Co2 Emissions: Statistical Methods for Discrete Response, Time Series, and Panel Data in R

Prathyusha Charagondla, Skyler Roh and Andrew Kiruluta

load libraries

```
knitr::opts_chunk$set(echo = TRUE, warning  = FALSE, message = FALSE)
```

```
library(car)
library(Hmisc)
library(ggplot2)
library(ggfortify)
library(mcprofile)
library(gridExtra)
library(nnet)
library(corrplot)
library(MASS)
library(tseries)
library(readr)
library(lubridate)
library(forecast)
library(fable)
library(fpp2)
library(fpp3)
library(astsa)
library(urca)
library(Hmisc)
library(zoo)
library(xts)
library(dplyr)
library(gridExtra)
```

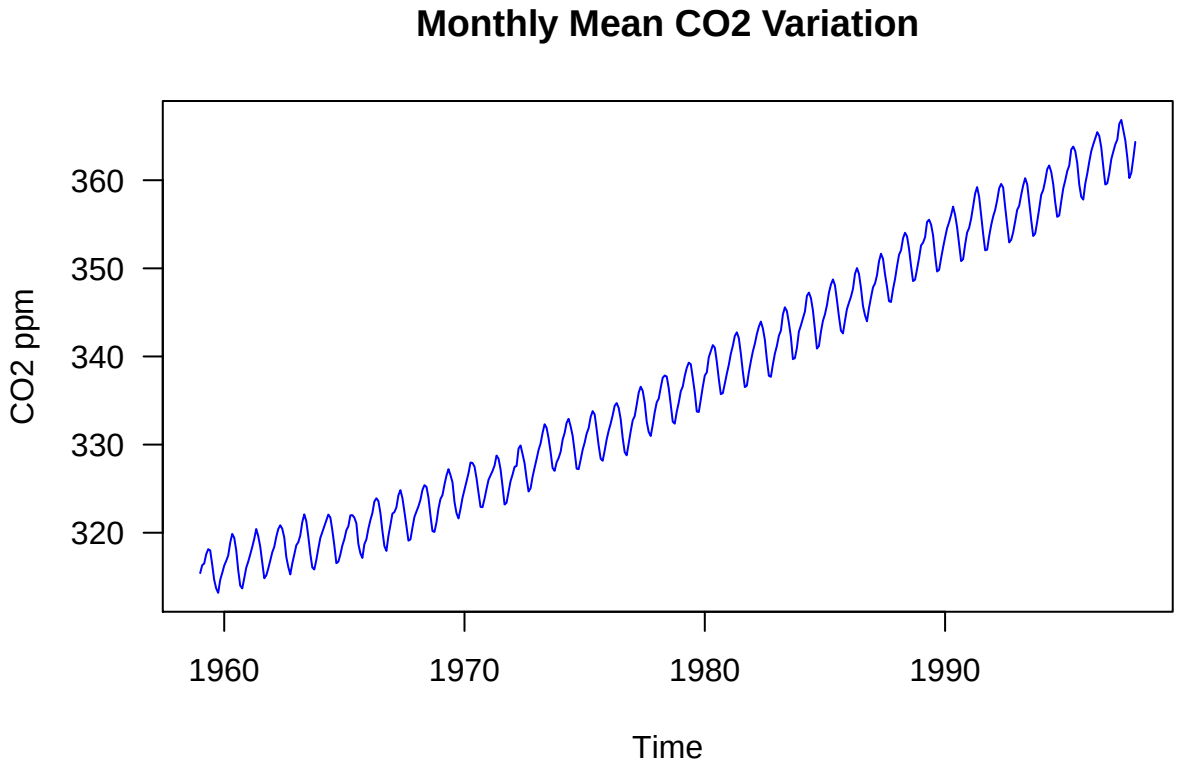
The Keeling Curve

In the 1950s, the geochemist Charles David Keeling observed a seasonal pattern in the amount of carbon dioxide present in air samples collected over the course of several years. He attributed this pattern to varying rates of photosynthesis throughout the year, caused by differences in land area and vegetation cover between the Earth's northern and southern hemispheres.

In 1958 Keeling began continuous monitoring of atmospheric carbon dioxide concentrations from the Mauna Loa Observatory in Hawaii. He soon observed a trend increase carbon dioxide levels in addition to the seasonal cycle, attributable to growth in global rates of fossil fuel combustion. Measurement of this trend at Mauna Loa has continued to the present.

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'.

```
plot(co2, col = 'blue', las = 1, ylab=expression("CO2 ppm"))
title(main = "Monthly Mean CO2 Variation")
```



Exploratory Data Analysis:

```
# inspect data
data("co2")
str(co2)
```

```
##  Time-Series [1:468] from 1959 to 1998: 315 316 316 318 318 ...
```

```
head(co2)
```

```
##         Jan    Feb    Mar    Apr    May    Jun
## 1959 315.42 316.31 316.50 317.56 318.13 318.00
```

```
tail(co2)
```

```
##         Jul    Aug    Sep    Oct    Nov    Dec
## 1997 364.52 362.57 360.24 360.83 362.49 364.34
```

Start date, and end date and frequency of the data collection:

```
tsp(co2)
```

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

1959 to 1997 in increments of one year with a montly data collection and reporting. Now check for any missing values if any:

```
sum(is.na(co2))
```

```
## [1] 0
```

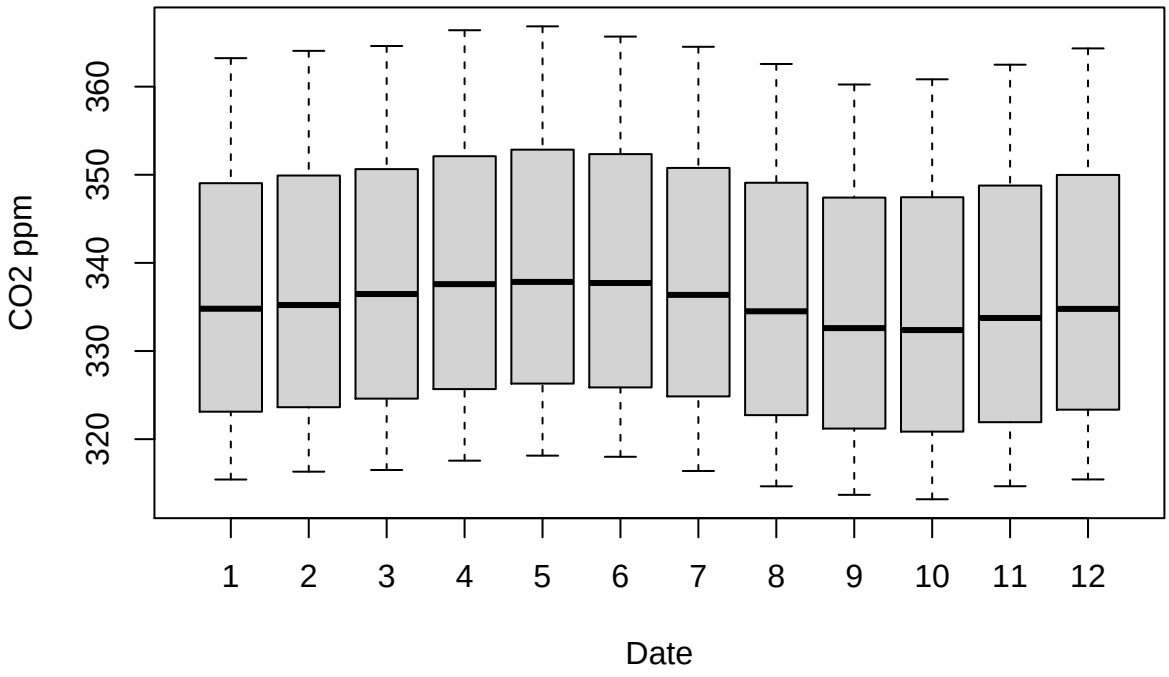
There are no missing values. Let check the summary:

```
summary(co2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   313.2   323.5   335.2   337.1   350.3   366.8
```

A box plot of the time series tracking CO_2 in the atomsphere is shown as:

```
boxplot(co2~cycle(co2),xlab="Date", ylab = "CO2 ppm" ,main ="Monthly CO2 Boxplot from 1959 to
```

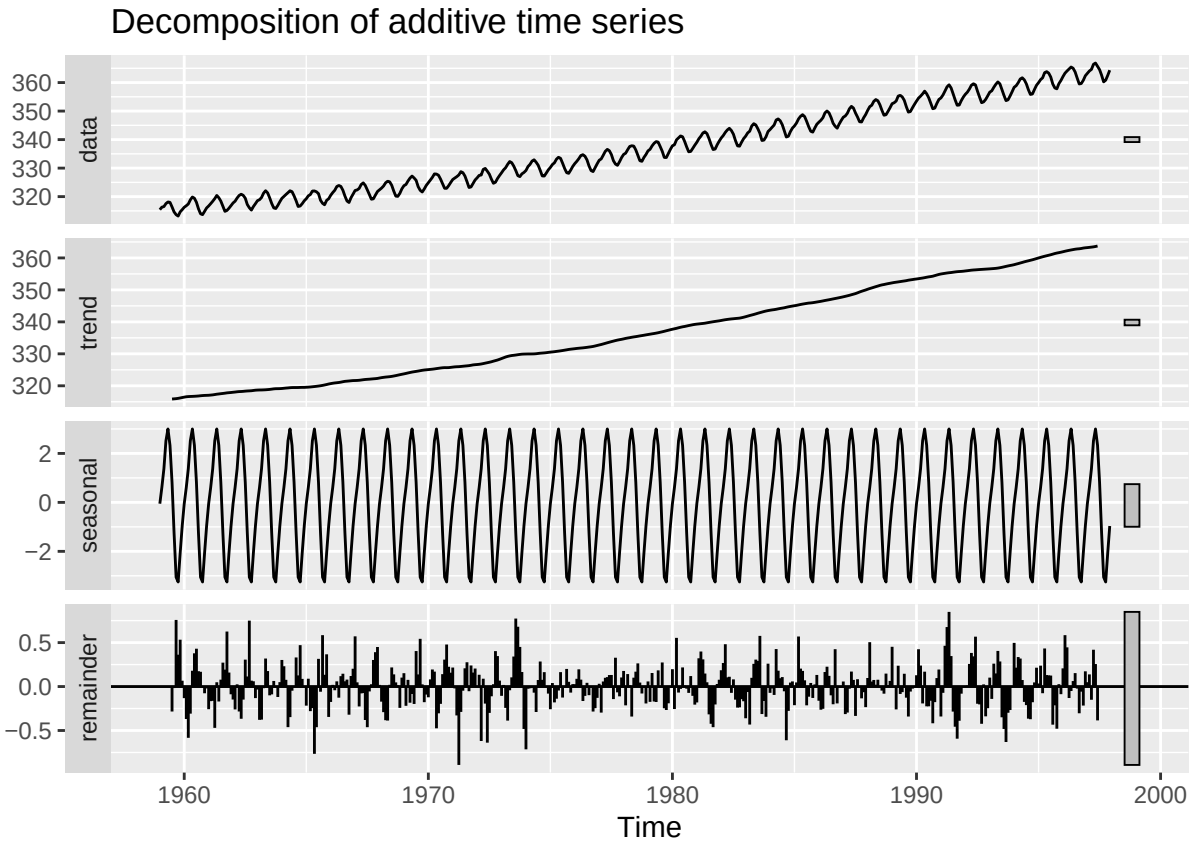


The date is units of months with 1 corresponding January while December is indicated as 12.

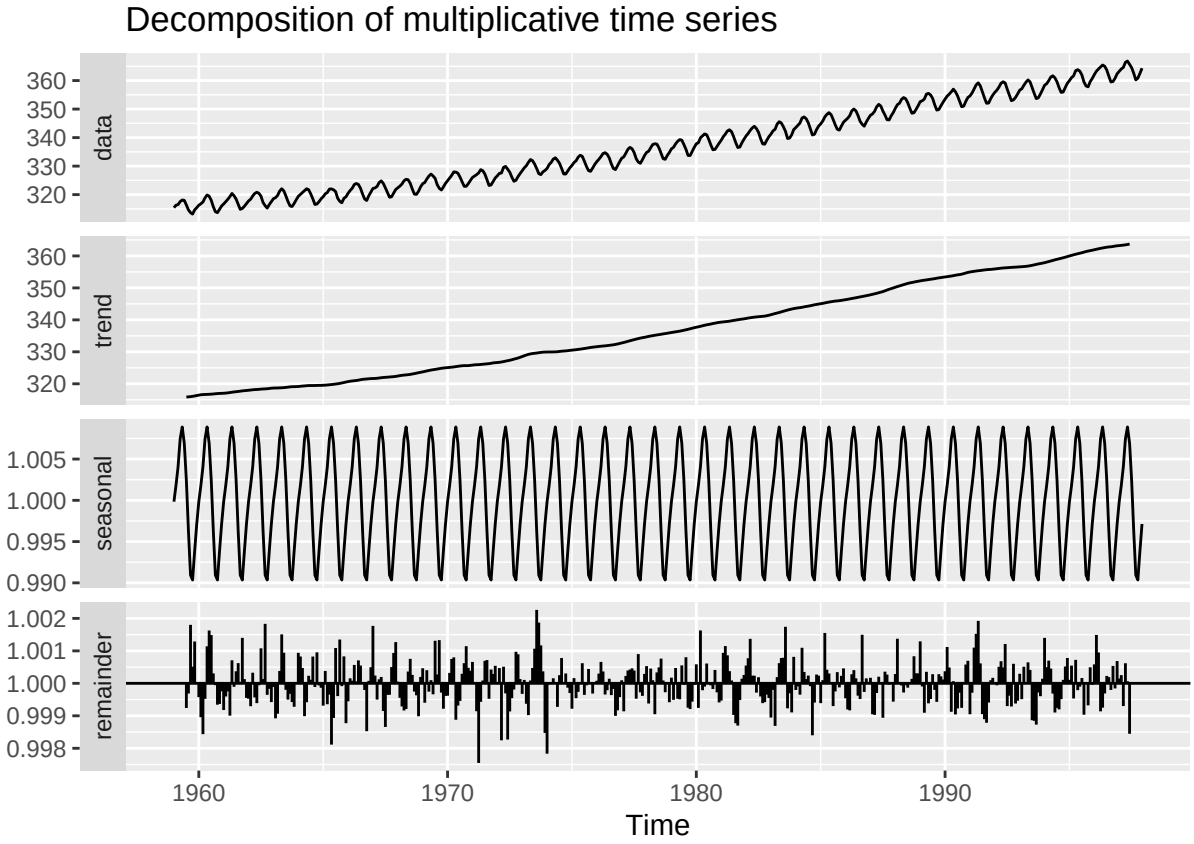
- The CO_2 concentration increase over time with each year which may be indicative of an increasing linear trend.
- In the boxplot there is more CO_2 concentration in April to June than other months, indicating seasonality with a apparent cycle of 12 months.
- CO_2 appears to be an additive time series as the CO2 concentration increases, the pattern of seasonality remains constant.
- There do not appear to be any outliers and there are no missing values. Therefore no data cleaning is required.

We will start by decomposing the time series to reveal the characteristics of the underlying trend, seasonal and error terms in it. All these components can be expressed as additive such that $y(t) = T(t) + S(t) + e(t)$, or as multiplicative, $y(t) = T(t) * S(t) * e(t)$ such that $\log y(t) = \log T(t) + \log S(t) + \log e(t)$ where $y(t)$ is the concentration of the measured CO_2 at time t, $T(t)$ is the underlying trend in the time series while $e(t)$ represents the random error term at time t in the model. We can use the decompose function to display these additive components of the time series:

```
decomposeCO2 <- decompose(co2,"additive")
autoplot(decomposeCO2)
```




```
decomposeCO2 <- decompose(co2,"multiplicative")
autoplot(decomposeCO2)
```



We see both an clear linear trend as well as a yearly periodicity in the time series with the seasons. We can also see the error components in the time series with a frequency component that is on the same scale as the seasonal periodicity.

As part of this EDA, we will also test the stationarity of the time series. A stationary time series has the conditions that the mean, variance and covariance are time invariant. Stationarity is a key requirement of the ARIMA models as we will see in parts 3-5 below. We will test for stationarity of the time series using the Augmented Dickey-Fuller Test and set the Hypothesis as:

- The null hypothesis H_0: that the time series is non-stationary
- The alternate Hypothesis H_A: that the time series is stationary

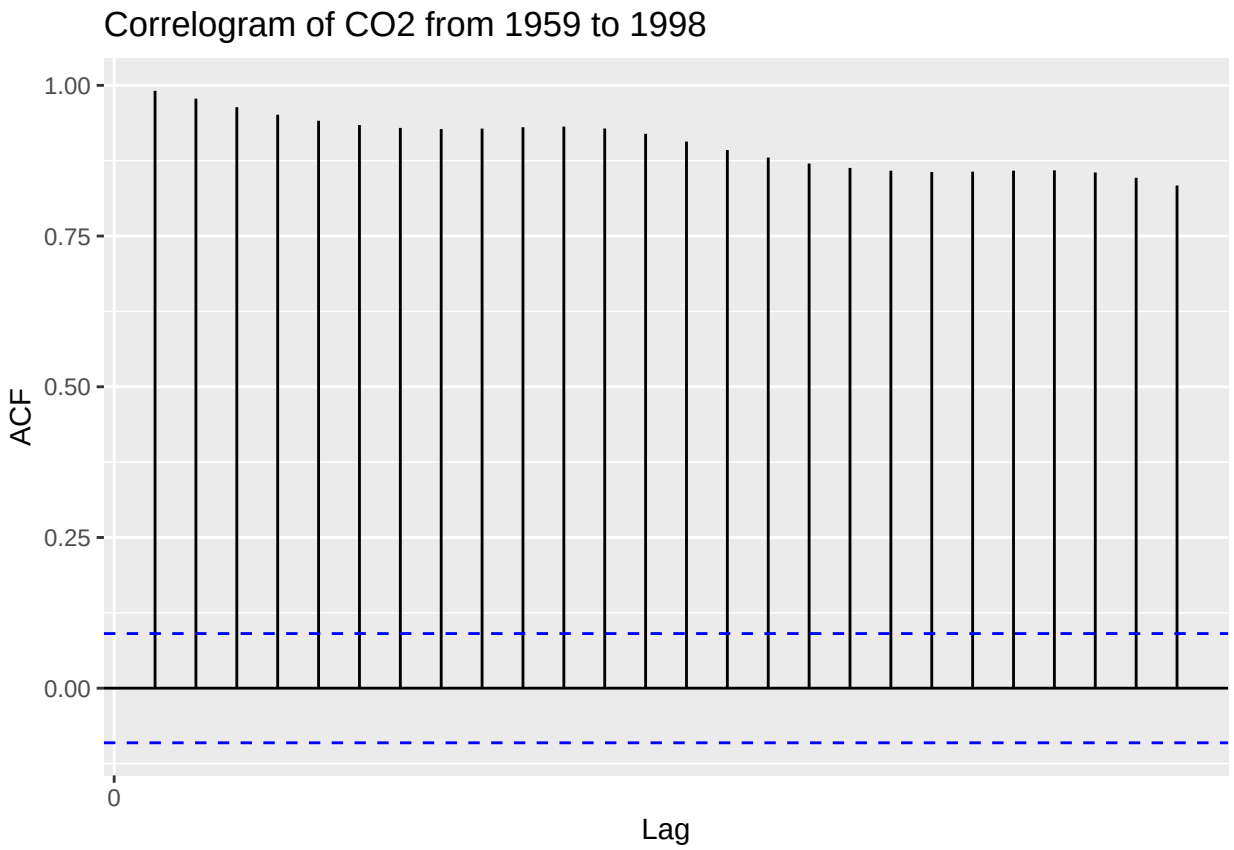
```
adf.test(co2)
```

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

We see that the p-value is greater than 0.05 at the 95% confidence interval and hence we do not have strong evidence to reject the null hypothesis and hence the time series is not stationary.

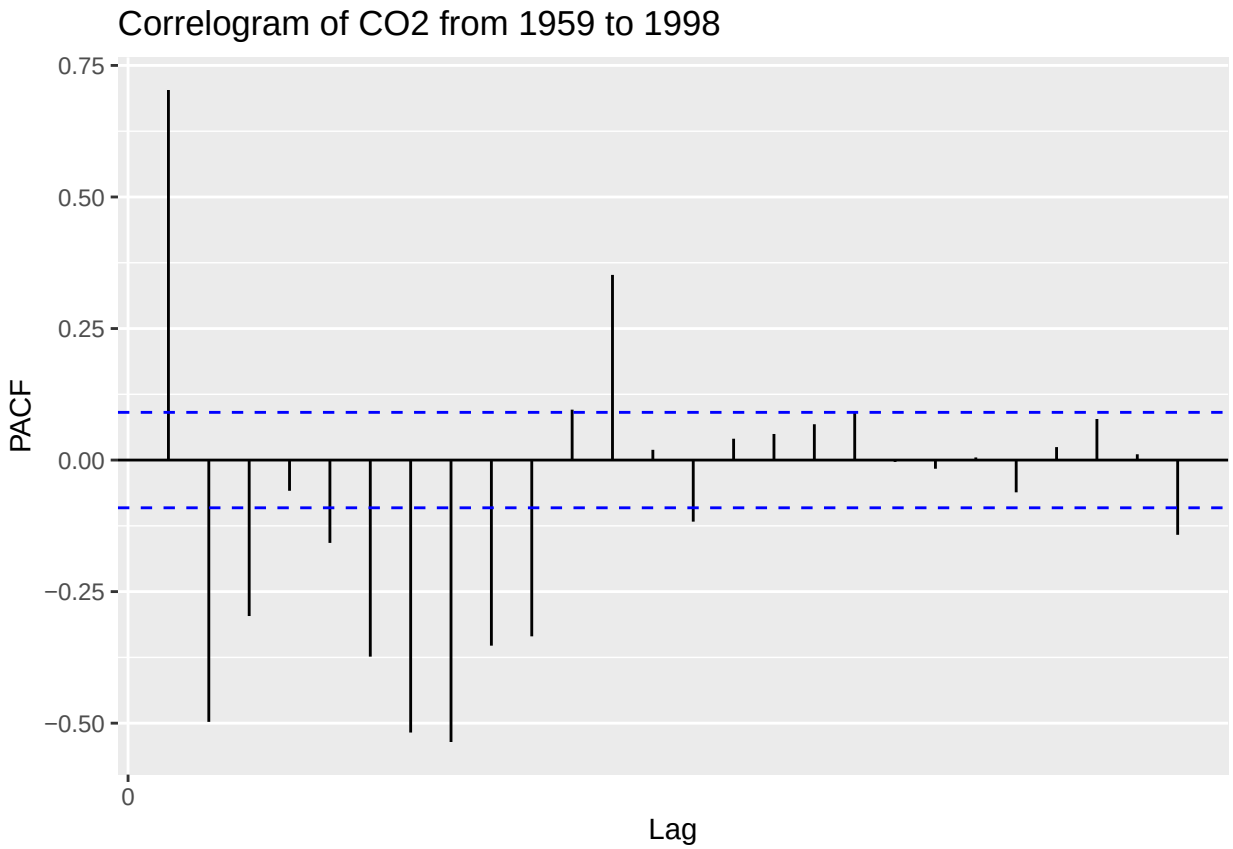
We can also test for stationarity using the autocorrelation function. We use the autocorrelation function acf and the partial autocorrelation function $pacf$.

```
autoplot(acf(co2,plot=FALSE))+ labs(title="Correlogram of CO2 from 1959 to 1998")
```



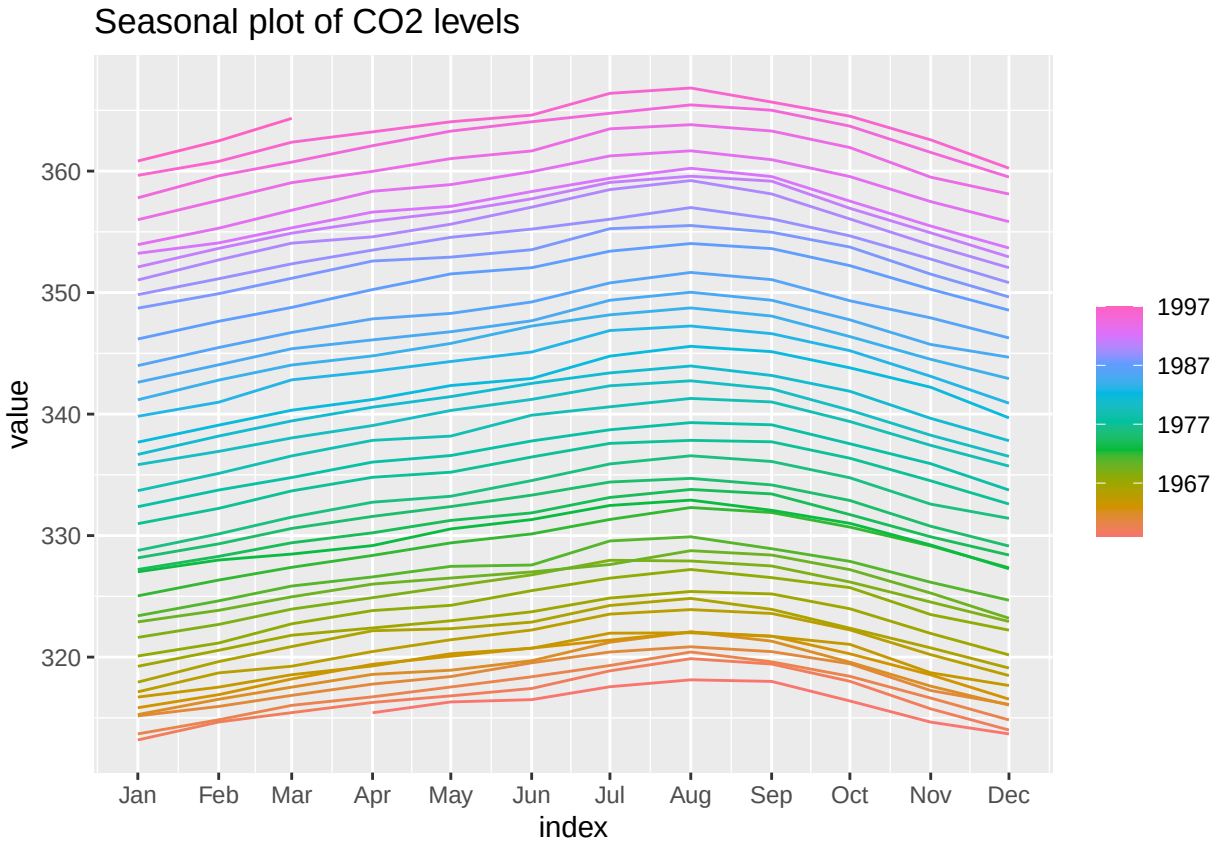
The line in blue indicates the 95% confidence interval and if the autocorrelation crosses this line, it means that specific lag is significantly correlated with the current series.

```
autoplot(pacf(diff(log(co2)),plot=FALSE))+ labs(title="Correlogram of CO2 from 1959 to 1998")
```



We see that the autocorrelation decays slowly while partial autocorrelation has significant spikes at seasonal lags exceeding the 95% confidence intervals which gives motivation for incorporating a seasonal component into both our regression and ARIMA models. Seen in the above decompositions and the seasonal plot below increasing seasonal variance is not apparent.

```
ts <- ts(co2, start = c(1950,100), frequency = 12) %>% as_tsibble()
gg_season(ts) +
  ggtitle("Seasonal plot of CO2 levels")
```



We need to remove the trend and seasonal effects from the time series before testing for stationarity via differencing.

```
adf.test(diff(diff(co2, 1), 12), alternative="stationary", k=0)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(co2, 1), 12)
## Dickey-Fuller = -29.027, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

These two operations result in a p-value for the null hypothesis of 0.01 and hence we reject the null hypothesis that the time series is not stationary. Hence the time series is considered weakly stationary enough for us to do times series modeling on it.

Linear Time Trend Model:

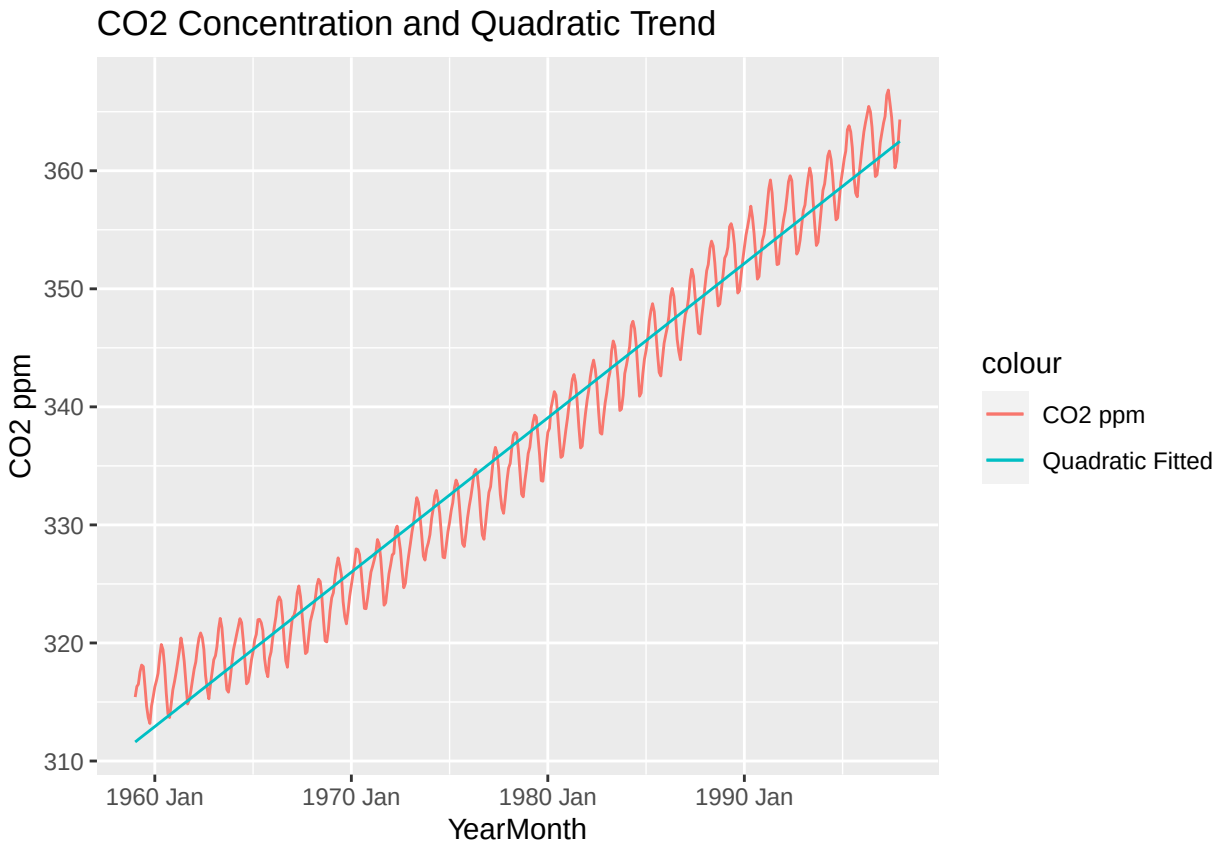
To start off, as seen in the additive and multiplicative decompositions explored in part 1, we do not have strong reason to need log transformation or similar non linear transformations such as box_cox as the variance across seasons over time is relatively constant and does not increase as the co2 levels rise.

Linear time trend model

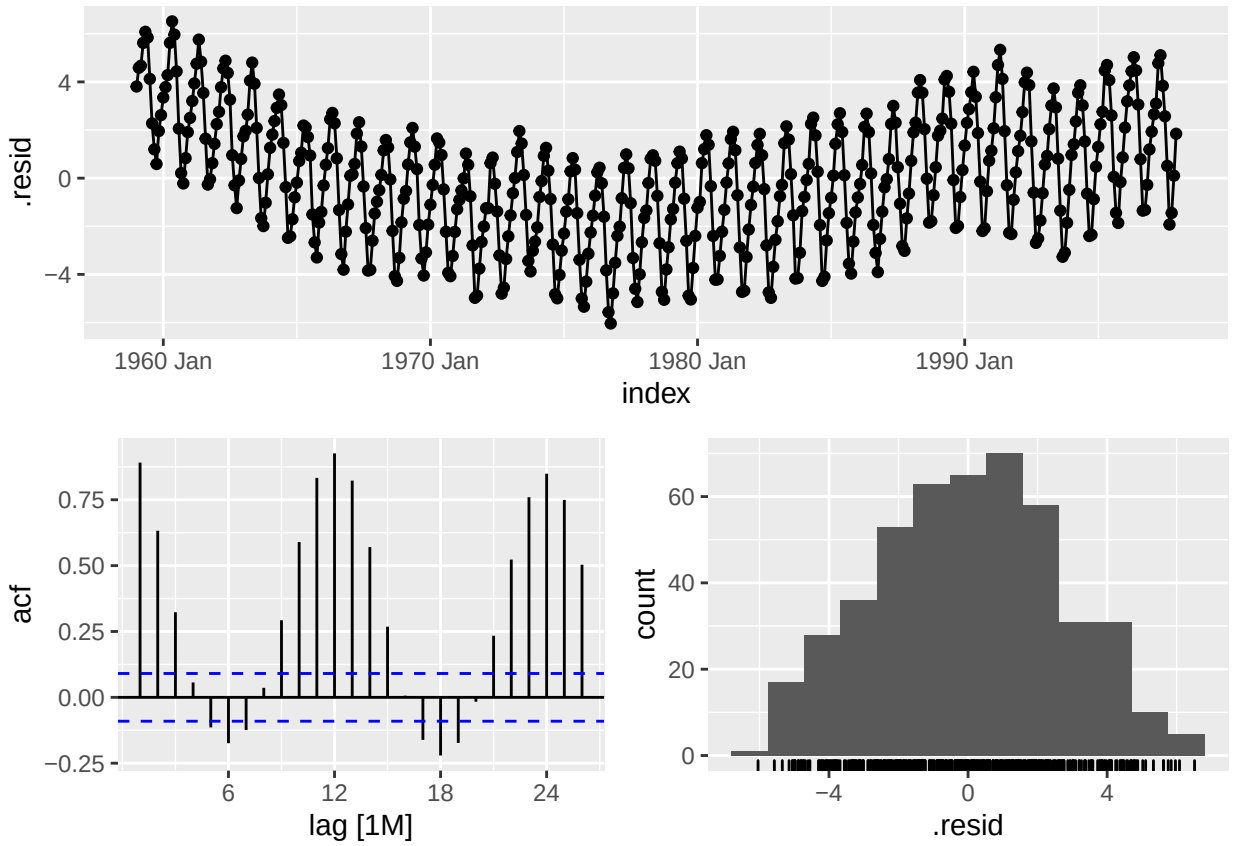
```
co2 = as_tsibble(co2) %>% mutate(time_index = row_number())

linear.trend.fit = co2 %>% model(TSLM(value ~ trend()))
linear.trend.fit %>% report()
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -6.039885 -1.947575 -0.001671  1.911271  6.514852
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.115e+02  2.424e-01  1284.9   <2e-16 ***
## trend()     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: < 2.22e-16
```

```
augment(linear.trend.fit) %>%
  ggplot(aes(x = index)) +
  geom_line(aes(y = value, colour = "CO2 ppm")) +
  geom_line(aes(y = .fitted, colour = "Quadratic Fitted")) +
  labs(x = "YearMonth", y = "CO2 ppm",
       title = "CO2 Concentration and Quadratic Trend")
```




```
linear.trend.fit %>% gg_tsresiduals()
```



The linear time trend model fitted to the co2 data is $CO_2 = 311.5 + 0.109t$ where t is the number of months since the beginning of 1959. The linear time trend coefficient is highly statistically significant with a p-value near 0 and thus we reject the null hypothesis that there is no trend present in the data. However, further examining the residuals of this model, it is clear that the residuals have significant autocorrelation and a quadratic trend still remains.

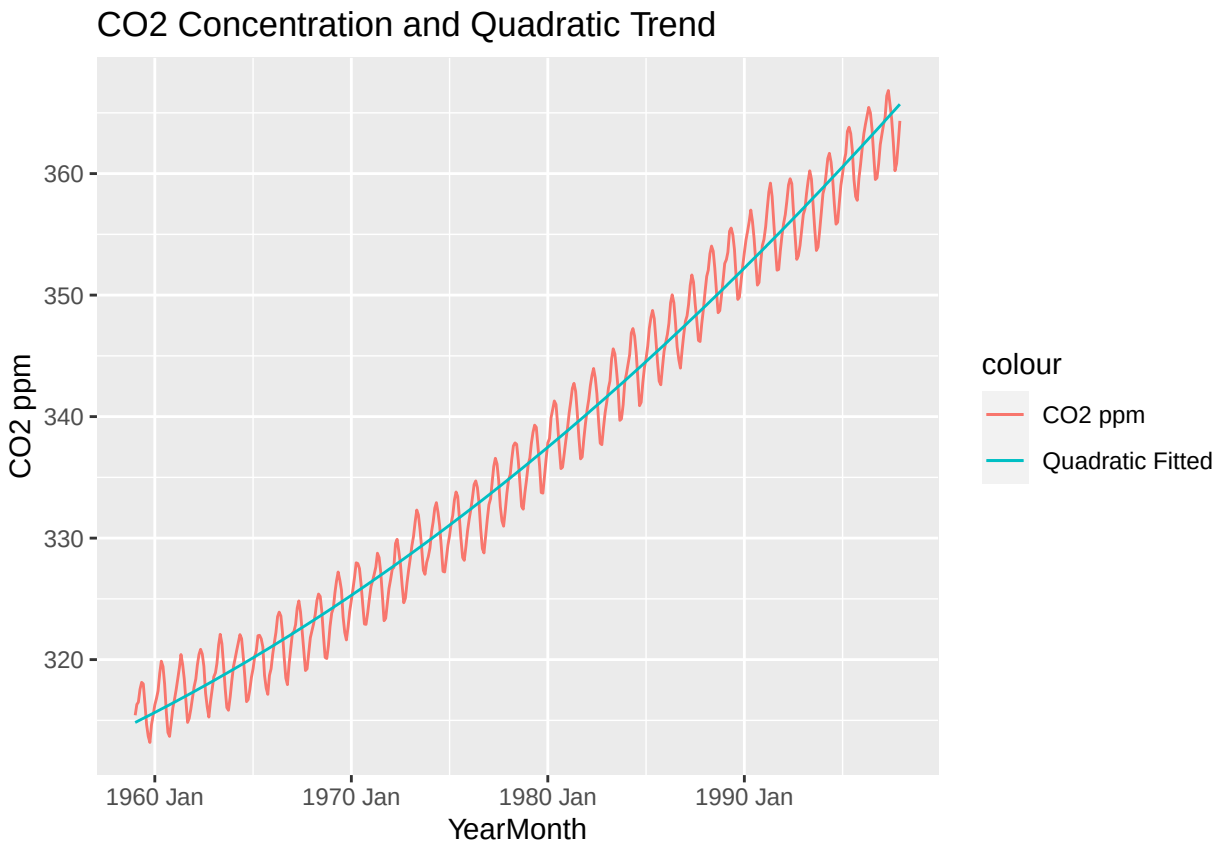
Quadratic time trend model

```
quadratic.trend.fit = co2 %>% model(TSLM(value ~ trend() + I(trend()^2)))
quadratic.trend.fit %>% report()
```

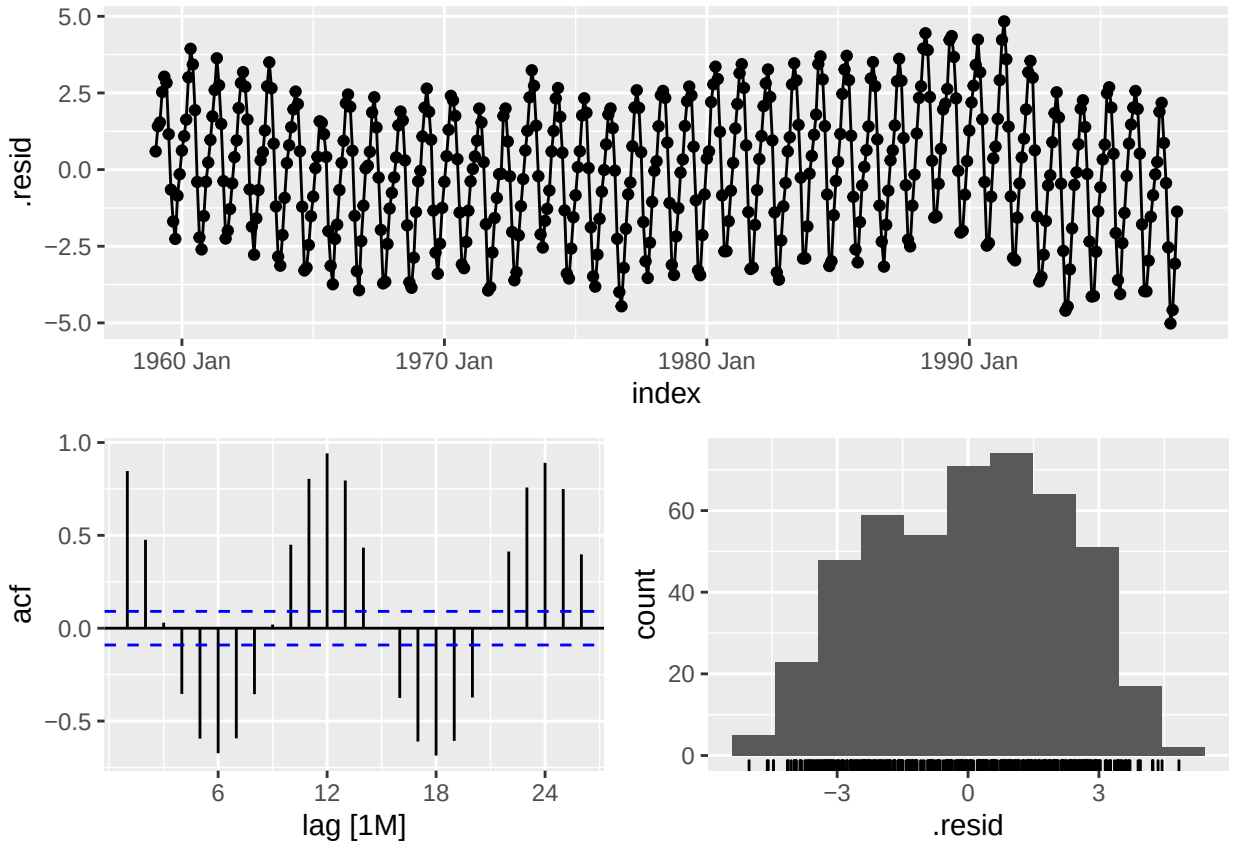
```
## Series: value
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.148e+02  3.039e-01 1035.65   <2e-16 ***
## trend()      6.739e-02  2.993e-03   22.52   <2e-16 ***
## I(trend()^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: < 2.22e-16
```

```
augment(quadratic.trend.fit) %>%
  ggplot(aes(x = index)) +
  geom_line(aes(y = value, colour = "CO2 ppm")) +
  geom_line(aes(y = .fitted, colour = "Quadratic Fitted")) +
  labs(x = "YearMonth", y = "CO2 ppm",
       title = "CO2 Concentration and Quadratic Trend")
```




```
quadratic.trend.fit %>% gg_tsresiduals()
```



The quadratic time trend model fitted to the co2 data is $CO_2 = 314.8 + 6.739\text{e-}02 * t + 8.862\text{e-}05 * t^2$ where t is the number of months since the beginning of 1959. Both components of this polynomial time trend are highly statistically significant with p-values near 0 and thus we reject the null hypothesis that there is no quadratic trend present in the data. Examining these residuals, compared to the linear time trend, the mean more closely centers around 0 at all time periods; however, still there is autocorrelation between residuals. This needs to be addressed with a seasonal component that will be addressed next.

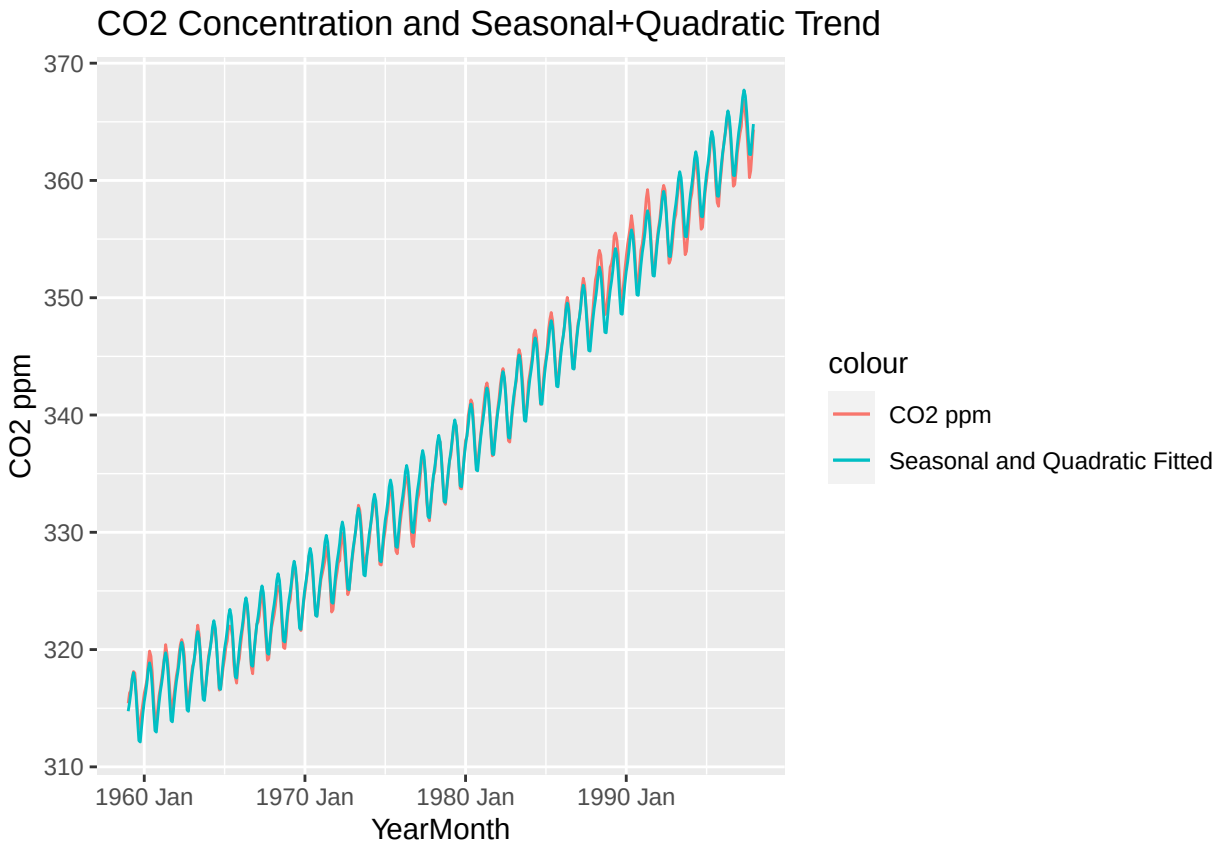
Quadratic time trend model w/ seasonal factors

```
co2 = co2 %>% mutate(month=replace(time_index%%12, time_index%%12==0, 12))
seasonal.and.quadratic = co2 %>% model(TSLM(value ~ trend() + I(trend()^2) + season()))
seasonal.and.quadratic %>% report()
```

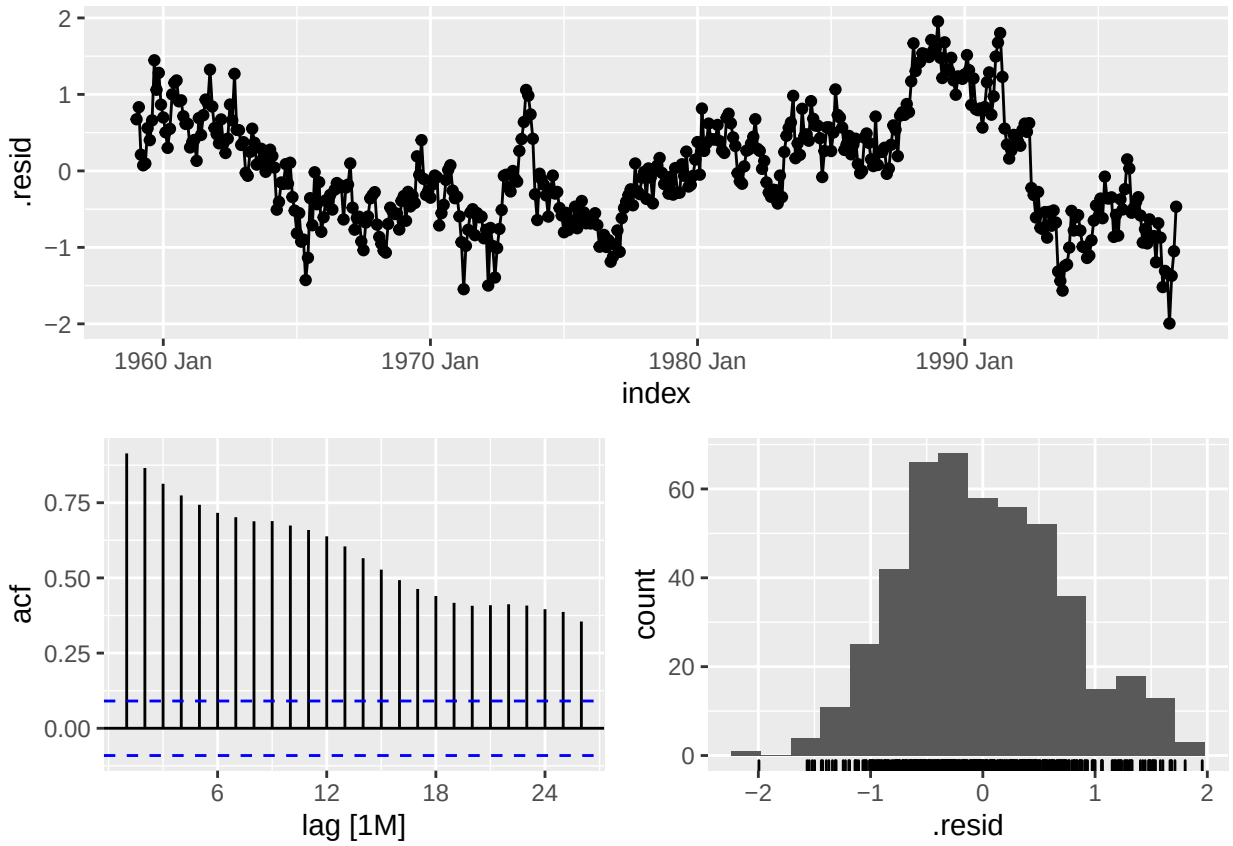
```
## Series: value
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99478 -0.54468 -0.06017  0.47265  1.95480
##
## Coefficients:
##                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)     3.147e+02  1.494e-01 2105.894  < 2e-16 ***
```

```
## trend()         6.763e-02  9.929e-04   68.114  < 2e-16 ***
## I(trend()^2)    8.865e-05  2.050e-06   43.242  < 2e-16 ***
## season()year2   6.642e-01  1.640e-01    4.051 5.99e-05 ***
## season()year3   1.407e+00  1.640e-01    8.582  < 2e-16 ***
## season()year4   2.538e+00  1.640e-01   15.480  < 2e-16 ***
## season()year5   3.017e+00  1.640e-01   18.400  < 2e-16 ***
## season()year6   2.354e+00  1.640e-01   14.357  < 2e-16 ***
## season()year7   8.331e-01  1.640e-01    5.081 5.50e-07 ***
## season()year8  -1.235e+00  1.640e-01   -7.531 2.75e-13 ***
## season()year9  -3.059e+00  1.640e-01  -18.659  < 2e-16 ***
## season()year10 -3.243e+00  1.640e-01  -19.777  < 2e-16 ***
## season()year11 -2.054e+00  1.640e-01  -12.526  < 2e-16 ***
## season()year12 -9.374e-01  1.640e-01   -5.717 1.97e-08 ***
## ---
## 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: < 2.22e-16
```

```
augment(seasonal.and.quadratic) %>%
  ggplot(aes(x = index)) +
  geom_line(aes(y = value, colour = "CO2 ppm")) +
  geom_line(aes(y = .fitted, colour = "Seasonal and Quadratic Fitted")) +
  labs(x = "YearMonth", y = "CO2 ppm",
       title = "CO2 Concentration and Seasonal+Quadratic Trend")
```




```
seasonal.and.quadratic %>% gg_tsresiduals()
```

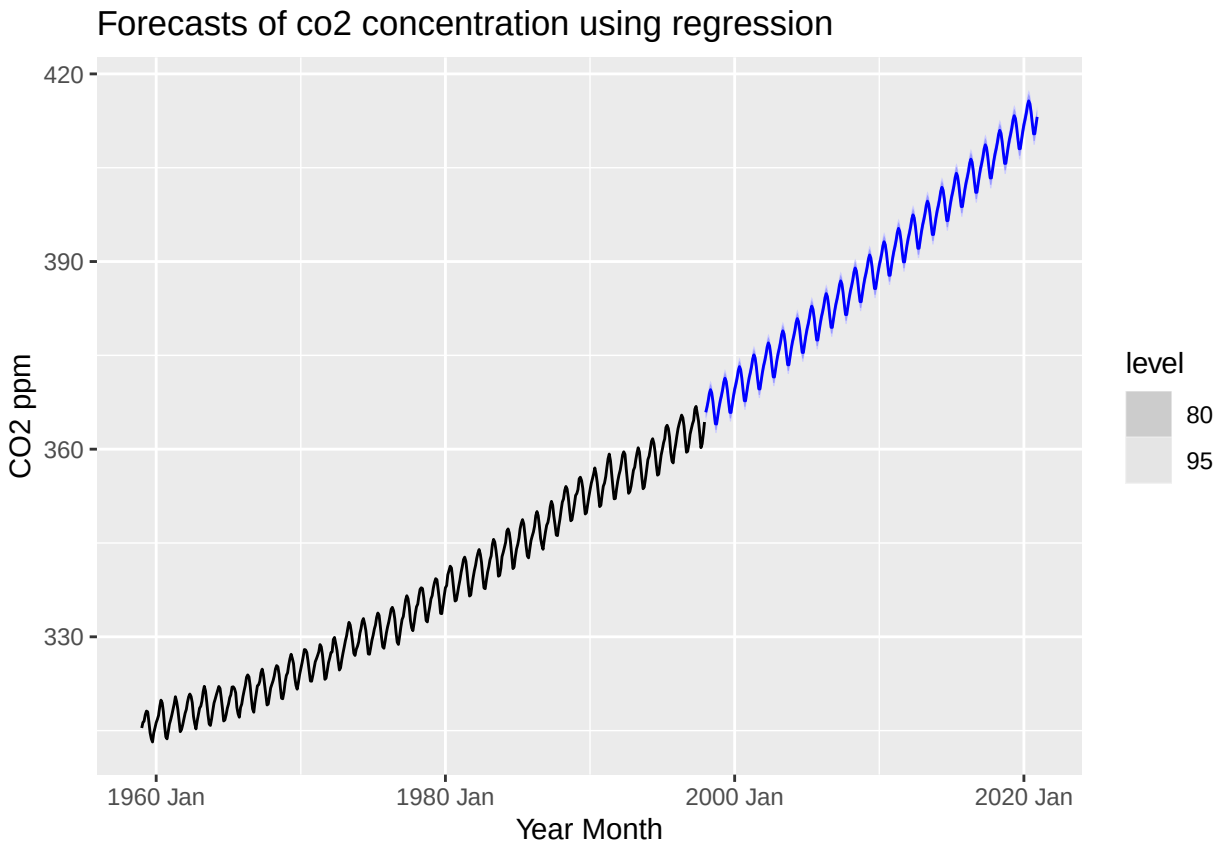


The quadratic time trend model with seasonal variables fitted to the co2 data is

$$\begin{aligned} CO_2 =& 314.7 + 6.763\text{e-}02 * t + 8.865\text{e-}05 * t^2 \\ &+ 0.664 * (month = Feb) + 1.41 * (month = Mar) + 2.538 * (month = Apr) \\ &+ 3.017 * (month = May) + 2.354 * (month = Jun) + 0.833 * (month = Jul) \\ &- 1.235 * (month = Aug) - 3.059 * (month = Sep) - 3.243 * (month = Oct) \\ &- 2.054 * (month = Nov) - 0.937 * (month = Dec) \end{aligned}$$

where t is the number of months since the beginning of 1959. The inclusion of the seasonal terms are statistically significant with p-values near 0 for each of the coefficients. The resulting fit is much closer to the true series however the remaining residuals still do not exhibit a white noise process as the ACF clearly shows a slow decay.

```
fc_co2 <- forecast(seasonal.and.quadratic, h=(2021-1998)*12)
fc_co2 %>%
  autoplot(co2) +
  ggtitle("Forecasts of co2 concentration using regression") +
  xlab("Year Month") + ylab("CO2 ppm")
```



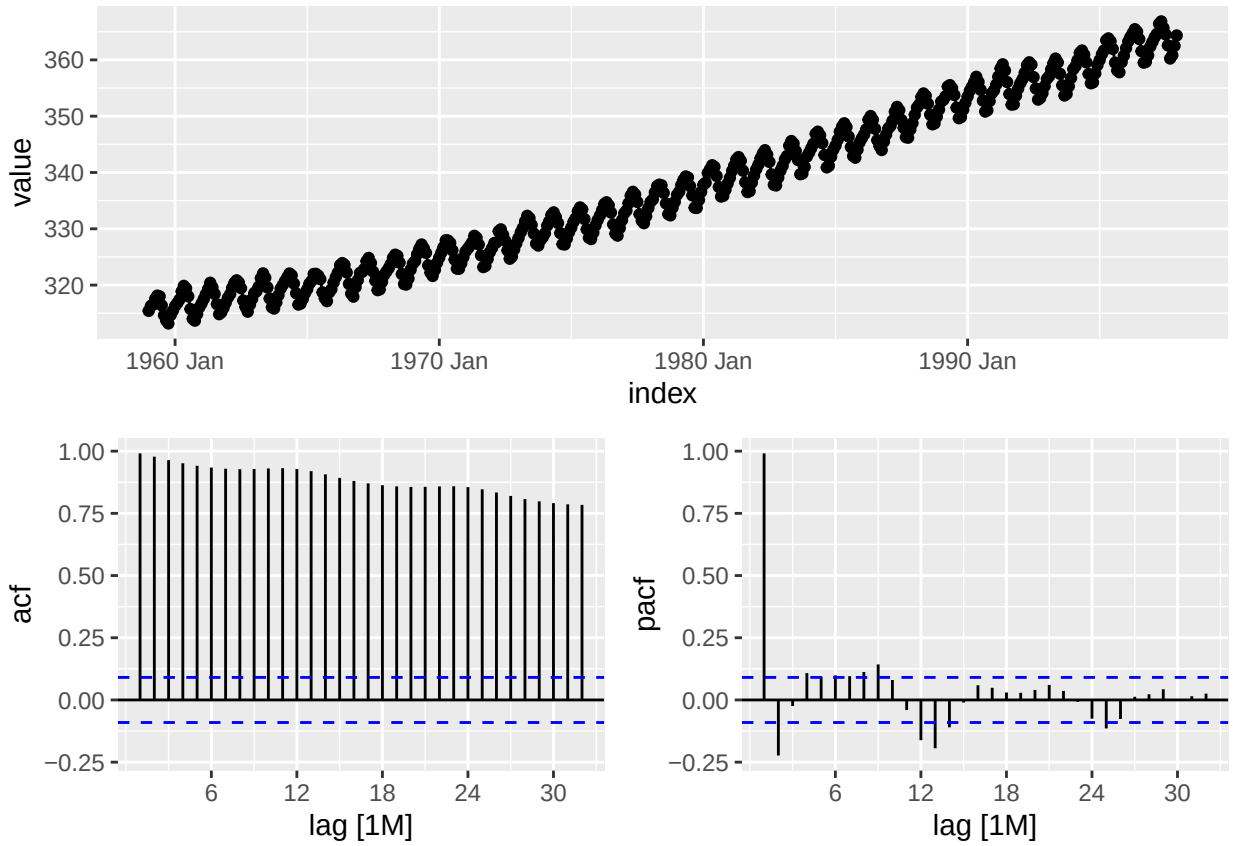

```
forecast(seasonal.and.quadratic, h=(2021-1998)*12) %>%
  hilo() %>%
  unpack_hilo("95%") %>%
  dplyr::select(index, .mean, '95%_lower', '95%_upper') %>%
  tail()
```

```
## # A tsibble: 6 x 4 [1M]
##      index .mean `95%_lower` `95%_upper`
##      <mth> <dbl>       <dbl>       <dbl>
## 1 2020 Jul  414.        412.        416.
## 2 2020 Aug  412.        410.        414.
## 3 2020 Sep  410.        409.        412.
## 4 2020 Oct  410.        409.        412.
## 5 2020 Nov  412.        410.        414.
## 6 2020 Dec  413.        411.        415.
```

Using the quadratic and seasonal regression, by the end of 2020, the co2 level is forecasted to be 413.1ppm with a 95% prediction interval of (411.4, 414.9).

ARIMA Model and Forecast to Fit Series:

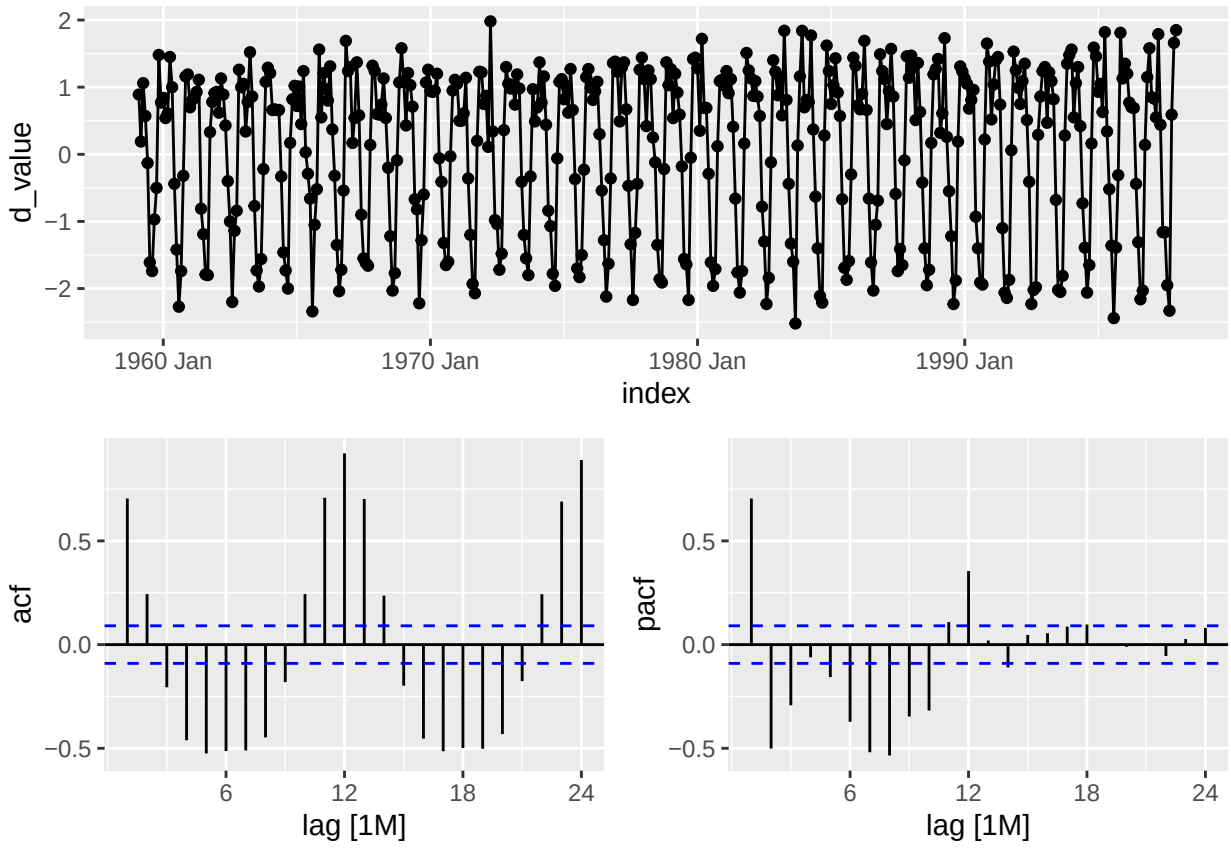
```
# plot autocorrelation and partial autocorrelation function
co2 %>% gg_tsdisplay(y = value, plot = 'partial', lag_max = 32)
```



There is a clear (possibly nonlinear) upward trend in the series, and a notable yearly seasonal pattern which can be both seen visually in the series and the significant partial autocorrelation at lag 12 and 13. Autocorrelation decays slowly while partial autocorrelation has significant spikes at seasonal lags.
As noted previously, increasing variance is not apparent in the seasonal plot. We will not use a box_cox transformation on the series as we do not have strong reason given the variance across seasons over time is relatively constant in the above additive decomposition and seasonal plot from part 1 eda.

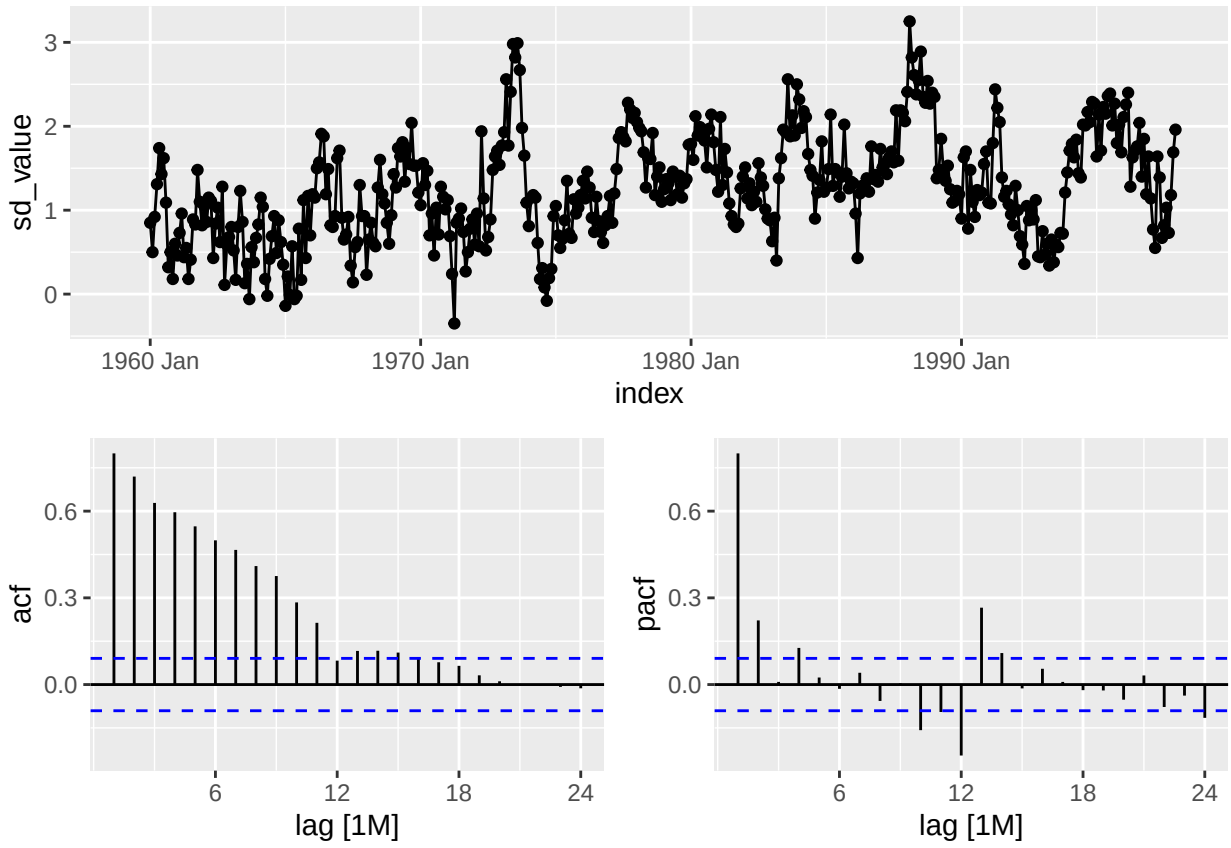
To address issues with non-stationarity due to trend, first apply first-differencing to stabilize the mean.

```
co2 <- co2 %>% mutate(d_value = difference(value, 1))
co2 %>% gg_tsdisplay(y=d_value, plot = 'partial', lag_max = 24)
```



After the first-differencing applied to the series, the trend is eliminated and the variance is largely subdued. Seasonality is still apparent visually in the series as well as noted by the spikes in pacf at lags 12. Next, we apply seasonal differencing for a period of 12 months to eliminate seasonality.

```
co2 <- co2 %>% mutate(sd_value = difference(value, 12))
co2 %>% gg_tsdisplay(y = sd_value, plot = 'partial', lag_max=24)
```



Under the KPSS unit root test, null hypotheses of stationarity would be rejected for both the first-differenced and seasonal-differenced series at least at a 0.1 level for both.

```
co2 %>% features(d_value, unitroot_kpss)
```

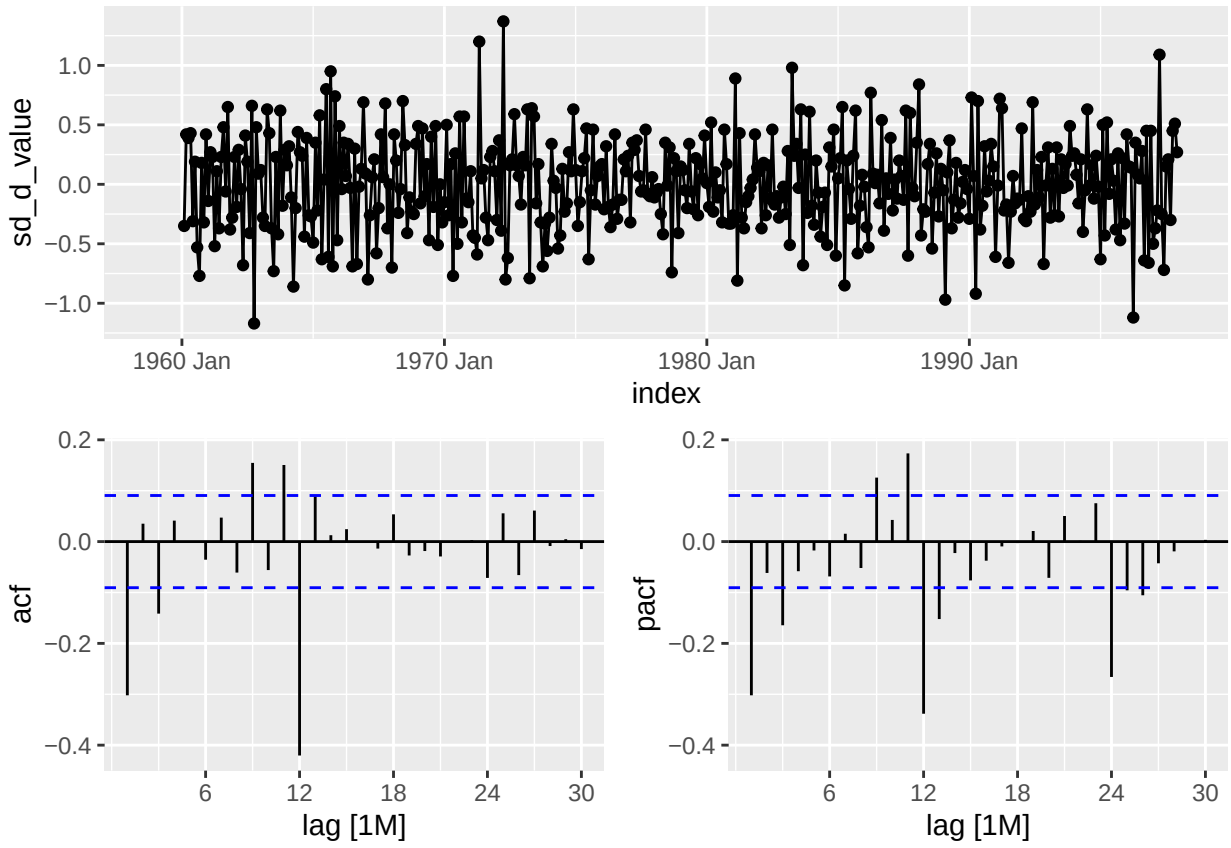
```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0124         0.1
```

```
co2 %>% features(sd_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      1.94        0.01
```

Applying both seasonal and nonseasonal differencing results in a series that is closer to stationary. A KPSS test would reject the null hypothesis of stationarity at 10% significance, but not 5% significance.

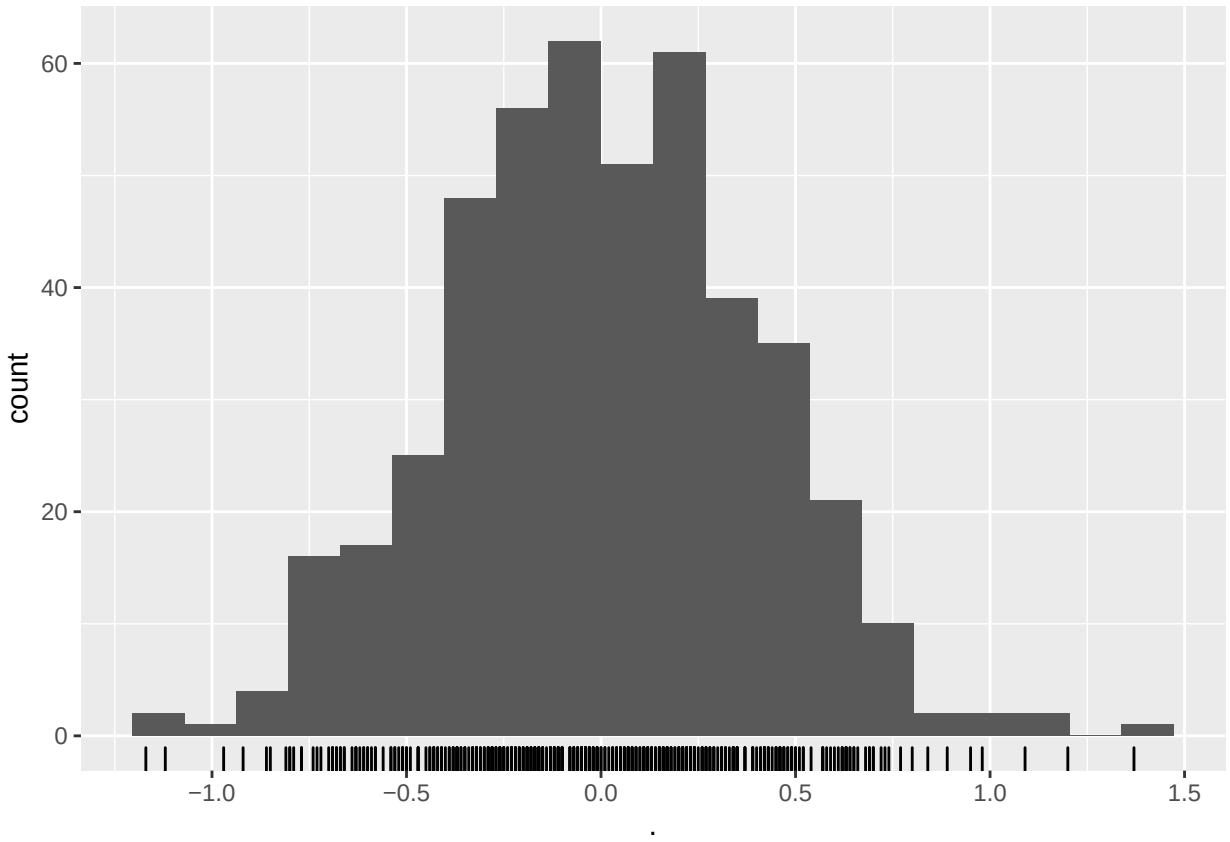
```
co2 <- co2 %>% mutate(sd_d_value = difference(d_value, 12))
co2 %>% gg_tsdisplay(y = sd_d_value, plot = 'partial', lag_max = 30)
```




```
co2 %>% features(sd_d_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0115         0.1
```

```
co2$sd_d_value %>% gghistogram()
```



The `ndiffs` and `nsdiffs` functions use KPSS tests to confirm the appropriate order of nonseasonal and seasonal differencing.

```
co2 %>% features(value, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

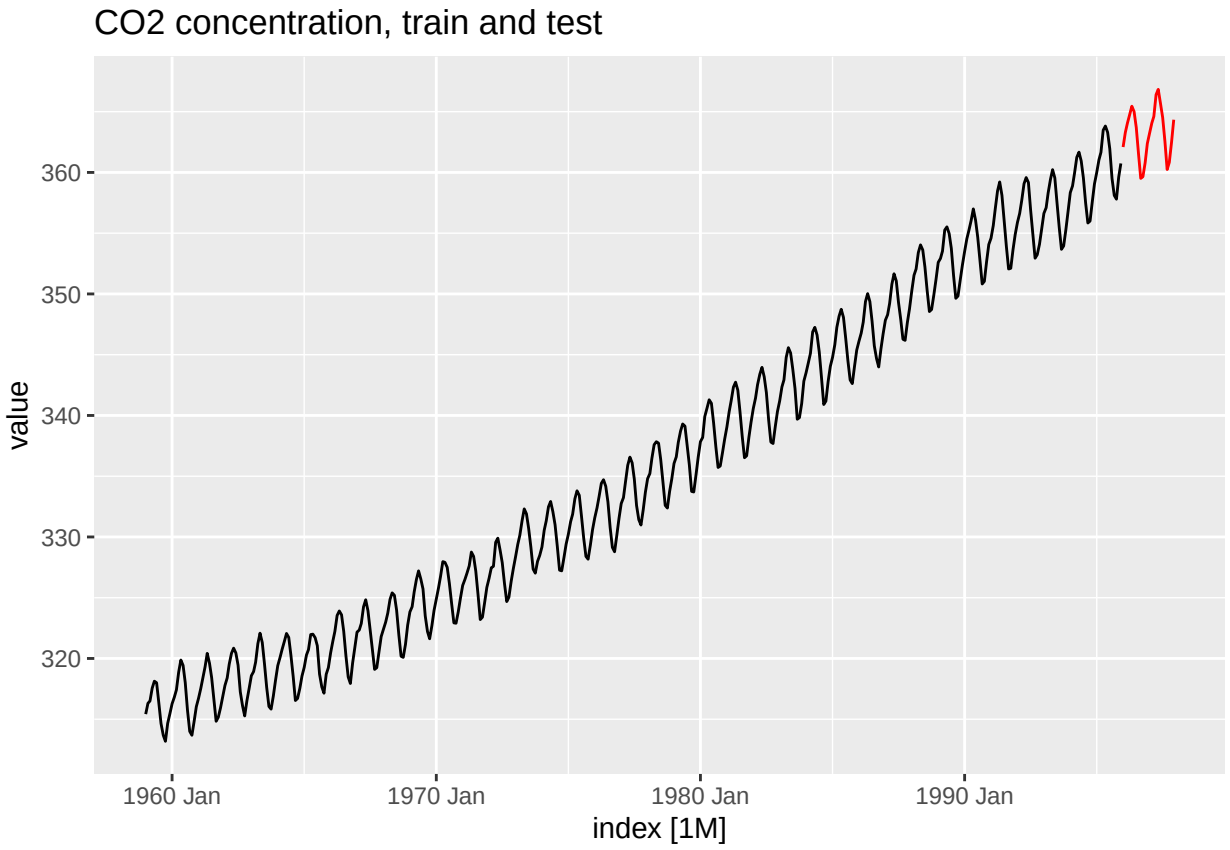
```
co2 %>% features(value, unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       1
```

The correlograms of the first and seasonal-differenced series do not show gradual decays associated with p AR terms. Rather significant spikes in ACF and PACF at lag 1 and 12, potentially suggesting seasonal MA(1) and non seasonal MA(1)_{12}. There may be other components–with a notable spike at the quarterly interval of 3 months–but we first compare an $ARIMA(0,1,0)(0,1,1)_{12}$, an $ARIMA(0,1,1)(0,1,1)_{12}$ and an $ARIMA(0,1,3)(0,1,1)_{12}$ among a few others, using in-sample and pseudo-out-of-sample accuracy comparisons.

We split the data into training and test sets, taking the final two years as a test set period for pseudo-out-of-sample forecasting performance.

```
co2.training <- co2 %>% filter_index(~'1995 Dec')
co2.test <- co2 %>% filter_index('1996 Jan'~.)
co2.training %>% autoplot(value) +
  autolayer(co2.test, value, colour = 'red') +
  ggtitle("CO2 concentration, train and test")
```

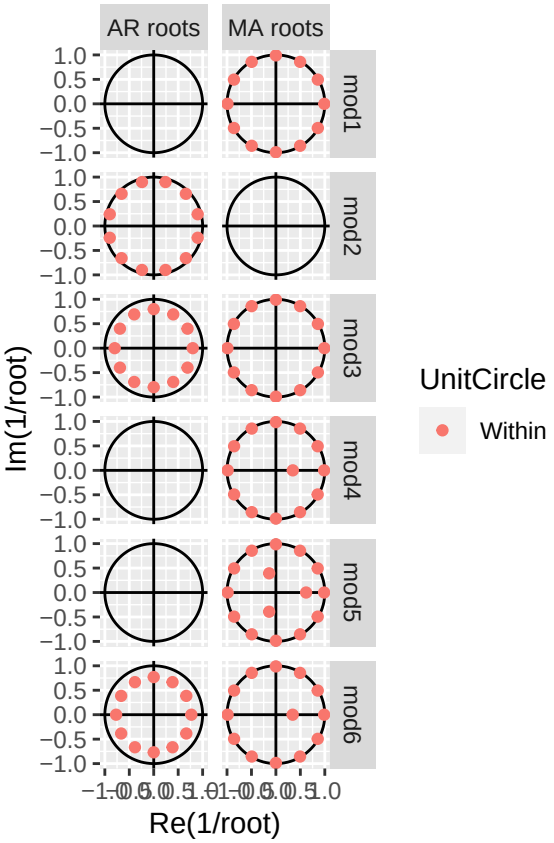


As previously mentioned, we fit the ARIMA models to the training series.

```
models <- co2.training %>%
  model(mod1 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(0,1,1, period=12)),
        mod2 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(1,1,0, period=12)),
        mod3 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(1,1,1, period=12)),
        mod4 = ARIMA(value ~ 0 +
                       pdq(0,1,1) + PDQ(0,1,1, period=12)),
        mod5 = ARIMA(value ~ 0 +
                       pdq(0,1,3) + PDQ(0,1,1, period=12)),
        mod6 = ARIMA(value ~ 0 +
                       pdq(0,1,1) + PDQ(1,1,1, period=12)),)

# The `glance()` function applied to a set of ARIMA models shows the variance of residuals (si
glance(models)
```

```
## # A tibble: 6 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 mod1   0.0918   -98.7  201.  201.  209. <cpl [0]>  <cpl [12]>
## 2 mod2   0.126   -160.   324.  324.  332. <cpl [12]> <cpl [0]>
## 3 mod3   0.0917   -97.9  202.  202.  214. <cpl [12]> <cpl [12]>
## 4 mod4   0.0840   -76.8  160.  160.  172. <cpl [0]>  <cpl [13]>
## 5 mod5   0.0834   -74.1  158.  158.  179. <cpl [0]>  <cpl [15]>
## 6 mod6   0.0842   -76.5  161.  161.  177. <cpl [12]> <cpl [13]>
```

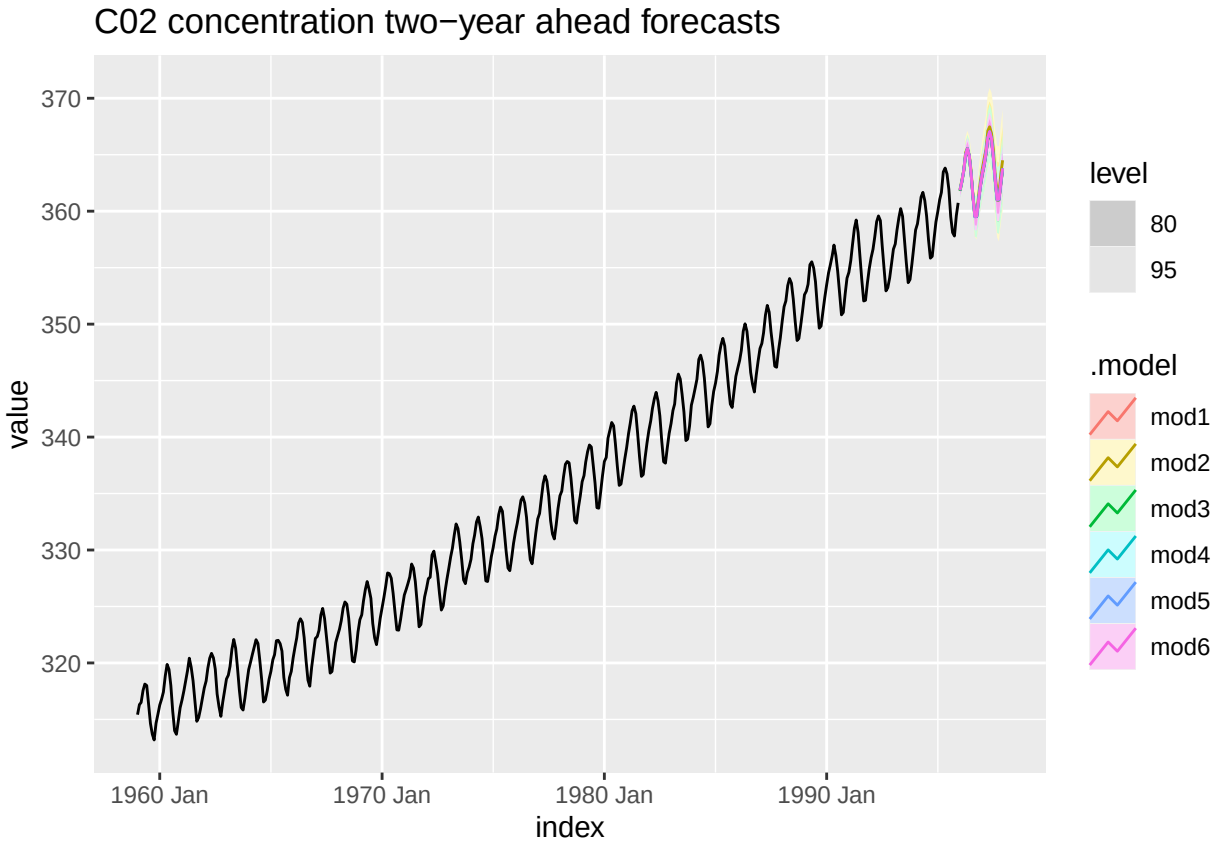
```
# Inverse roots all lie within the unit circle
gg_arma(models)
```



The glance() function provides various measures of in-sample performance. As with the information criteria, the $ARIMA(0,1,1)(0,1,1)_12$ model is preferred on a combination of low AICc and lower BIC than the next best model $ARIMA(0,1,3)(0,1,1)_12$.

Two-year forecasts for the four models generated are very similar.

```
models_forecast <- models %>% forecast(h = 24)
models_forecast %>% autoplot(co2.training) +
  ggtitle('C02 concentration two-year ahead forecasts')
```



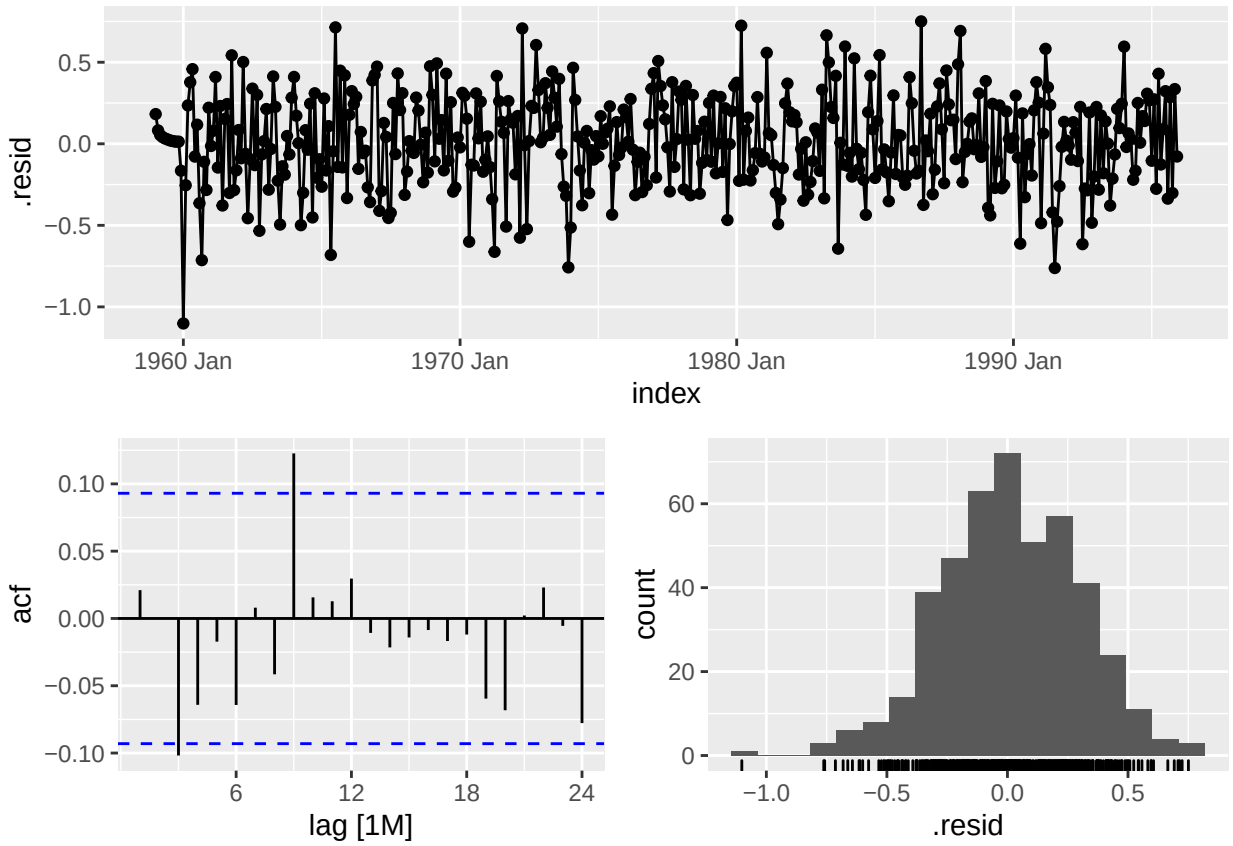
Pseudo-out-of-sample performance can be assessed applying the accuracy function to the test set.

```
models_forecast %>% accuracy(co2.test)
```

```
## # A tibble: 6 x 10
##   .model .type      ME  RMSE   MAE      MPE   MAPE  MASE RMSSE  ACF1
##   <chr>  <chr>   <dbl> <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 mod1   Test   0.0281 0.345 0.259  0.00773 0.0712   NaN   NaN 0.397
## 2 mod2   Test  -0.399  0.643 0.529 -0.110   0.146    NaN   NaN 0.617
## 3 mod3   Test   0.0323 0.347 0.261  0.00886 0.0718   NaN   NaN 0.388
## 4 mod4   Test  -0.0357 0.354 0.277 -0.00980 0.0761   NaN   NaN 0.426
## 5 mod5   Test  -0.0252 0.352 0.273 -0.00692 0.0752   NaN   NaN 0.419
## 6 mod6   Test  -0.0392 0.356 0.279 -0.0108  0.0768   NaN   NaN 0.425
```

The $ARIMA(0,1,0)(1,1,1)_12$ model is marginally preferred in out of sample forecasting in RMSE and MAPE however the difference is quite small. Based on these results, we will continue to use $ARIMA(0,1,1)(0,1,1)_12$ model as the best ARIMA candidate. We will check its residuals and perform a portmanteau test with a suitable number of lags. There is insufficient evidence to reject a null hypothesis of no autocorrelation.

```
models %>% dplyr::select(mod4) %>% gg_tsresiduals(lag_max = 24)
```




```
models %>% dplyr::select(mod4) %>% augment() %>%  features(.resid, ljung_box, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 mod4      24.5     0.434
```

```
# The ARIMA function in the fable package without specification of pdq/PDQ works similarly to
model.auto <- co2.training %>%
  model(ARIMA(value))
model.auto %>% report()
```

```
## Series: value
## Model: ARIMA(1,1,0)(1,1,2)[12]
##
## Coefficients:
##           ar1     sar1    sma1     sma2
##       -0.2949  -0.8438  0.0197  -0.7609
## s.e.   0.0469   0.2479  0.2279   0.1896
##
## sigma^2 estimated as 0.08473:  log likelihood=-77.95
## AIC=165.91   AICc=166.05   BIC=186.24
```

```
glance(model.auto)
```

```
## # A tibble: 1 x 8
##   .model       sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>         <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 ARIMA(value) 0.0847   -78.0  166.  166.  186. <cpl [13]> <cpl [24]>
```

```
model.auto.forecast<- model.auto %>% forecast(new_data=co2.test)
model.auto.forecast %>% accuracy(co2.test)
```

```
## # 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(value) Test  -0.0419 0.359 0.282 -0.0115 0.0776   NaN   NaN 0.423
```

The model produced by auto ARIMA produces neither minimum of in-sample or out-of-sample metrics compared to other models thus far explored. Continuing ARIMA parameter search, we explore a more exhaustive search of parameters.

```
test.ARIMA.params = function(p_range, q_range, P_range, Q_range,
                             train.data, test.data) {
  results <- data.frame(p=integer(),
                        q=integer(),
                        P=integer(),
                        Q=integer(),
                        AIC=double(),
                        AICc=double(),
                        BIC=double(),
                        RMSE=double(),
                        MAPE=double())
  for (p in p_range){
    for (q in q_range){
      for(P in P_range){
        for (Q in Q_range){
          tryCatch(
            {
              mod <- train.data %>%
                model(ARIMA(value ~ 0 +
                              pdq(p,1,q) + PDQ(P,1,Q, period=12)))

              oos_metrics = mod %>%
                forecast(new_data=test.data) %>%
                accuracy(test.data)

              results <- results %>% add_row(p=p, q = q, P=P, Q=Q,
                   AIC = as.numeric(glance(mod)$AIC),
                   AICc = as.numeric(glance(mod)$AICc),
                   BIC = as.numeric(glance(mod)$BIC),
                   RMSE = as.numeric(c(oos_metrics$RMSE)),
```

```
                    MAPE = as.numeric(c(oos_metrics$MAPE)))
                print(paste(p, q, P, Q, as.numeric(glance(mod)$AICc)))
            },
                error=function(e) {
                  print(paste('error encountered for', p, q, P, Q))
              }
            )
          }
        }
      }
    }
  }
  results
}
```

The following code is run and produced `part3_arima_param_results.csv` as a result. For knitting, this file is read in instead of redoing the loop

```
### Comment out to skip loop and read results from previous run

# results = test.ARIMA.params(p_range=0:3,
#                             q_range=0:3,
#                             P_range=0:3,
#                             Q_range=0:3,
#                             train.data=co2.training,
#                             test.data=co2.test)
# write.csv(results, 'part3_arima_param_results.csv')
```

```
results = read.csv('part3_arima_param_results.csv')
head(results[order(results$AICc),], 5)
```

```
##       X p q P Q      AIC     AICc      BIC      RMSE       MAPE
## 59   59 0 3 2 2 151.9552 152.2964 184.4840 0.4180352 0.09517732
## 56   56 0 3 1 3 152.1054 152.4466 184.6343 0.4131769 0.09368838
## 174 174 2 3 2 2 151.9301 152.4539 192.5912 0.4219943 0.09637820
## 171 171 2 3 1 3 152.0939 152.6177 192.7550 0.4162405 0.09473920
## 142 142 2 1 1 3 153.2234 153.5646 185.7522 0.4005522 0.09050558
```

```
head(results[order(results$BIC),], 5)
```

```
##     X p q P Q      AIC     AICc      BIC      RMSE       MAPE
## 18 18 0 1 0 1 159.5804 159.6367 171.7788 0.3540707 0.07613800
## 66 66 1 0 0 1 163.9023 163.9585 176.1006 0.3548180 0.07652639
## 81 81 1 1 0 1 160.4785 160.5724 176.7429 0.3523944 0.07559345
## 19 19 0 1 0 2 160.8578 160.9517 177.1222 0.3556869 0.07669119
## 22 22 0 1 1 1 160.9827 161.0766 177.2471 0.3559029 0.07678980
```

```
# (0, 1, 1)(0, 1, 1)_{12}
```

Lastly, we perform a more exhaustive search on the parameter space for p, q, P, Q and select based on AICc and BIC selection criteria. Although the $ARIMA(0,1,3)(2,1,2)_{12}$ produced lowest AICc, its

BIC and test RMSE value was notably worse than $ARIMA(0,1,1)(0,1,1)_{12}$, which was significantly lower than all other models on this BIC metric. Thus again, based on the combination of selection criteria, $ARIMA(0,1,1)(0,1,1)_{12}$ will be used going forward. The resulting out of sample metrics are non significantly different amongst those with the best insample results.

```
part.3.ARIMA = co2.training %>%
  model(ARIMA(value ~ 0 +
                pdq(0,1,1) + PDQ(0,1,1, period=12)))

part.3.ARIMA %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.3459  -0.8495
## s.e.   0.0499   0.0267
##
## sigma^2 estimated as 0.08405:  log likelihood=-76.79
## AIC=159.58   AICc=159.64   BIC=171.78
```

```
part.3.ARIMA %>% forecast(new_data=co2.test) %>% accuracy(co2.test)
```

```
## # 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(value ~ 0 +~ Test  -0.0357 0.354 0.277 -0.00980 0.0761   NaN   NaN 0.426
```

The final estimated ARIMA model to be used for forecasting is

$$(1-B)(1-B^{12})y_t = (1-0.3459B)(1-0.8495B^{12})e_t$$

where y_t is the co2 ppm at time t.

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). We convert these data into a suitable time series object, conduct a thorough EDA on the data, and address the problem of missing observations. We then investigate the Keeling Curve which evolved from 1997 to the present and compare this to the predictions from the previous forecasts.

```
# read in text file
cwm_df <- read.table("co2_weekly_mlo.txt", quote="\"")

# Change column names
colnames(cwm_df) <- c("year", "mon", "day", "decimal", "co2", "days", "one yr", "ten yr", "inc

# Creating a time column combining year, month and day
cwm_df$time <- as.Date(with(cwm_df, paste(year, mon, day,sep="-")), "%Y-%m-%d")
```

```
# New dataframe with time and co2 levels
cwm <- cwm_df[, c('time', 'co2')]
```

```
# Examine the data structure
describe(cwm)
```

Initial Exploratory Analysis

```
## cwm
##
##  2  Variables      2441  Observations
## ---------------------------------------------------------------------------
## time
##          n    missing   distinct       Info       Mean        Gmd        .05
##       2441          0       2441          1 1997-10-05       5698 1976-09-19
##        .10        .25        .50        .75        .90        .95
## 1979-01-21 1986-01-26 1997-10-05 2009-06-14 2016-06-19 2018-10-21
##
## lowest : 1974-05-19 1974-05-26 1974-06-02 1974-06-09 1974-06-16
## highest: 2021-01-24 2021-01-31 2021-02-07 2021-02-14 2021-02-21
## ---------------------------------------------------------------------------
## co2
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2441        0     2129        1    357.9     47.7    332.4    336.0
##      .25      .50      .75      .90      .95
##    347.0    364.9    387.6    404.0    409.7
##
## lowest : -999.99  326.72  326.99  327.07  327.23
## highest:  417.06  417.12  417.21  417.44  417.67
##
## Value      -1000   320   340   360   380   400   420
## Frequency     18    45   638   662   527   435   116
## Proportion 0.007 0.018 0.261 0.271 0.216 0.178 0.048
##
## For the frequency table, variable is rounded to the nearest 20
## ---------------------------------------------------------------------------
```

2441 total observations time: 1974-05-19 to 2021-02-21 with no missing values for time co2: there are missing values, denoted by -999.99, the range is from 326.72 to 417.67

Observing the parsed dataset above, there are 2441 total observations and two variables - time and co2. The time is from 1974-05-19 to 2021-02-21 and there are no missing values. The co2 variable ranges from 326.72 to 417.67. There are missing values, denoted by -999.99 that may need to be interpolated.

```
# observe missing values
nan <- table(cwm$co2)
nan[names(nan)==-999.99]
```

```
## -999.99
##      18
```

There are 18 missing values. Notice the original file says (-999.99 = no data). The value -999.99 is being used to mean that data isn't available. Since the presence of missing values is not very frequent, and we see that week to week variation is relatively low, we make a linear interpolation of the missing data.

```
# replace missing values with NA
cwm$co2[cwm$co2 == -999.99] <- NA
```

Convert to time series

```
# Convert to Time Series
cwm_ts.1 <- ts(cwm$co2, start=c(1974,week(cwm$time)[1]), frequency = 52)
```

```
# Replace missing values by interpolating
library(zoo)
cwm_ts <- na.approx(cwm_ts.1)
```

Exploratory Time Series Data Analysis

```
str(cwm_ts)
```

```
##  Time-Series [1:2441] from 1974 to 2021: 333 333 332 332 332 ...
```

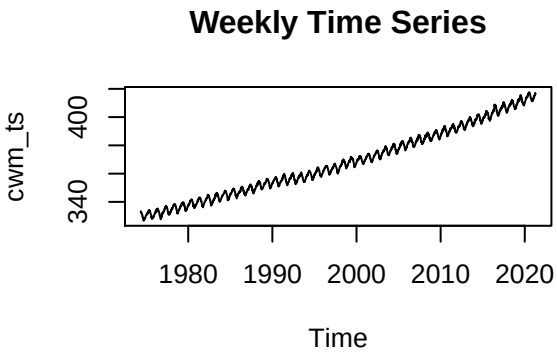
```
head(cwm_ts)
```

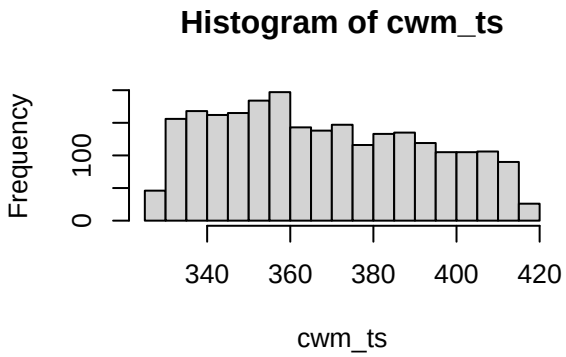
```
## Time Series:
## Start = c(1974, 20)
## End = c(1974, 25)
## Frequency = 52
## [1] 333.37 332.95 332.35 332.20 332.37 331.73
```

```
tail(cwm_ts)
```

```
## Time Series:
## Start = c(2021, 11)
## End = c(2021, 16)
## Frequency = 52
## [1] 415.42 416.01 417.12 416.92 416.51 416.31
```

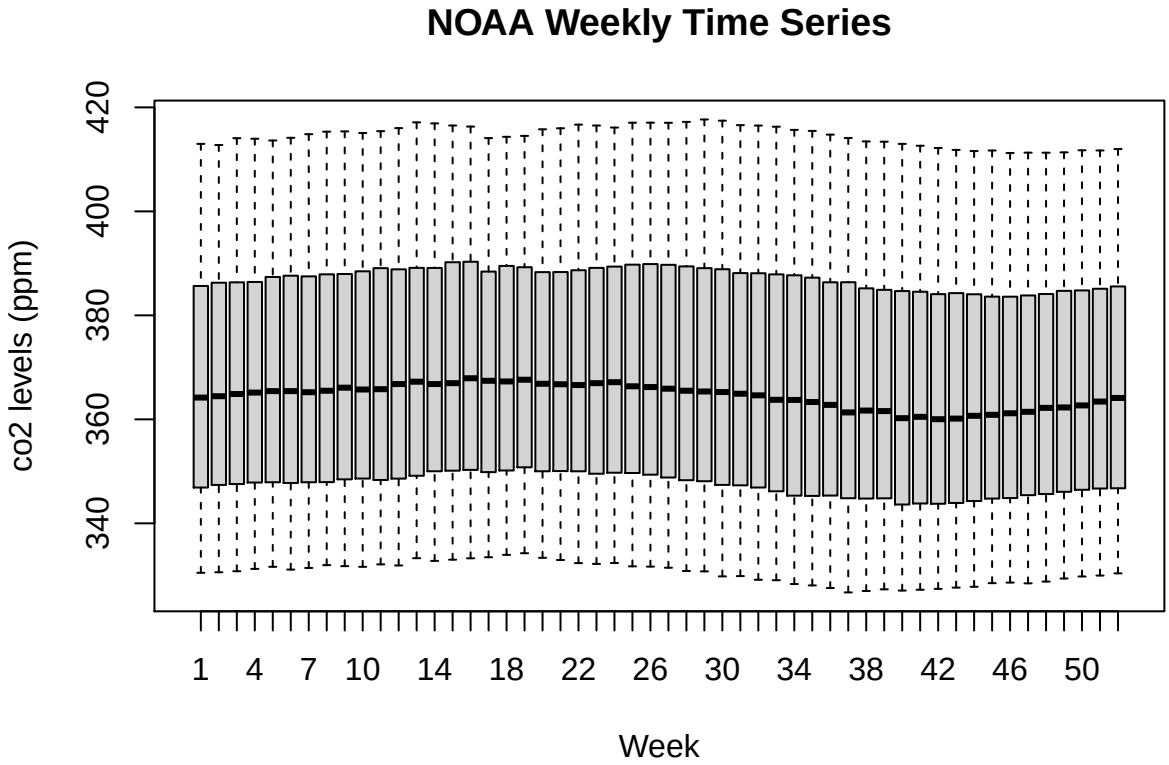
```
par(mfrow=c(2,2))
plot(cwm_ts, main = "Weekly Time Series")
hist(cwm_ts)
```





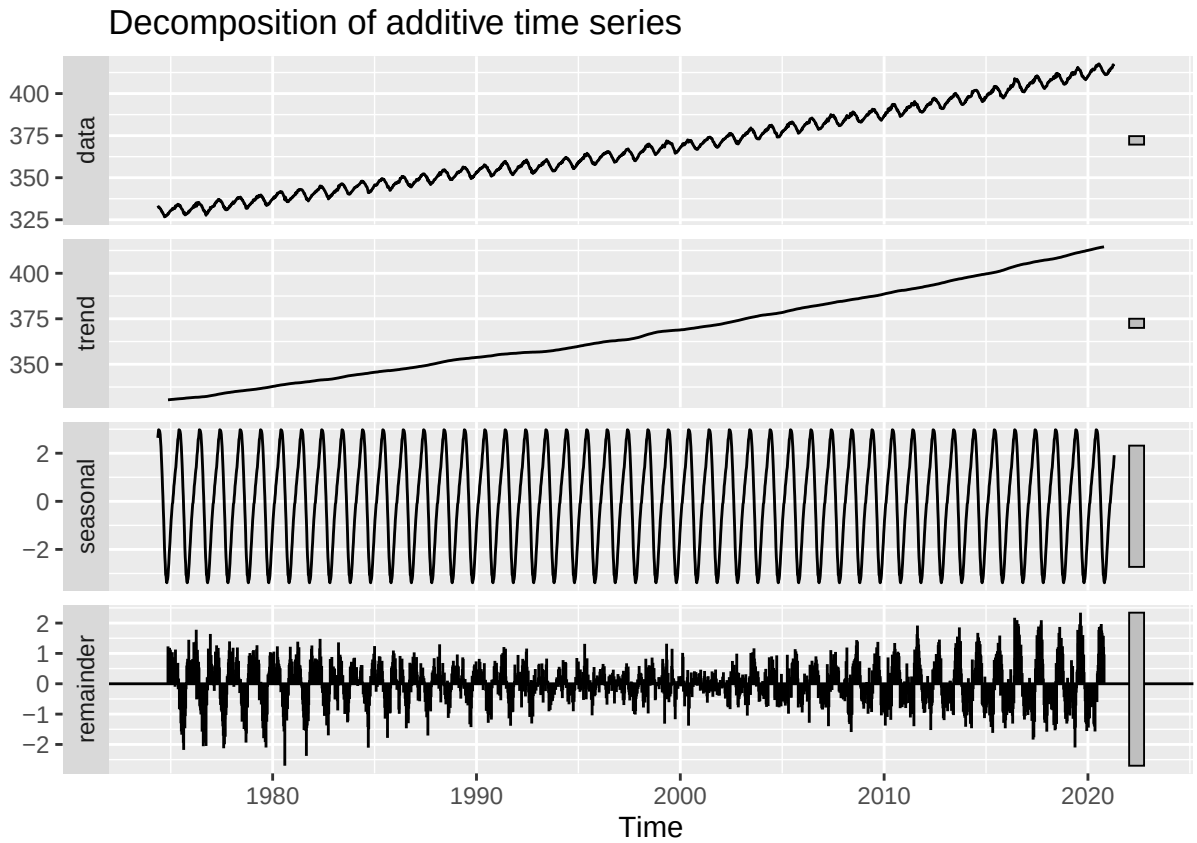
As seen in the plot of the original data, the data seems to follow what we saw earlier in the original Keeling Curve. There is an upward trend in the plot. Additionally, the co2 levels seems to decrease and increase in cycles which might be indicative of seasonality. To observe this further, we will be creating a boxplot looking at the distribution of co2 levels across each week in a year. Here we see that the distribution does not stay constant throughout the weeks and is slightly lower in the middle of the

```
boxplot(cwm_ts ~ cycle(cwm_ts), main="NOAA Weekly Time Series", xlab='Week', ylab='co2 levels
```



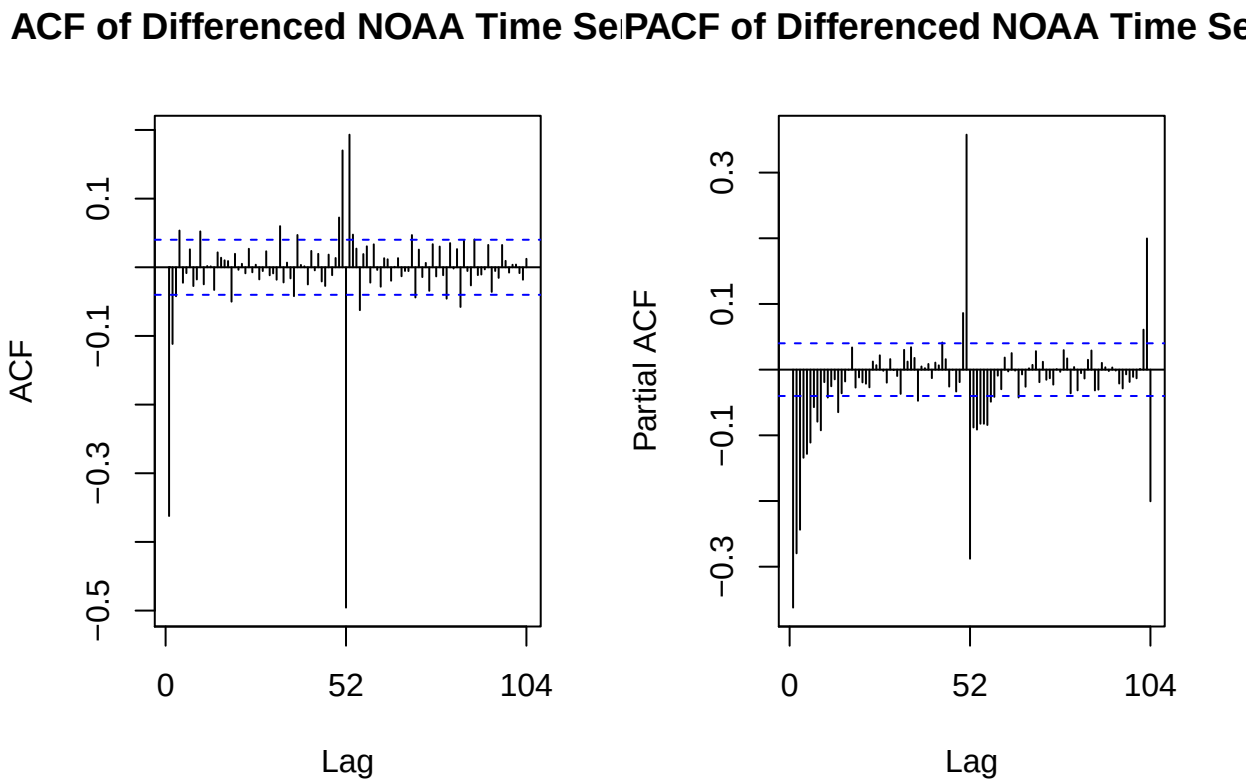
For the boxplot above, in the x-axis, 1 denotes the first week in the year and 52 denotes the last week of the year and the y-axis is the co2 levels in ppm. One observation we can see from above is that the co2 levels seem to peak in the middle of the year around week 26 and dip to their lowest point in the fall around week 45, which gives evidence for seasonality in the data. As done in Part 1, we will decompose the time series. As the seasonal variances do not seem to vary by the level of the time series, the time series is additive, therefore we do not have reason for non linear transformations such as log or box cox.

```
# Additive Decomposition
cwm_decomp <- decompose(cwm_ts,"additive")
autoplot(cwm_decomp)
```



Next, we will examine the ACF and PACF of the Time Series.

```
par(mfrow=c(1,2))
Acf(diff(diff(cwm_ts, 1), 52), main="ACF of Differenced NOAA Time Series")
Pacf(diff(diff(cwm_ts, 1), 52), main="PACF of Differenced NOAA Time Series")
```



After taking the first and 52-week seasonal difference, the autocorrelation is decreases rapidly close to 0 by lag 4 while the partial autocorrelation decays until about lag 9 before spiking again at the yearly mark. Similar to the monthly data, we expect the generating process to potentially have AR(4) or MA(4) terms and a MA(1)_{52} term. Similar to part 1, we will now test for stationarity of the differenced time series using the Augmented Dickey-Fuller Test. The hypotheses are: H_0: The time series is not stationary H_a: The time series is stationary

```
adf.test(diff(diff(cwm_ts, 1), 52))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(diff(cwm_ts, 1), 52)
## Dickey-Fuller = -18.855, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

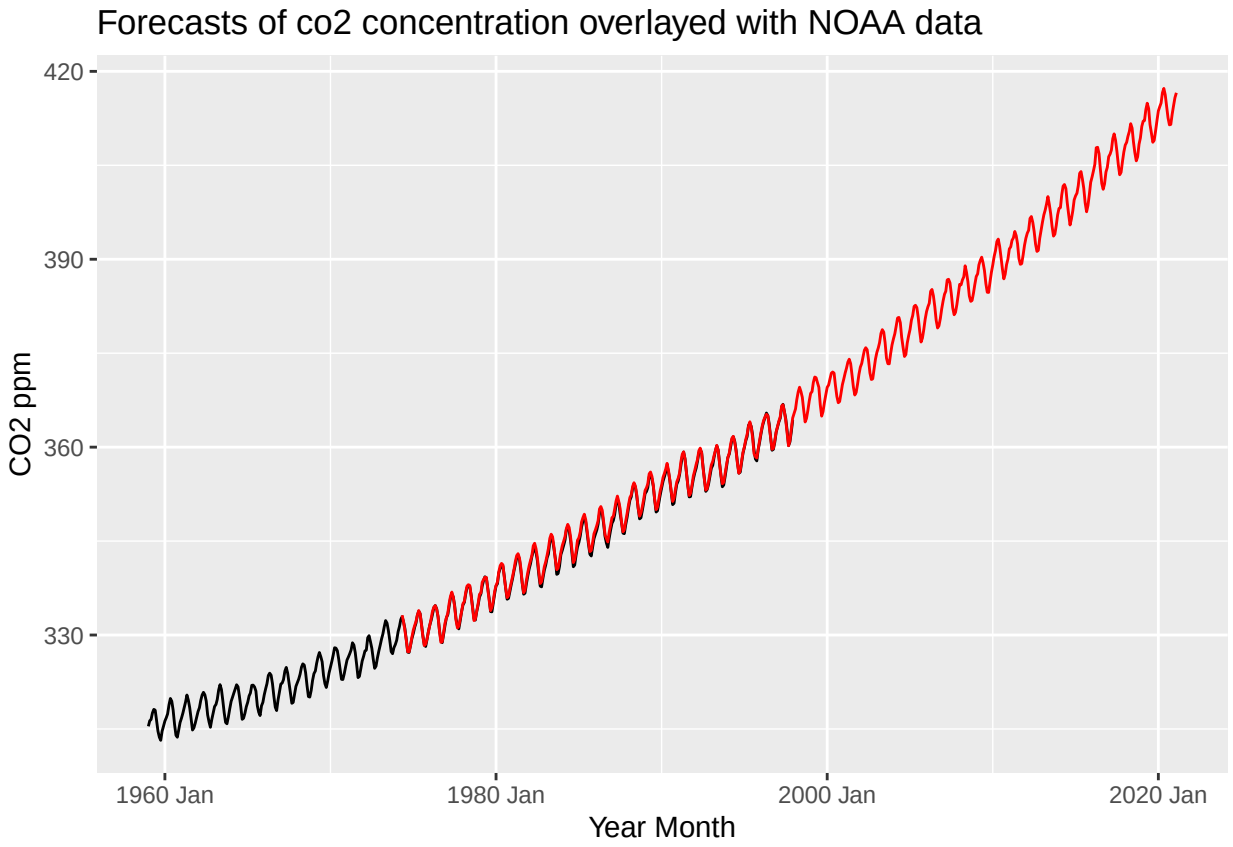
The p-value is 0.01, which is less than the alpha value (0.05), thus we reject the null hypothesis that the time series is not stationary. There is evidence to suggest that the time series is stationary are likely to contain unit roots.

Evolution of Keeling Curve:

```
# Convert to Monthly
mon <- xts(cwm$co2, order.by = cwm$time)
mon <- na.approx(mon, maxgap=31)
mon_ep <- endpoints(mon, on = "months")
cwm_mon <- period.apply(mon, INDEX = mon_ep, FUN = mean)
```

```
index(cwm_mon) = as.yearmon(index(cwm_mon))
# index(cwm_mon) = as.Date(index(cwm_mon))

cwm_tsibble = cwm_mon %>%
  fortify.zoo(melt = TRUE) %>%
  mutate(Index=yearmonth(Index)) %>%
  as_tsibble(index = "Index") %>%
  dplyr::select(-Series)
names(cwm_tsibble) = tolower(names(cwm_tsibble))
```

```
# Forecasts Plot from Part 2 overlayed with Plot from NOAA Data (Part 4)
autoplot(co2) + autolayer(cwm_tsibble, color='red') +
  ggtitle("Forecasts of co2 concentration overlayed with NOAA data") +
  xlab("Year Month") + ylab("CO2 ppm")
```



First, I will start by observing the Keeling Curve from 1997 to 2021. Above is the Original Keeling curve (black), overlayed with the NOAA dataset from 1974 to 2021 (in red). Overall, the same trend from the original Keeling Curve continued from 1997 on - there is an upward trend with cycles of season effects. The kneeling curve returns to a slight convex shape following 1998 after a slight plateau from around 1990 to 1995.

Next, we will be comparing the accuracy metrics of the forecasts produced by models in part 2 and 3.

```
data_to_forecast = cwm_tsibble %>% filter_index('1998 Jan'~.)
head(data_to_forecast)
```

```
## # A tsibble: 6 x 2 [1M]
##      index value
##      <mth> <dbl>
## 1 1998 Jan  365.
## 2 1998 Feb  366.
## 3 1998 Mar  368.
## 4 1998 Apr  369.
## 5 1998 May  370.
## 6 1998 Jun  369.
```

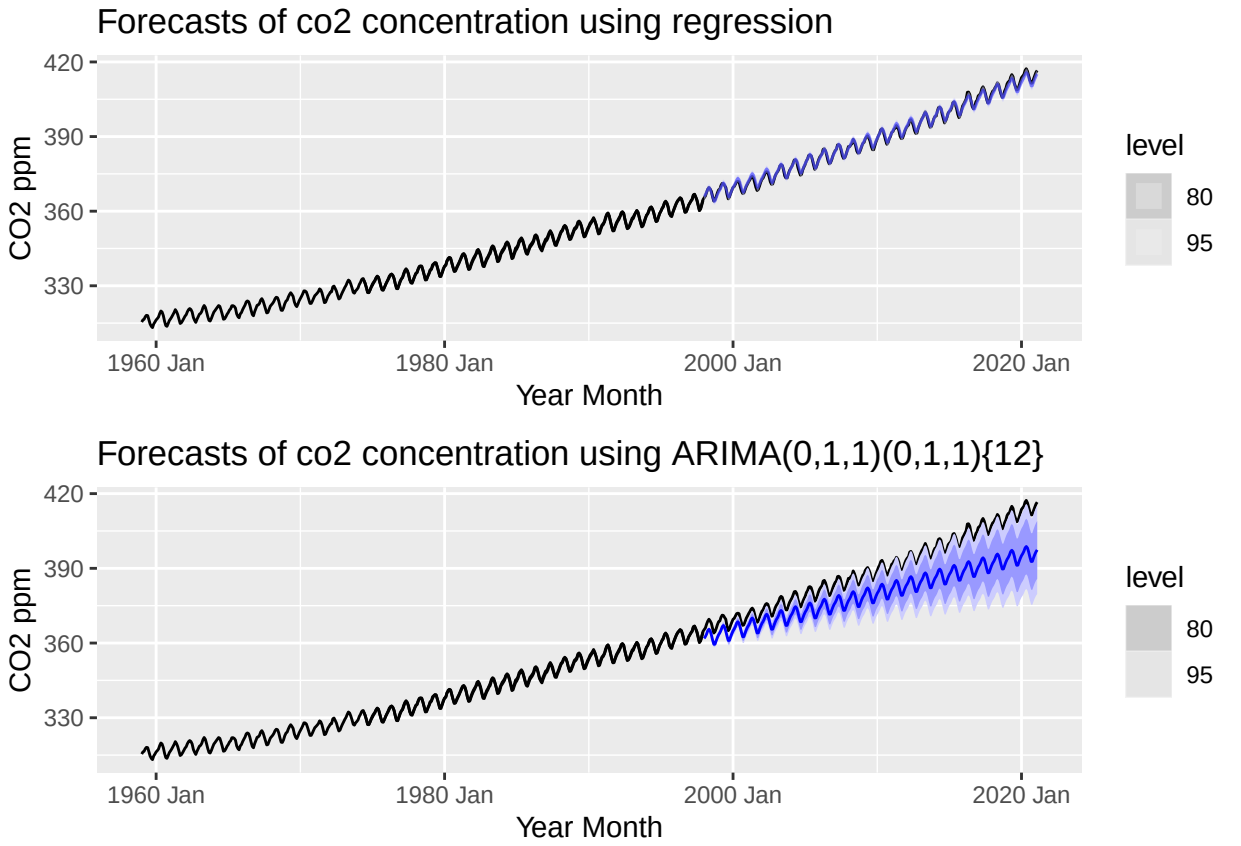
```
seasonal.and.quadratic.forecast = seasonal.and.quadratic %>%
  forecast(new_data=data_to_forecast, level=95)
seasonal.and.quadratic.accuracy = seasonal.and.quadratic.forecast%>%
  accuracy(data_to_forecast)

part.3.ARIMA.forecast = part.3.ARIMA %>%
  forecast(new_data=data_to_forecast, level=95)
part.3.ARIMA.accuracy = part.3.ARIMA.forecast %>% accuracy(data_to_forecast)

seasonal.and.quadratic.accuracy
```

```
## # 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(value ~ trend(~ Test  -0.0990 0.767 0.616 -0.0294 0.157   NaN   NaN 0.837
```

```
part.3.ARIMA.accuracy
```

```
## # 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(value ~ 0 + pdq(0~ Test   9.70  10.7  9.70  2.46  2.46   NaN   NaN 0.984
```

```
fc1 = autoplot(co2) +
  autolayer(cwm_tsibble) +
  autolayer(seasonal.and.quadratic.forecast, alpha=0.4) +
  ggtitle("Forecasts of co2 concentration using regression") +
  xlab("Year Month") + ylab("CO2 ppm")
fc2 = autoplot(co2) +
  autolayer(cwm_tsibble) +
  autolayer(part.3.ARIMA.forecast, alpha=1) +
  ggtitle("Forecasts of co2 concentration using ARIMA(0,1,1)(0,1,1){12}") +
  xlab("Year Month") + ylab("CO2 ppm")
grid.arrange(fc1, fc2, nrow = 2)
```

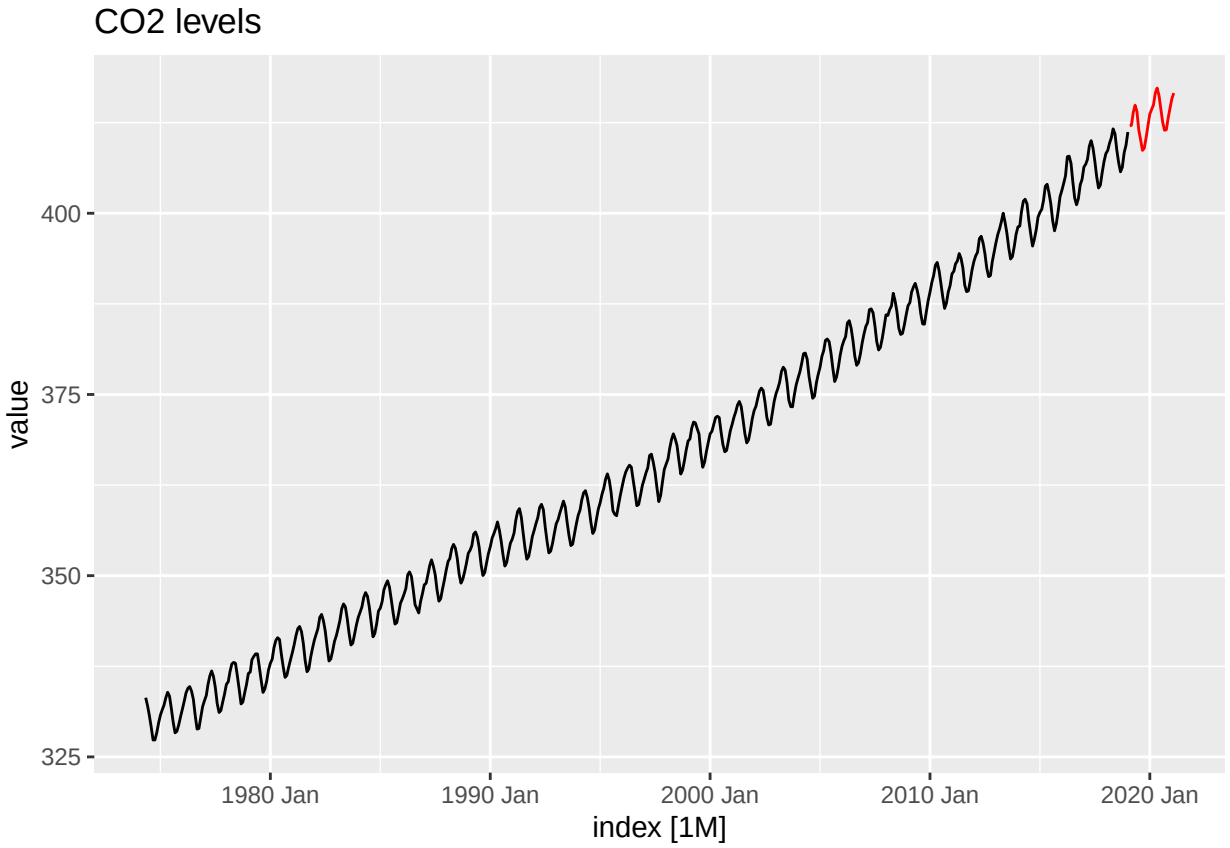


The return to the convex shape seen in the observed Keeling curve is followed much closer by the seasonal and quadratic regression model compared to the forecast of the $ARIMA(0,1,1)(0,1,1)_{12}$ produced in part 3. The seasonal and quadratic regression produces a better forecast from 1998 to present based on all metrics. The seasonal and quadratic regression has a mean absolute percentage error (MAPE) of 0.157 which is significantly better than the 2.455 of the SARIMA model. Both models contain the true series within the 95% prediction interval band.

We split the NOAA series into training and test sets, using the final two years of observations as the test set. We then fit an ARIMA model to the series following all appropriate steps. This includes measurement and discussion of how the model performs in-sample and (psuedo-) out-of-sample, comparing candidate models and explaining our choices. We then generate predictions for when atmospheric CO2 is expected to be at 420 ppm and 500 ppm levels, considering prediction intervals as well as point estimates in the calculations. We also generate a prediction for atmospheric CO2 levels in the year 2100 and show confidence intervals.

We split the data into training and test sets, taking the final two years as a test set period for pseudo-out-of-sample forecasting performance as follows:

```
mauna_loa_monthly <- cwm_tsibble
mlm.co2.training <- mauna_loa_monthly %>% filter_index(~'2019 Jan')
mlm.co2.test <- mauna_loa_monthly %>% filter_index('2019 Feb'~.)
mlm.co2.training %>% autoplot(value) +
  autolayer(mlm.co2.test, value, colour = 'red') + ggtitle("CO2 levels")
```



Where the training set is represented in black and the test set for the pseudo-out-of-sample is in red.

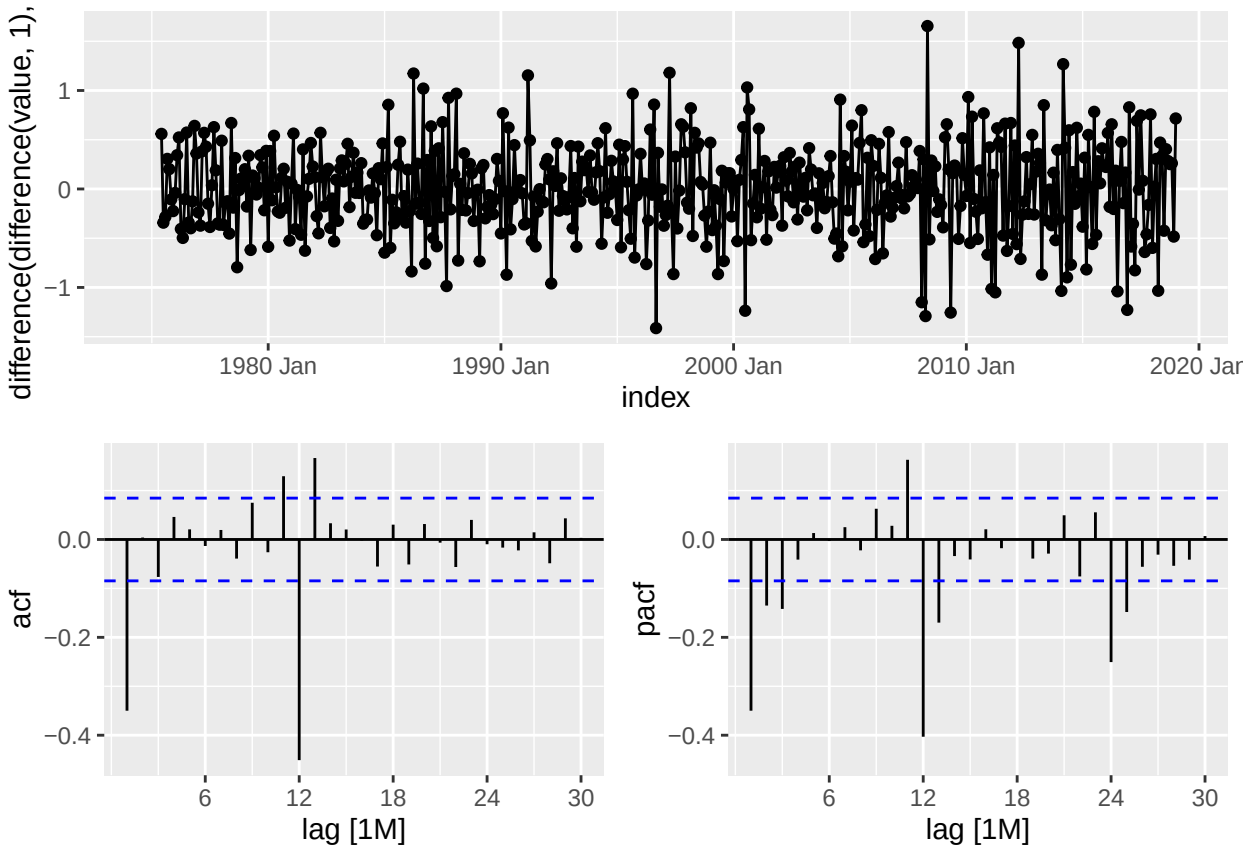
```
mlm.co2.training %>% features(value, unitroot_ndiffs)
```

```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      1
```

```
mlm.co2.training %>% features(value, unitroot_nsdiffs)
```

```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       1
```

```
mlm.co2.training %>% gg_tsdisplay(y=difference(difference(value, 1), 12),
                                  plot = 'partial', lag_max = 30)
```



Examining the correlograms of the first difference and first seasonal difference series. There is significant auto and partial auto correlation at lag 1 and 12. There are some smaller pacf values for lag 2, 3 as well. We will use these findings in initial examination of few models following an auto arima step as

Use the auto.arima function to fit the best model and coefficients, given the default parameters including seasonality as TRUE. This becomes a baseline for our search for the best model in the subsequent section below. Due the fact that our times series exhibits seasonality, we will use a model called SARIMA, a seasonality ARIMA. We write SARIMA as ARIMA(p,d,q)(P, D, Q)m, where p is the number of autoregressive components, d is the degree of differencing, q is the number of moving average terms, m refers to the number of periods in each season and (P,D,Q) represents the (p,d,q) for the seasonal part of the time series. :

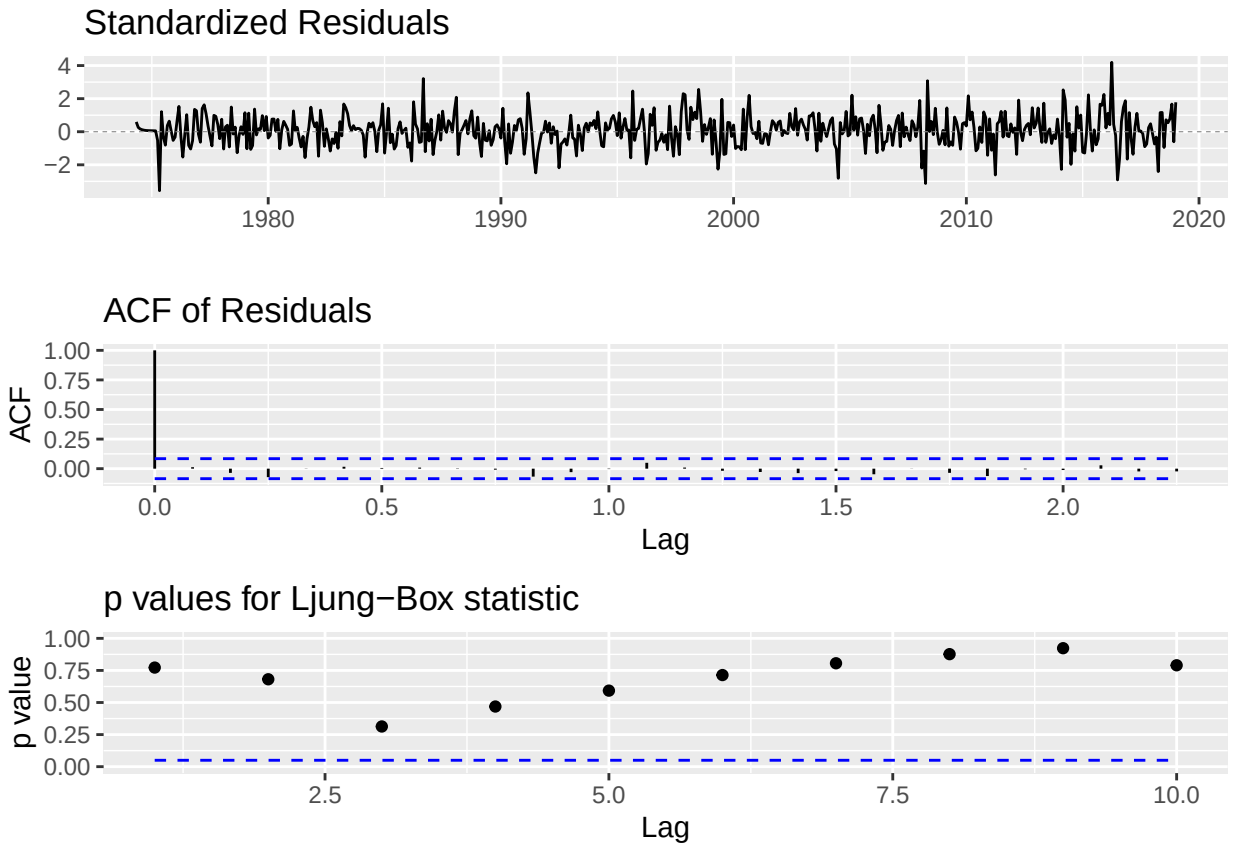
```
arimaCO2 <- mlm.co2.training %>% auto.arima()
arimaCO2
```

```
## Series: .
## ARIMA(0,1,1)(2,1,2)[12]
##
## Coefficients:
##           ma1     sar1    sar2     sma1     sma2
##       -0.4322  -0.4806  0.0166  -0.3494  -0.4678
## s.e.   0.0432   1.0267  0.0733   1.0255   0.9040
##
## sigma^2 estimated as 0.1051:  log likelihood=-152.38
```

```
## AIC=316.77   AICc=316.93   BIC=342.34
```

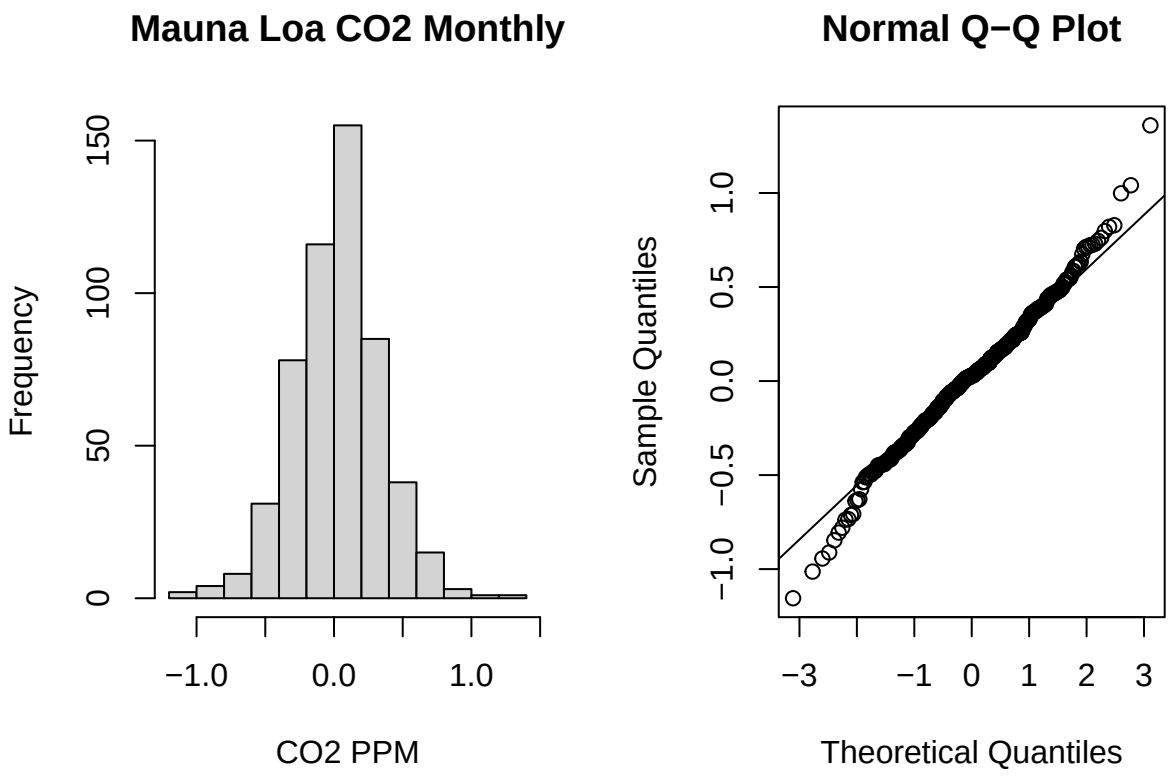
We expected model parameters as lag 1 for differencing (d), an autoregressive term of first lag (p) and a moving average model of order 1 (q). The second part of the ARIMA model (P,D,Q) corresponds to the seasonal component (12 indicates the number of periods per season, in this case months).

```
ggtsdiag(arimaCO2)
```



The above figure shows the ACF of the residuals for the model. The "lag" (time span between observations) is shown along the horizontal axis, and the autocorrelation is on the vertical axis. The blue lines indicate bounds for a 95% statistical significance. The residual plots appear to be centered around 0 as noise, with no pattern and all bounded within the 95% significance level. Ljung-Box test show a p-value that is pretty high; the SARIMA model is a fairly good fit. ARIMA() fits the model using maximum likelihood estimation (assuming Gaussian residual series). Now, plot the Q–Q plots, which measures the agreement of a fitted distribution with observed data:

```
# qqnorm is a generic function the default method of which produces a normal QQ plot of the va
par(mfrow=c(1,2))
hist(residuals(arimaCO2), main='Mauna Loa CO2 Monthly', xlab='CO2 PPM')
qqnorm(residuals(arimaCO2))
qqline(residuals(arimaCO2))
```



The linearity of the points suggests that the data are normally distributed with mean = 0.

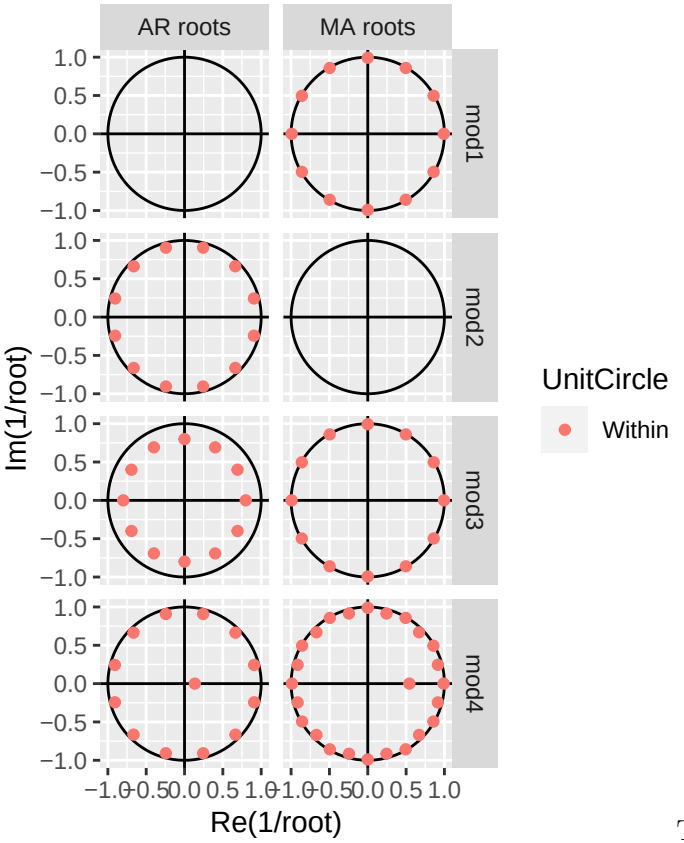
Rather than relying on auto.arima as the optimal ARIMA model, we will use it as as initial point for searching the parameter space of p,q,d, m and (P,D,Q) to iteratively scan this space and pick the best model based on metrics AICc and BIC. The correlograms of the first and seasonal-differenced series show significant spikes in ACF and PACF at lag 12, potentially suggesting a seasonal MA(1) and/or AR(1) component. There may be other components but let's first compare an $ARIMA(0,1,0)(0,1,1)_{12}$, an $ARIMA(0,1,0)(1,1,0)_{12}$, $ARIMA(0,1,0)(1,1,1)_{12}$ and $ARIMA(1,1,1)(1,1,2)_{12}$, using in-sample and pseudo-out-of-sample accuracy comparisons. Let us start with three candidate models. We fit the ARIMA models to the Box-Cox transformed series; when forecasting, fable will take care of the back-transformation and bias adjustment automatically.

```
mlm.co2.models <- mlm.co2.training %>%
  model(mod1 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(0,1,1, period=12)),
        mod2 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(1,1,0, period = 12)),
        mod3 = ARIMA(value ~ 0 +
                       pdq(0,1,0) + PDQ(1,1,1, period = 12)),
        mod4 = ARIMA(value ~ 0 +
                       pdq(1,1,1) + PDQ(1,1,2, period = 12)))

# The `glance()` function applied to a set of ARIMA models shows the variance of residuals (si

glance(mlm.co2.models)
```

```
## # A tibble: 4 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
```

```
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 mod1    0.120   -192.  387.  387.  396. <cpl [0]>  <cpl [12]>
## 2 mod2    0.162   -263.  529.  529.  538. <cpl [12]> <cpl [0]>
## 3 mod3    0.119   -191.  387.  387.  400. <cpl [12]> <cpl [12]>
## 4 mod4    0.105   -151.  315.  315.  341. <cpl [13]> <cpl [25]>
```

```
# Inverse roots all lie within the unit circle
gg_arma(mlm.co2.models)
```



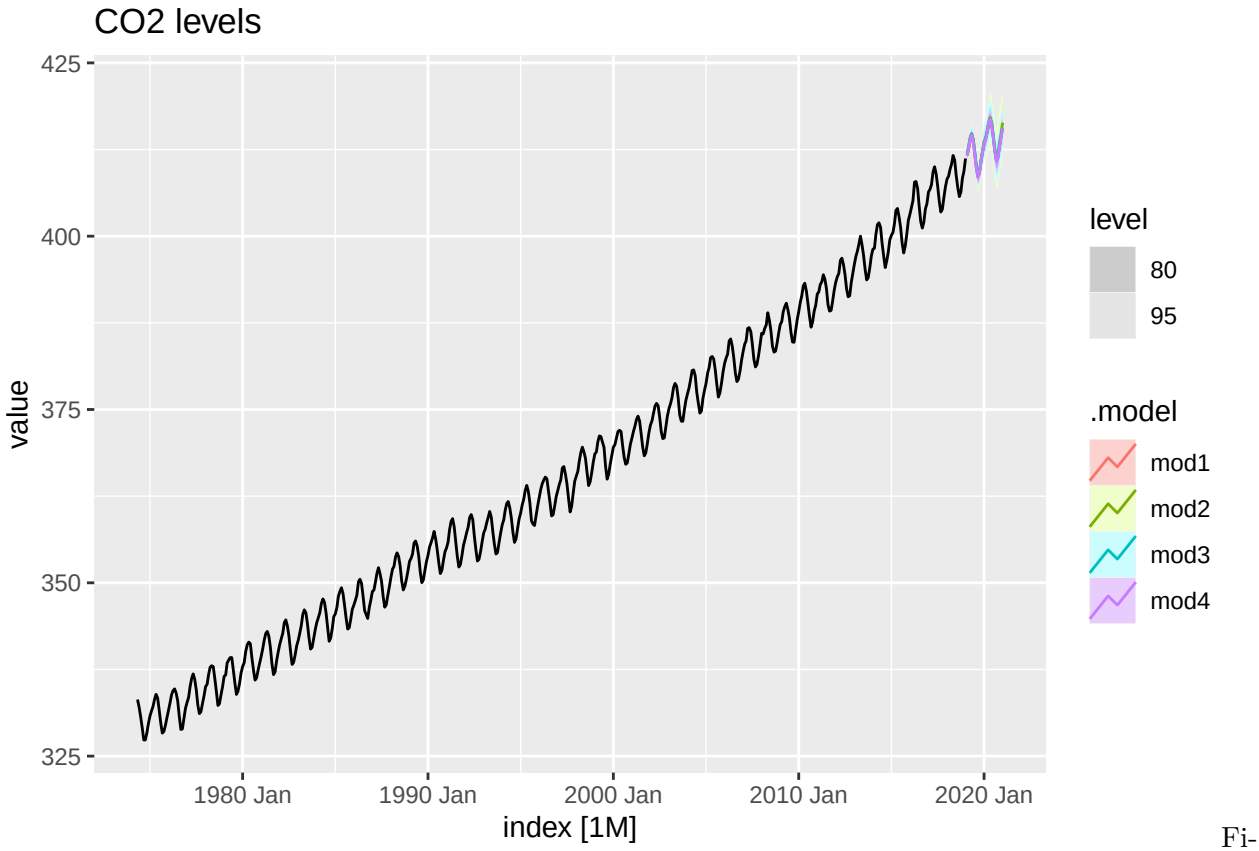
The accuracy() function provides various comparative measures of in-sample performance for the four models as:

```
mlm.co2.models %>% accuracy()
```

```
## # A tibble: 4 x 10
##   .model .type          ME  RMSE   MAE      MPE   MAPE  MASE RMSSE    ACF1
##   <chr>  <chr>       <dbl> <dbl> <dbl>    <dbl>  <dbl> <dbl> <dbl>   <dbl>
## 1 mod1   Training 0.0184   0.341 0.262 0.00486   0.0712 0.146 0.178 -0.323
## 2 mod2   Training 0.00355  0.397 0.303 0.000921  0.0823 0.169 0.207 -0.360
## 3 mod3   Training 0.0185   0.341 0.262 0.00487   0.0711 0.146 0.177 -0.322
## 4 mod4   Training 0.0281   0.318 0.243 0.00746   0.0662 0.135 0.166 -0.0146
```

The performance of the models is very similar with a slight advantage of model4 ($ARIMA(1,1,1)(1,1,2)_12$) based on AICc, BIC, and rmse performing better than other parameters.

```
mlm.co2.models_forecast <- mlm.co2.models %>% forecast(level = c(95), h = 24)
mlm.co2.training %>% autoplot(value) +
  autolayer(mlm.co2.models_forecast, value, colour = 'red') + ggtitle("CO2 levels")
```



Finally we can plot a forecast of the time series, for all 4 candidate models, using the forecast function, with a 95% confidence interval where h is the forecast horizon periods in months. The two-year forecasts for the three models generated are very similar.

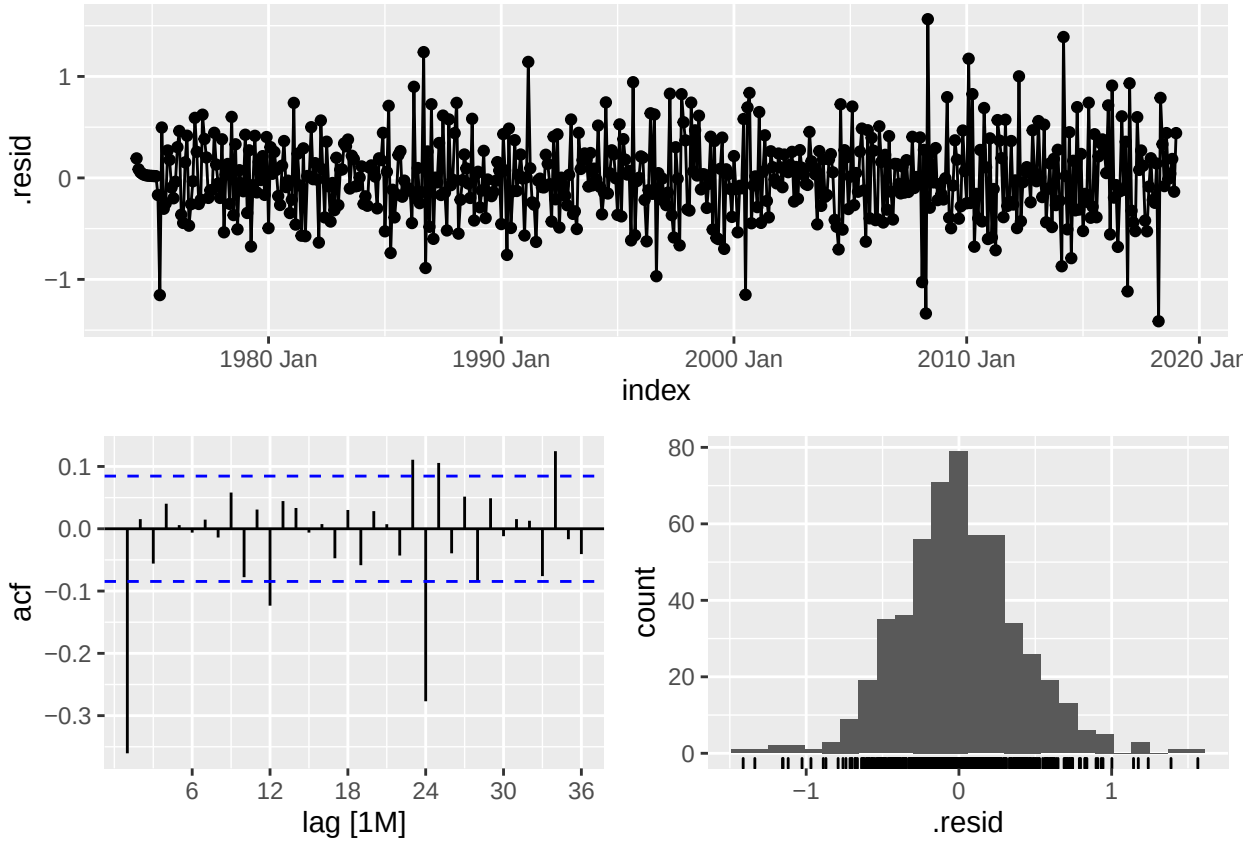
Pseudo-out-of-sample performance can be assessed applying the accuracy function to the test set.

```
mlm.co2.models_forecast %>% accuracy(mlm.co2.test)
```

```
## # A tibble: 4 x 10
##   .model .type      ME  RMSE   MAE     MPE   MAPE  MASE RMSSE   ACF1
##   <chr>  <chr>   <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl> <dbl>  <dbl>
## 1 mod1   Test  0.00820 0.287 0.223 0.00184 0.0541   NaN   NaN 0.214
## 2 mod2   Test  0.0179  0.291 0.244 0.00430 0.0591   NaN   NaN 0.0866
## 3 mod3   Test  0.0191  0.284 0.224 0.00446 0.0544   NaN   NaN 0.196
## 4 mod4   Test  0.207   0.327 0.292 0.0501  0.0706   NaN   NaN 0.0722
```

Using the RMSE metric, the $ARIMA(0,1,0)(1,1,1)_{12}$ model(2) is marginally preferred. We will check its residuals and perform a portmanteau test with a suitable number of lags. Based on this test there is statistically significant evidence to reject the null hypothesis of no autocorrelation between residuals however so this model fails the assumption that residuals should be similar to white noise. Using the model(4) $ARIMA(1,1,1)(1,1,2)_1 2$ with best insample performance, we do not see any evidence to reject the same null hypothesis.

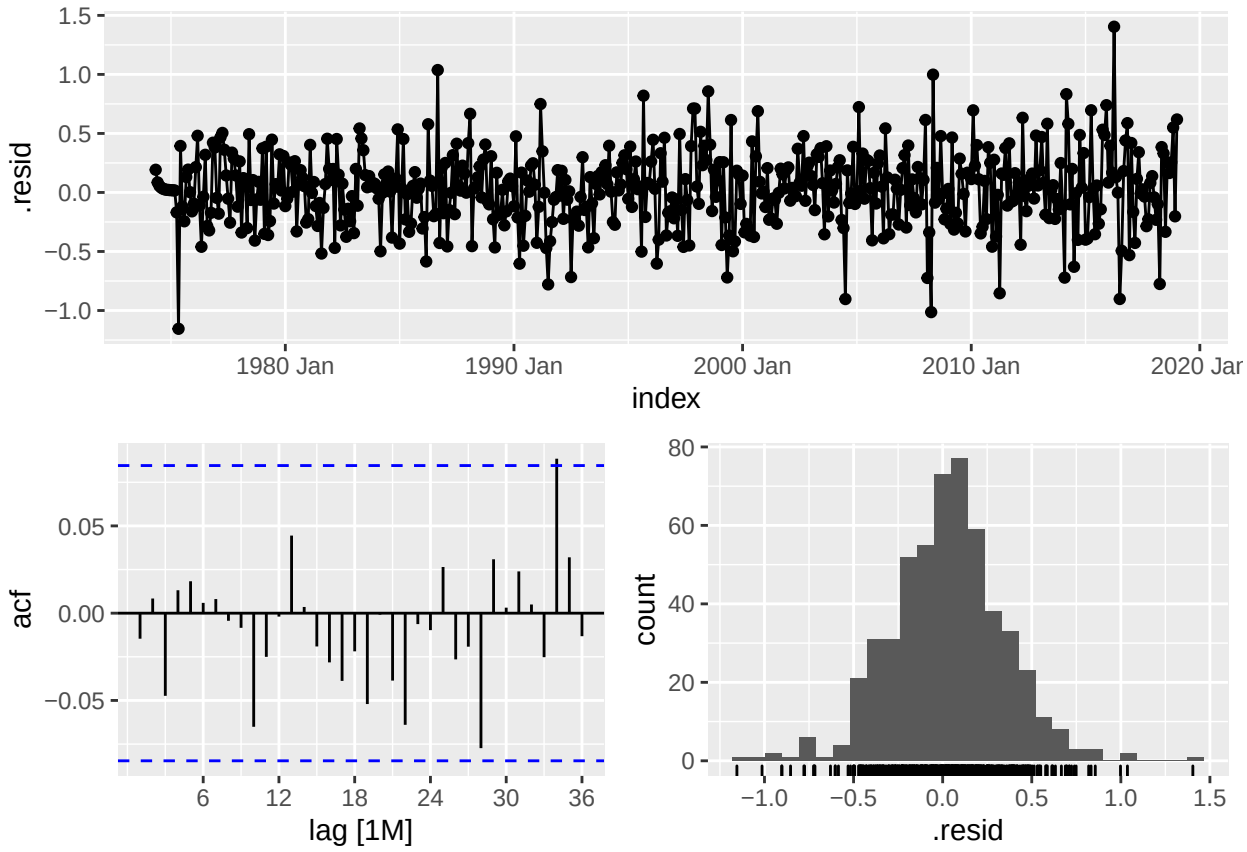
```
mlm.co2.models %>% dplyr::select(mod2) %>% gg_tsresiduals(lag_max = 36)
```




```
mlm.co2.models %>% dplyr::select(mod2) %>% augment() %>%  features(.resid, ljung_box, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 mod2      144.         0
```

```
mlm.co2.models %>% dplyr::select(mod4) %>% gg_tsresiduals(lag_max = 36)
```




```
mlm.co2.models %>% dplyr::select(mod4) %>% augment() %>%  features(.resid, ljung_box, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 mod4      12.0     0.980
```

The residuals are pretty small and relatively uniform across the entire time series period as well, the correlations are all within the 95% confidence intervals indicating a stationary process for the entire time series period.

However, we still need to establish if this is indeed the best model for parameters p, q, P and Q. A thorough parameter search is needed to scan through all candidate models and pick the best one based on a given criteria. The following code performs a loop of $ARIMA(p,1,q)(P,1,Q)_12$ models up to a maximum value of 3 for p, q, P and Q similar to part 3. A model is selected based on AICc, however this type of loop could be used to assess candidate models on the basis of other in-sample or pseduo-out-of-sample accuracy measures. Complete the given code, and modify it to include alternative measures in addition to AICc.

```
### Comment out to skip loop and read results from previous run

# part.5.results = test.ARIMA.params(p_range=0:3,
#                                    q_range=0:3,
#                                    P_range=0:3,
#                                    Q_range=0:3,
```

```
#                               train.data=mlm.co2.training,
#                               test.data=mlm.co2.test)
# write.csv(part.5.results, 'part5_arima_param_results.csv')
```

We inspect the results of the parameter search for the models based on combination of insample criteria and out of sample metrics.

```
results = read.csv('part5_arima_param_results.csv')
head(results[order(results$RMSE),], 5)
```

```
##       X p q P Q      AIC     AICc      BIC      RMSE       MAPE
## 41   41 0 2 2 0 391.6991 391.8149 413.0066 0.2529248 0.04791760
## 101 101 1 2 2 0 393.6535 393.8159 419.2224 0.2530666 0.04795615
## 57   57 0 3 2 0 393.6479 393.8104 419.2169 0.2530853 0.04794264
## 89   89 1 1 2 0 391.3932 391.5090 412.7007 0.2535379 0.04804589
## 116 116 1 3 2 0 395.2479 395.4650 425.0784 0.2537574 0.04800771
```

```
# ARIMA(0, 1, 1)(0, 1, 1)_12 best performs best on combination of AICc and BIC
```

Based on the AICc and BIC evaluation metrics, we select $ARIMA(0,1,1)(0,1,1)_{12}$ as the best model on insample metrics. This model was 3rd on AICc by less than 0.55 difference compared to the lowest AICc model and 20 points better on BIC. AICc is nearly identical amongst the top models however, the $ARIMA(0,1,1)(0,1,1)_{12}$ does exhibit a notably better BIC. Thus, we will train the final model on the full dataset with these parameters.

```
part.5.ARIMA = mauna_loa_monthly %>%
  model(ARIMA(value ~ 0 +
                pdq(0,1,1) + PDQ(0,1,1, period=12)))
part.5.ARIMA %>% report()
```

```
## Series: value
## Model: ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4472  -0.8644
## s.e.   0.0411   0.0242
##
## sigma^2 estimated as 0.1038:  log likelihood=-157.53
## AIC=321.06   AICc=321.1   BIC=333.98
```

```
augment(part.5.ARIMA) %>% features(.innov, ljung_box, lag=24, dof=2)
```

```
## # A tibble: 1 x 3
##   .model                                                lb_stat lb_pvalue
##   <chr>                                                   <dbl>     <dbl>
## 1 ARIMA(value ~ 0 + pdq(0, 1, 1) + PDQ(0, 1, 1, period = 12))    17.7     0.725
```

The final estimated ARIMA model to be used for forecasting on the month-averaged NOAA dataset is

$$(1-B)(1-B^{12})y_t = (1-0.4472B)(1-0.8644B^{12})e_t$$

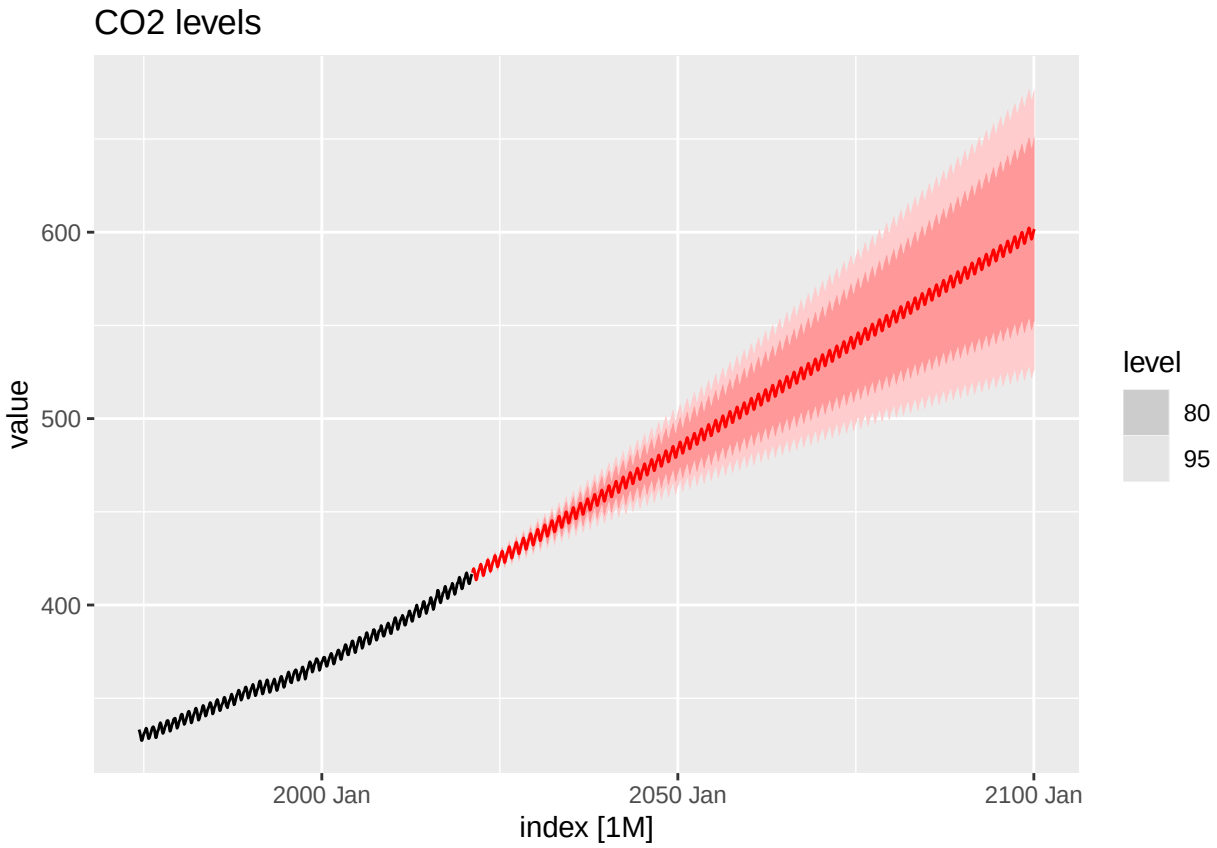
where y_t is the co2 ppm at time t. This model has a large p-value under the Ljung-Box test, confirming that the residuals are similar to white noise.

Holt Winters to Evaluate Seasonality:

```
# mod.holt.winters<-HoltWinters(mlm.co2.training)
# mod.holt.winters
```

Let us generate predictions for when atmospheric CO2 is expected to be at 420 ppm and 500 ppm levels. We show a plot of such a forecast along with the 95% confidence forecast interval as depicted in the plot below:

```
# prediction from single model with confidence levels
mlm.co2.models_forecast2 <- part.5.ARIMA %>% forecast(level = c(95),h=(2100-2021)*12)
mauna_loa_monthly %>% autoplot(value) +
  autolayer(mlm.co2.models_forecast2, value, colour = 'red') + ggtitle("CO2 levels")
```



Obtain point forecast:

```
# forecast in increments of 1 month starting from present.
fc.mod.forecast<- part.5.ARIMA %>%
  forecast(level = c(95),h=(2100-2021)*12) %>%
  hilo() %>%
  unpack_hilo("95%")
head(fc.mod.forecast)
```

```
## # A tsibble: 6 x 7 [1M]
```

```
## # Key:       .model [1]
##   .model             index        value .mean                  `80%` `95%_lower`
##   <chr>              <mth>       <dist> <dbl>                 <hilo>       <dbl>
## 1 ARIMA(value ~ ~ 2021 Mar  N(417, 0.1)  417. [416.9373, 417.7630]80        417.
## 2 ARIMA(value ~ ~ 2021 Apr N(419, 0.14)  419. [418.3934, 419.3368]80        418.
## 3 ARIMA(value ~ ~ 2021 May N(420, 0.17)  420. [419.0055, 420.0536]80        419.
## 4 ARIMA(value ~ ~ 2021 Jun  N(419, 0.2)  419. [418.0583, 419.2014]80        418.
## 5 ARIMA(value ~ ~ 2021 Jul N(417, 0.23)  417. [416.1210, 417.3519]80        416.
## 6 ARIMA(value ~ ~ 2021 Aug N(415, 0.26)  415. [414.1143, 415.4272]80        414.
## # ... with 1 more variable: 95%_upper <dbl>
```

```
tail(fc.mod.forecast)
```

```
## # A tsibble: 6 x 7 [1M]
## # Key:       .model [1]
##   .model             index        value .mean                  `80%` `95%_lower`
##   <chr>              <mth>       <dist> <dbl>                 <hilo>       <dbl>
## 1 ARIMA(value ~ ~ 2099 Sep N(596, 1460)  596. [547.2823, 645.2035]80        521.
## 2 ARIMA(value ~ ~ 2099 Oct N(597, 1464)  597. [547.6427, 645.7065]80        522.
## 3 ARIMA(value ~ ~ 2099 Nov N(598, 1468)  598. [549.1938, 647.4000]80        523.
## 4 ARIMA(value ~ ~ 2099 Dec N(600, 1472)  600. [550.5234, 648.8717]80        524.
## 5 ARIMA(value ~ ~ 2100 Jan N(601, 1477)  601. [551.7583, 650.2486]80        526.
## 6 ARIMA(value ~ ~ 2100 Feb N(602, 1481)  602. [552.3705, 651.0025]80        526.
## # ... with 1 more variable: 95%_upper <dbl>
```

```
lower_95_420 = fc.mod.forecast$index[min(which(fc.mod.forecast['95%_lower'] >= 420))]
upper_95_420 = fc.mod.forecast$index[min(which(fc.mod.forecast['95%_upper'] >= 420))]
# lower and upper bound of month when co2 levels pass 420ppm based on prediction intervals
c(upper_95_420, lower_95_420)
```

```
## <yearmonth[2]>
## [1] "2021 May" "2022 May"
```

```
point_420 = fc.mod.forecast[min(which(fc.mod.forecast['.mean'] >= 420)),
                            c('index', '.mean', '95%_lower', '95%_upper')]
point_420
```

```
## # A tsibble: 1 x 4 [1M]
##      index .mean `95%_lower` `95%_upper`
##      <mth> <dbl>       <dbl>       <dbl>
## 1 2022 Apr  421.        420.        423.
```

```
lower_95_500 = fc.mod.forecast$index[min(which(fc.mod.forecast['95%_lower'] >= 500))]
upper_95_500 = fc.mod.forecast$index[min(which(fc.mod.forecast['95%_upper'] >= 500))]
# lower and upper bound of month when co2 levels pass 500ppm based on prediction intervals
c(upper_95_500, lower_95_500)
```

```
## <yearmonth[2]>
## [1] "2048 Apr" "2075 Apr"
```

```
point_500 = fc.mod.forecast[min(which(fc.mod.forecast['.mean'] >= 500)),
                            c('index', '.mean', '95%_lower', '95%_upper')]
point_500
```

```
## # A tsibble: 1 x 4 [1M]
##      index .mean `95%_lower` `95%_upper`
##      <mth> <dbl>       <dbl>       <dbl>
## 1 2056 Apr  501.        475.        527.
```

Based on the 95% prediction intervals, we see that the forecast level will reach a co2 concentration of 420ppm between May 2021 and May 2022 with a point estimate of April 2022 with a 95% prediction interval 421.21(419.76, 422.65) while the forecast for 500 ppm level is between April 2048 and April 2075 with a point estimate of March 2056 with 95% prediction interval of 500.89(475.06, 526.71).

```
fc.mod.forecast %>%
  filter_index('2100 Jan'~.) %>%
  dplyr::select('index', '.mean', '95%_lower', '95%_upper') %>%
  head(1)
```

```
## # A tsibble: 1 x 4 [1M]
##      index .mean `95%_lower` `95%_upper`
##      <mth> <dbl>       <dbl>       <dbl>
## 1 2100 Jan  601.        526.        676.
```

The point forecast for CO2 levels by Jan 2100 is 601.0pmm with 95 prediction interval of (525.69, 676.21). This prediction interval is notably wide and there is no sufficient evidence to consider this prediction as being accurate.

Conclusion

Based on the nature of the prediction interval that grows with forecast length, it is not very helpful to produce a point forecast 80 years out into 2100. The prediction range is over 150ppm wide from (525.69, 676.21) and as seen in the 23 year forecast produced in part 3 from 1998 to 2021, the actual change, while may fall within the prediction interval, can have considerable deviation from our mean estimates. While higher order ARIMA models may produce more accurate short term forecasts, keeping the model fairly parsimonious also reduces the prediction interval due to less s.e. that is contributed to the forecast that is associated with additional terms.

Additionally, since the slight plateau in C02 growth from 1990 to 1995, the C02 concentration has been steadily increasing with a faster than linear trend in recent years. Although the first and first-seasonal differenced series is weakly stationary, the increasing slope of the series may warrant further exploration of a second order difference term. This may also be captured with the inclusion of an AR term however based on our selection metrics this did not perform best on our insample and two year test data. For a more robust model, we could perform other model selection methods such as time based nested cross validation.

As such, these models are best used for estimating macro-environmental forecasts rather than single point estimates.