonYear<-yearmonth(paste(Year, Month_Num)))
head(carbon)
```

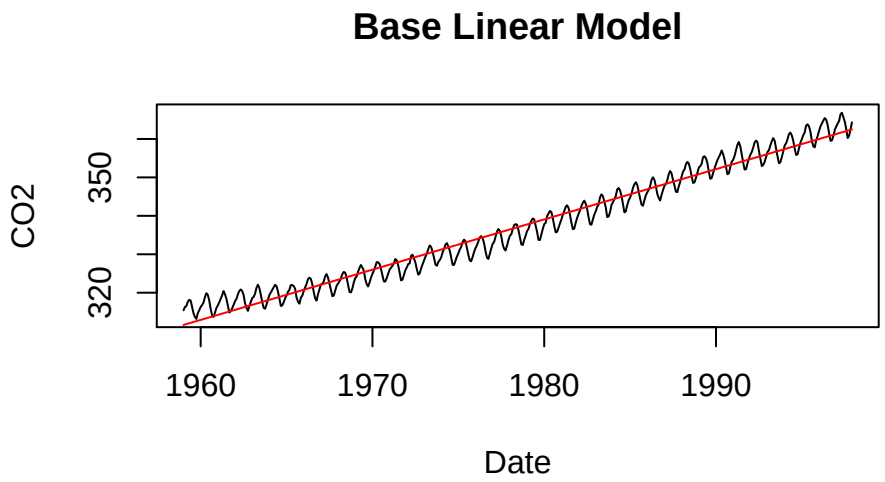
```
##      CO2 Month Year Time Month_Num       Date  MonYear
## 1 315.42   Jan 1959    1         1 1959-01-01 1959 Jan
## 2 316.31   Feb 1959    2         2 1959-02-01 1959 Feb
## 3 316.50   Mar 1959    3         3 1959-03-01 1959 Mar
## 4 317.56   Apr 1959    4         4 1959-04-01 1959 Apr
## 5 318.13   May 1959    5         5 1959-05-01 1959 May
## 6 318.00   Jun 1959    6         6 1959-06-01 1959 Jun
```

So first, we can fit a linear time trend model to the co2 series.

```
linear.trend.fit <- lm(CO2 ~ Time, data=carbon)
summary(linear.trend.fit)
```

```
##
## Call:
## lm(formula = CO2 ~ Time, data = carbon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0399 -1.9476 -0.0017  1.9113  6.5149
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.115e+02  2.424e-01  1284.9   <2e-16 ***
## Time        1.090e-01  8.958e-04   121.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.618 on 466 degrees of freedom
## Multiple R-squared:  0.9695, Adjusted R-squared:  0.9694
## F-statistic: 1.479e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

```
plot(CO2 ~ Date, data = carbon, type = "l", main = "Base Linear Model")
lines(fitted(linear.trend.fit) ~ Date, data = carbon, col = "red")
```



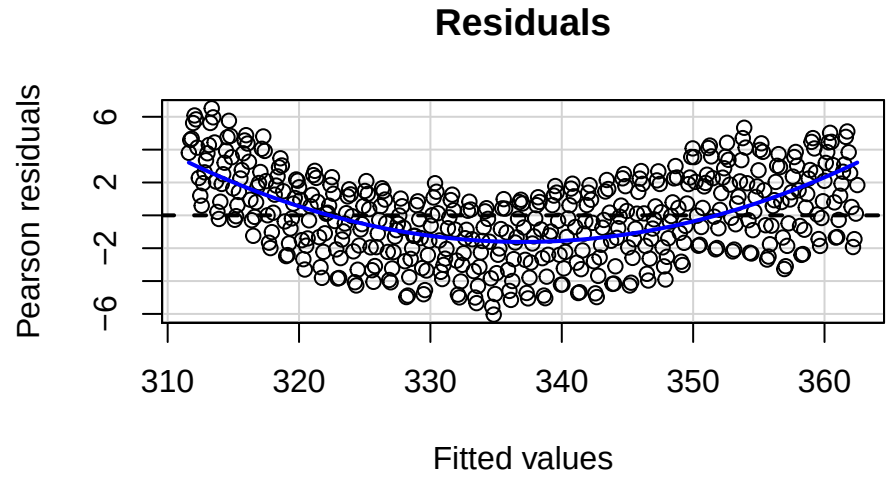
From the plot, we can see that the linear model fits the trend of the series fairly well. We see that time has a strongly significant relationship with the trend in CO2 concentration, although it appears not to capture some upwards concavity.

Then, we can examine the performance further by observing the characteristics of the residuals. The residuals, in a time series model, are what is left over after fitting a model. The residuals are equal to the difference between the observations and the corresponding fitted values. Residuals are useful in checking whether a model has adequately captured the information in the data.

In order to produce good forecasts, the residuals must:

- be uncorrelated. If there are correlations, than there is still information in them. So, they should not be used for forecasting.
- have zero mean. If they do not, then the forecasts will be biased.
- have constant variance.
- be normally distributed.

```
residualPlot(linear.trend.fit, main = "Residuals")
```

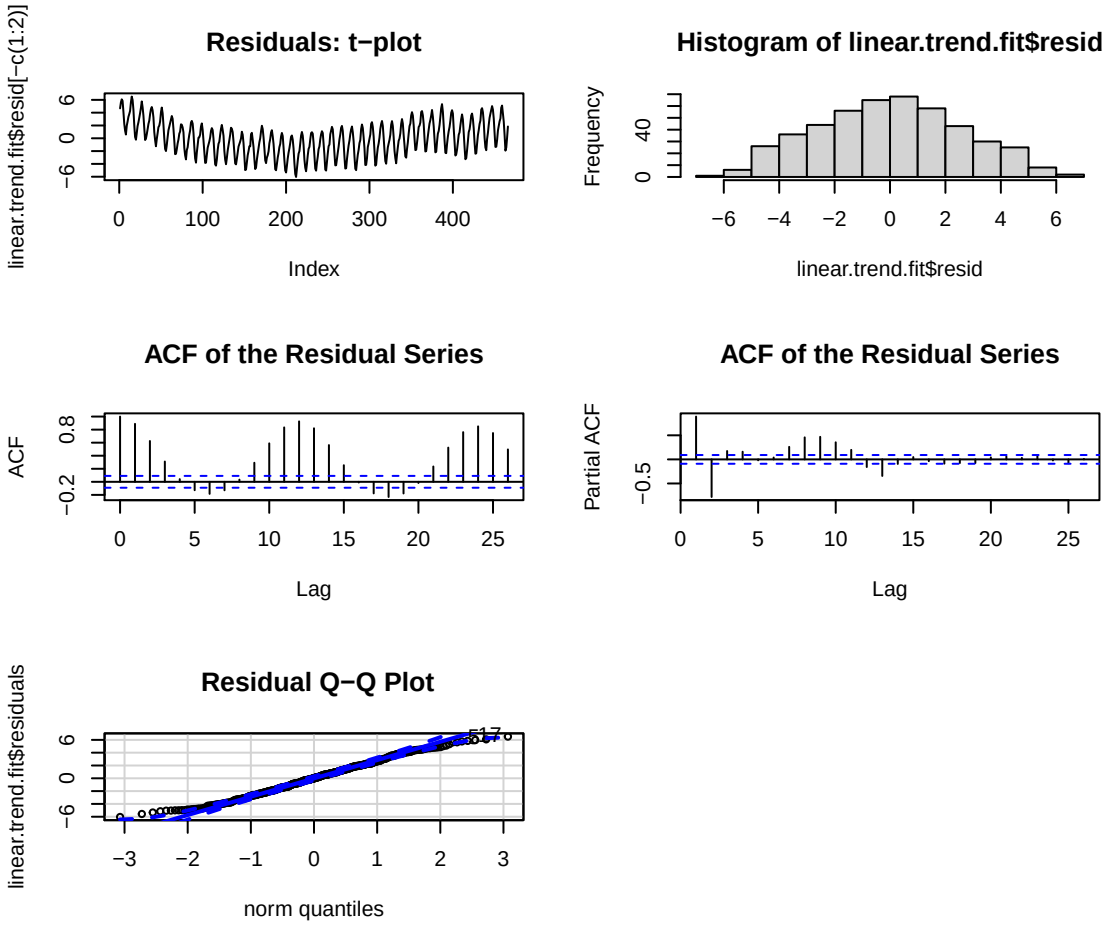


From the residual plot, we can see that the residuals follow a curve very well. We see the leftover concavity in the plot of the residuals, indicating the presence of a higher order relationship. The residuals are also disperse since we are not yet taking seasonality into account. Overall, we can not confirm the assumption of having a zero mean through the Residuals vs Fitted plot as there is an apparent pattern among the data points. The data is not centered around the zero-line. The mean changes drastically from left to right. So,

values are different for different values of x. There is not a flat band of points for the plot and our error is not zero in expectation. This means that the model is not actually linear. Yet, we can also take a look at the time series, the histogram, the ACF, and the PACF of the residuals in order to further analyze model fit.

```
par(mfrow=c(3,2))
plot(linear.trend.fit$resid[-c(1:2)], type="l", main="Residuals: t-plot")
hist(linear.trend.fit$resid)
acf(linear.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
pacf(linear.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
qqPlot(linear.trend.fit$residuals, main=" Residual Q-Q Plot")
```

```
## [1] 17  5
```



From the residual plots, we can see that:

- the time series still has a trend that's unaccounted for as well as seasonality. However, the trend is not as strong as in the original series. Yet, the variation of the residuals stays much the same across the data.
- The histogram and Q-Q plot show a roughly normal distribution, with a slight departure around extreme residual values. This is an indication of the normality of the residuals.
- the ACF appears to gradually decline and oscillate around zero. There appears to be some correlation, suggesting the forecasts are not reliable.
- the PACF appears to cut off sharply after two lags, then gradually declines and oscillates around zero.

We can also test for the significance of the correlation using the Ljung–Box test.

```
Box.test(linear.trend.fit$residuals)
```

```
##
##  Box-Pierce test
##
## data:  linear.trend.fit$residuals
## X-squared = 371.55, df = 1, p-value < 2.2e-16
```

As a result, we can reject the null hypothesis that the series is uncorrelated. This means that correlation exists and our model assumptions are violated. So, we must use a different model or de-trend our model, which could be accomplished by including seasonality.

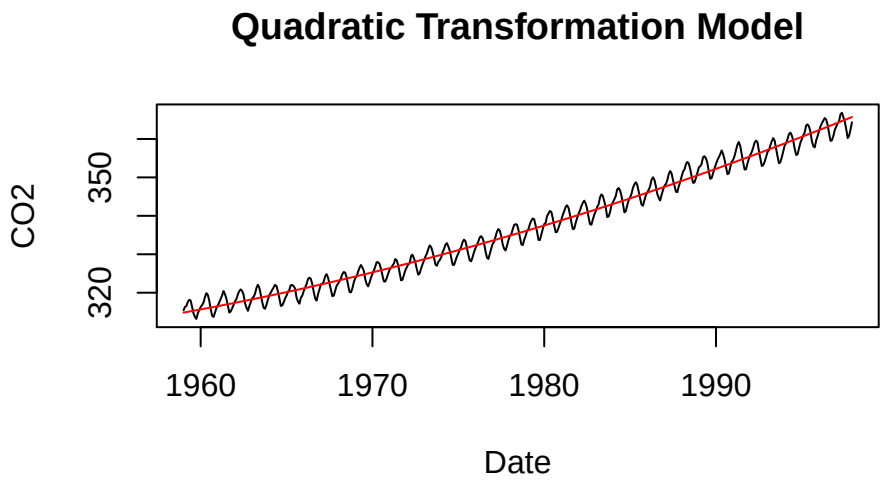
Although our residuals generally follow a normal distribution and have constant variance, the other model assumptions of zero mean and non-correlation are violated. Therefore, this model is not good for our series and we must consider other models for our series.

So next, we can compare this to a higher-order quadratic time trend model.

```
quadratic.trend.fit <- lm(CO2 ~ Time + I(Time^2), data=carbon)
summary(quadratic.trend.fit)
```

```
##
## Call:
## lm(formula = CO2 ~ Time + I(Time^2), data = carbon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.148e+02  3.039e-01 1035.65   <2e-16 ***
## Time        6.739e-02  2.993e-03   22.52   <2e-16 ***
## I(Time^2)   8.862e-05  6.179e-06   14.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.182 on 465 degrees of freedom
## Multiple R-squared:  0.9788, Adjusted R-squared:  0.9787
## F-statistic: 1.075e+04 on 2 and 465 DF,  p-value: < 2.2e-16
```

```
plot(CO2 ~ Date, data = carbon, type = "l", main = "Quadratic Transformation Model")
lines(fitted(quadratic.trend.fit) ~ Date, data = carbon, col = "red")
```

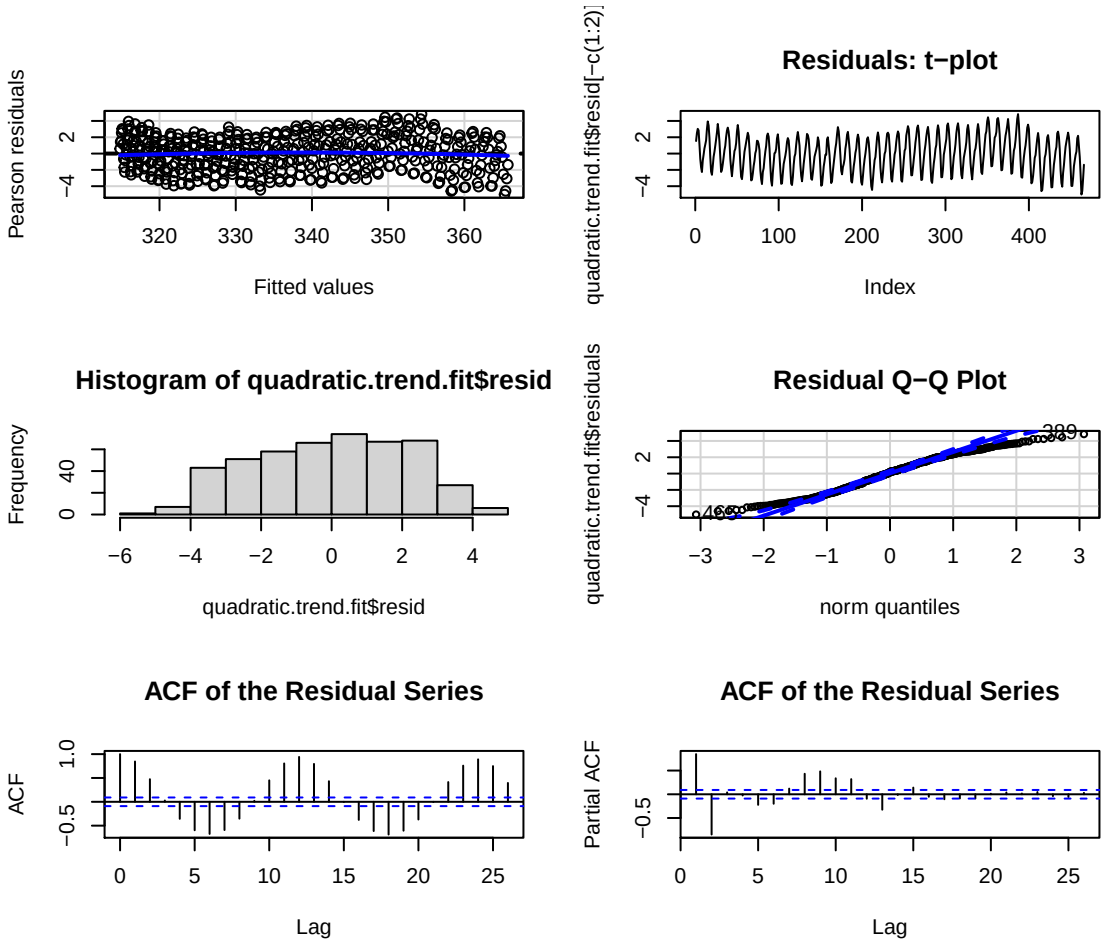


From the plot, we can see that the quadratic model fits the trend of the series slightly better than the linear model. We can, once again, take a look at the residuals to assess the model fit.

```
par(mfrow=c(3,2))
residualPlot(quadratic.trend.fit)
plot(quadratic.trend.fit$resid[-c(1:2)], type="l", main="Residuals: t-plot")
hist(quadratic.trend.fit$resid)
qqPlot(quadratic.trend.fit$residuals, main="Residual Q-Q Plot")
```

```
## [1] 465 389
```

```
acf(quadratic.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
pacf(quadratic.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
```



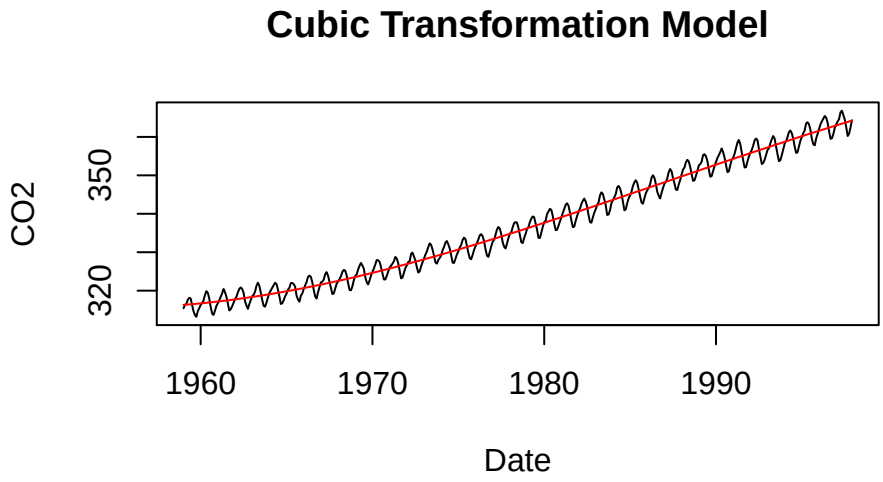
We see that a second order relationship with time is strongly significant and accounts for the curvature seen in the base linear model. Visually, the quadratic model fits the trend of the series better than the linear model. The shape of the residuals indicate a possible third order relationship, which we model below.

```
cubic.trend.fit <- lm(CO2 ~ Time + I(Time^2) + I(Time^3), data=carbon)
summary(cubic.trend.fit)
```

```
##
## Call:
## lm(formula = CO2 ~ Time + I(Time^2) + I(Time^3), data = carbon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5786 -1.7299  0.2279  1.8073  4.4318
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.163e+02  3.934e-01 804.008  < 2e-16 ***
## Time         2.905e-02  7.256e-03   4.004 7.25e-05 ***
## I(Time^2)    2.928e-04  3.593e-05   8.149 3.44e-15 ***
## I(Time^3)   -2.902e-07  5.036e-08  -5.763 1.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 464 degrees of freedom
```

```
## Multiple R-squared:  0.9802, Adjusted R-squared:  0.9801
## F-statistic:  7674 on 3 and 464 DF,  p-value: < 2.2e-16
```

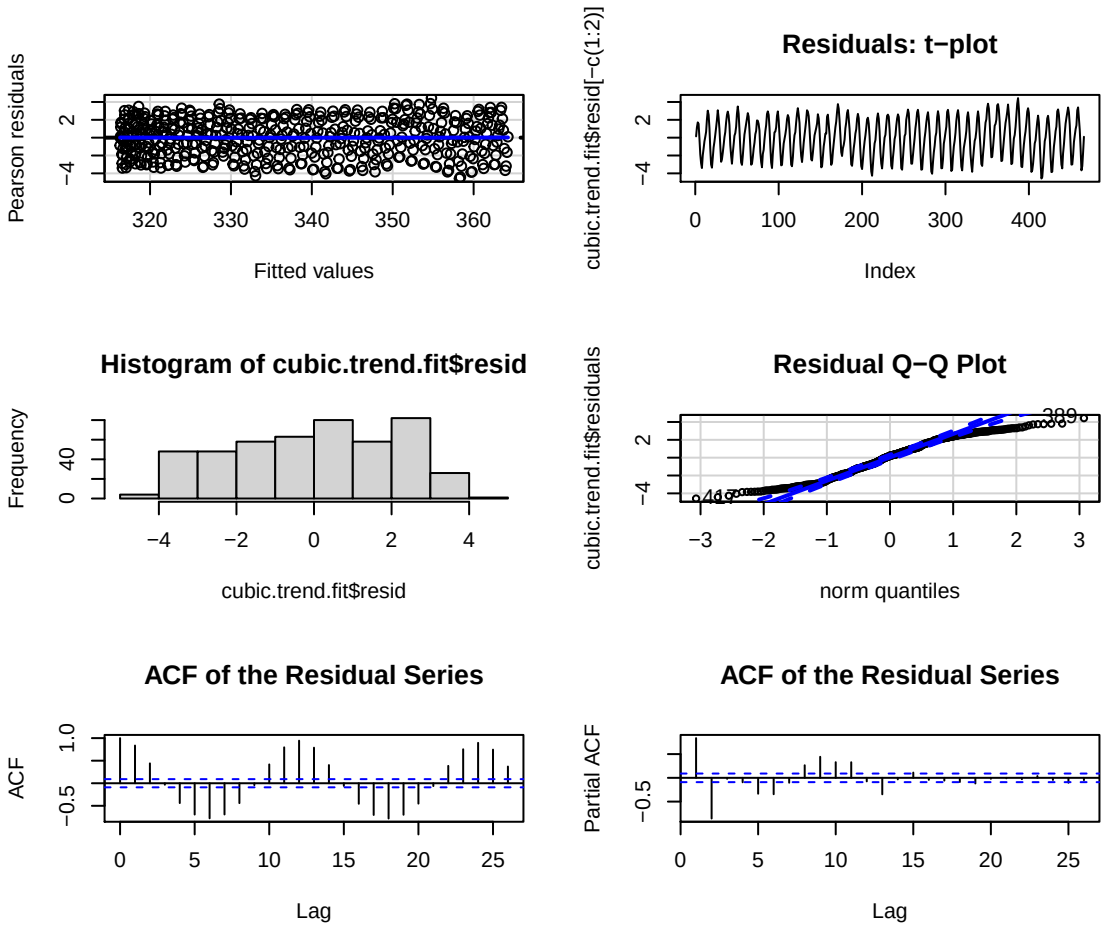
```
plot(CO2 ~ Date, data = carbon, type = "l", main = "Cubic Transformation Model")
lines(fitted(cubic.trend.fit) ~ Date, data = carbon, col = "red")
```




```
par(mfrow=c(3,2))
residualPlot(cubic.trend.fit)
plot(cubic.trend.fit$resid[-c(1:2)], type="l", main="Residuals: t-plot")
hist(cubic.trend.fit$resid)
qqPlot(cubic.trend.fit$residuals, main="Residual Q-Q Plot")
```

```
## [1] 417 389
```

```
acf(cubic.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
pacf(cubic.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
```



The cubic term is statistically significant, and the model fits the data very well. However, it's performance is only slightly better than the quadratic model and it's likely overfitting the data. Physically, a quadratic term would be modeling the acceleration in buildup of CO2 concentration while a cubic term would model the time change in that acceleration. That appears to be a small effect on the time scale of our data set and the third order term is probably only fitting variation in the trend. We will use the linear and quadratic terms only for our predictive model.

We can also discuss whether a logarithmic transformation of the data would be appropriate.

The logarithmic model would take the form as follows:

$$log(y_t) = \beta_0 + \beta_1 t + \epsilon_t y_t = e^{\beta_0 + \beta_1 t + \epsilon_t}$$

Therefore, we can take the `log()` of our explanatory variable in order to build the model.

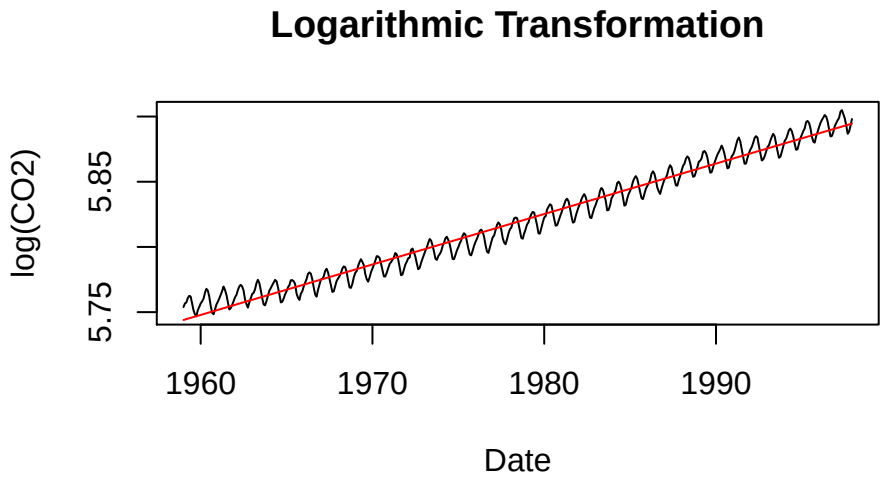
```
exp.trend.fit <- lm(log(CO2) ~ Time, data=carbon)
summary(exp.trend.fit)
```

```
##
## Call:
## lm(formula = log(CO2) ~ Time, data = carbon)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0172650 -0.0056145  0.0002764  0.0053760  0.0187770
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.744e+00  6.829e-04  8410.5   <2e-16 ***
## Time        3.224e-04  2.523e-06   127.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007375 on 466 degrees of freedom
## Multiple R-squared:  0.9722, Adjusted R-squared:  0.9722
## F-statistic: 1.633e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

A logarithmic transformation of the CO2 data does not appear to be appropriate based on the distribution of the data itself. In our EDA, we saw that it is roughly normally distributed and does not display any significant tailedness. Including the transformation in the model above, we can see that there is a statistically significant relationship. Visualizing the model below shows that it is only slightly different than the base linear model which excludes the transformation. The logarithmic transformation is not necessary for our data and does not result in an improved model.

```
plot(log(CO2) ~ Date, data = carbon, type = "l", main = "Logarithmic Transformation")
lines(fitted(exp.trend.fit) ~ Date, data = carbon, col = "red")
```



From the plot, we can see that the exponential model fits the log-trend of the series fairly well. It appears to have a similar fit to the linear model. Thus, we can expect to see similar results from the residuals. However, our goal is not necessarily to measure the percent change in CO2. We determined this earlier when looking at expressing longer-run growth rates as annualized averages. Therefore, we can expect to use a different model for our analysis.

In order to get an overall look at model fit, we can observe the AIC and BIC metrics for each of the models.

```
cbind(AIC(linear.trend.fit, quadratic.trend.fit, cubic.trend.fit, exp.trend.fit),
      BIC(linear.trend.fit, quadratic.trend.fit, cubic.trend.fit, exp.trend.fit))
```

```
##                    df       AIC df       BIC
## linear.trend.fit    3  2232.961  3  2245.406
## quadratic.trend.fit 4  2063.536  4  2080.129
## cubic.trend.fit     5  2033.187  5  2053.929
## exp.trend.fit       3 -3263.316  3 -3250.870
```

From the metrics, we can observe that each model has varying AIC and BIC values as well as different `df` values. We need to consider both accuracy and generalization in determining the best fit for our model.

Last, we can fit a polynomial time trend model that incorporates seasonal dummy variables. In this context,

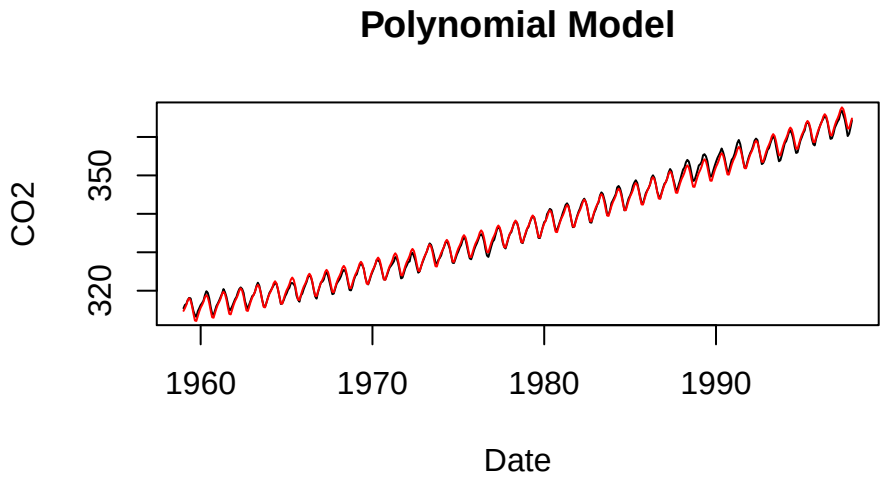
the seasons are expressed by `Month`. So, we start by fitting a simple linear model including components for the trend and season.

```
poly.trend.fit <- lm(CO2 ~ Time + Month + I(Time^2), data=carbon)
summary(poly.trend.fit)
```

```
##
## Call:
## lm(formula = CO2 ~ Time + Month + I(Time^2), data = carbon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99478 -0.54468 -0.06017  0.47265  1.95480
##
## Coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  3.172e+02  1.497e-01 2118.503  < 2e-16 ***
## Time         6.763e-02  9.929e-04   68.114  < 2e-16 ***
## MonthAug    -3.773e+00  1.640e-01  -23.011  < 2e-16 ***
## MonthDec    -3.476e+00  1.640e-01  -21.196  < 2e-16 ***
## MonthFeb    -1.874e+00  1.640e-01  -11.429  < 2e-16 ***
## MonthJan    -2.538e+00  1.640e-01  -15.480  < 2e-16 ***
## MonthJul    -1.705e+00  1.640e-01  -10.399  < 2e-16 ***
## MonthJun    -1.840e-01  1.640e-01   -1.122  0.26226
## MonthMar    -1.131e+00  1.640e-01   -6.899 1.77e-11 ***
## MonthMay     4.788e-01  1.640e-01    2.920  0.00367 **
## MonthNov    -4.592e+00  1.640e-01  -28.006  < 2e-16 ***
## MonthOct    -5.781e+00  1.640e-01  -35.257  < 2e-16 ***
## MonthSep    -5.598e+00  1.640e-01  -34.139  < 2e-16 ***
## I(Time^2)    8.865e-05  2.050e-06   43.242  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.724 on 454 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9977
## F-statistic: 1.531e+04 on 13 and 454 DF,  p-value: < 2.2e-16
```

Nearly all seasonal terms are statistically significant and the resulting adjusted r-squared value is greater than 0.99.

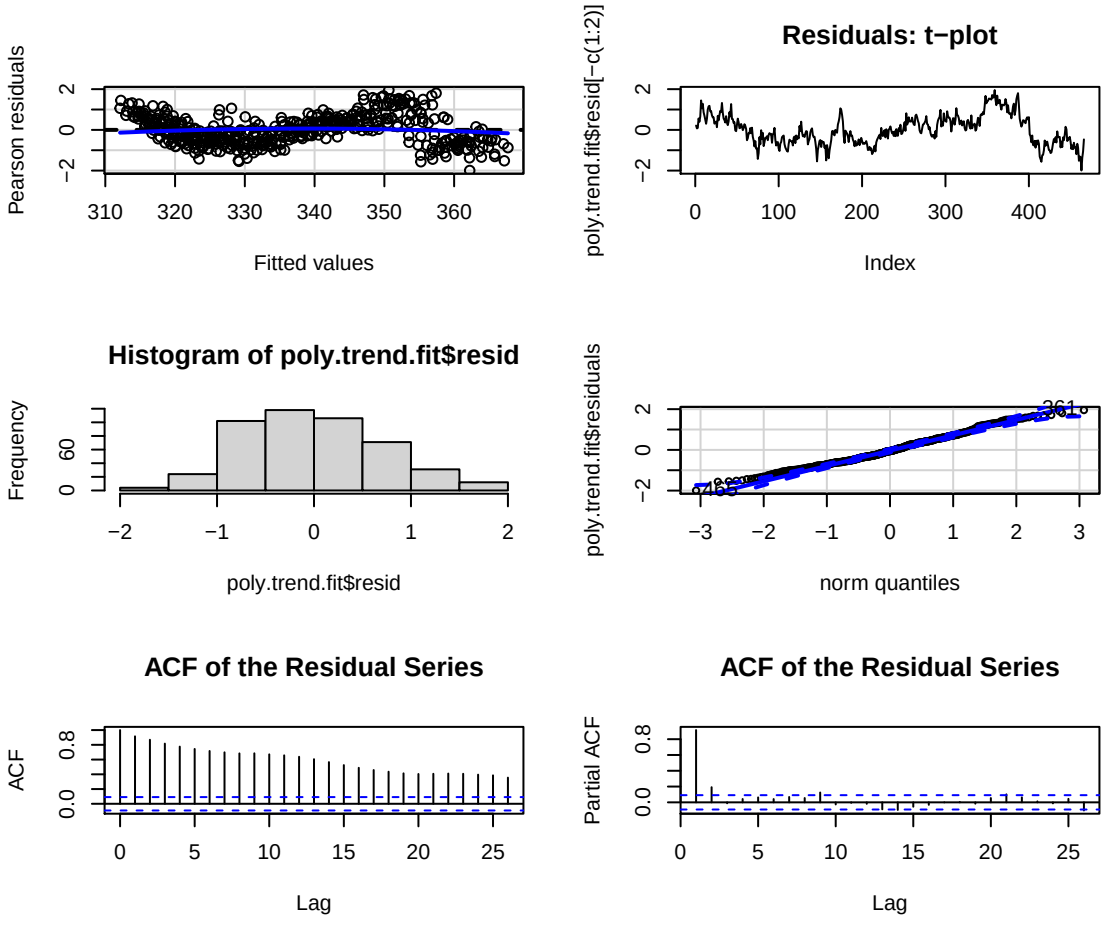
```
plot(CO2 ~ Date, data = carbon, type = "l", main = "Polynomial Model")
lines(fitted(poly.trend.fit) ~ Date, data = carbon, col = "red")
```




```
par(mfrow=c(3,2))
residualPlot(poly.trend.fit)
plot(poly.trend.fit$resid[-c(1:2)], type="l", main="Residuals: t-plot")
hist(poly.trend.fit$resid)
qqPlot(poly.trend.fit$residuals)
```

```
## [1] 465 361
```

```
acf(poly.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
pacf(poly.trend.fit$resid[-c(1:2)], main="ACF of the Residual Series")
```



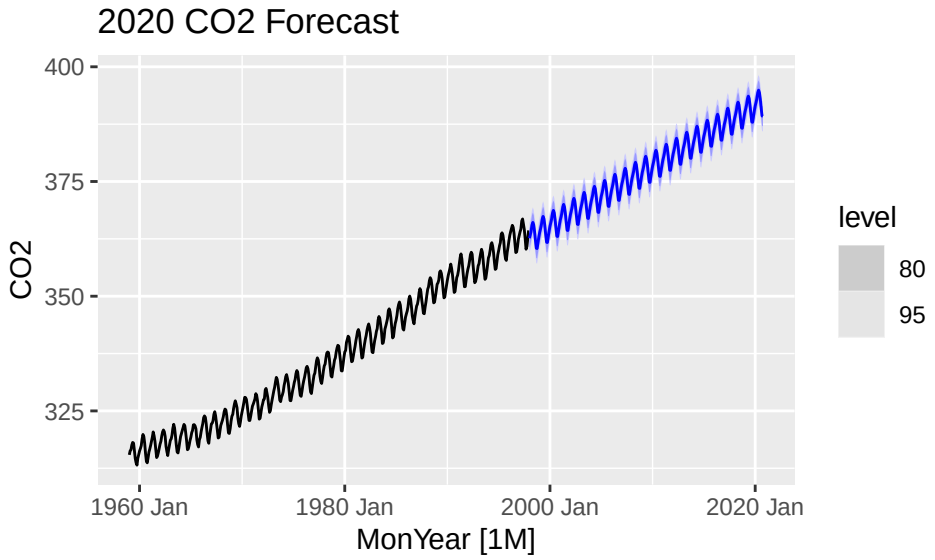
We see that a polynomial relationship that incorporates seasonality with time is strongly significant and accounts for the curves and variations seen in the original series. Visually, the polynomial model fits the trend of the series best out of all the models without over-fitting the data. The residual plots indicate a good model fit as the assumptions do not appear to be violated.

Therefore, we are able to use this model to generate forecasts to the year 2020.

```
carbon_ts <- as_tsibble(carbon, index = MonYear, regular = TRUE)
```

```
poly.trend.fcast<-carbon_ts%>%model(TSLM(CO2~trend()+season()))%>%forecast(h=274)
carbon_ts %>% autoplot(CO2) + autolayer(rbind(poly.trend.fcast)) +
  ggtitle('2020 CO2 Forecast')
```

```
## Warning: `rbind.fbl_ts()` is deprecated as of fabletools 0.2.0.
## Please use `bind_rows()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```



Our forecast shows that the model's trend and seasonality will continue persistently through the year 2020. The CO2 levels are forecasted to continue to oscillate in a similar manner. So, the projections predict that the CO2 levels will be around 380 to 400 ppm in 2020.

Part 3

Following all appropriate steps, choose an ARIMA model to fit to the series. Discuss the characteristics of your model and how you selected between alternative ARIMA specifications. Use your model (or models) to generate forecasts to the year 2020.

To start, we determine the appropriate level of differencing using a KPSS test to evaluate stationarity.

```
print("Original series: ")
```

```
## [1] "Original series: "
```

```
kpss.test(carbon_ts$CO2)
```

```
## Warning in kpss.test(carbon_ts$CO2): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  carbon_ts$CO2
## KPSS Level = 7.8173, Truncation lag parameter = 5, p-value = 0.01
```

```
print("First Difference: ")
```

```
## [1] "First Difference: "
```

```
kpss.test(diff(carbon_ts$CO2, 1))
```

```
## Warning in kpss.test(diff(carbon_ts$CO2, 1)): p-value greater than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(carbon_ts$CO2, 1)
## KPSS Level = 0.012352, Truncation lag parameter = 5, p-value = 0.1
```

The p-value for the original series is not significant, showing that series is not stationary. Testing the first difference, we see that the series becomes stationary and no higher degree of differencing is warranted. We will evaluate our ARIMA models using a first difference only.

Next, we build seasonal ARIMA models by varying the orders of autoregressive and moving average modeling. In order to optimize our model, we can run multiple `arima` models with different parameter values in a nested loop. We already know from our previous EDA and tests that we should take the first order difference of the series, so `d` and `D` will be set to 1. Therefore, our focus will be on optimizing the `AR` & `MA` components of the model (for both non-seasonal and seasonal).

There are several different metrics that we can use in order to determine the best fit for our model. These include the AIC, the AICc, and the BIC.

The AIC is an estimator of out-of-sample prediction error and thereby relative quality of statistical models for a given set of data. The AICc is AIC with a correction for small sample sizes. Unlike the AIC, the BIC penalizes the model more for its complexity, meaning that more complex models will have a worse (larger) score and will, in turn, be less likely to be selected.

In this context, we will use the AIC measure in order to determine the optimal parameters for our model. This is because we want to generate out-of-sample forecasts and do not have complex explanatory variables.

We evaluated each parameter from 0 to 4 with the loop below. But, for time considerations in knitting, the parameter search is limited to include parameters up to the best performing model only.

```
results <- data.frame(p=integer(),q=integer(),P=integer(),Q=integer(),
                      AICc=double(),AIC=double(),BIC=double())

for (p in 0:1){
  for (q in 0:1){
    for(P in 1:2){
      for (Q in 1:2){
        tryCatch(
        {
            mod <- carbon_ts %>% model(ARIMA(CO2 ~ 0 + pdq(p,1,q) + PDQ(P,1,Q)))
            results <- results %>% add_row(p=p,q=q,P=P,Q=Q,
                                           AICc=as.numeric(glance(mod)$AICc),
                                           AIC=as.numeric(glance(mod)$AIC),
                                           BIC=as.numeric(glance(mod)$BIC))
         },
         error=function(e) {
            print(paste('error encountered for', p, q, P, Q))
         }
        )
      }
    }
  }
}

min_aic <- results[which.min(results$AIC), ]
min_aicc <- results[which.min(results$AICc), ]
min_bic <- results[which.min(results$BIC), ]
df <- rbind(min_aic, min_aicc,min_bic)
rownames(df) <- c("Min. AIC", "Min. AICc", "Min. BIC")
df
```

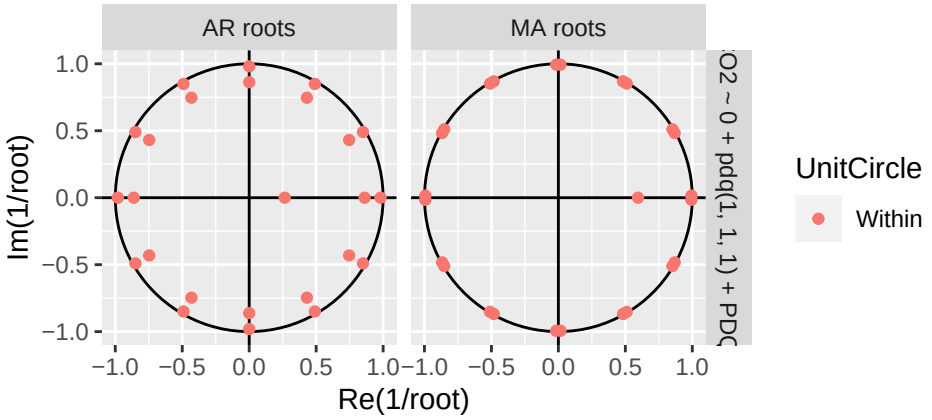
```
##           p q P Q     AICc      AIC      BIC
## Min. AIC  1 1 2 2 172.9589 172.7083 201.5504
## Min. AICc 1 1 2 2 172.9589 172.7083 201.5504
## Min. BIC  0 1 1 1 180.0124 179.9235 196.4047
```

The model which minimizes AICc is `ARIMA(1,1,1)(2,1,2)[12]`. As a result, we will build a model with parameters `p=1, q=1, P=2, & Q=2` as it produces the lowest AIC score. Then, we can evaluate the model below.

```
mod <- carbon_ts %>% model(ARIMA(CO2 ~ 0 + pdq(1,1,1) + PDQ(2,1,2)))
mod %>% report()
```

```
## Series: CO2
## Model: ARIMA(1,1,1)(2,1,2)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1    sma2
##       0.2652  -0.5950  0.9613  -0.1335  -1.8169  0.8564
## s.e.  0.1358   0.1154  0.0787   0.0603   0.0710  0.0622
##
## sigma^2 estimated as 0.08303:  log likelihood=-79.35
## AIC=172.71   AICc=172.96   BIC=201.55
```

```
gg_arma(mod)
```



The selected model in back-shift notation is below.

$$(1-0.2652B)(1-0.9613B^{12}+0.1335B^{24})(1-B)(1-B^{12})y_t = (1-0.5950B)(1-1.8169B^{12}+0.8564B^{24})\epsilon_t$$

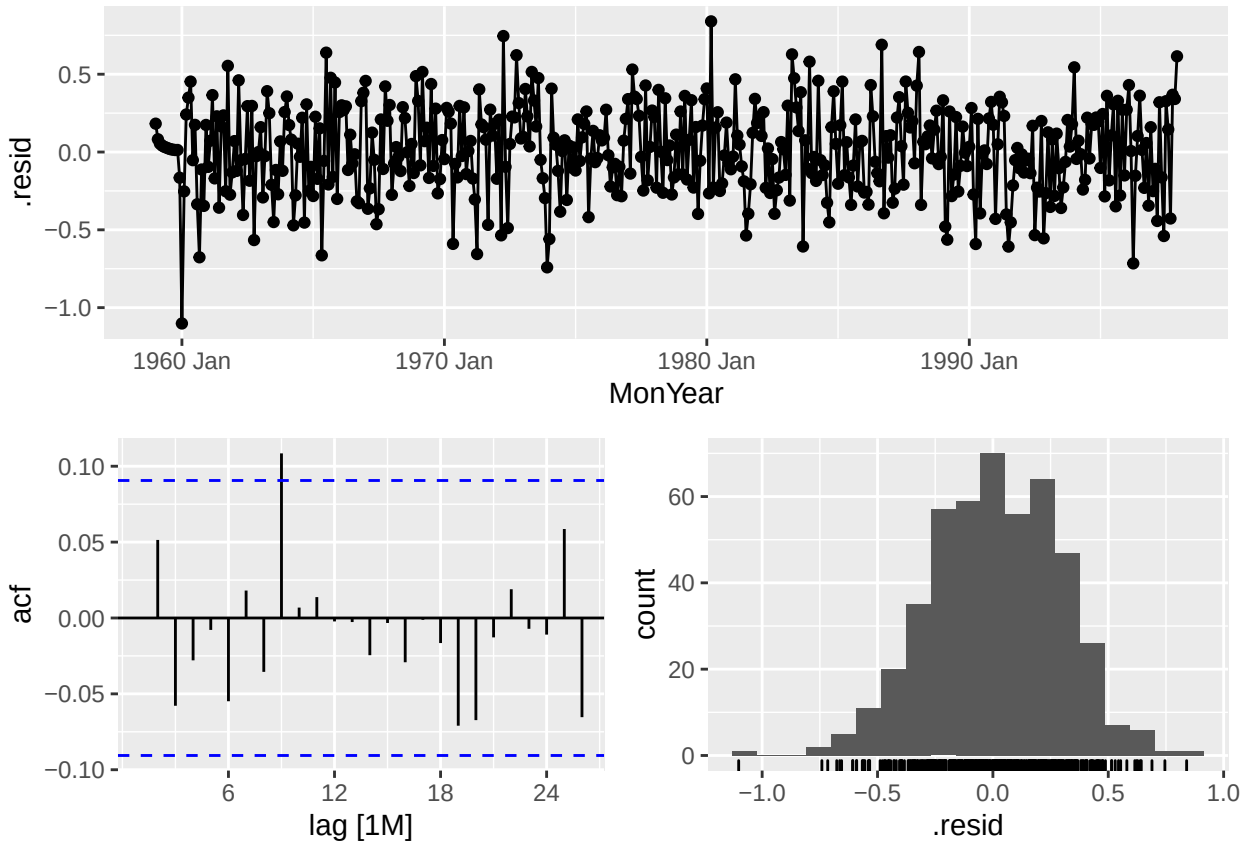
```
mod %>% accuracy()
```

```
## # A tibble: 1 x 9
##   .model                 .type      ME  RMSE   MAE     MPE   MAPE  MASE     ACF1
##   <chr>                  <chr>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>
## 1 ARIMA(CO2 ~ 0 + pdq(1~ Train~ 0.0101 0.282 0.228 0.00302 0.0677 0.180 -1.50e-4
```

All inverse characteristic roots are within the unit circle, which means that our model is indeed stationary. The model also appears to fit the data well with a root mean square error or 0.28, which indicates that the we should be able to generate accurate forecasts.

We can also evaluate the model residuals to assess the fit.

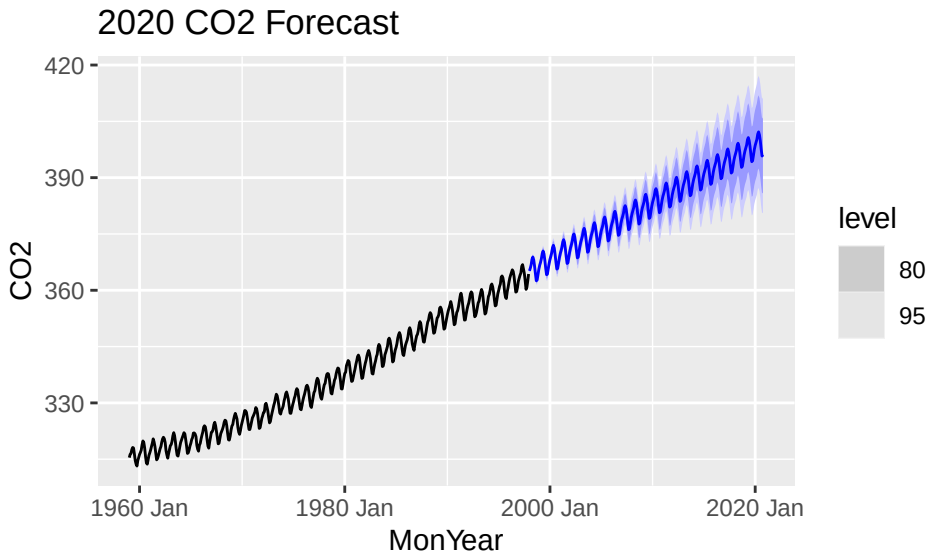
```
mod %>% gg_tsresiduals()
```



The residuals are normally distributed and do not have an apparent trend, indicating leftover white noise. The ACF does not have an apparent trend and oscillates around zero. As a result, our model satisfies all of the necessary assumptions.

We will then use our model below to forecast 2020 CO2 levels.

```
model_forecast <- mod %>% forecast(h = 274)
model_forecast %>% autoplot(carbon_ts) + ggtitle('2020 CO2 Forecast')
```



Visually, the model matches the persistent trend and seasonality that we observe in the data. One of the

limitations in the model is that CO2 is shown to grow linearly because of the use of first differencing only. Additionally, the confidence interval is extremely close for the initial years following the end of the series. The confidence interval does grow wider over time, but it stays within a relatively small range (+/- about 15 ppm in 2020). So, over a long period of time the model may predict lower than actual CO2 concentrations if the increase in CO2 concentration continues to accelerate.

Overall, we are able to fit an optimized ARIMA model for our `co2` time series data and forecast into 2020.

Part 4

Convert these data into a suitable time series object, conduct a thorough EDA on the data, and address the problem of missing observations. Describe how the Keeling Curve evolved from 1997 to the present and compare this to the predictions from your forecasts in Parts 2 and 3. Use the weekly data to generate a month-average series from 1997 to the present, and use this to generate accuracy metrics for the forecasts generated by your models from Parts 2 and 3.

The file `co2_weekly_mlo.txt` contains weekly observations of atmospheric carbon dioxide concentrations measured at the Mauna Loa Observatory from 1974 to 2020, published by the National Oceanic and Atmospheric Administration (NOAA).

```
co2_w <- read.table(paste0(here::here(),"/labs/lab_2/co2_weekly_mlo.txt"),
                    header = TRUE, quote="\"")
```

We can start by doing some basic exploration of the data set itself.

```
head(co2_w)
```

```
##   X1974 X5 X19 X1974.3795 X333.34 X6 X.999.99 X.999.99.1 X50.36
## 1  1974  5  26   1974.399  332.95  6  -999.99    -999.99  50.06
## 2  1974  6   2   1974.418  332.32  5  -999.99    -999.99  49.57
## 3  1974  6   9   1974.437  332.18  7  -999.99    -999.99  49.63
## 4  1974  6  16   1974.456  332.37  7  -999.99    -999.99  50.07
## 5  1974  6  23   1974.475  331.59  6  -999.99    -999.99  49.60
## 6  1974  6  30   1974.495  331.68  6  -999.99    -999.99  50.04
```

We can then perform some data transformations to reformat the necessary information from our data set.

```
co2_w <- co2_w[, c(1, 2, 3, 5)]
names(co2_w) <- c('year', 'month', 'day', 'co2ppm')
co2_w$date <- as.Date(paste(co2_w$year, co2_w$month, co2_w$day,
                            sep = '-'),format = '%Y-%m-%d')
co2_w$week <- lubridate::week(ymd(co2_w$date))
head(co2_w)
```

```
##   year month day co2ppm       date week
## 1 1974     5  26 332.95 1974-05-26   21
## 2 1974     6   2 332.32 1974-06-02   22
## 3 1974     6   9 332.18 1974-06-09   23
## 4 1974     6  16 332.37 1974-06-16   24
## 5 1974     6  23 331.59 1974-06-23   25
## 6 1974     6  30 331.68 1974-06-30   26
```

```
summary(co2_w)
```

```
##       year          month             day            co2ppm
##  Min.   :1974   Min.   : 1.000   Min.   : 1.00   Min.   :-1000.0
##  1st Qu.:1985   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.:  346.7
##  Median :1997   Median : 7.000   Median :16.00   Median :  364.5
##  Mean   :1997   Mean   : 6.532   Mean   :15.71   Mean   :  356.2
##  3rd Qu.:2009   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:  386.9
##  Max.   :2020   Max.   :12.000   Max.   :31.00   Max.   :  417.4
##       date                 week
##  Min.   :1974-05-26   Min.   : 1.00
##  1st Qu.:1985-12-29   1st Qu.:14.00
##  Median :1997-08-03   Median :27.00
```

```
##  Mean   :1997-08-03   Mean   :26.63
##  3rd Qu.:2009-03-08   3rd Qu.:40.00
##  Max.   :2020-10-11   Max.   :53.00
```

There seems to be an issue with the `co2ppm` variable as there is a minimum of -1000.

The original file says that [-999.99 = no data]. So, the value -999.99 is being used to mean that data isn't available. We don't want that value in our calculations, so in R, we use a special value called NA instead. Then, we can use the `na.locf()` function in order to replace each NA with the most recent non-NA prior to it. Each observation is important, so we can't just remove the null values. We also know that there is a strong linear trend in the series. As a result, we can justifiably replace the null value with the previous non-null value in our series.

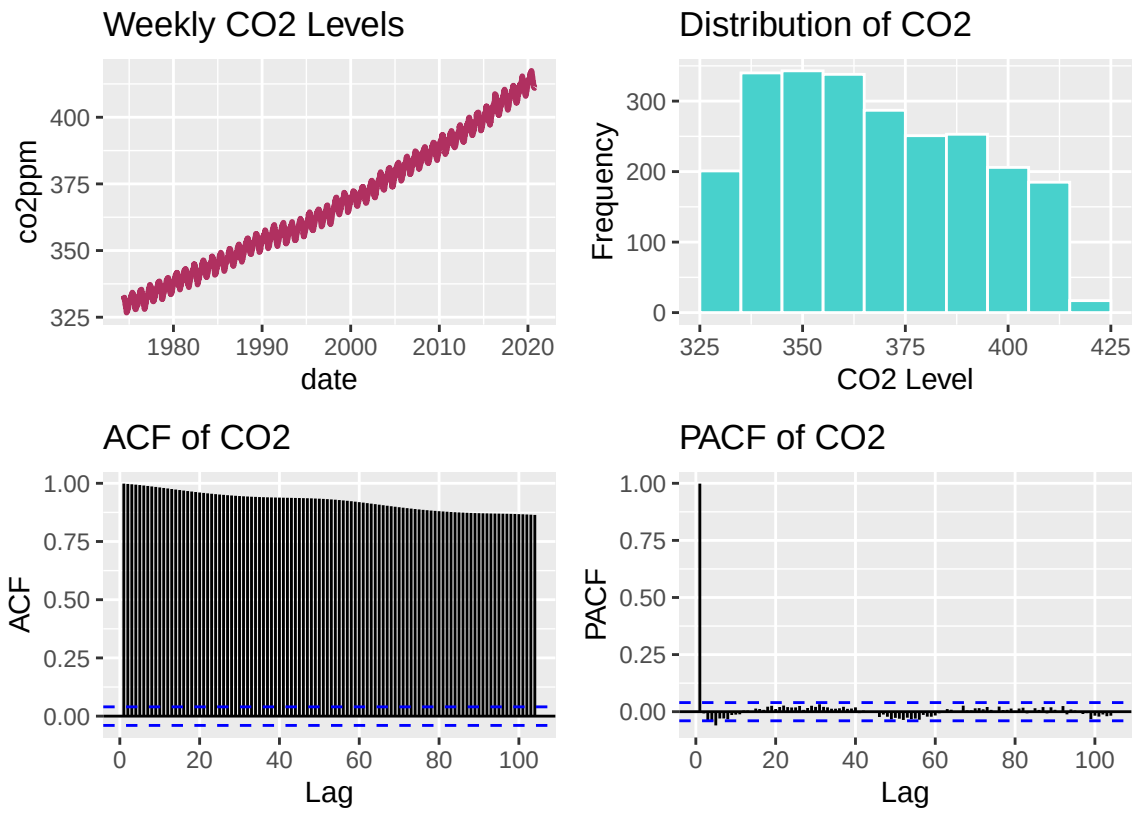
```
co2_w[co2_w$co2ppm == -999.99, ]$co2ppm = NA
co2_w$co2ppm <- na.locf(co2_w$co2ppm)
```

After cleaning our data, we can convert the data into suitable time series objects.

```
carbon_ts_we <- as_tsibble(co2_w, index = date, regular = TRUE)
cts <- ts(co2_w$co2ppm, frequency=52.18, start=c(1974,21))
```

Now, we can conduct a thorough EDA on the data.

```
p1 <- co2_w %>% ggplot(aes(x=date,y=co2ppm)) + geom_line(colour="maroon",size=1) +
  ggtitle('Weekly CO2 Levels') + theme(legend.position = "none")
p2 <- ggplot(co2_w, aes(co2ppm)) +
  geom_histogram(binwidth = 10, fill = "mediumturquoise",col="white",size = 0.5)+
  labs(title="Distribution of CO2", x = "CO2 Level",y="Frequency")
p3 <- ggAcf(co2_w$co2ppm, lag.max = 104, main = "ACF of CO2")
p4 <- ggPacf(co2_w$co2ppm, lag.max = 104, main = "PACF of CO2")
egg::ggarrange(p1, p2, p3, p4, nrow = 2)
```



From the plots, we can see that:

- the time series of `co2_w` displays a very strong upward trend with strong seasonality. The fluctuation between each season appears to be quite consistent and pronounced.
- the histogram of `co2_w` does not necessarily have a normal distribution. Instead, the values appear to be somewhat uni-modal with a slight positive right skew towards lower CO2 levels.
- the correlogram of ACF of `co2_w` has a gradual decline, with a small seasonal effect. Yet, the ACF does display some persistence. There are signs of non-stationarity (and the need for differencing) as we see the ACF tapering very slowly.
- the correlogram of PACF of `co2_w` has a sharp decline after the first lag. However, the PACF then gradually oscillates and declines.

Collectively, these plots demonstrate the non-stationary of the series. This is a result of trend and seasonality, yet there are no obvious irregular elements.

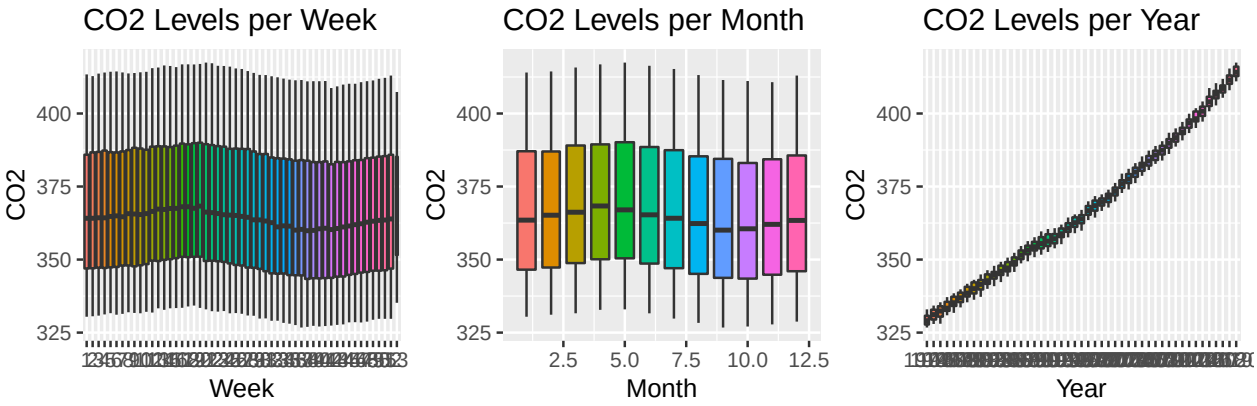
As there is apparent trend and seasonality in our series, we can take a greater look into those features. So, we can start by computing the weekly, monthly, and annual averages to give gross descriptions of the series.

```
t1 <- head(aggregate(co2_w$co2ppm,by=list(Week=co2_w$week),mean))
t2 <- head(aggregate(co2_w$co2ppm,by=list(Month=co2_w$month),mean))
t3 <- head(aggregate(co2_w$co2ppm,by=list(Year=co2_w$year),mean))
grid.arrange(tableGrob(t1), tableGrob(t2), tableGrob(t3), ncol = 3)
```

	Week	x		Month	x		Year	x
1	1	367.142826086957	1	1	367.339704433498	1	1974	329.3678125
2	2	367.319565217392	2	2	368.080268817204	2	1975	331.039615384615
3	3	367.510869565217	3	3	369.382352941176	3	1976	332.085769230769
4	4	367.726956521739	4	4	370.500812182741	4	1977	333.830576923077
5	5	367.967391304348	5	5	370.693317073171	5	1978	335.411886792453
6	6	368.059130434783	6	6	369.296435643564	6	1979	336.896346153846

From the tables, we can see how the CO2 values gradually change overtime. We can get a better understanding of this through box-plots.

```
p1 <- ggplot(co2_w,aes(as.factor(week),co2ppm))+theme(legend.position="none") +
  geom_boxplot(varwidth=T, aes(fill = as.factor(week))) + labs(x="Week",y="CO2") +
  labs(title="CO2 Levels per Week")
p2 <- ggplot(co2_w,aes(month,co2ppm))+theme(legend.position="none") +
  geom_boxplot(varwidth=T, aes(fill = as.factor(month))) + labs(x="Month",y="CO2") +
  labs(title="CO2 Levels per Month")
p3 <- ggplot(co2_w,aes(as.factor(year),co2ppm))+theme(legend.position="none") +
  geom_boxplot(varwidth=T, aes(fill = as.factor(year))) + labs(x="Year",y="CO2") +
  labs(title="CO2 Levels per Year")
egg::ggarrange(p1, p2, p3, nrow = 1)
```



From the box-plots, we can see that:

- with respect to change by the week, there is some seasonal fluctuation in the CO2 levels. The levels appear to increase from Week 42 to 24 and then decrease from Week 25 to 41. So it appears, that the average of the CO2 levels is larger in spring and summer months than in fall and winter months.
- with respect to the change by the month, there is some seasonality in the CO2 levels. There appears to be a fluctuation in levels as they increase from October to May then decrease from June to September. So it also appears, that the average of the CO2 levels is larger in spring and summer months than in fall and winter months.
- with respect to the change by the year, there is a very strong and consistent upward trend in the CO2 levels.

These results further indicate trend and seasonality in our series, making it non-stationary. This is in line with Kealing's conclusions as well. However, the seasonality does not change too drastically over the weeks and months, so there is no need to apply the Box-Cox method to our ARMIA model, as the Box-Cox transformation is designed to reduce non-normality of the errors in a model.

Therefore, we can statistically test for a unit root in our series. In the context of an ADF test, H_0 is the presence of a unit root, with rejection (H_A) indicating that the series does not have a unit root.

```
adf.test(cts)
```

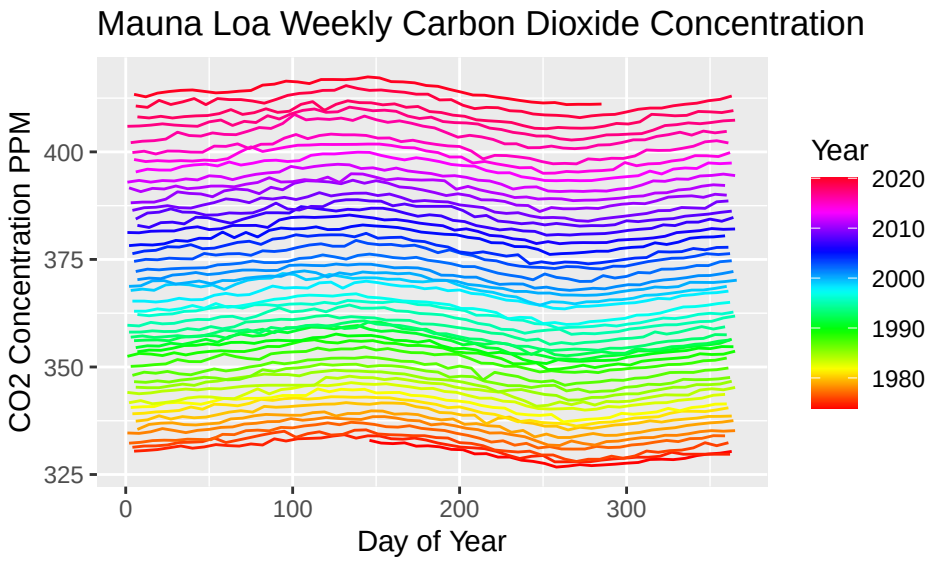
```
## Warning in adf.test(cts): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cts
## Dickey-Fuller = -7.3439, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

The resulting p-value of the ADF test is 0.01, which is greater than $\alpha = 0.05$. So, we can reject the null hypothesis that the series does have a unit root. This is an indication that the series is actually stationary.

Yet, we can still explore different aspects of the data. So, we can also visualize the time series for each individual year in order to compare how the seasonality impacts the series.

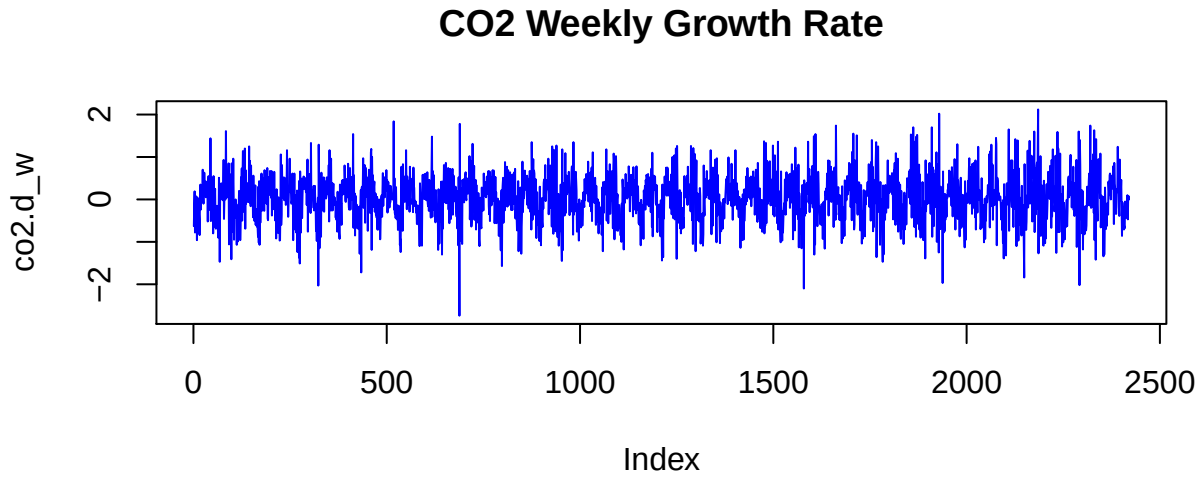
```
co2_w$yday <- yday(co2_w$date)
ggplot(data = co2_w, aes(yday, co2ppm, colour = year, group = year)) +
  geom_line() + xlab('Day of Year') + ylab('CO2 Concentration PPM') +
  ggtitle('Mauna Loa Weekly Carbon Dioxide Concentration') +
  scale_color_gradientn('Year', colors = rainbow(length(unique(co2_w$year))))
```



From the plot above, we can see that the weekly and monthly seasonality is very consistent across each year as each line generally curves in the same way. We can also observe that the CO2 levels increase with each year.

We must also consider taking the difference of our series, by a certain order, to ensure stationarity. So, to start, we can look at the first difference of our series. That way we can investigate the trend, seasonality, and irregular elements in the differenced series as well. However, we will now be examining the CO2 growth rates rather than the levels, which we had previously been looking at.

```
co2.d_w <- diff(co2_w$co2ppm)
plot(co2.d_w, main = "CO2 Weekly Growth Rate", col = "blue", type = "l")
```



As mentioned earlier, the original data's ACF suggested the need to difference the series to remove the stochastic drift. This plot shows that after we take the first difference, there's no obvious stochastic drift noticeable over longer time intervals. Although the seasonal averages and box-plots do not show the strong seasonal component exhibited in the raw data, we can see the strong seasonality very clearly in the first order difference of the series. Additionally, with first-differencing applied to the series, the trend is eliminated and the variance is largely subdued. Thus, the first order difference is stationary. This will be important for our model moving forward.

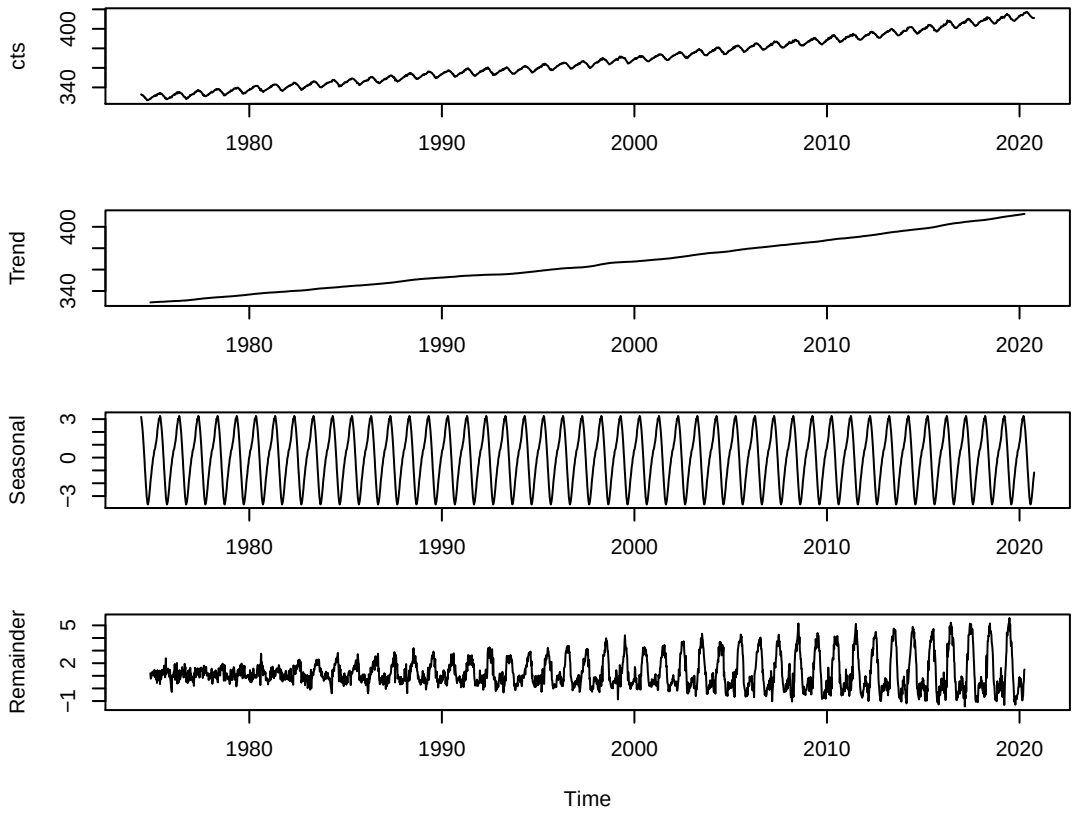
We can also decompose the original time series into trend, seasonal and error components. This will help improve our understanding of the time series and it can also be used to improve forecast accuracy. We will,

once again, use employ the additive decomposition for our model.

```
co2.deco_w <- decompose(cts, type = "additive")
```

We can then plot our results.

```
layout(matrix(1:4, ncol = 1))
op <- par(oma = c(2,0,0,1) + 0.1, mar = c(2,4,2,2) + 0.1, xpd = NA)
plot(cts, xlab = "")
plot(co2.deco_w$trend, xlab = "", ylab = "Trend")
plot(co2.deco_w$seasonal, xlab = "", ylab = "Seasonal")
plot(co2.deco_w$random, ylab = "Remainder")
```




```
par(op)
```

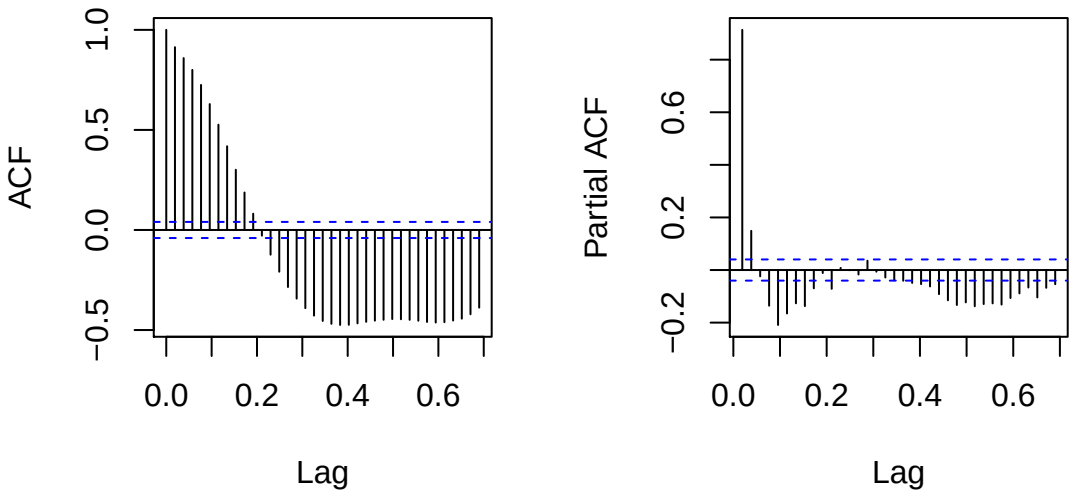
From the plots, we can see that:

- the overall CO2 levels have an upward trend with seasonality.
- the CO2 level trend is strong and upward.
- the CO2 level seasonality fluctuates consistently.
- the variation in the remainder no longer appears like white noise. There is persistent oscillation that seems to grow larger in frequency over time.

Therefore, we must consider the strong trend as well as the random element in building our model. Additionally, the consistency of the seasonal component provides further evidence that we do not need to apply the Box-Cox method to our ARIMA model.

Due to its non-random behavior, we can take a deeper look at the ACF and PACF of the random elements of the series.

```
layout(matrix(1:2, ncol = 2))
acf(co2.deco_w$random, na.action = na.omit, lag.max = 36, main = "")
pacf(co2.deco_w$random, na.action = na.omit, lag.max = 36, main = "")
```



From the correlograms, we can see that:

- the ACF very gradually declines and somewhat oscillates around zero. This means we can expect a negative AR parameter for the random elements.
- the PACF also gradually declines and somewhat oscillates around zero. Yet, the PACF drastically cuts off after the first lag, so we know we should use a first order model.

As a result, the random element of the series still does not appear to act like white noise. This will be important to consider for building our model.

However, from 1997, the Keeling Curve seems to have not changed much as it continued a similar trajectory as before. In general, the curve still rises steadily with fluctuations of seasonality. These results are also reflected in our forecasts from `Part 2 & 3`.

Now, we can use the weekly data to generate a month-average series from 1997 to the present.

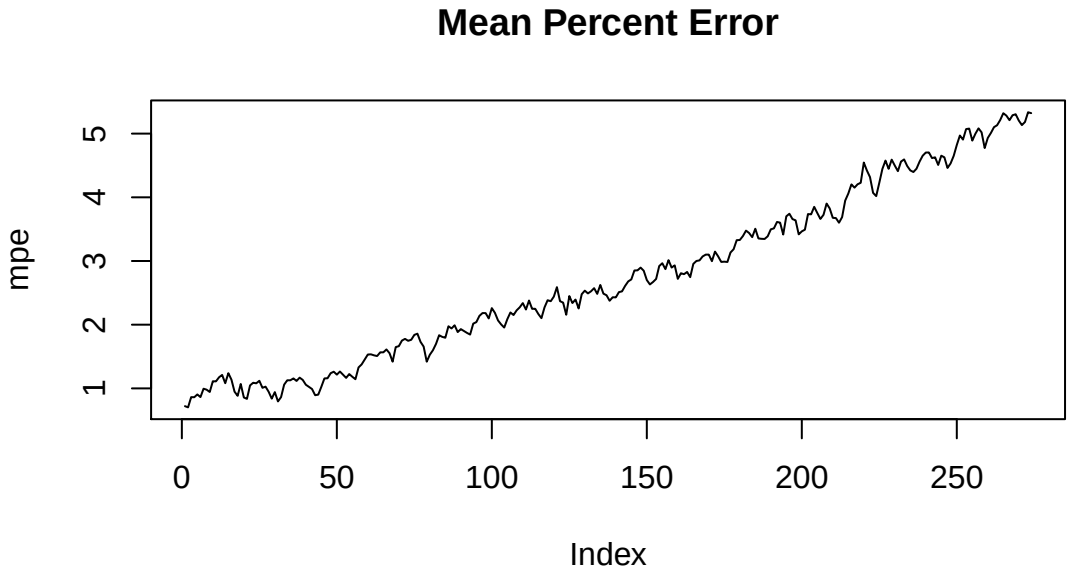
```
Month <- aggregate(co2_w$co2ppm, by = list(Month = co2_w$month,
                                           Year = co2_w$year), mean, na.rm=TRUE)
```

This can be used to generate accuracy metrics for the forecasts generated by our models from `Parts 2 & 3`.

In order to measure accuracy, we will use the Mean Percent Absolute Error (MAPE) metric. Percentage errors have the advantage of being unit-free, and so are frequently used to compare forecast performances between data sets. In statistics, the mean absolute percentage error is the computed absolute average of percentage errors by which forecasts of a model differ from actual values of the quantity being forecast. This metric looks at the relative error in the forecast (based on monthly data obtained in Parts 2, 3) and 'actual' measurement based on taking monthly average of the weekly data. This metric performs better in the case when the forecast and actual measurements have a trend, as we expect the MSE or RMSE to be biased due the trend. The MPE should address that limitation to some extent.

```
fcast_time <- Month[which(Month$Year > 1997),]
fcast_resid <- fcast_time$x - abs(poly.trend.fcast$.mean)
```

```
mpe <- ((fcast_time$x - poly.trend.fcast$.mean)/fcast_time$x) * 100
plot(mpe, type="l", main="Mean Percent Error")
```



From the plot above, we can see that, over time, the forecast accuracy got worse. Compared to our forecasts generated by our models from `Parts 2` and `3`, the difference between the actual value and the forecasted value is initially small. Yet, this percent difference grows almost linearly over time. The percent error doesn't grow too large though, as it goes from about 0.2% to around 5% error from 1997 to the present. These results makes sense with regards to long term forecasting. To some extent, we do expect the MPE to increase as our prediction become increasing further out of the original data on which the model was estimated. This could be due to changes in the trend itself, perhaps due to increased carbons emissions rates as emerging economies become larger contributors to emissions.

Part 5

Seasonally adjust the weekly NOAA data, and split the seasonally-adjusted (SA) series into training and test sets, using the final two years of observations as the test set.

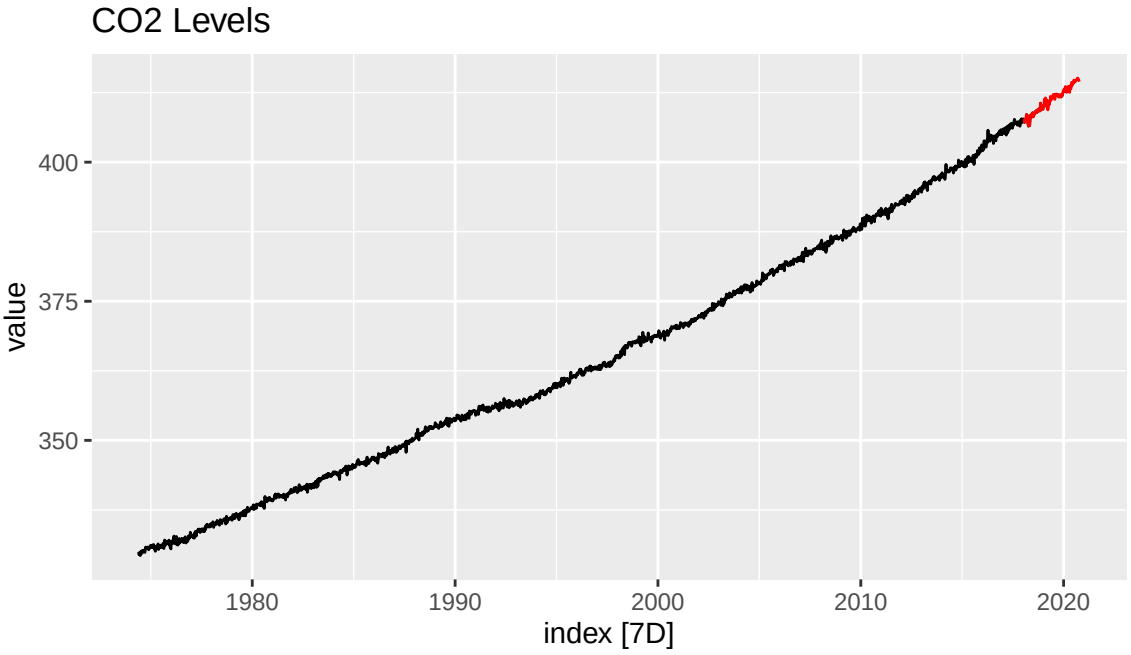
To start, we will remove the seasonality from the model.

```
carbon_ts_we <- as_tsibble(co2_w, index = date, regular = TRUE)
dcmp <- carbon_ts_we %>% model(STL(co2ppm))
my_co2 <- components(dcmp)[, c('date', 'season_adjust')]
my_co2 <- rename(my_co2, index = date, value = season_adjust)
```

Then, we can split the seasonally-adjusted series into a training and test set and plot the results.

```
co2.training <-  my_co2 %>% filter_index('1974'~'2018')
co2.test <- my_co2 %>% filter_index('2018'~.)
```

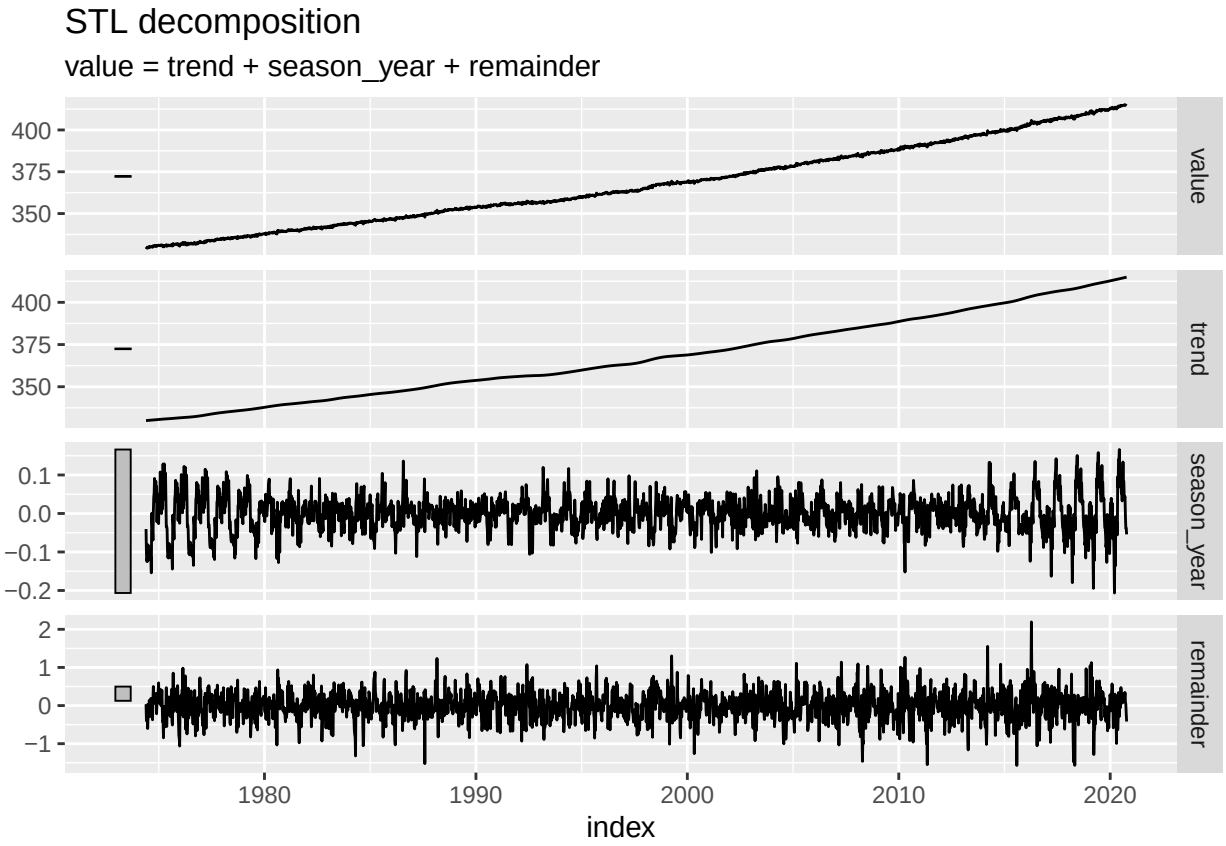
```
co2.training %>% autoplot(value) +
  autolayer(co2.test, value, colour = 'red') + ggtitle("CO2 Levels")
```



In the seasonally-adjusted series, we can still see the persistent trend but without any of the seasonal fluctuations.

We can also decompose the model in order to assess the different components.

```
my_co2 %>% model(STL(value)) %>% components() %>% autoplot()
```



From the plots, we can see that:

- the overall CO2 levels have an upward trend with seasonality.
- the CO2 level trend is strong and upward.
- the CO2 level seasonality fluctuates, but not consistently. The variation is greater at each end and slightly smaller in the middle. But, the overall variation scale is quite small.
- the CO2 levels appear to have a random white noise element in the series as well. The variation in the remainder component is small compared to the variation in the data.

Therefore, we must consider the strong trend in building our model. Additionally, the seasonal and remainder components provide evidence that we do not need to apply the Box-Cox method to our ARIMA model.

Next, we can fit an ARIMA model to the SA-series. We will do this by utilizing another loop to vary the model parameters. We will optimize these parameters according to how the model performs in-sample and pseudo out-of-sample. This will be accomplished by observing both the AIC and the MAPE metrics. Our goal will be to minimize the two metrics.

```
results <- data.frame(p=integer(),q=integer(),P=integer(),Q=integer(),
                      AIC=double(), mpe=double())

for (p in 1:2){
  for(q in 1:2){
    for(P in 0:1){
      for(Q in 0:1){
        tryCatch(
        {
            mod <- co2.training %>% model(ARIMA(value ~ 0 + pdq(p,1,q) + PDQ(P,1,Q)))
            mpe <- mod %>% accuracy()
```

```
            mpe <- mpe$MAPE

            results <- results %>% add_row(p=p,q=q,P=P,Q=Q,
                                           AIC=as.numeric(glance(mod)$AIC),
                                           mpe=mpe)
         },
         error=function(e) {
            print(paste('error encountered for', p, q, P, Q))
         }
        )
      }
    }
  }
}

min_aic <- results[which.min(results$AIC), ]
min_mpe <- results[which.min(results$mpe), ]
df <- rbind(min_aic, min_mpe)
rownames(df) <- c("Min. AIC", "Min. MAPE")
df
```

```
##           p q P Q      AIC       mpe
## Min. AIC  2 2 0 0 2009.069 0.0773977
## Min. MAPE 2 2 0 0 2009.069 0.0773977
```

The model selection function results in non-seasonal `p` and `q` values of 2 and 2, with seasonal `P` and `Q` values of 0 and 0. This makes sense in this context as we have removed the seasonality from our series.

```
mod <- co2.training %>% model(ARIMA(value ~ 0 + pdq(2,1,2) + PDQ(0,1,0)))
mod %>% report()
```

```
## Series: value
## Model: ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2033  -0.2042  -1.8288  0.8309
## s.e.  0.0274   0.0275   0.0168  0.0170
##
## sigma^2 estimated as 0.1384:  log likelihood=-999.53
## AIC=2009.07   AICc=2009.09   BIC=2037.83
```

This model in back-shift notation is below.

$$(1 - 1.2033B + 0.2042B^2)(1 - B)y_t = (1 - 1.8288B + 0.8309B^2)\epsilon_t$$

```
mod %>% accuracy()
```

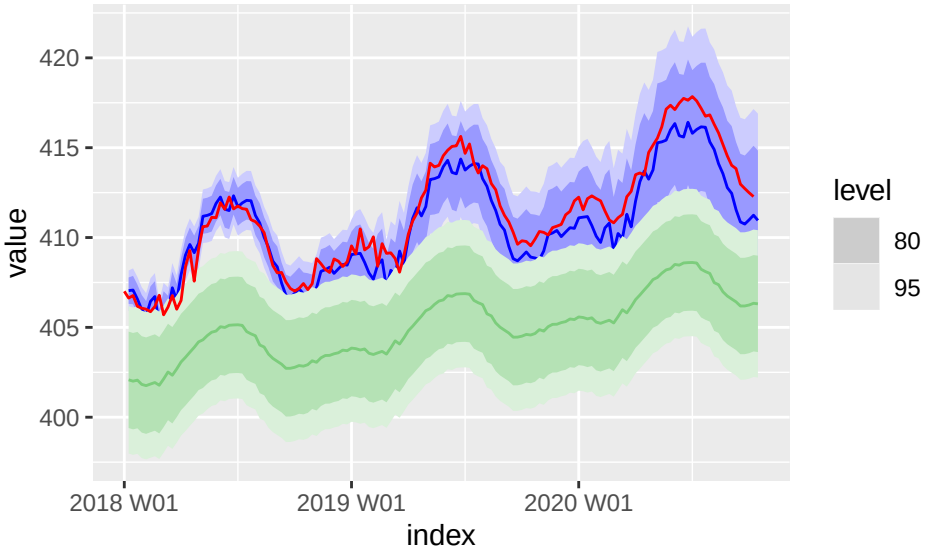
```
## # A tibble: 1 x 9
##   .model                   .type      ME  RMSE   MAE     MPE   MAPE  MASE   ACF1
##   <chr>                    <chr>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>  <dbl>
## 1 ARIMA(value ~ 0 + pdq(2~ Train~ 0.0177 0.372 0.283 0.00474 0.0774 0.852 0.0159
```

Our model has an AIC value of 2009.07 and a pseudo out-of-sample MAPE of 0.08 and RMSE of 0.37.

In addition, we can fit a polynomial time-trend model to the SA series and compare its performance to that of our ARIMA model.

```
carbon_ts_we <- as_tsibble(co2_w, index = date, regular = TRUE)
cts <- ts(co2_w$co2ppm, frequency=52.18, start=c(1974,21))
co2.deco_w <- decompose(cts, type = "additive")
cts <- cts - co2.deco_w$seasonal
my_co2 <- as_tsibble(cts, index = date, regular = TRUE)
co2.training <-  my_co2 %>% filter_index('1974'~'2018')
co2.test <- my_co2 %>% filter_index('2018'~.)
mod <- co2.training %>% model(ARIMA(value ~ 0 + pdq(2,1,2) + PDQ(0,1,0)))
```

```
poly.trend<-co2.training%>%model(TSLM(value~trend()+season()))%>%forecast(h=145)
model_forecast <- mod %>% forecast(h = 145)
model_forecast %>% autoplot() + autolayer(rbind(poly.trend),colour="palegreen3") +
  autolayer(co2.test, value, colour = 'red')
```



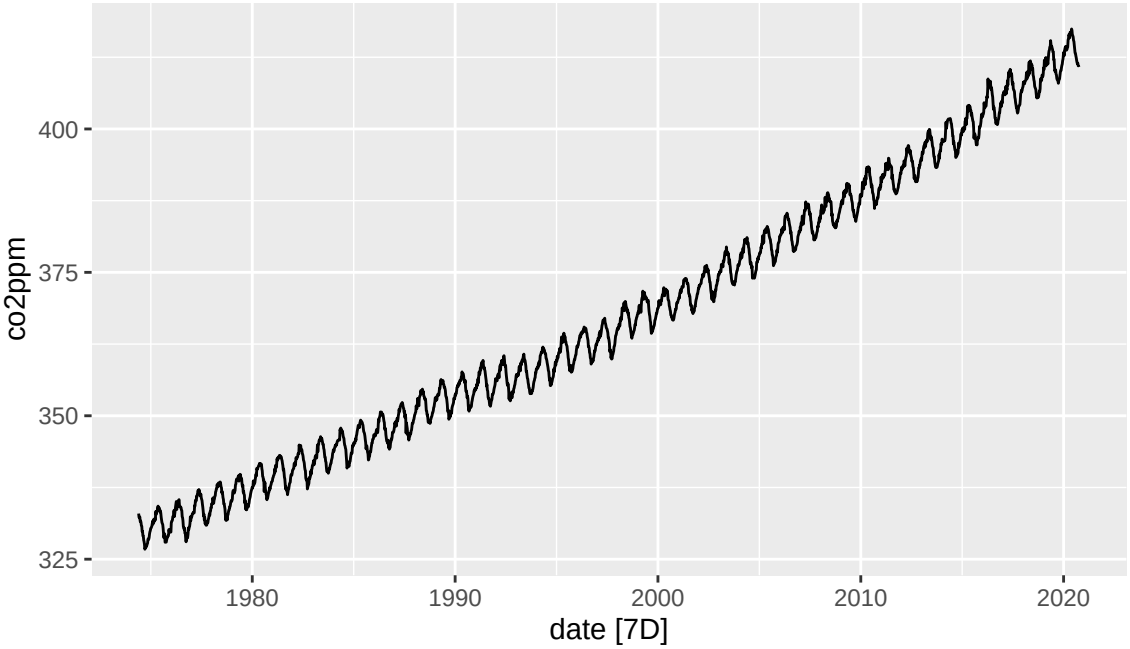
As a result, we can see that our model (blue) performs better than the polynomial model (green) with respect to the actual test values (red). Therefore, our ARIMA model is a better fit and produces better forecasts for our time series.

Part 6

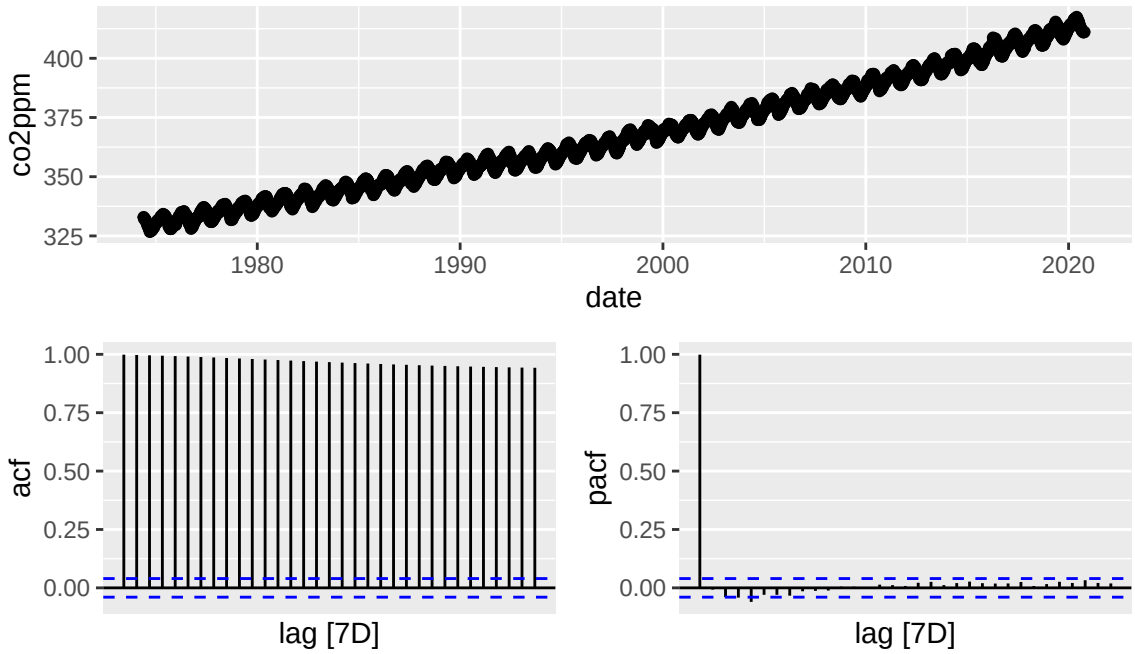
Fit an ARIMA model to the original (NSA) weekly series following all appropriate steps. Generate predictions for when atmospheric CO2 is expected to be at 420 ppm and 500 ppm levels for the first and final times, considering prediction intervals as well as point estimates in your answer. Generate a prediction for atmospheric CO2 levels in the year 2100. How confident are you that these will be accurate predictions?

We begin the ARIMA model fitting process with an initial examination of the weekly carbon dioxide concentration measurements, as well as it's autocorrelation and partial autocorrelation plots.

```
carbon_ts_we <- as_tsibble(co2_w, index = date, regular = TRUE)
carbon_ts_we %>% autoplot(co2ppm)
```

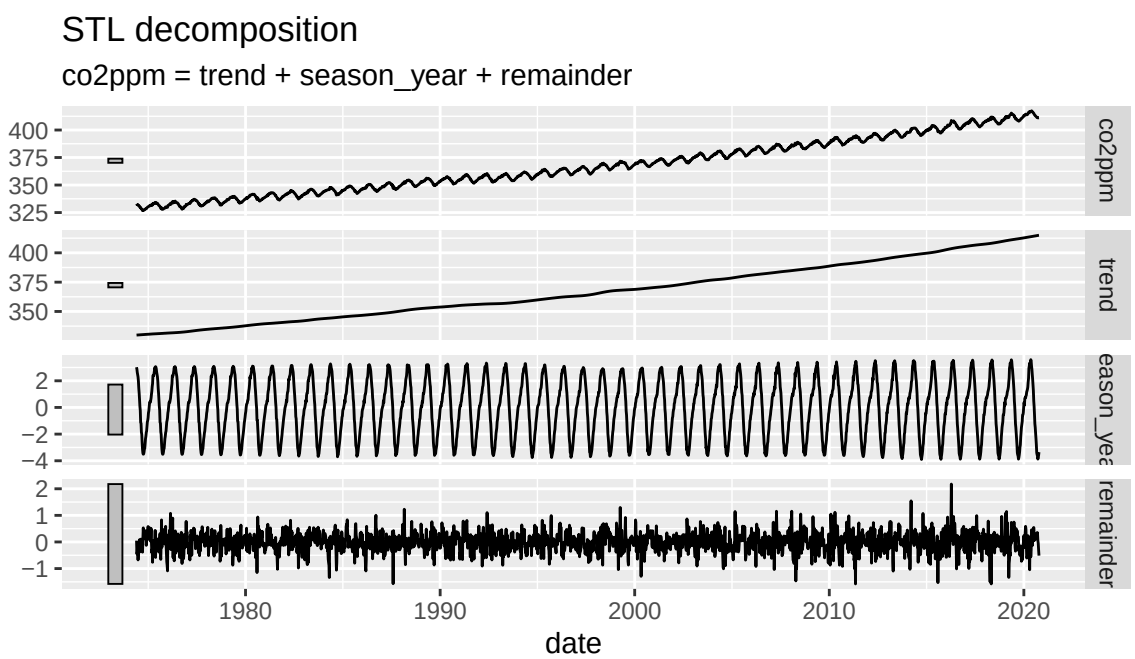



```
carbon_ts_we %>% gg_tsdisplay(y = co2ppm, plot = 'partial')
```



The carbon dioxide concentration measurements show a clear upward trend, as well as clear yearly seasonality. The autocorrelation decays slowly and the variance over time appears relatively unchanging. To expand an understanding of the different components of the data, we graph decomposition of trend, seasonality, and random components using the STL additive decomposition method.

```
carbon_ts_we %>% model(STL(co2ppm)) %>% components() %>% autoplot()
```

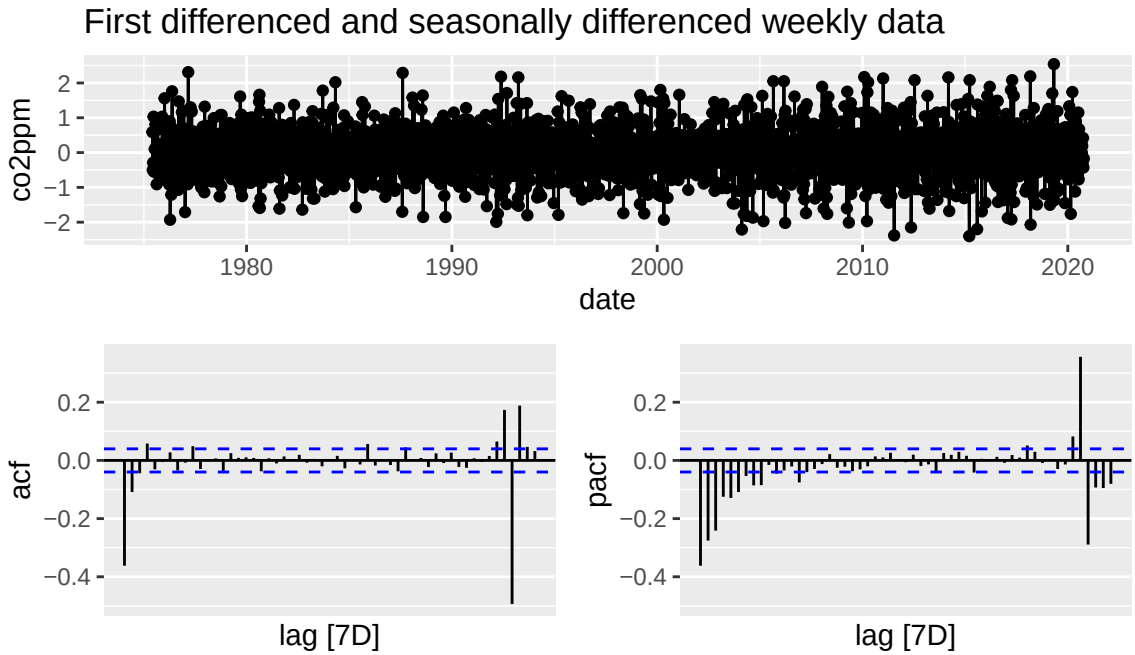


The above plots reiterate the general positive trend, the yearly seasonality with relatively unchanging variance, and a random looking remainder section. The unchanging variance eliminates the need to do any log or Box-Cox transformations.

In order to meet the requirements of stationarity for ARIMA models, we will next examine the carbon dioxide concentration data seasonally differenced by 52 weeks (as a reasonable estimate of yearly seasonality), and

then first difference the series in order to stabilize the mean.

```
carbon_ts_we %>% gg_tsdisplay(co2ppm %>% difference(52) %>% difference(),
                              plot_type='partial', lag_max = 55) + ylab("co2ppm") +
  ggtitle("First differenced and seasonally differenced weekly data")
```



The correlograms of the first and seasonal-differenced series show significant spikes in ACF and PACF around lag 52, potentially suggesting a seasonal MA(1) and/or AR(1) component. The PACF plot also exponentially decays, possibly suggesting a nonseasonal MA(1) component. As a result, we will start with a base model of $ARIMA(0,1,1)(1,1,1)_{52}$.

We can also determine the appropriate level of differencing using a KPSS test to evaluate stationarity.

```
carbon_ts_we %>% features(co2ppm %>% difference(52) %>% difference(),
                          unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1   0.00316         0.1
```

Running a KPSS test results in an inability to reject the null hypothesis of stationarity at five percent significance, confirming that no more differencing is required for ARIMA model selection.

Next, we split the data into training and test sets. From the initial examination of the data and it's components, the overall trend and seasonality do not change significantly over time. Therefore, for simplification of ARIMA model choice, we limit the scope of analysis to data from the last 20 years. Of this most recent 20 years, we divide data from October of 1998 to October of 2018 to the training set, and data from October of 2018 to October of 2020 to the test set.

```
carbon_ts_we_training <- carbon_ts_we %>%filter_index('1998-10'~'2018-10')
carbon_ts_we_test <- carbon_ts_we %>%filter_index('2018-10'~.)
```

Now, we can create our base model and examine in-sample-performance and out-of-sample performance.

```
base_mod <- carbon_ts_we_training %>% model(ARIMA(co2ppm ~ pdq(0,1,1) +
                                                  PDQ(1,1,1, period=52)))
```

```
base_mod %>% report()
```

```
## Series: co2ppm
## Model: ARIMA(0,1,1)(1,1,1)[52]
##
## Coefficients:
##           ma1     sar1     sma1
##       -0.6624  -0.0456  -0.8248
## s.e.   0.0318   0.0391   0.0287
##
## sigma^2 estimated as 0.2322:  log likelihood=-700.2
## AIC=1408.39   AICc=1408.43   BIC=1428
```

```
base_mod %>% accuracy()
```

```
## # A tibble: 1 x 9
##   .model                    .type      ME  RMSE   MAE     MPE   MAPE  MASE  ACF1
##   <chr>                     <chr>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl> <dbl>
## 1 ARIMA(co2ppm ~ pdq(0, 1,~ Train~ 0.0121 0.469 0.350 0.00292 0.0900 0.765 0.102
```

```
base_mod %>% forecast(h = 102) %>% accuracy(carbon_ts_we_test)
```

```
## # A tibble: 1 x 9
##   .model                        .type    ME  RMSE   MAE   MPE  MAPE  MASE  ACF1
##   <chr>                         <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(co2ppm ~ pdq(0, 1, 1) +~ Test  0.792  1.01 0.850 0.192 0.206  2.16 0.609
```

This model in back-shift notation is below.

$$(1 + 0.0456B^{52})(1 - B)(1 - B^{52})y_t = (1 - 0.6624B)(1 - 0.0456B^{52})\epsilon_t$$

Our base model has an AICc value of 1408.43, in sample RMSE of 0.4688399, and pseudo out-of-sample RMSE of 1.005171.

Next, we examine other potential ARIMA models by looping through multiple different values for `p` and `q` values for both the non-seasonal and seasonal parts of the optimized ARIMA model. We restrict `d` values to 1 since our previous EDA requires only one degree of first differencing for stationarity. We will use the AICc value to determine the best model while restricting `p` and `q` values to 0 through 2 and limiting the sum of seasonal and nonseasonal `p` and `q` components to 6 or less. This facilitates a wide enough breadth of model exploration, while limiting the model selection time to a reasonable length.

```
results <- data.frame(p=integer(),q=integer(),P=integer(),Q=integer(),AICc=double())

for (p in 0:1){
  for (q in 1:2){
    for(P in 0:1){
      for (Q in 1:2){
        tryCatch({
          if (p + q + P + Q <= 6) {
            mod <- carbon_ts_we_training %>% model(ARIMA(co2ppm ~ 0 +
                                        pdq(p,1,q) + PDQ(P,1,Q, period = 52)))

            results <- results %>% add_row(p=p, q = q, P=P, Q=Q,
                                           AICc = as.numeric(glance(mod)$AICc))
          }
        },
```

```
        error=function(e) {
          print(paste('error encountered for', p, q, P, Q))
        })
      }
    }
  }
}

results[which.min(results$AICc), ]
```

The model selection function results in non-seasonal `p` and `q` values of 0 and 2, with seasonal `P` and `Q` values of 1 and 2.

```
mod <- carbon_ts_we_training %>% model(ARIMA(co2ppm ~ 0 + pdq(0,1,2) +
                                               PDQ(1,1,2, period = 52)))
report(mod)
```

```
## Series: co2ppm
## Model: ARIMA(0,1,2)(1,1,2)[52]
##
## Coefficients:
##           ma1      ma2    sar1     sma1    sma2
##       -0.5898  -0.1621  0.8851  -1.7999  0.8426
## s.e.   0.0313   0.0311  0.0989   0.1196  0.0979
##
## sigma^2 estimated as 0.2189:  log likelihood=-680.29
## AIC=1372.59   AICc=1372.67   BIC=1402.01
```

The selected model in back-shift notation is below.

$$(1 - 0.8851B^{52})(1 - B)(1 - B^{52})y_t = (1 - 0.5898B - 0.1621B^2)(1 - 1.7999B^{52} + 0.8426B^{104})\epsilon_t$$

Next, we will use this model to assess in-sample and pseudo out-of-sample performance. Then, we can generate predictions for carbon dioxide concentration levels in the future for 420 ppm and 500 ppm

```
mod %>% accuracy()
```

```
## # A tibble: 1 x 9
##   .model                   .type      ME  RMSE   MAE     MPE   MAPE  MASE   ACF1
##   <chr>                    <chr>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>  <dbl>
## 1 ARIMA(co2ppm ~ 0 + pdq(~ Train~ 0.0139 0.455 0.339 0.00341 0.0872 0.741 0.0109
```

```
mod %>% forecast(h = 102) %>% accuracy(carbon_ts_we_test)
```

```
## # A tibble: 1 x 9
##   .model                        .type    ME  RMSE   MAE    MPE  MAPE  MASE  ACF1
##   <chr>                         <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 ARIMA(co2ppm ~ 0 + pdq(0, 1,~ Test  0.390 0.651 0.538 0.0946 0.130  1.37 0.425
```

The second model has an AICc value of 1372.67, in-sample RMSE of 0.4547748, and pseudo out-of-sample RMSE of 0.6508385. All three of these values are better than our base model, so we will use the second model generated by the loop for future predictions. Next, we will forecast to year 2100 and also find the period where the mean and interval estimates for CO2 concentration are around 420 ppm and 500 ppm.

```
model_forecast <- mod %>% forecast(h = 5000)
model_forecast_unpacked <- model_forecast %>% hilo(level = c(80, 95)) %>%
  unpack_hilo("80%") %>% unpack_hilo("95%")
```

```
model_forecast_420 <- model_forecast_unpacked %>% filter(.mean > 419 &
                                                           .mean < 421)
summary(model_forecast_420$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2021-04-18" "2022-06-17" "2023-01-11" "2023-02-11" "2023-10-30" "2024-09-29"
```

```
model_forecast_420_interval <- model_forecast_unpacked %>%
  filter(`80%_lower` < 420 & `80%_upper` > 420)
summary(model_forecast_420_interval$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2021-04-04" "2022-05-29" "2023-03-26" "2023-05-18" "2024-01-14" "2025-10-19"
```

```
model_forecast_500 <- model_forecast_unpacked %>%
  filter(.mean > 499 & .mean < 501)
summary(model_forecast_500$date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2051-02-26" "2051-05-03" "2052-06-02" "2052-06-06" "2053-07-09" "2053-09-21"
```

```
model_forecast_500_interval <- model_forecast_unpacked %>%
  filter(`80%_lower` < 500 & `80%_upper` > 500)
summary(model_forecast_500_interval$date)
```

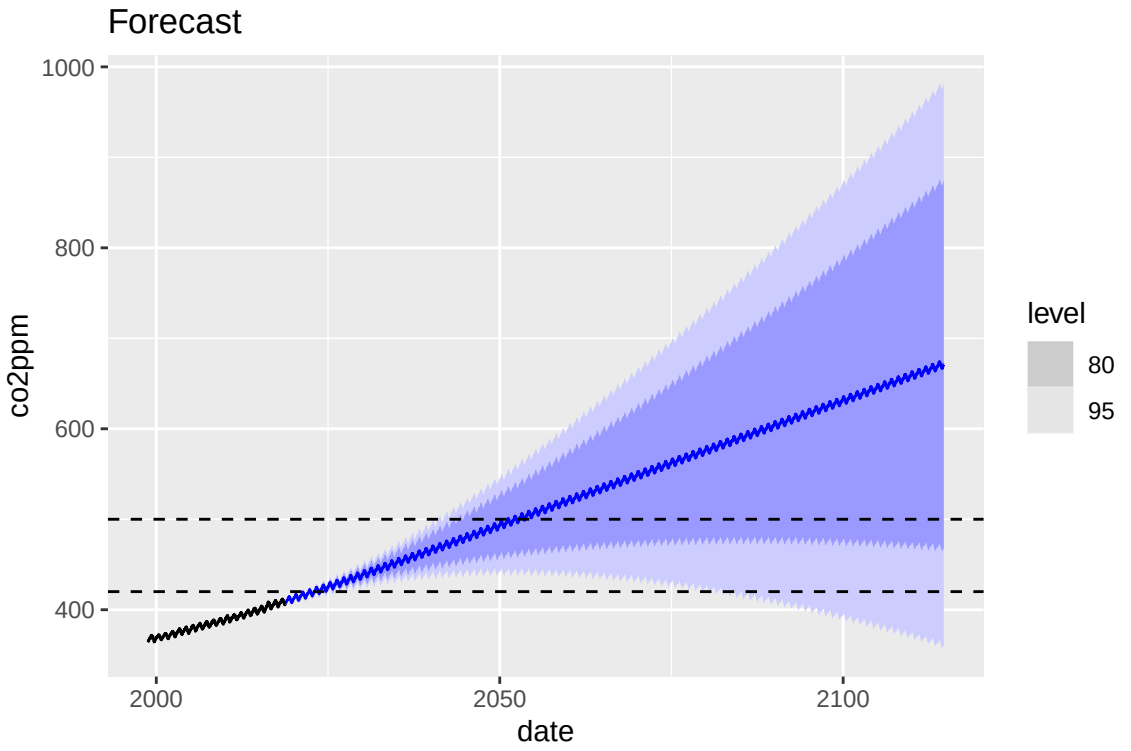
```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2043-03-15" "2061-07-20" "2079-04-02" "2079-03-31" "2096-12-12" "2114-08-26"
```

```
model_forecast_2100 <- model_forecast %>% filter_index('2100-01'~'2100-12')
summary(model_forecast_2100$.mean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   628.0   630.5   633.1   632.5   634.5   636.8
```

We also plot the forecast along with the base training data.

```
model_forecast %>% autoplot(carbon_ts_we_training) + ggtitle('Forecast') +
  geom_hline(yintercept=420,lty="dashed") + geom_hline(yintercept=500,lty="dashed")
```



When considering point estimates, our model predicts atmospheric C02 to reach 420 ppm for the first time in between March 20th, 2022 and March 27th, 2022 and for the last time between November 5th, 2023 and November 12th, 2023. When considering 80% confidence intervals, our model predicts atmospheric C02 could reach 420 ppm for the first time as early as April 4th, 2021 and for the last time as late as October 19th 2025. When considering point estimates, our model predicts atmospheric C02 to reach 500 ppm for the first time in between March 5th, 2051 and March 12th, 2051 and for the last time between September 14th, 2053 and September 21st, 2053. When considering 80% prediction intervals, our model predicts atmospheric C02 could reach 500 ppm for the first time as early as March 15th, 2043. Because the confidence intervals widen as forecasts go farther into the future, there is no definitive projected last date in the confidence intervals for when the CO2 will be at 500 ppm as is apparent in the previous forecast graph.

When considering point estimates, our model predicts atmospheric C02 levels to be between 627.9867 and 636.8326 for the year of 2100. However, when considering prediction intervals, our model predicts atmospheric C02 could range from 470.2484 to 796.2185 ppm. When the model attempts to predict data that far in the future, the uncertainty compounds enough that the prediction does not tend to be very useful.

Although the model may predict CO2 concentration levels well if the past trends were to continue, the model does not take into effect potential underlying factors that may influence a change in the levels of CO2 in the atmosphere. One key change could be the current transition from fossil fuels as a primary energy source to renewable energy sources that have a smaller carbon footprint. This could result in a significant change in the direction of the trend component of the forecasted data series. Another potential variable not captured by the ARIMA model is natural longer term trends in the CO2 levels of the Earth's atmosphere. Overall, we are relatively confident in the predictions of the model in the short term (15-30 years), but less confident in much longer term trends.