W271 Group Lab 2

Due 11:59pm Pacific Time, Sunday Mar 15, 2020

Mayukh Dutta, Wade Holmes, Hersh Solanki

Contents

```r
# Insert the function to *tidy up* the code when they are printed out
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

# Load required libraries
library(car)
library(dplyr)
library(readr)
library(astsa)
library(xts)
library(forecast)
library(ggplot2)
library(plotly)
library(tsibble)
library(fable)
library(fpp2)
library(fpp3)
library(stargazer)
library(feasts)
```

Abstract

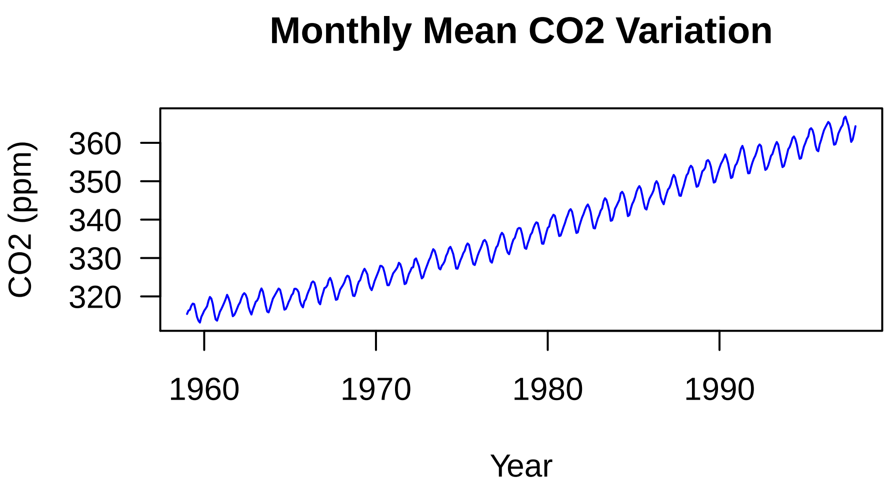
In this analysis we consider two datasets containing automospheric concentration of co2 measured in parts per million (ppm) from the Mauna Loa observatory in Hawaii. Mauna Loa is presumed to be a largely unbiased point of measurement on the globe due to it's distance from source emitters and ability to capture samples from a well-mixed global atmosphere. One dataset contains monthly values from 1956 through 1997, while the other dataset collected by NOAA begins in 1974 and includes weekly samples through 2019.

Using linear, quadratic and ARIMA models in the first three parts of this analysis, we forecast that the worst case 95% confidence ppm is at 412 in 2020. Using seasonally adjusted data from the second dataset we can confirm that actual ppm measurements exceeded our 95% confidence estimate and were measured at a seasonal peak of 414ppm in the month of May.

Using data available to year-end 2019, we can predict that co2 concentration will reach 600ppm in the year 2100 with 95% confidence. The concentration may reach levels as high as 750, with a floor at 450 given 95% confidence. The atmospheric concentration may reach levels of 420ppm as early as March 2022, and will surpass that level by the end of October 2024.

We can conclude with high confidence based on the two datasets used for this study that co2 concentration is rising in the atmosphere at predictable rates. We as a society should be alarmed.


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



Part 1 (4 points): EDA of 'co2' Series

Conduct a comprehensive Exploratory Data Analysis on the `co2` series. This should include thorough analyses of the trend, seasonal and irregular elements. Trends both in levels and growth rates should be discussed.

Let's take a look at the format of the data in the dataset

```
glimpse(co2)
```

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

The data stored as a time series with 468 observations. The data are monthly data starting from year 1959 until year 1998.

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

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

There are no missing (na) observations in the data. However, upon researching the dataset, the values of Feb, March and April of 1964 are interpolated between the values of January and May of 1964.

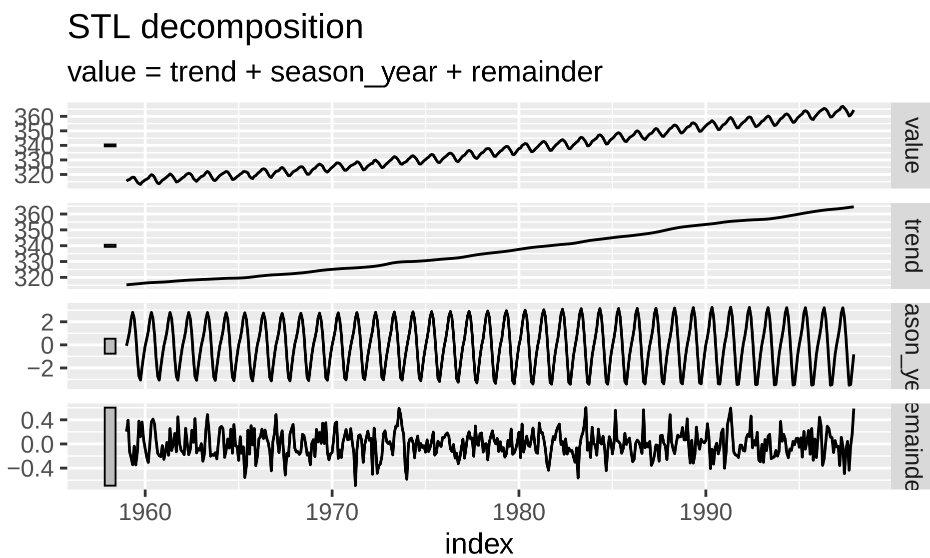
```
summary(co2)
```

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

There are no unusual values in the data, so there is no indication of missing or bad data. All data seem to be in a reasonable range between 313.2 ppm to 366.8 ppm.

A look at the trend of the series We will need to decompose the series into its components, visualize the trend, season and the noise

```
weather_tsbl <- as_tsibble(co2, index = date)
dcmp <- weather_tsbl %>% model(STL(value))
comps <- components(dcmp)
comps %>% autoplot()
```



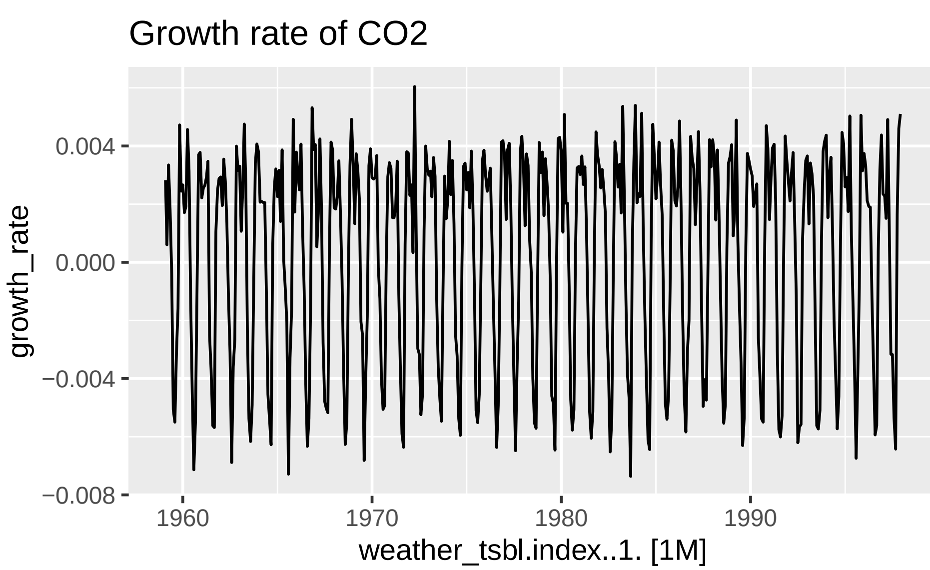
The levels of CO2 has been definitely trending upwards. There is also a very strong seasonal effect seen in the series. There are certain shocks in the irregular component of the series. The first shock was seen in 1973/1974, and the shocks seem to be more prevalent during the later years, especially in the mid 1980s.

Let's examine the growth rate of CO2 levels over time

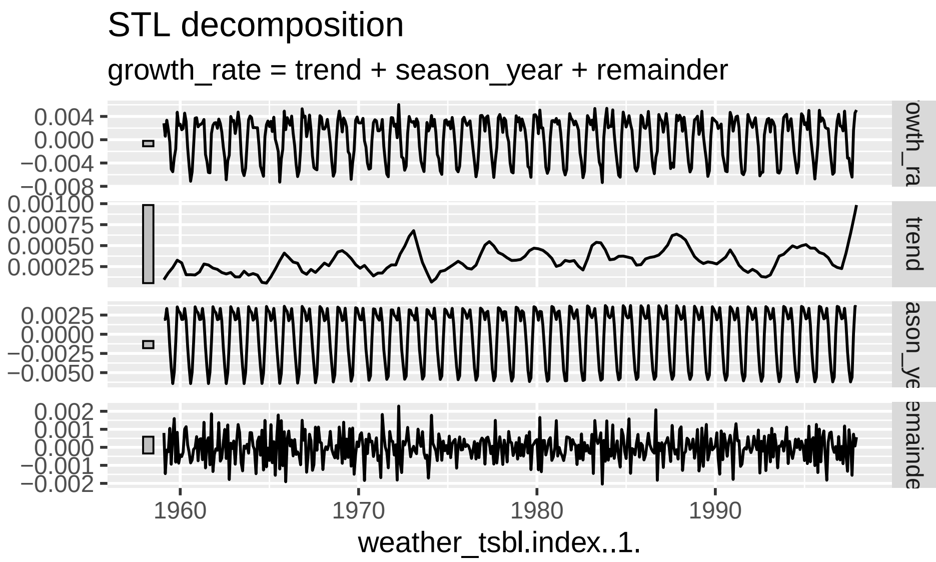
```
growth_rate <- diff(weather_tsbl$value)/(weather_tsbl$value)[-length(weather_tsbl$value)]
gr_tsibble <- as_tsibble(data.frame(growth_rate, weather_tsbl$index[-1]))
```

```
## Using `weather_tsbl.index..1.` as index variable.
```

```
gr_tsibble %>% autoplot(growth_rate) + ggtitle("Growth rate of CO2")
```



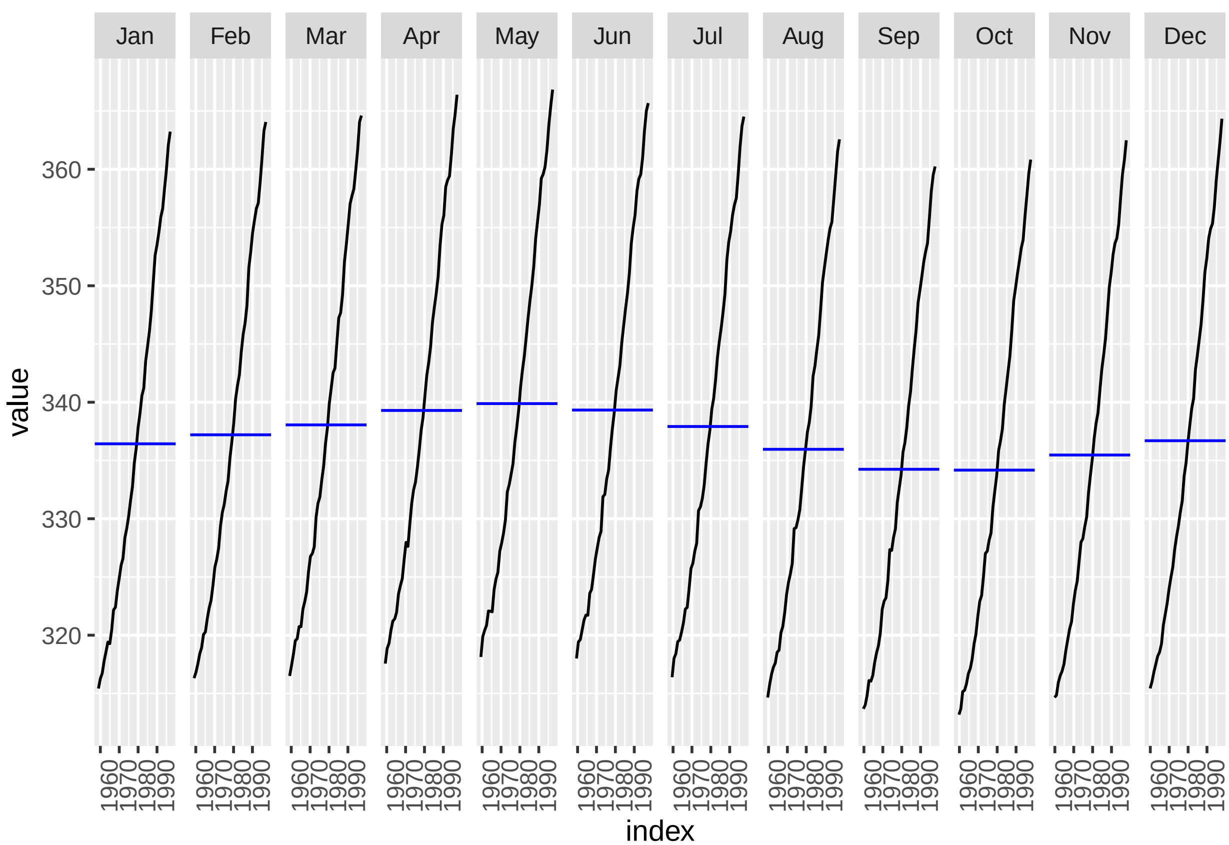

```
dcmp <- gr_tsibble %>% model(STL(growth_rate))
comps <- components(dcmp)
comps %>% autoplot()
```



The growth rate seems to have been uniform over the years from the time series plot. However when we decompose the series into its components, we can clearly see an upward trend in the growth rates of CO2 level.

A closer look at the seasonal effects

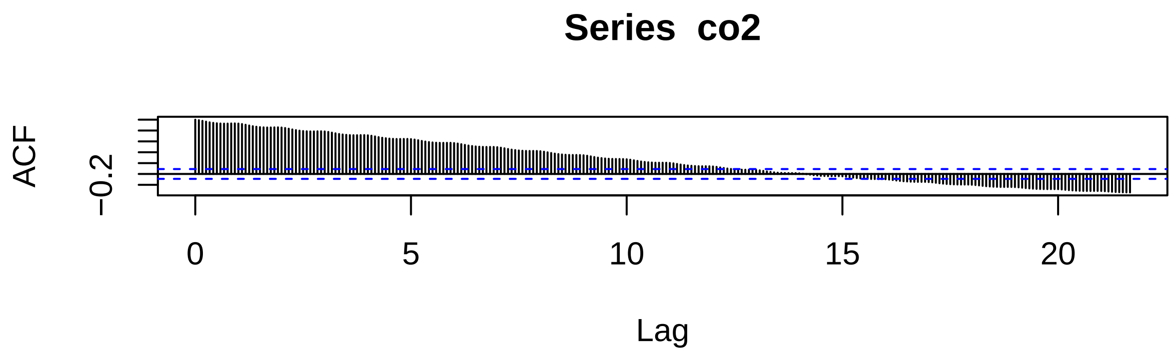
```
weather_tsbl <- as_tsibble(co2, key = origin)
weather_tsbl %>% gg_subseries(value)
```

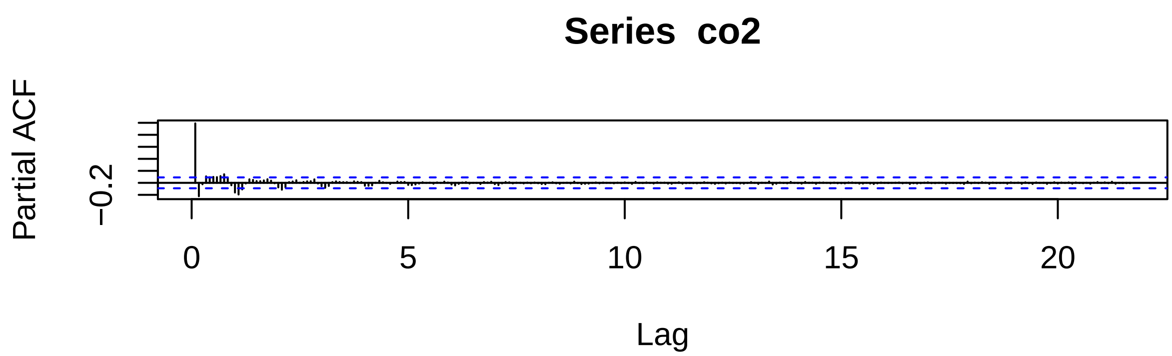


From the seasonal plot, it is clear that there is a gradual rise in the CO2 levels between February and May with the levels reaching the peak in May and then gradually falling to a low level in October. Again this is been the same pattern every single year starting from 1959.

Examine the ACF and PACF charts

```r
par(mfrow = c(2, 1))
acf(co2, lag.max = 260)
pacf(co2, lag.max = 260)
```





The ACF decays down very slowly and the lags all the way back to 5 years ago also shows high correlations. The PACF drops rapidly after the 1st lag. This shows that this may be an AR(1) process. However since the series has a very strong trend this will have to be de-trended first.

Part 2 (3 points): Linear Time Trend to 'co2' series

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

For fitting a linear time trend model, we will need to convert the date index to a numeric field

```r
weather_data <- weather_tsbl %>% mutate(time_index = row_number())
summary(weather_data)
```

```
##      index                 value           time_index
```

```
##  Min.   :1959 Jan   Min.   :313.2   Min.   :  1.0
##  1st Qu.:1968 Sep   1st Qu.:323.5   1st Qu.:117.8
##  Median :1978 Jun   Median :335.2   Median :234.5
##  Mean   :1978 Jun   Mean   :337.1   Mean   :234.5
##  3rd Qu.:1988 Mar   3rd Qu.:350.3   3rd Qu.:351.2
##  Max.   :1997 Dec   Max.   :366.8   Max.   :468.0
```

We fit a linear model to the data using the lm() function

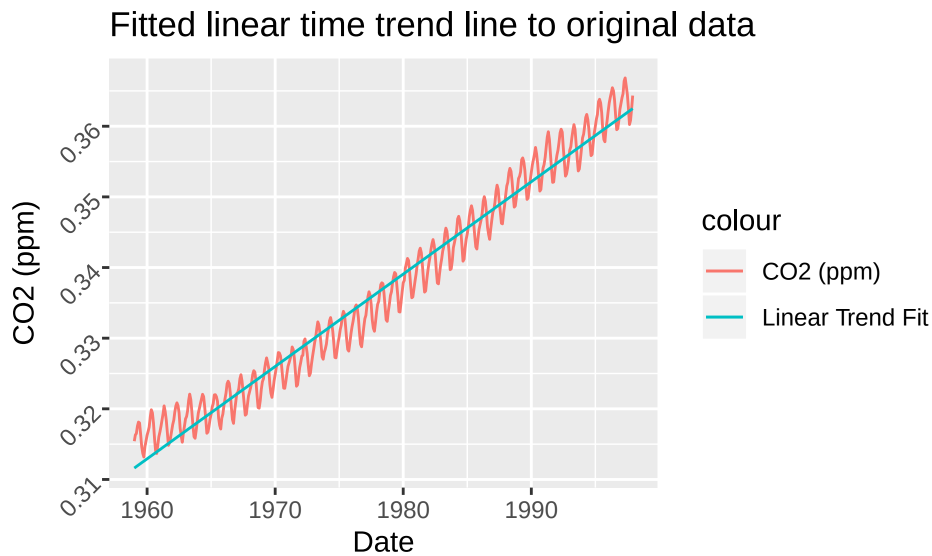
```
x.lm <- lm(value ~ time_index, data = weather_data)
summary(x.lm)
```

```
##
## Call:
## lm(formula = value ~ time_index, data = weather_data)
##
## 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_index  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
```

The time index turns out to be significant to the regression which is what we expected.

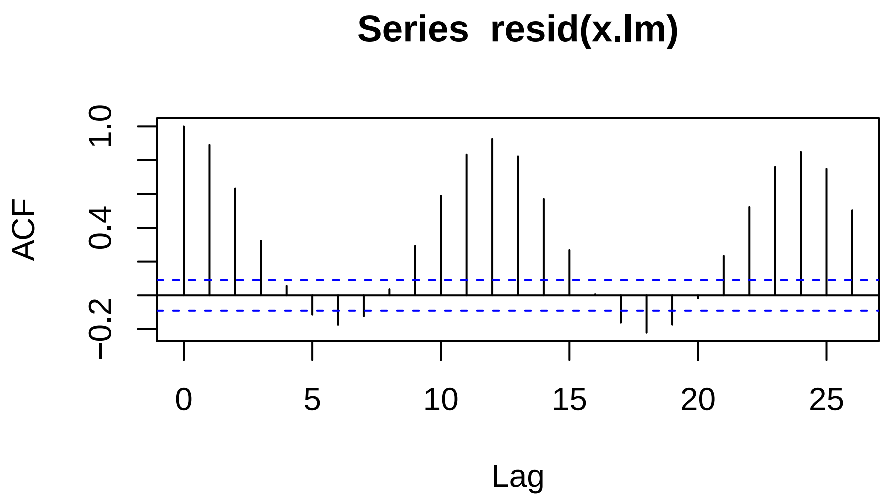
A plot of the fitted line on the original data

```
co2.df = data.frame(time = weather_data$index, val = weather_data$value,
    fitted.values = x.lm$fitted.values, residuals = x.lm$residuals)

ggplot(data = co2.df, aes(x = time, y = val)) + xlab("Date") +
    ylab("CO2 (ppm)") + geom_line(aes(y = val, col = "CO2 (ppm)")) +
    geom_line(aes(y = fitted.values, col = "Linear Trend Fit")) +
    scale_y_continuous(labels = function(x) format(x/1000, scientific = FALSE)) +
    theme(title = element_text(size = rel(1)), axis.text.y = element_text(angle = 45,
        hjust = 1)) + ggtitle("Fitted linear time trend line to original data")
```

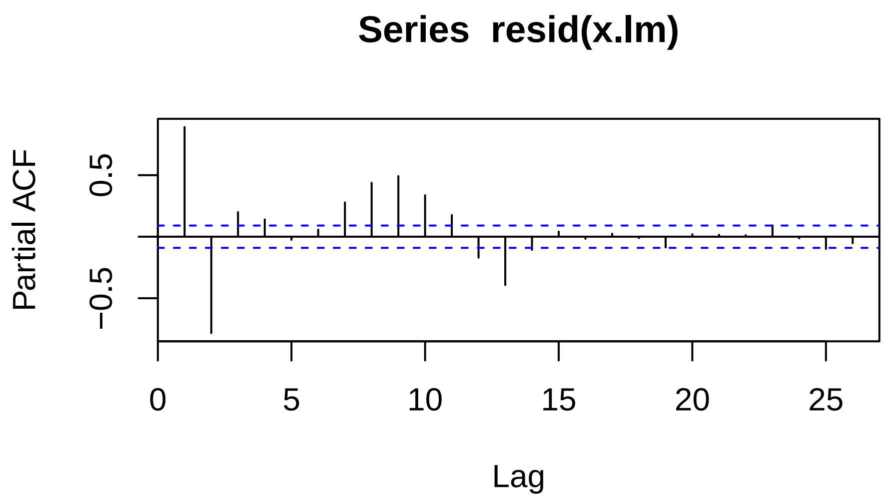


Examine the residuals of the linear fit

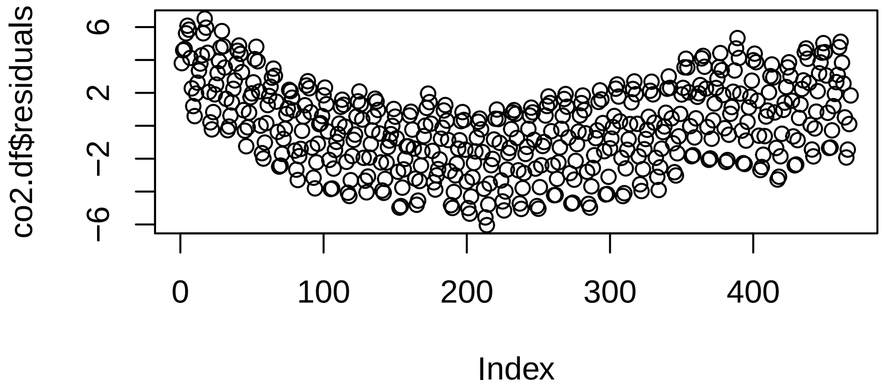
```
acf(resid(x.lm))
```




```
pacf(resid(x.lm))
```



```r
plot(co2.df$residuals)
```



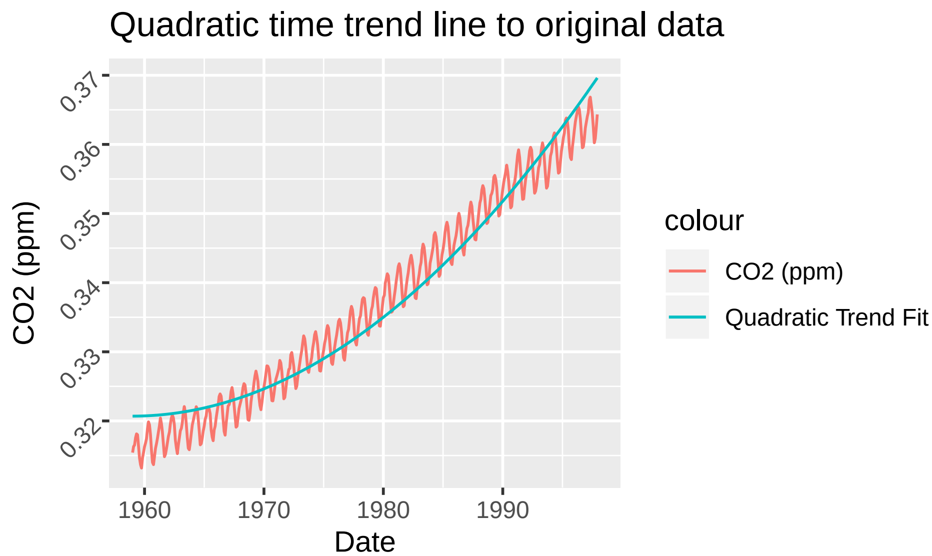
We can see this is highly seasonal, especially of the ACF. Clear swooping pattern in residuals

Fit a quardratic trend to the series

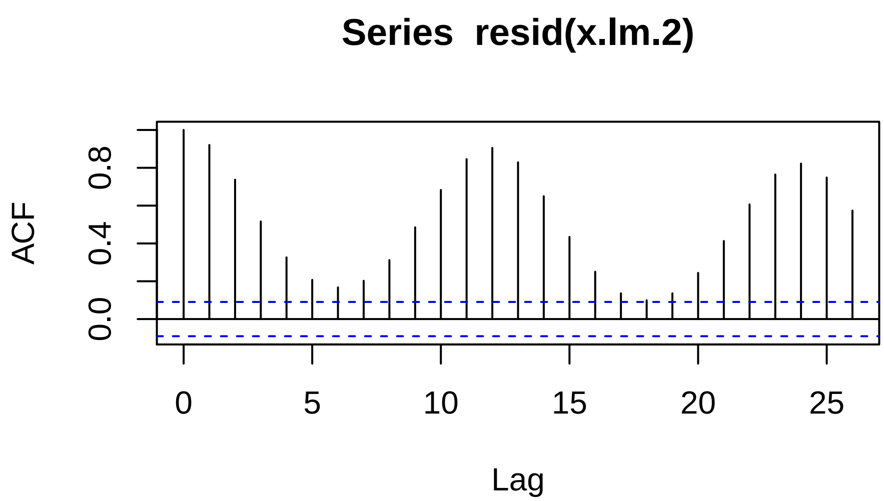
```r
x.lm.2 <- lm(value ~ I(time_index^2), data = weather_data)
summary(x.lm.2)
```

```
##
## Call:
## lm(formula = value ~ I(time_index^2), data = weather_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7523 -1.9818 -0.0023  2.4092  5.8833
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.207e+02  2.187e-01  1466.4   <2e-16 ***
## I(time_index^2) 2.234e-04  2.227e-06   100.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.152 on 466 degrees of freedom
## Multiple R-squared:  0.9557, Adjusted R-squared:  0.9556
## F-statistic: 1.006e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

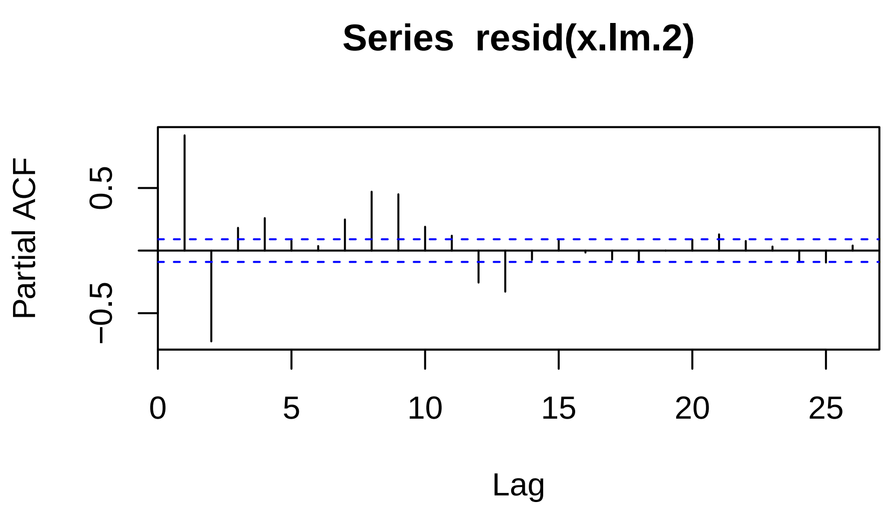
```r
co2.df.2 = data.frame(time = weather_data$index, val = weather_data$value,
    fitted.values = x.lm.2$fitted.values, residuals = x.lm.2$residuals)

ggplot(data = co2.df.2, aes(x = time, y = val)) + xlab("Date") +
    ylab("CO2 (ppm)") + geom_line(aes(y = val, col = "CO2 (ppm)")) +
    geom_line(aes(y = fitted.values, col = "Quadratic Trend Fit")) +
    scale_y_continuous(labels = function(x) format(x/1000, scientific = FALSE)) +
    theme(title = element_text(size = rel(1)), axis.text.y = element_text(angle = 45,
        hjust = 1)) + ggtitle("Quadratic time trend line to original data")
```



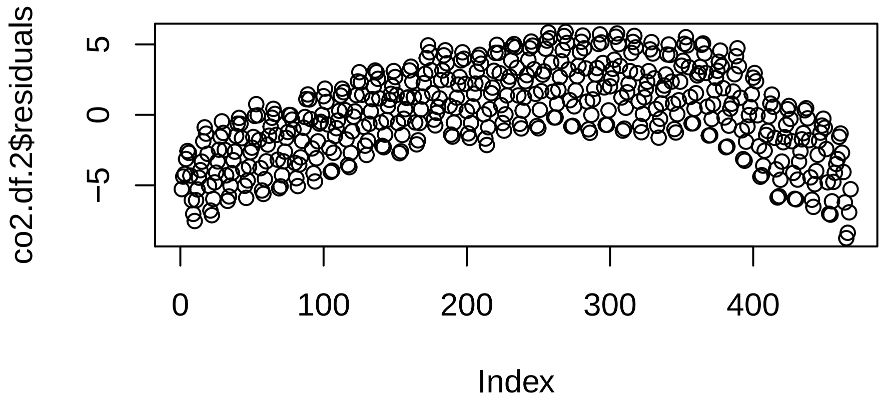

```
acf(resid(x.lm.2))
```




```
pacf(resid(x.lm.2))
```



```r
plot(co2.df.2$residuals)
```



$time^2$ is significant to the regression model, however, once again, clear pattern in residuals resulting from the seasonal trend.

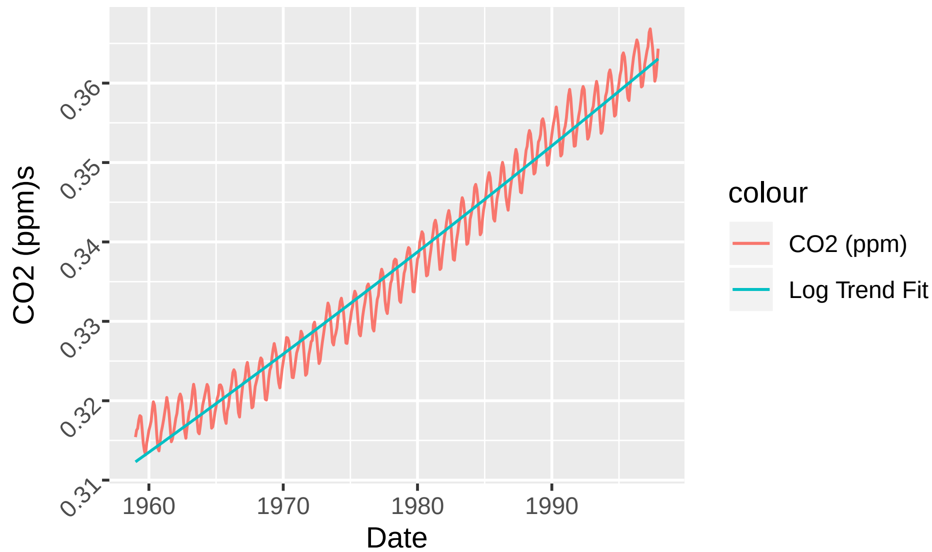
Fit a linear time trend model with a logarithmic transformation to the response variable (CO2 level)

```r
exp.x.lm <- lm(log(value) ~ time_index, data = weather_data)
summary(exp.x.lm)
```

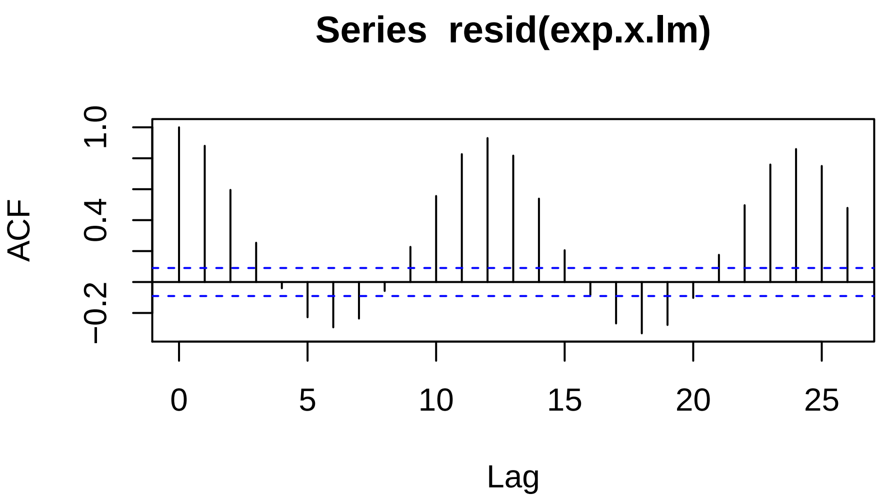
```
##
## Call:
## lm(formula = log(value) ~ time_index, data = weather_data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0172650 -0.0056145  0.0002764  0.0053760  0.0187770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.744e+00  6.829e-04  8410.5   <2e-16 ***
## time_index  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
```

```r
exp.co2.df = data.frame(time = weather_data$index, val = weather_data$value,
    fitted.values = exp(exp.x.lm$fitted.values), residuals = exp(exp.x.lm$residuals))


ggplot(data = exp.co2.df, aes(x = time, y = val)) + xlab("Date") +
    ylab("CO2 (ppm)s") + geom_line(aes(y = val, col = "CO2 (ppm)")) +
    geom_line(aes(y = fitted.values, col = "Log Trend Fit")) +
    scale_y_continuous(labels = function(x) format(x/1000, scientific = FALSE)) +
```

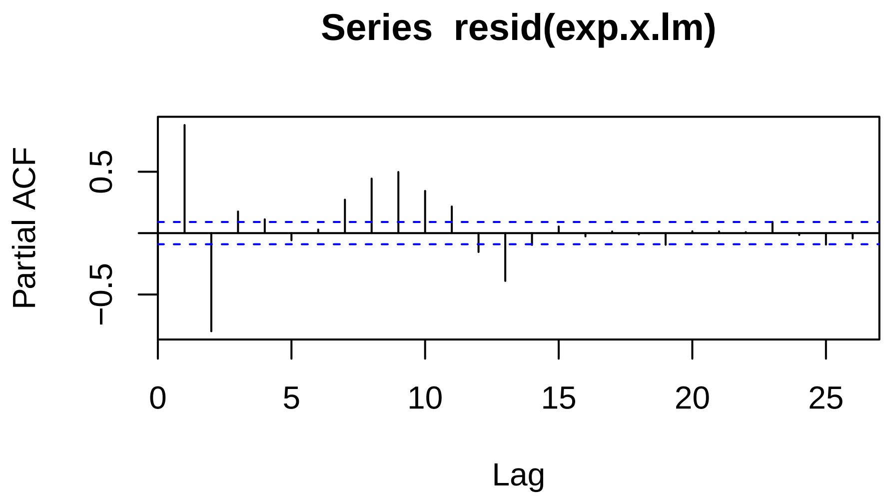
```r
    theme(title = element_text(size = rel(1)), axis.text.y = element_text(angle = 45,
        hjust = 1))
```



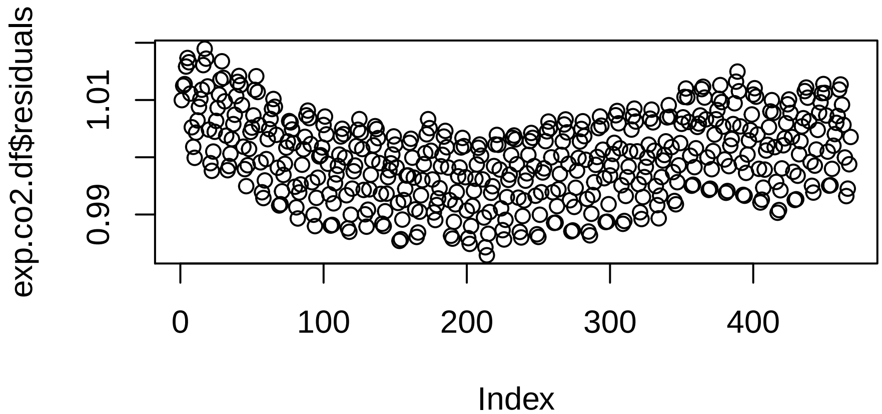

```r
acf(resid(exp.x.lm))
```




```r
pacf(resid(exp.x.lm))
```




```
plot(exp.co2.df$residuals)
```



Taking the log does not seem to change the residuals. However the heteroskedasticity situation seems to have improved over the previous approaches.

Fit a polynomial time trend model with seasonal dummy variables. For this we will use the TSLM functions. This will use the trend, season and the random components to fit a regression on the original data series.

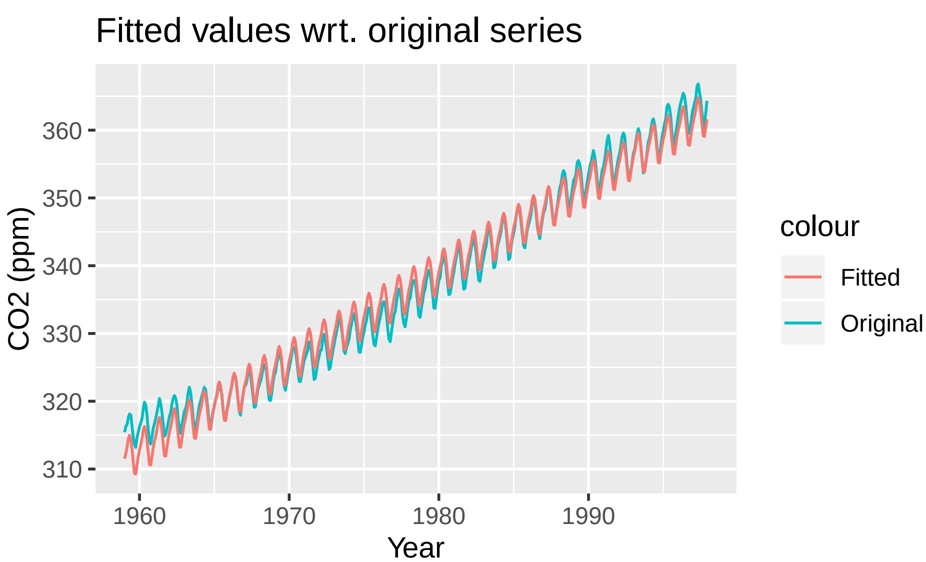
```
fit_weather <- weather_tsbl %>% model(TSLM(value ~ trend() +
    season()))
report(fit_weather)
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##     Min      1Q Median      3Q     Max
## -2.768 -1.284 -0.405  1.261  4.337
##
## Coefficients:
##                    Estimate Std. Error  t value Pr(>|t|)
## (Intercept)      311.42208    0.29171 1067.565  < 2e-16 ***
```

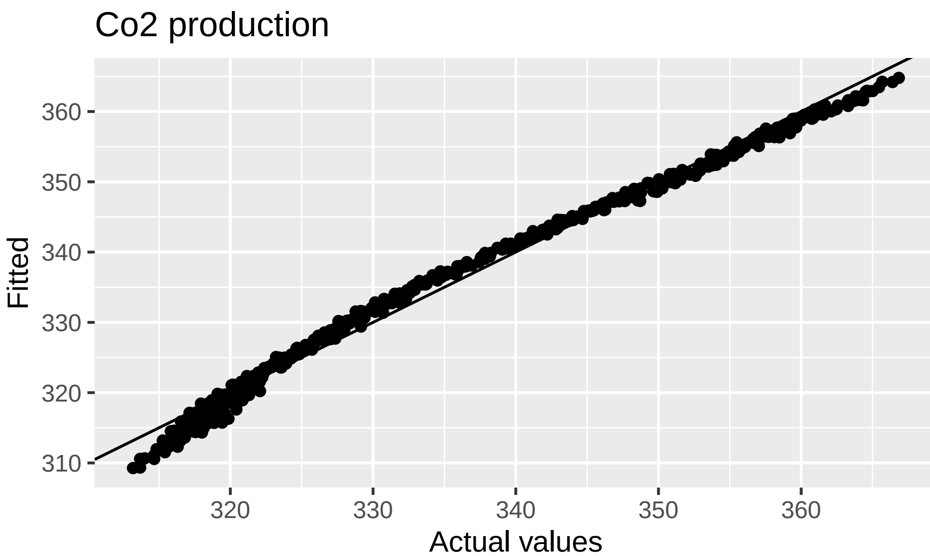
```
## trend()           0.10921    0.00056  195.003  < 2e-16 ***
## season()year2     0.66336    0.37054    1.790 0.074078 .
## season()year3     1.40543    0.37054    3.793 0.000169 ***
## season()year4     2.53597    0.37054    6.844 2.50e-11 ***
## season()year5     3.01445    0.37054    8.135 3.95e-15 ***
## season()year6     2.35139    0.37055    6.346 5.36e-10 ***
## season()year7     0.83039    0.37055    2.241 0.025510 *
## season()year8    -1.23728    0.37056   -3.339 0.000910 ***
## season()year9    -3.06161    0.37056   -8.262 1.58e-15 ***
## season()year10   -3.24441    0.37057   -8.755  < 2e-16 ***
## season()year11   -2.05490    0.37058   -5.545 4.99e-08 ***
## season()year12   -0.93744    0.37059   -2.530 0.011755 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.636 on 455 degrees of freedom
## Multiple R-squared: 0.9884,  Adjusted R-squared: 0.988
## F-statistic:  3218 on 12 and 455 DF, p-value: < 2.22e-16
```

The trend component seems to be significant and positively influences the series. This is no surprise since this a a trending series as is evident from the previous EDA. This shows that 1 unit increase in trend component leads to a 0.1 unit increase in the CO2 levels.

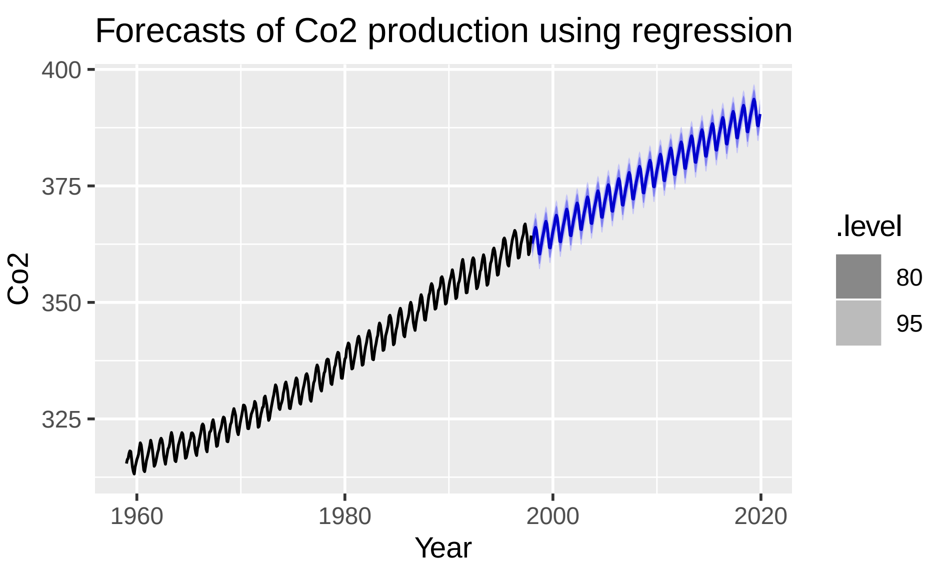
```r
augment(fit_weather) %>% ggplot(aes(x = index)) + geom_line(aes(y = value,
    colour = "Original")) + geom_line(aes(y = .fitted, colour = "Fitted")) +
    labs(x = "Year", y = "CO2 (ppm)", title = "Fitted values wrt. original series")
```




```r
augment(fit_weather) %>% ggplot(aes(x = value, y = .fitted)) +
    geom_point() + ylab("Fitted") + xlab("Actual values") + ggtitle("Co2 production") +
    scale_colour_brewer(palette = "Dark2", name = "Quarter") +
    geom_abline(intercept = 0, slope = 1)
```




```
fc_weather <- forecast(fit_weather, h = "22 years")

fc_weather %>% autoplot(weather_tsbl) + ggtitle("Forecasts of Co2 production using regression")
    xlab("Year") + ylab("Co2")
```



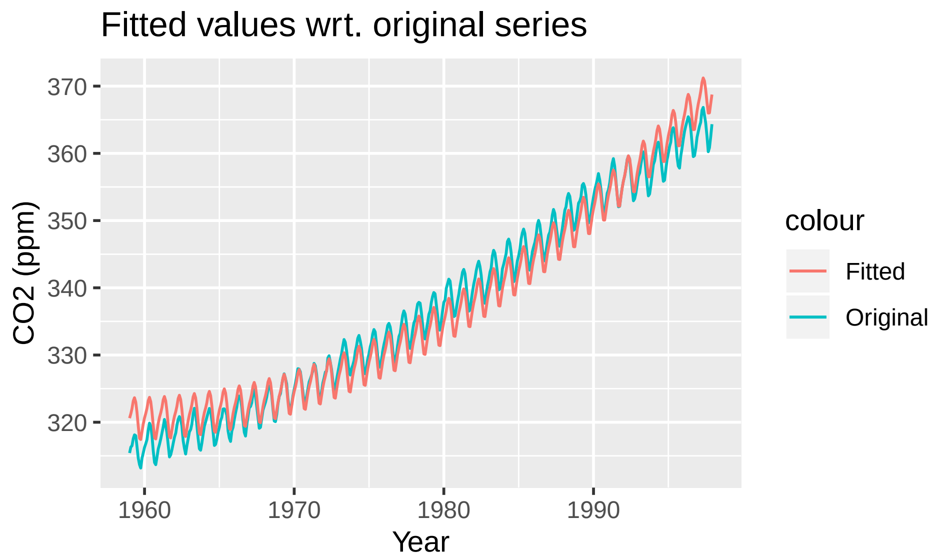
The fit of the prior model is good, but does not follow the non-linear trend of the model, especially in the 80's and may not capture the growing trend into the 2000s. Using the TSLM model, we attempt to capture the non-linear growth with a quadratic factor for the trend and seasonal component. Results below indicate a slightly better adusted R-squared, although it has a harder time in some years - resulting in perhaps a better overall fit but missing precision in certian specific years. From a practical standpoint, we don't care about accuracy in specific years - we want to best model the overall trend of co2 ppm in the atmosphere.

```
fit_weather <- weather_tsbl %>% model(TSLM(value ~ I(trend()^2) +
    I(season()^2)))
report(fit_weather)
```

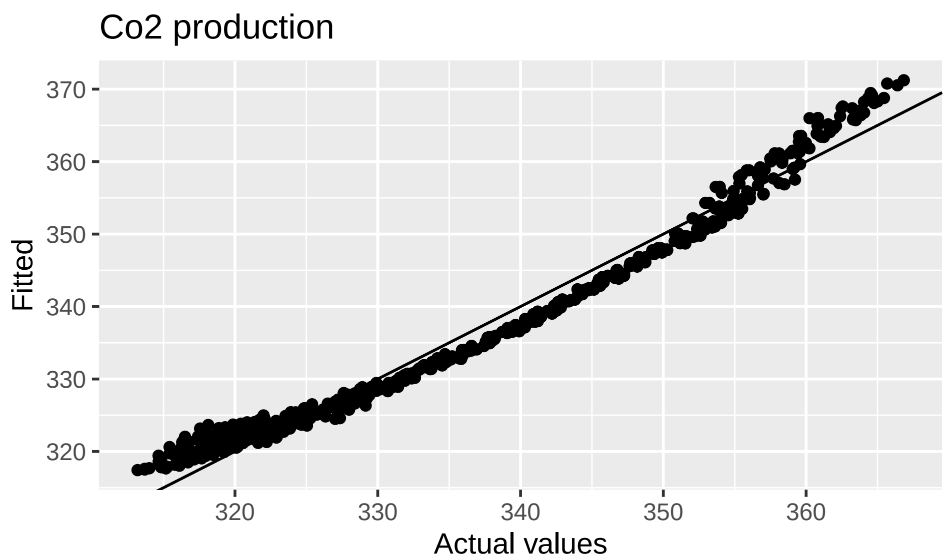
```
## Series: value
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7514 -2.2333  0.6857  2.0903  3.3276
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.206e+02  4.063e-01 789.029  < 2e-16 ***
## I(trend()^2)        2.239e-04  1.712e-06 130.774  < 2e-16 ***
## I(season()^2)year2   6.698e-01  5.486e-01   1.221 0.222717
## I(season()^2)year3   1.418e+00  5.486e-01   2.585 0.010057 *
## I(season()^2)year4   2.554e+00  5.486e-01   4.656 4.24e-06 ***
## I(season()^2)year5   3.038e+00  5.486e-01   5.537 5.21e-08 ***
## I(season()^2)year6   2.379e+00  5.486e-01   4.337 1.78e-05 ***
## I(season()^2)year7   8.624e-01  5.486e-01   1.572 0.116646
## I(season()^2)year8  -1.202e+00  5.486e-01  -2.190 0.029019 *
## I(season()^2)year9  -3.023e+00  5.486e-01  -5.509 6.03e-08 ***
## I(season()^2)year10 -3.202e+00  5.486e-01  -5.837 1.01e-08 ***
## I(season()^2)year11 -2.011e+00  5.486e-01  -3.665 0.000277 ***
## I(season()^2)year12 -8.911e-01  5.486e-01  -1.624 0.105024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.422 on 455 degrees of freedom
## Multiple R-squared: 0.9745,   Adjusted R-squared: 0.9738
## F-statistic:  1448 on 12 and 455 DF, p-value: < 2.22e-16
```

Below we plot the fitted results to the time series linear regression model and find that the opposite regions of the graph are estimated incorrectly. In the last model, we under estimated early and late years in the data. In this model we over estimate early and late perhaps, but capture the non-linear nature of the plot. You could consider the prior model a conservative model and this one more agressive. Understanding both could be valuable.

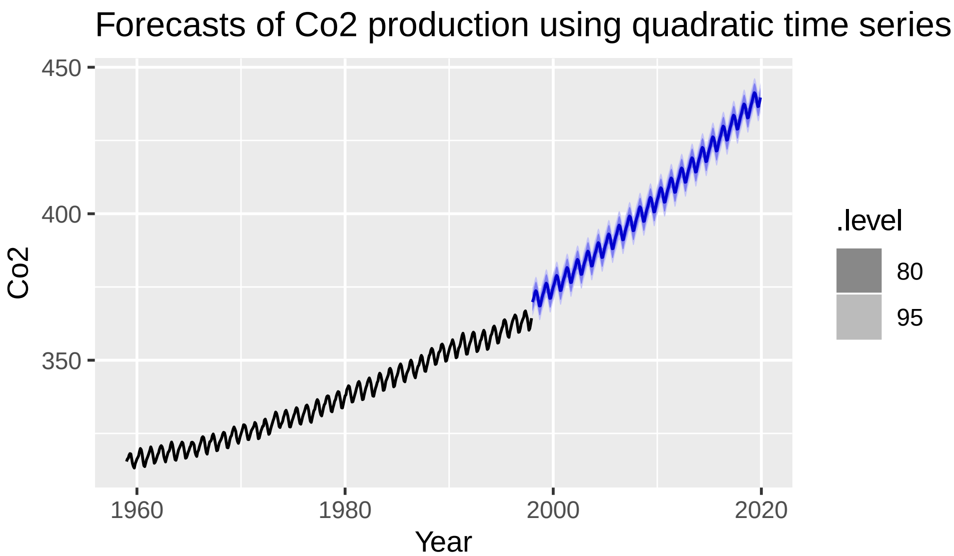
```r
augment(fit_weather) %>% ggplot(aes(x = index)) + geom_line(aes(y = value,
    colour = "Original")) + geom_line(aes(y = .fitted, colour = "Fitted")) +
    labs(x = "Year", y = "CO2 (ppm)", title = "Fitted values wrt. original series")
```




```
augment(fit_weather) %>% ggplot(aes(x = value, y = .fitted)) +
    geom_point() + ylab("Fitted") + xlab("Actual values") + ggtitle("Co2 production") +
    scale_colour_brewer(palette = "Dark2", name = "Quarter") +
    geom_abline(intercept = 0, slope = 1)
```




```
fc_weather <- forecast(fit_weather, h = "22 years")

fc_weather %>% autoplot(weather_tsbl) + ggtitle("Forecasts of Co2 production using quadratic t
    xlab("Year") + ylab("Co2")
```



Part 3 (3 points): Choose ARIMA model for 'co2' series

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. Write your model (or models) using backshift notation. Use your model (or models) to generate forecasts to the year 2020.

ARIMA model for 'co2' series

The model has a strong annual seasonality indicated in the series residuals at month 1, 13, etc.

We will use a 12-month backshift operator for the series to adjust for seasonality

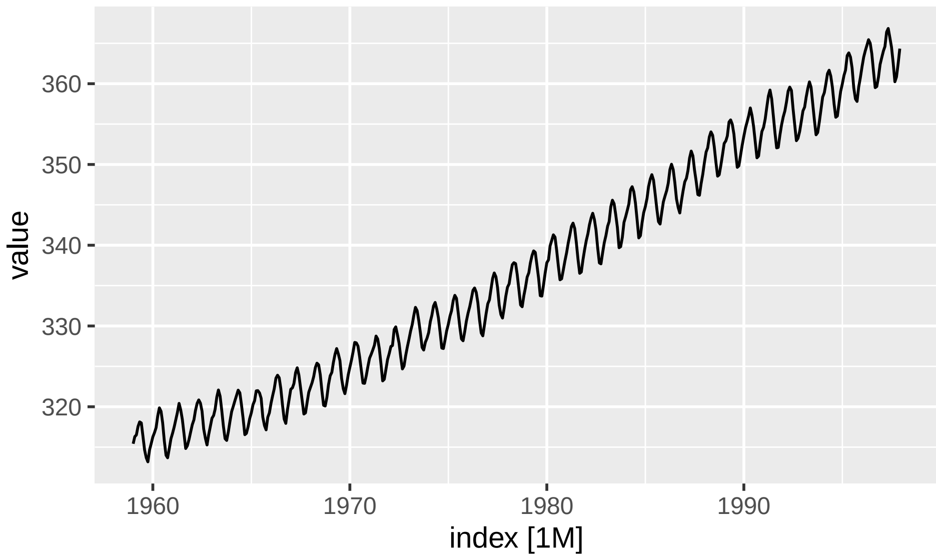
$$B_{y_t}^{12} = y_{t-12}$$

Model Procedure Steps

Plot the data

In the raw form, as discussed above, the general trend is not stationary. We will adjust the trend with first difference.

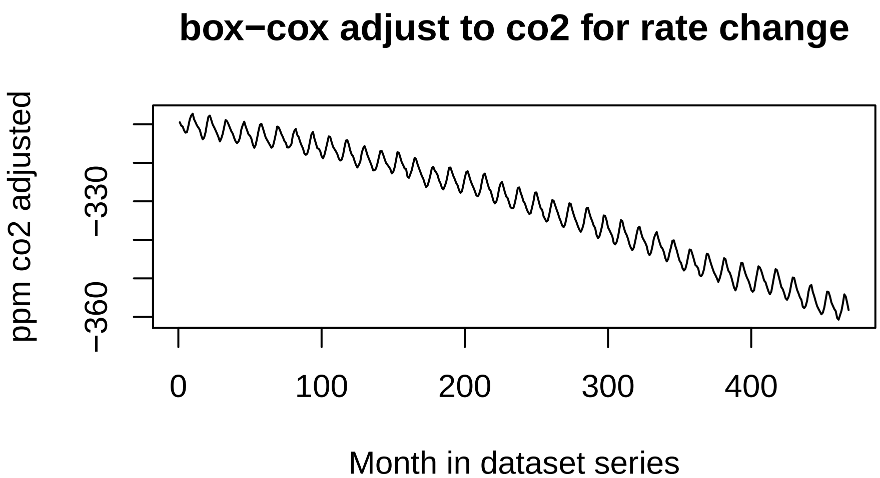
```
fc_weather_dcmp <- weather_tsbl %>% model(STL(value ~ season(window = "periodic"))) %>%
    components() %>% select(-.model) %>% as_tsibble()
fc_weather_dcmp %>% autoplot(value)
```



Transform the data

There is evidence of changing rate, especially in more recent years. Evaluate a box-cox transformation. Knowing that we'll be fitting an ARIMA model, we should be cautious about any transformation given that the ARIMA model will work best on untransformed data. However, the trend will provide insight into the rate of change and how that may be handled in the ARIMA.

```
optim_lambda = BoxCox.lambda(weather_tsbl$value)
no_boxcox = BoxCox(weather_tsbl$value, lambda = optim_lambda)
delta = no_boxcox - weather_tsbl$value
plot(ts(delta), xlab = "Month in dataset series", ylab = "ppm co2 adjusted",
    main = "box-cox adjust to co2 for rate change")
```



The boxcox optimization, with optimized lambda, would adjust the ppm value down about 50ppm over the sample period to flatten the growth. This indicates that there is indeed a non-linear trend to the data, growth is increasing as years progress through the data as we would expect. We will allow the ARIMA model to fit the data rather than update the values based on the box-cox.

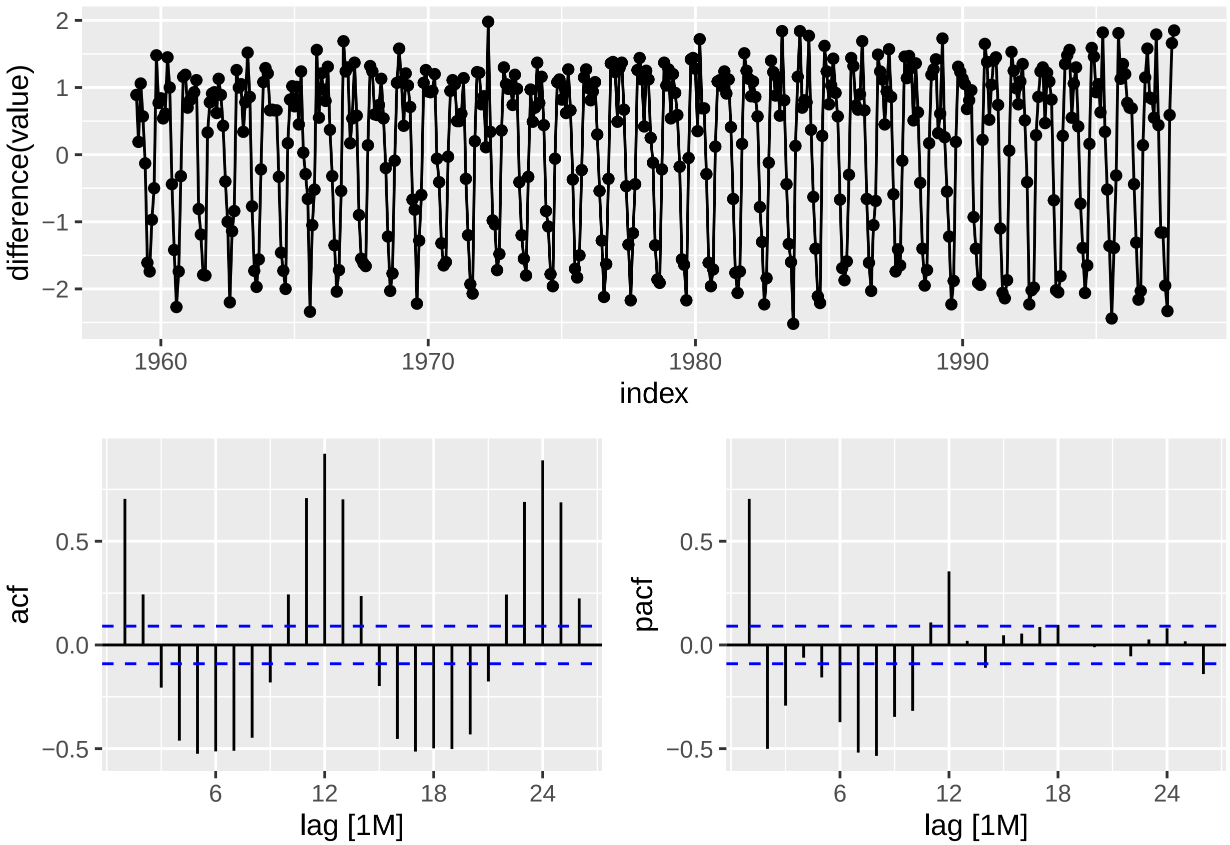
If the data is non-stationary, take the first difference(s) until stationary

Check to see if the first difference is sufficient to create stationarity. There appears to be strong stationarity here, but let's plot the trend to be sure. The residuals vary slightly more in the 1980s, than in the 1960s, but remain constant through the 1980-2000 timeframe. The lag in the AFC continues an alternating pattern with equal intensity and the pacf drops off at lag 12.

```
weather_tsbl %>% gg_tsdisplay(difference(value), plot_type = "partial")
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Examine ACF/PACF to see if ARIMA(p,d,0) or ARIMA(0,d,q) is sufficient

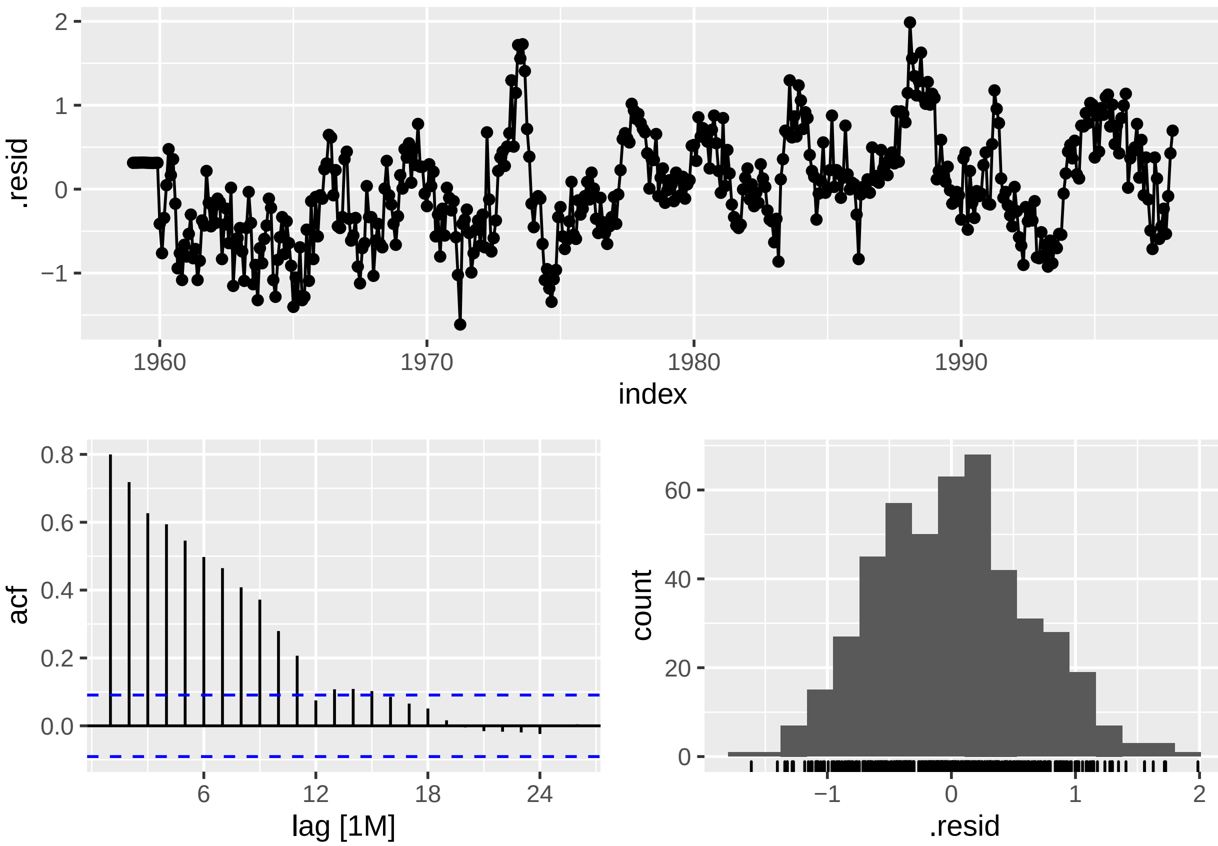
Start with a very simple model that takes the first seasonal difference, but no other parameters. We are using this estimation as a simple starting point.

```
fit <- weather_tsbl %>% model(arima = ARIMA(value ~ pdq(0, 0,
    0) + PDQ(0, 1, 0)))
report(fit)
```

```
## Series: value
## Model: ARIMA(0,0,0)(0,1,0)[12] w/ drift
##
```

```
## Coefficients:
##       constant
##         1.2629
## s.e.    0.0291
##
## sigma^2 estimated as 0.3908:  log likelihood=-430.77
## AIC=865.54   AICc=865.56   BIC=873.78
```

```
fit %>% gg_tsresiduals()
```



If we just look at the MA seasonal term, the ACF slowly decays over 12 periods to zero. There does not appear to be any polynomial form to the model, so we will start with a p=1 and d=1 for both seasonal and q=2. There is quite a bit of negative residual in the early years from 1960-1975, then the residuals become stationary.

Try a better model and test with AICs

A better model appears to be AR(1) with MA(1) and the same for seasonal. To capture the accelerated growth, we'll include a quadratic parameter, again in both the base and seasonal component of the ARIMA model. We can express this model with the 6 non-zero parameters below.

$$(1-\phi_1 B)(1-\Phi_1 B^1)(1-B)(1-B^1)_{y_t} = (1+\phi_2 B)(1+\Theta_2 B^2)\epsilon_t$$

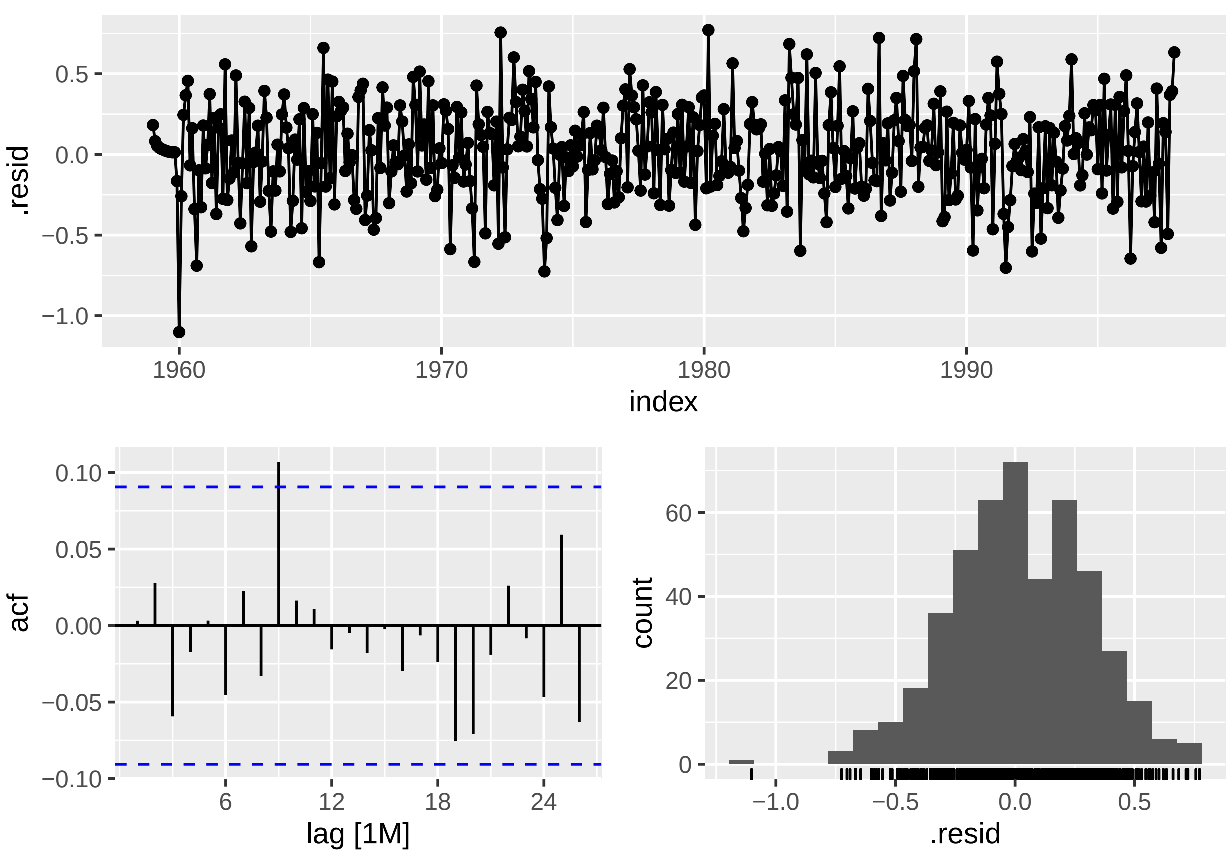
```r
fit <- weather_tsbl %>% model(arima = ARIMA(value ~ pdq(1, 1,
    2) + PDQ(1, 1, 2)))
```

check residuals by plotting and comparing against white noise

```r
report(fit)
```

```
## Series: value
## Model: ARIMA(1,1,2)(1,1,2)[12]
##
## Coefficients:
##          ar1      ma1     ma2     sar1     sma1     sma2
##       0.5959  -0.9284  0.1413  -0.5055  -0.3059  -0.4784
## s.e.  0.2402   0.2445  0.1101   0.4995   0.4856   0.4117
##
## sigma^2 estimated as 0.08565:  log likelihood=-83.66
## AIC=181.31   AICc=181.56   BIC=210.15
```

```r
fit %>% gg_tsresiduals()
```



Test to see if the fit behaves like white noise and it does. The large p-value (0.29) suggests that although we do see noise in the ACF, that is white noise. The acf plot looks like white noise and the residuals are normally distributed with an a notch at 1 which is a little suspect. Let's see if we can do better after a review of the ljung-box test.

The null hypothesis for the ljung-box is that the model is a good fit. The p-value is 0.28, larger than the critical value, so there is no evidence to reject the null hypothesis. That said, the diagnostic plot indicates that we havn't found an optimal solution - the distribution of residuals isn't uniform and the p-value, which large, isn't big enough to lay down the pen.

```
# set the seasonal lag to 12
augment(fit) %>% features(.resid, ljung_box, lag = 12, dof = 4)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 arima     9.69     0.288
```

Using a search algorithm, walk through the reasonable potential p,d,q and P,D,Q values. We're walk over values that we've already seen in prior tests above, but keep track of the AICc for each and report back the best. We'll also keep a data frame of every result in case we need to evaluate diagnostic data - this will take some time to run.

Note that we have to use tryCatch to keep the search running. Not all parameters will successfully converge. Although we want to track which parameters do not converge, we don't want to babysit the loop - so we'll just report those back as a NA AICc.

Grab a soda and lets this run.

```
best_aic = data.frame(aic = 1000, status = "I", p = 0, d = 0,
    q = 0, P = 0, D = 0, Q = 0)
result_list = NULL
for (p in seq(0:2)) {
    for (d in seq(0:2)) {
        for (q in seq(0:1)) {
            for (P in seq(0:2)) {
                for (D in seq(0:2)) {
                  for (Q in seq(0:1)) {

                    fit <- weather_tsbl %>% model(arima = ARIMA(value ~
                      pdq(p, d, q) + PDQ(P, D, Q)))

                    aicc = tryCatch({
                      fit$arima[[1]]$fit[[3]]$AICc
                    }, error = function(cond) {
                      return(NA)
                    }, warn = function(cond) {
                      return(NA)
                    })

                    result = data.frame(aic = aicc, p = p, d = d,
                      q = q, P = P, D = D, Q = Q)
                    result_list = rbind(result_list, result)
                    if (!is.na(aicc)) {
```

```
                      if (aicc < best_aic["aic"]) {
                        best_aic = result
                      }
                    }

                  }
                }
            }
        }
    }
}

print("Best AIC combination:")
```

```
## [1] "Best AIC combination:"
```

```
print(best_aic)
```

```
##        aic p d q P D Q
## 1 172.9589 1 1 1 2 1 2
```

It is also helpful to see the variation of the parameters and AICs associated with each. NA indicates that an error or warning resulted in no AICs available for the model - typically because the model could not converge.

Sort the AIC in ascending order and look at the few smallest values

```
# The list of AIC values tesed is large, and to save the
# reader, we will not exaust the list

# df <-result_list[order(result_list$aic),] head(df)
```

Use the p, d, q, P, D, Q values for the best AIC to train the $ARIMA(pdq(1,1,1), PDQ(2,1,2))_{12}$ model

The best AICc identifies a formula defined below:

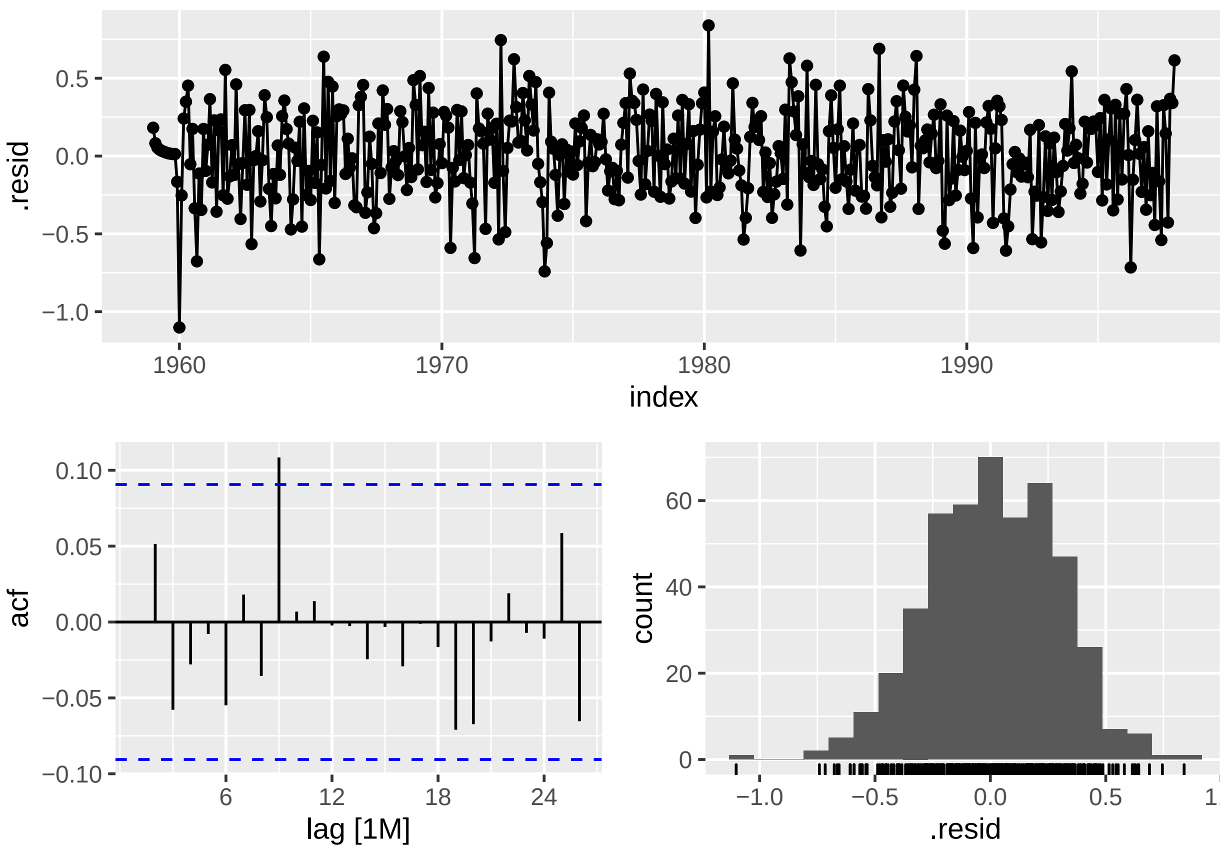
$$(1 - \phi_1 B)(1 - \Phi_2 B^{12})(1 - B)(1 - B^{12})_{y_t} = (1 + \phi_1 B)(1 + \Theta_2 B^{12})\epsilon_t$$

```
# Best AICc at 172.9589

fit <- weather_tsbl %>% model(arima = ARIMA(value ~ pdq(1, 1,
    1) + PDQ(2, 1, 2)))

# validate that we found a reasonable module visually
report(fit)
```

```
## Series: value
## 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
```

```r
fit %>% gg_tsresiduals()
```



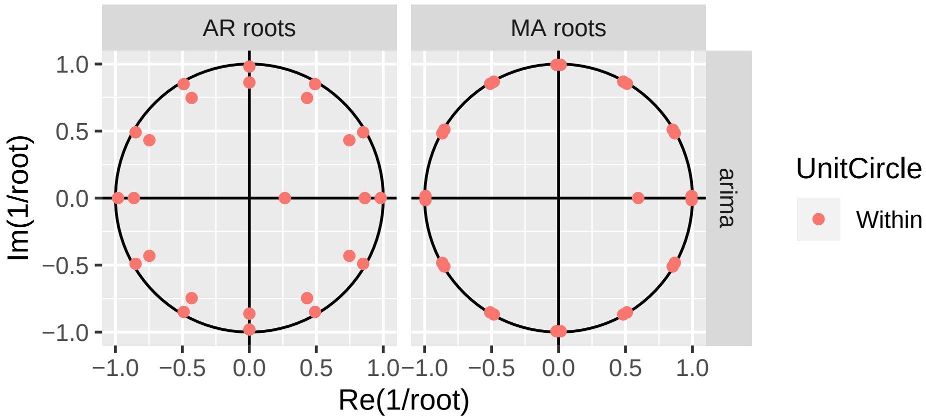
The diagnostics on this model look much better. We've archived nice uniform residuals for the entire model duration, the residuals are uniform, and the acf looks like white noise. Verify with a Ljung-box. The p-value in this case is well above the critical value, it is large and so we can conclude that a new model isn't required. Without the diagnostic plots one might assume that the prior model was better given the p-value, but the distribution is much better with this model and AICc.

```r
augment(fit) %>% features(.resid, ljung_box, lag = 12, dof = 4)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 arima     11.2     0.192
```

Ensure that the AR and MA roots are inside the unit circle (the model converged, we believe this to be true)

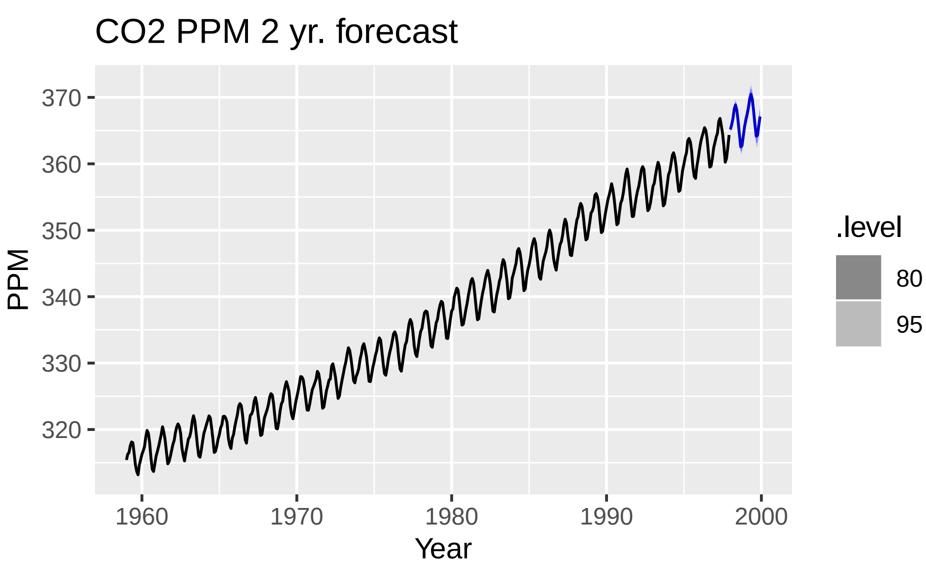
```r
gg_arma(fit)
```



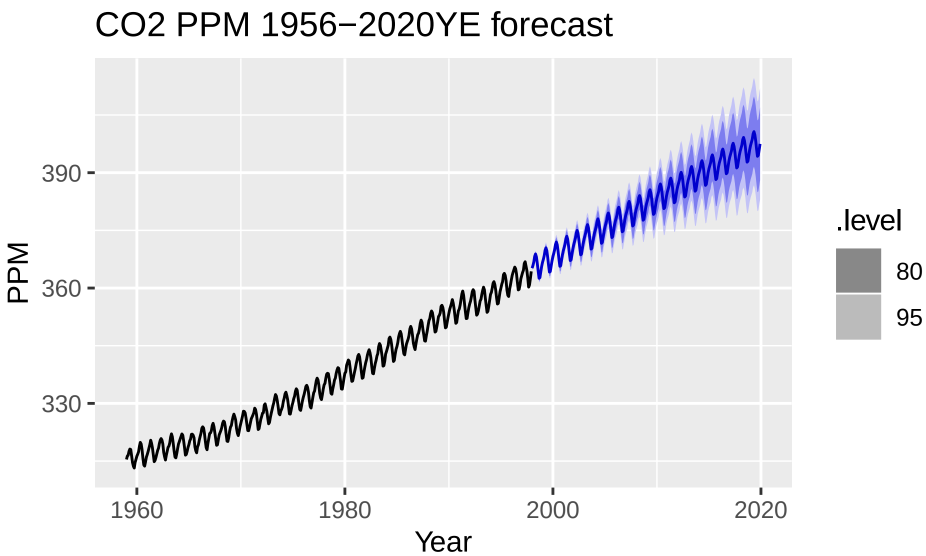
Forecast

Let's start with a very simple forecast to make sure the two-year forecast appears reasonable

```r
# simple forecast out 24 months
fit %>% forecast() %>% autoplot(weather_tsbl) + xlab("Year") +
    ylab("PPM") + ggtitle("CO2 PPM 2 yr. forecast")
```




```r
# now let's forecast out 22 years
fit %>% forecast(h = 264, level = c(95)) %>% autoplot(weather_tsbl) +
    xlab("Year") + ylab("PPM") + ggtitle("CO2 PPM 1956-2020YE forecast")
```




```
fc = fit %>% forecast(h = 264, level = c(95))

# save this data out for part 4
hilo_res = hilo(fc, level = 95)
df_part3 = data.frame(index = hilo_res$index, value = hilo_res$value)
ts_values_p3 = c(weather_tsbl$value, df_part3$value)
ts_values_p3 = ts(ts_values_p3, start = 1959, frequency = 12)

tail(hilo_res$`95%`, 12)
```

```
## [383.4377, 410.7327]95
## [384.1866, 411.6438]95
## [384.8657, 412.4816]95
## [386.2164, 413.9896]95
## [386.7207, 414.6501]95
## [385.8225, 413.9072]95
## [384.1705, 412.4095]95
## [381.9192, 410.3118]95
## [379.9775, 408.5229]95
## [380.1752, 408.8725]95
## [381.5928, 410.4412]95
## [382.9995, 411.9983]95
```

The 2020 forecast appears to fit the model well, although it tends to become highly linear as compared to a slight acceleration observable between 1980 and 1990. The confidence interval does capture the acceleration with a 95% confidence that the co2 level is between 383 and 412 ppm in 2020.

In order to do in-sample fit test, we need to build a training and test segment of the plot. The future prediction will be from 1998 to 2020, and our frequency is months. That is 264 periods.

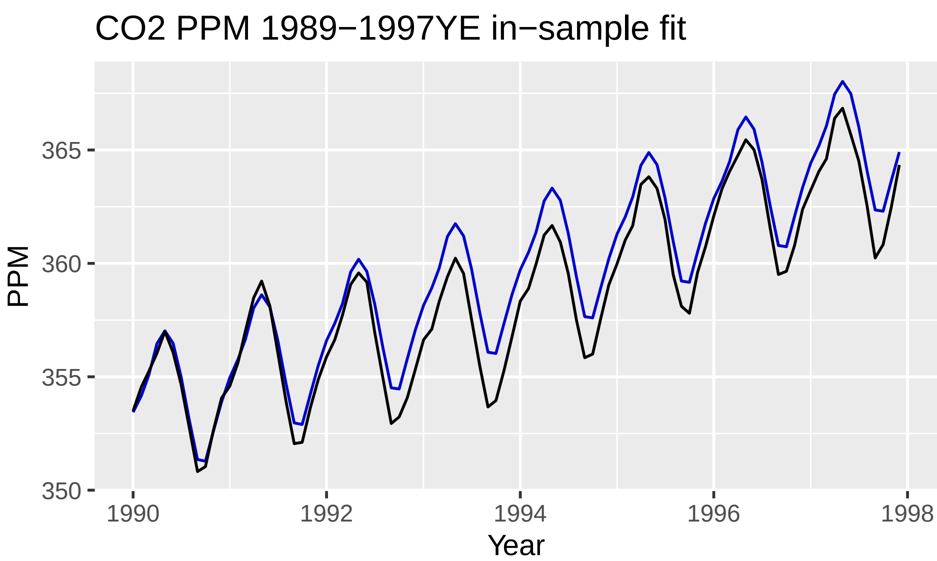
There are 38 years, or 456 months in the data. If we set the training set to equal size of the predicted range, it would be 60% of the plot, which is too large, and it may not capture the recent periods. We'll take 20% of the time period for the in-sample data, which is fractional, so we'll use

8 years, 96 periods.

```
# split the data
tsbl_test = weather_tsbl %>% filter(year(index) >= 1990)
tsbl_train = weather_tsbl %>% filter(year(index) < 1990)

# validate that we have Jan 1959 to Dec 1988 in the train
# data
summary(tsbl_train)
```

```
##       index              value
##  Min.   :1959 Jan   Min.   :313.2
##  1st Qu.:1966 Sep   1st Qu.:321.7
##  Median :1974 Jun   Median :329.9
##  Mean   :1974 Jun   Mean   :331.5
##  3rd Qu.:1982 Mar   3rd Qu.:340.9
##  Max.   :1989 Dec   Max.   :355.5
```

```
# fit the model to the training data (1956-1987)
fit <- tsbl_train %>% model(arima = ARIMA(value ~ pdq(1, 1, 1) +
    PDQ(2, 1, 2)))

# plot the test data from 1990 thru 1997 (8 years)
fit_training = fit %>% forecast(h = 96)
fit_training %>% autoplot(tsbl_test, level = NULL) + xlab("Year") +
    ylab("PPM") + ggtitle("CO2 PPM 1989-1997YE in-sample fit")
```



The model, using pre 1990 data fits the observed values very well. There tends to be slight slight error in the exceedence of observation early on, but we don't see that growing and it was appropriatly representing the long-term growth in the 2020 model above.

Part 4: Mauna Loa Data from 1974 to 2020

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). 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 current atmospheric CO2 levels to those predicted by your forecasts in Parts 2 and 3. Use the weekly data to generate a month-average series from 1997 to the present, and compare the overall forecasting performance of your models from Parts 2 and 3 over the entire period.

Load and transform data

Load data and glimpse

```
# the data file has a header, so we will read from the row in
# the file when the data starts
co2 <- read.csv("co2_weekly_mlo.txt", header = FALSE, sep = "",
    skip = 49, stringsAsFactors = FALSE)
names(co2) <- c("yr", "mon", "day", "decimal", "ppm", "days",
    "1yr", "10yr", "since1800")
# glimpse(co2)
paste("Number of observations = ", nrow(co2))
```

```
## [1] "Number of observations =  2388"
```

```
paste("Number of columns = ", ncol(co2))
```

```
## [1] "Number of columns =  9"
```

```
summary(co2)
```

```
##        yr            mon              day           decimal
##  Min.   :1974   Min.   : 1.000   Min.   : 1.00   Min.   :1974
##  1st Qu.:1985   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.:1986
##  Median :1997   Median : 7.000   Median :16.00   Median :1997
##  Mean   :1997   Mean   : 6.539   Mean   :15.71   Mean   :1997
##  3rd Qu.:2008   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:2009
##  Max.   :2020   Max.   :12.000   Max.   :31.00   Max.   :2020
##       ppm              days            1yr               10yr
##  Min.   :-1000.0   Min.   :0.000   Min.   :-1000.0   Min.   : -999.99
##  1st Qu.:  346.4   1st Qu.:5.000   1st Qu.:  344.9   1st Qu.:  330.66
##  Median :  363.9   Median :6.000   Median :  361.8   Median :  348.87
##  Mean   :  355.4   Mean   :5.858   Mean   :  326.3   Mean   :   49.23
##  3rd Qu.:  385.9   3rd Qu.:7.000   3rd Qu.:  384.1   3rd Qu.:  366.55
##  Max.   :  415.4   Max.   :7.000   Max.   :  412.7   Max.   :  390.67
##    since1800
##  Min.   : -999.99
##  1st Qu.:   66.38
```

```
##  Median :   83.28
##  Mean   :   77.76
##  3rd Qu.:  106.20
##  Max.   :  133.61
```

```r
head(co2)
```

```
##     yr mon day  decimal    ppm days     1yr    10yr since1800
## 1 1974   5  19 1974.380 333.34    6 -999.99 -999.99     50.36
## 2 1974   5  26 1974.399 332.95    6 -999.99 -999.99     50.06
## 3 1974   6   2 1974.418 332.32    5 -999.99 -999.99     49.57
## 4 1974   6   9 1974.437 332.18    7 -999.99 -999.99     49.63
## 5 1974   6  16 1974.456 332.37    7 -999.99 -999.99     50.07
## 6 1974   6  23 1974.475 331.59    6 -999.99 -999.99     49.60
```

```r
tail(co2)
```

```
##        yr mon day  decimal    ppm days    1yr   10yr since1800
## 2383 2020   1  12 2020.031 412.82    6 410.66 388.41    132.51
## 2384 2020   1  19 2020.051 413.65    7 412.19 388.27    133.17
## 2385 2020   1  26 2020.070 414.09    7 411.06 389.37    133.47
## 2386 2020   2   2 2020.089 414.33    7 411.11 390.67    133.61
## 2387 2020   2   9 2020.108 414.40    6 412.70 390.32    133.58
## 2388 2020   2  16 2020.127 414.01    7 411.22 390.45    133.10
```

Observations: There are no NA values in the dataset. The date column will need to be generates using the day, month and year data. There are several values for the PPM metric that are set to -999 which seems like missing values. These values will need to be either imputed or removed. The data is reported weekly starting from 19/5/1974 till 16/2/2020.

Convert to timeseries

```r
# create the date column, use the ymd from the lubridate
# package to convert to date data type
co2_withdate <- mutate(co2, date = ymd(paste(yr, mon, day, sep = "_")))
summary(co2_withdate)
```

```
##        yr            mon              day           decimal
##  Min.   :1974   Min.   : 1.000   Min.   : 1.00   Min.   :1974
##  1st Qu.:1985   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.:1986
##  Median :1997   Median : 7.000   Median :16.00   Median :1997
##  Mean   :1997   Mean   : 6.539   Mean   :15.71   Mean   :1997
##  3rd Qu.:2008   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:2009
##  Max.   :2020   Max.   :12.000   Max.   :31.00   Max.   :2020
##       ppm              days            1yr              10yr
##  Min.   :-1000.0   Min.   :0.000   Min.   :-1000.0   Min.   : -999.99
##  1st Qu.:  346.4   1st Qu.:5.000   1st Qu.:  344.9   1st Qu.:  330.66
##  Median :  363.9   Median :6.000   Median :  361.8   Median :  348.87
##  Mean   :  355.4   Mean   :5.858   Mean   :  326.3   Mean   :   49.23
##  3rd Qu.:  385.9   3rd Qu.:7.000   3rd Qu.:  384.1   3rd Qu.:  366.55
##  Max.   :  415.4   Max.   :7.000   Max.   :  412.7   Max.   :  390.67
```

```
##    since1800              date
##  Min.   : -999.99   Min.   :1974-05-19
##  1st Qu.:   66.38   1st Qu.:1985-10-25
##  Median :   83.28   Median :1997-04-02
##  Mean   :   77.76   Mean   :1997-04-02
##  3rd Qu.:  106.20   3rd Qu.:2008-09-08
##  Max.   :  133.61   Max.   :2020-02-16
```

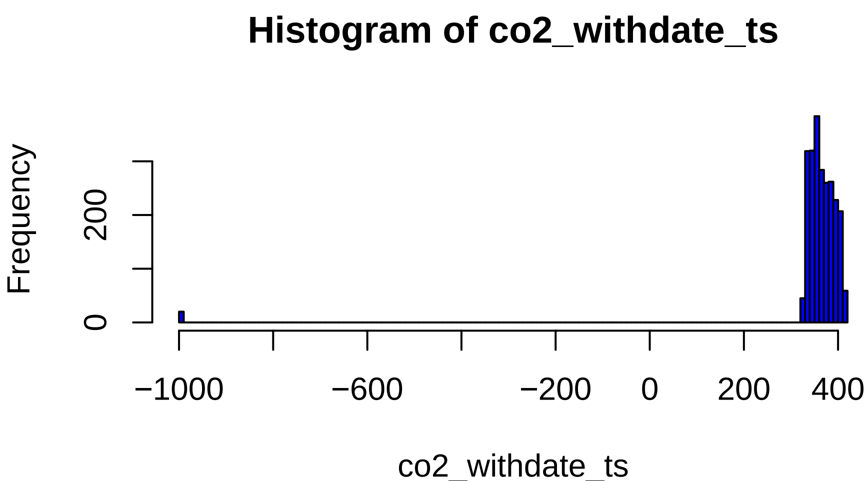
```
co2_withdate_ts <- ts(co2_withdate$ppm, frequency = 365.25/7)
```

EDA for the weekly CO2 data

Treat missing observations

Histogram of the PPM variable

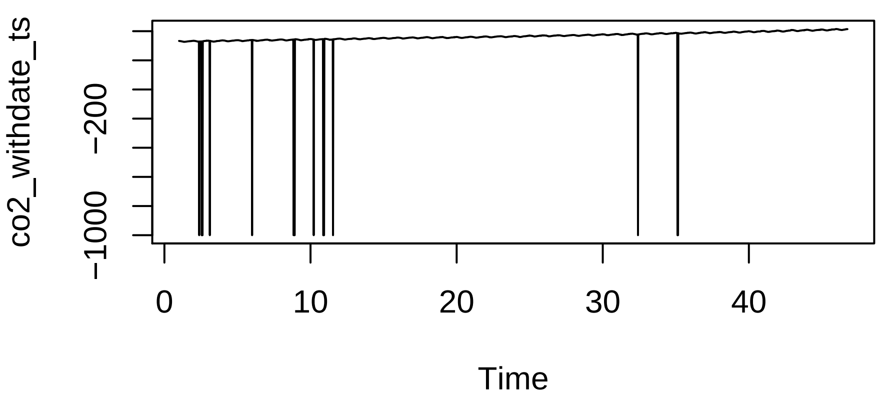
```
hist(co2_withdate_ts, breaks = 200, col = "blue")
```



The histogram provides an idea about the number of missing observations, however it does not tell us where these values are placed in the duration over time, i.e. in the time series. So we will look at a time series plot of the data to understand if these values are clustered around a specific duration of time.

Plot the series (with missing values)

```
ts.plot(co2_withdate_ts)
```



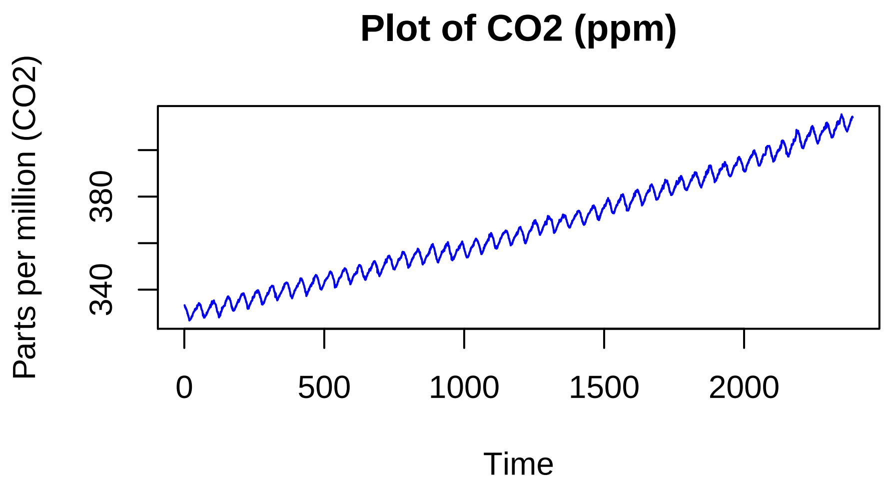
The missing values seem to be distributed all over the time duration and a look at the chart tells us that it would be safe to impute the values.

Impute missing values

```
# install.packages('imputeTS')
library(imputeTS)
```

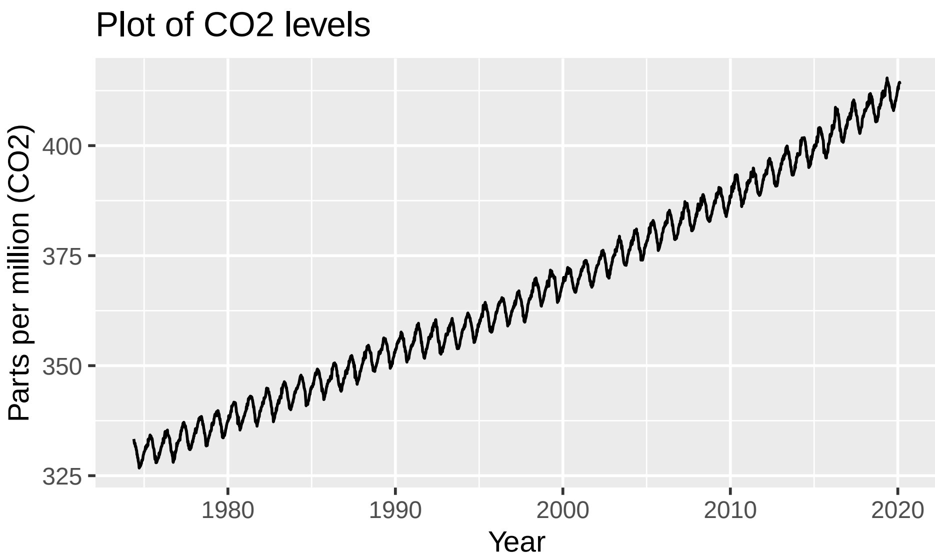
```
##
## Attaching package: 'imputeTS'

## The following object is masked from 'package:zoo':
##
##     na.locf
```

```
# replace -999 with NA
co2_withdate_mut <- co2_withdate %>% mutate(ppm = sub(-999, NA,
    ppm))
# replace NA with an approximation method
co2_withdate_mut$ppm <- na.approx(co2_withdate_mut$ppm)
ts.plot(co2_withdate_mut$ppm, ylab = "Parts per million (CO2)",
    col = "blue")
title("Plot of CO2 (ppm)")
```



Use the tsibble object so that the plot shows the x-axis formatted with the year

```
library(fpp2)
co2_tsibble <- as_tsibble(co2_withdate_mut, index = date)
co2_tsibble %>% autoplot(ppm) + ylab("Parts per million (CO2)") +
    xlab("Year") + ggtitle("Plot of CO2 levels")
```



The above plot is the real observations for the Mauna Loa dataset. We can see that actual values in May were approx 414 ppm, which is slightly less than the May peak 2019 estimate at 95% of 415 ppm using the quadratic model that in part 3. All things considered, it is astounding how well the model predicted the maximum ppm of the 2020 values.

Convert the mutated and imputed series to time series

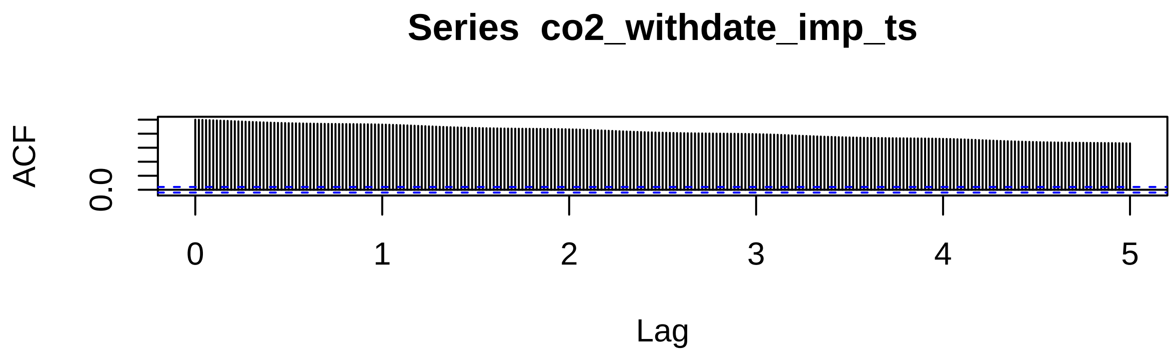
Examine the data in the time series object

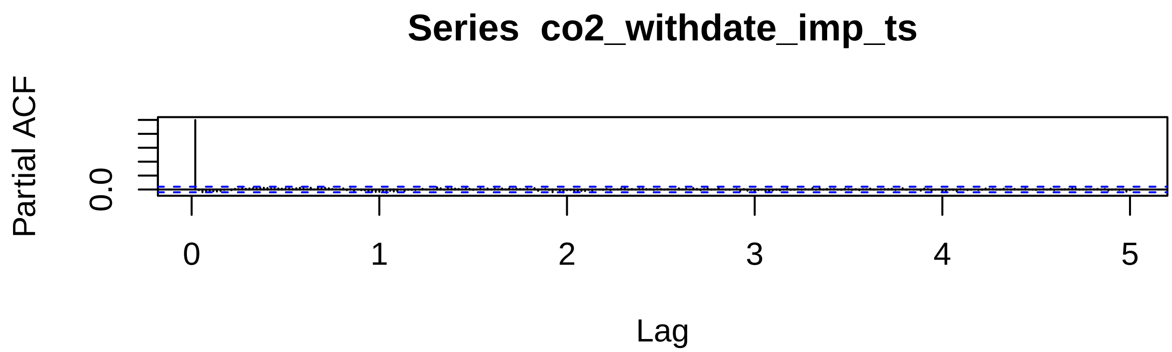
```
co2_withdate_imp_ts <- ts(co2_withdate_mut$ppm, frequency = 52)
str(co2_withdate_imp_ts)
```

```
##  Time-Series [1:2388] from 1 to 46.9: 333 333 332 332 332 ...
```

Examine the ACF and PACF charts

```
par(mfrow = c(2, 1))
acf(co2_withdate_imp_ts, lag.max = 260)
pacf(co2_withdate_imp_ts, lag.max = 260)
```





The ACF decays down very slowly and the lags all the way back to 5 years ago also shows high correlations. The PACF drops down to zero after the 1st lag. This shows that this may be an AR(1) process. However since the series has a very strong trend this will have to be de-trended first.

Perform the unit root tests to check if the series is stationary

```
library(tseries)
```

```
##
## Attaching package: 'tseries'

## The following object is masked from 'package:imputeTS':
##
##     na.remove
```

```
adf.test(co2_withdate_imp_ts)
```

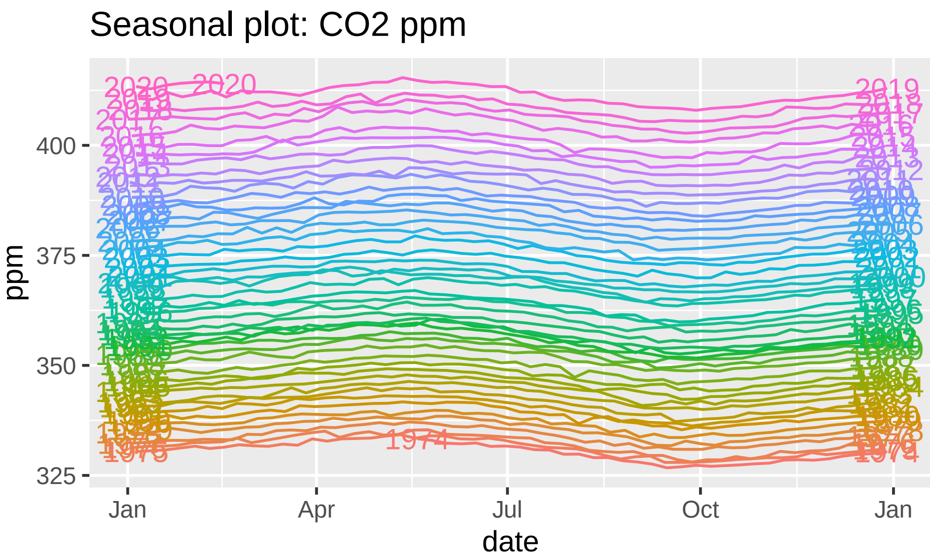
```
## Warning in adf.test(co2_withdate_imp_ts): p-value smaller than printed p-
## value

##
##  Augmented Dickey-Fuller Test
##
## data:  co2_withdate_imp_ts
## Dickey-Fuller = -7.4309, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

We reject the null hypothesis that the series has a unit root. Therefore the series is stationary.

Seasonal effects

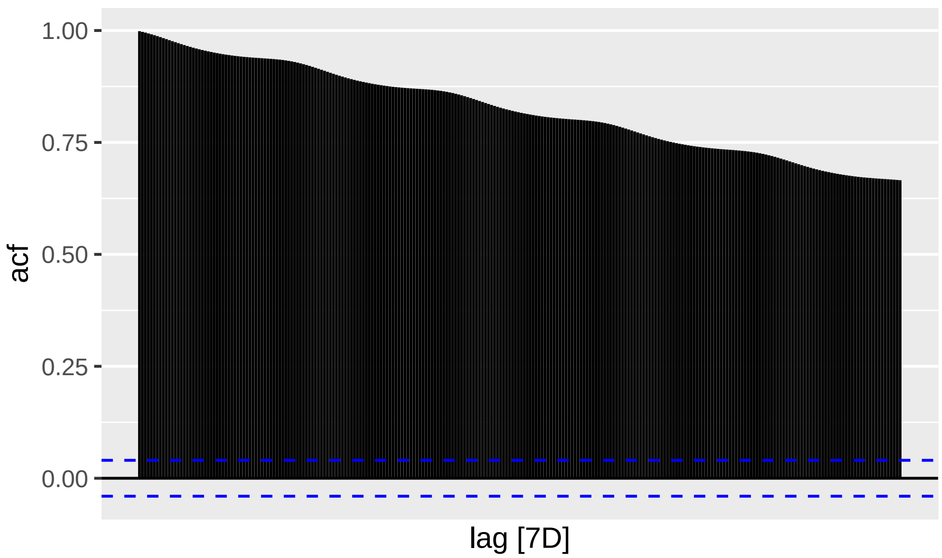
Looking for seasonal effects, we understand that this is weekly data, so we take a first seasonal difference and plot the relevant charts

```
co2_tsibble %>% gg_season(ppm, labels = "both") + ylab("ppm") +
    ggtitle("Seasonal plot: CO2 ppm")
```

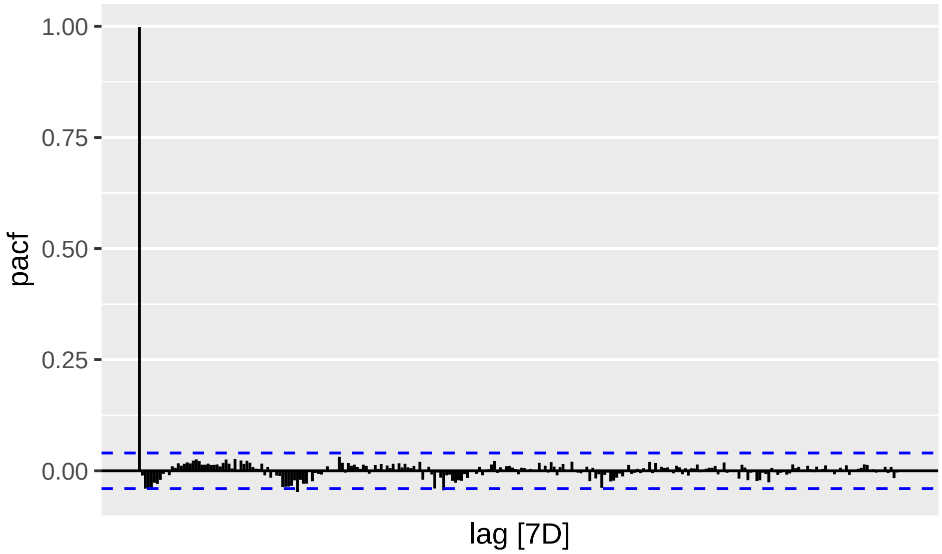


From the seasonal plot, it is clear that there is a gradual rise in the CO2 levels between February and May with the levels reaching the peak in May and then gradually falling to a low level in October. Again this is been the same pattern every single year starting from 1974. It is also clear that the levels have been rising with each year.

```
par(mfrow = c(2, 1))
co2_tsibble %>% ACF(ppm, lag_max = 256) %>% autoplot()
```

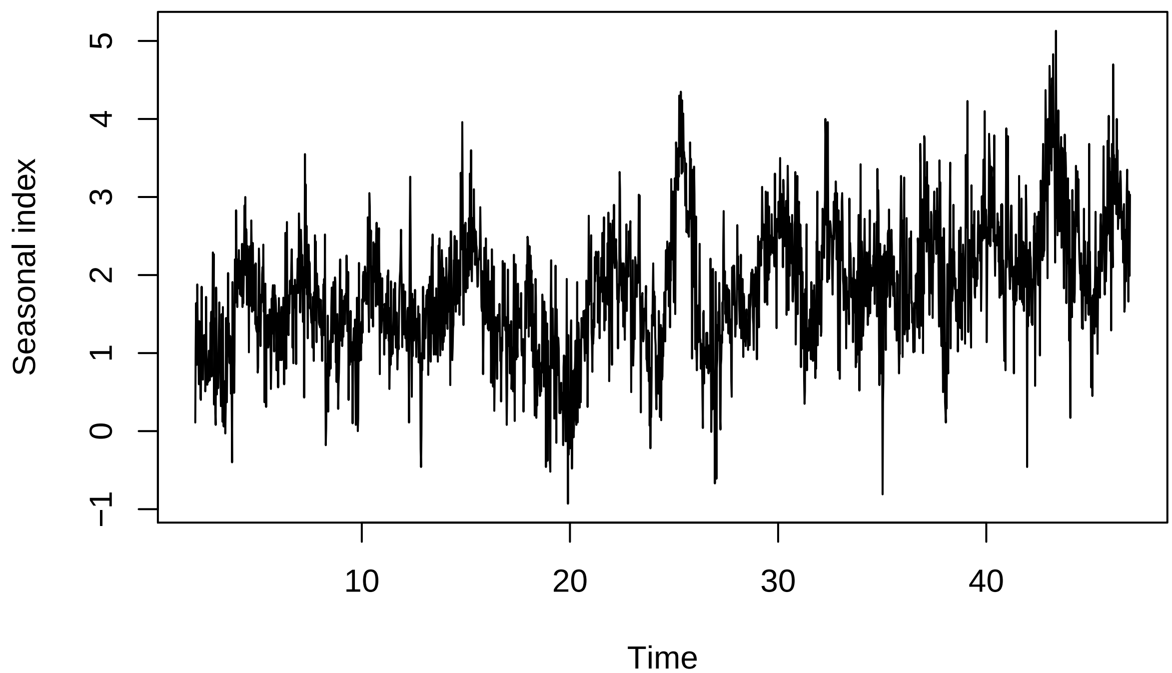



```
co2_tsibble %>% PACF(ppm, lag_max = 256) %>% autoplot()
```

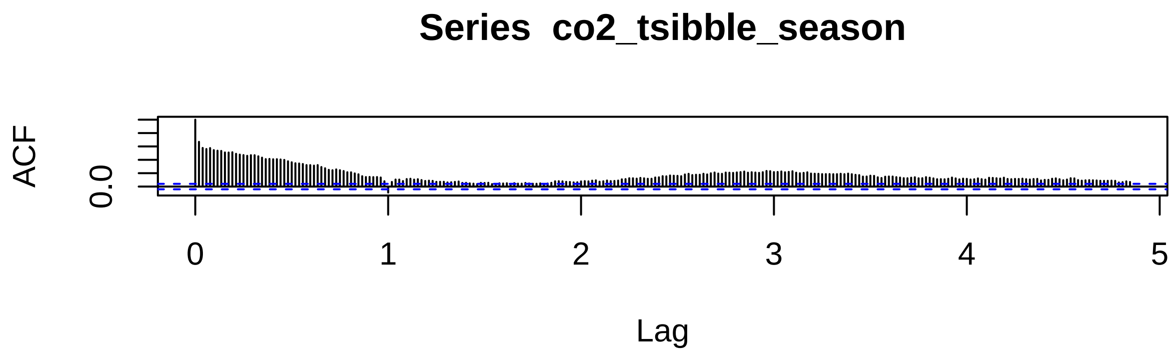


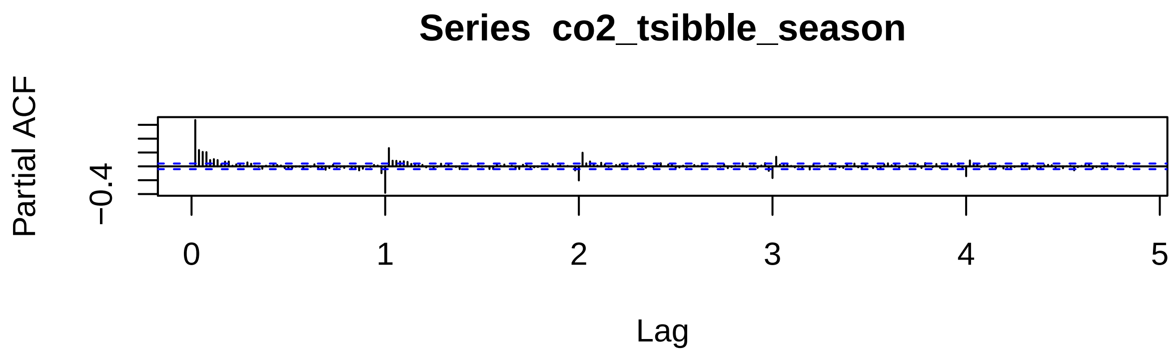
The ACF plot shows both trend and seasonality. Since this is a trending series, there is a slow decrease in autocorrelations.

```
# take the first seasonal difference, since this is weekly
# data, 52 lags in the past we have the same week
co2_tsibble_season <- diff(co2_withdate_imp_ts, lag = 52)
plot(co2_tsibble_season, type = "l", ylab = "Seasonal index")
```




```
par(mfrow = c(2, 1))
acf(co2_tsibble_season, lag.max = 252)
pacf(co2_tsibble_season, lag.max = 252)
```





The ACF of the sesonal chart shows strong oscillating patterns in the data which indicates the presence of strong seasonal effects.

Part 5: Seasonally Adjust NOAA data

Seasonally adjust the weekly NOAA data, and split both seasonally-adjusted (SA) and non-seasonally-adjusted (NSA) series into training and test sets, using the last two years of observations as the test sets. For both SA and NSA series, fit ARIMA models using all appropriate steps. Measure and discuss how your models perform in-sample and (psuedo-) out-of-sample, comparing candidate models and explaining your choice. In addition, fit a polynomial time-trend model to the seasonally-adjusted series and compare its performance to that of your ARIMA model.

Seasonal adjustments

Split into training and test sets

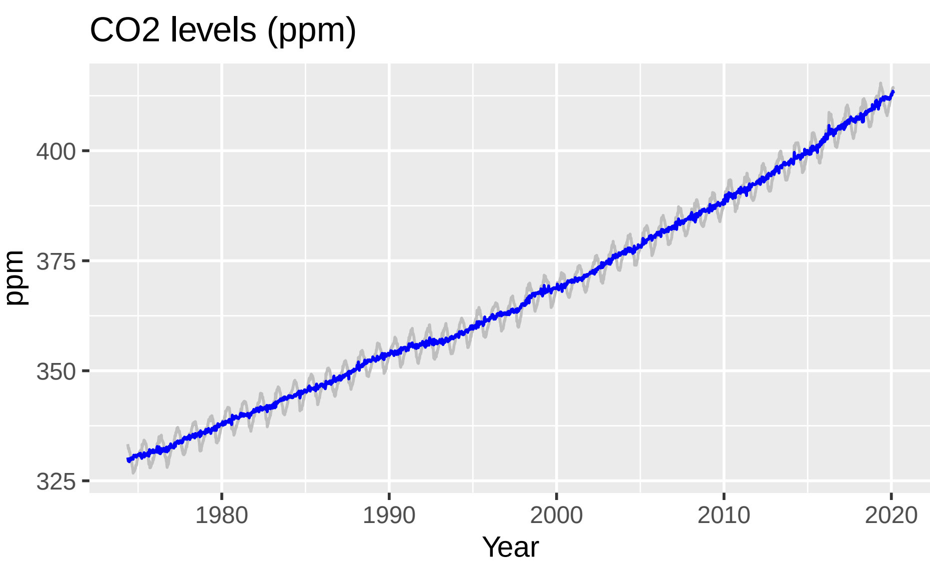
For seasonal adjustments, the seasonal component needs to be removed from the original data.

```
dcmp <- co2_tsibble %>% model(STL(ppm))
components(dcmp)
```

```
## # A dable:            2,388 x 7 [7D]
## # Key:                .model [1]
## # STL Decomposition: ppm = trend + season_year + remainder
```

```
##    .model   date          ppm trend season_year remainder season_adjust
##    <chr>    <date>      <dbl> <dbl>       <dbl>     <dbl>         <dbl>
##  1 STL(ppm) 1974-05-19  333.  330.        3.05   0.399           330.
##  2 STL(ppm) 1974-05-26  333.  330.        3.04  -0.000272        330.
##  3 STL(ppm) 1974-06-02  332.  330.        2.85  -0.466           329.
##  4 STL(ppm) 1974-06-09  332.  330.        2.60  -0.381           330.
##  5 STL(ppm) 1974-06-16  332.  330.        2.45  -0.0573          330.
##  6 STL(ppm) 1974-06-23  332.  330.        2.28  -0.689           329.
##  7 STL(ppm) 1974-06-30  332.  330.        1.93  -0.274           330.
##  8 STL(ppm) 1974-07-07  331.  330.        1.43  -0.0389          330.
##  9 STL(ppm) 1974-07-14  331.  330.        1.06  -0.282           330.
## 10 STL(ppm) 1974-07-21  331.  330.        0.593  0.0744          330.
## # ... with 2,378 more rows
```

```
co2_tsibble %>% autoplot(ppm, color = "gray") + autolayer(components(dcmp),
    season_adjust, color = "blue") + xlab("Year") + ylab("ppm") +
    ggtitle("CO2 levels (ppm)")
```

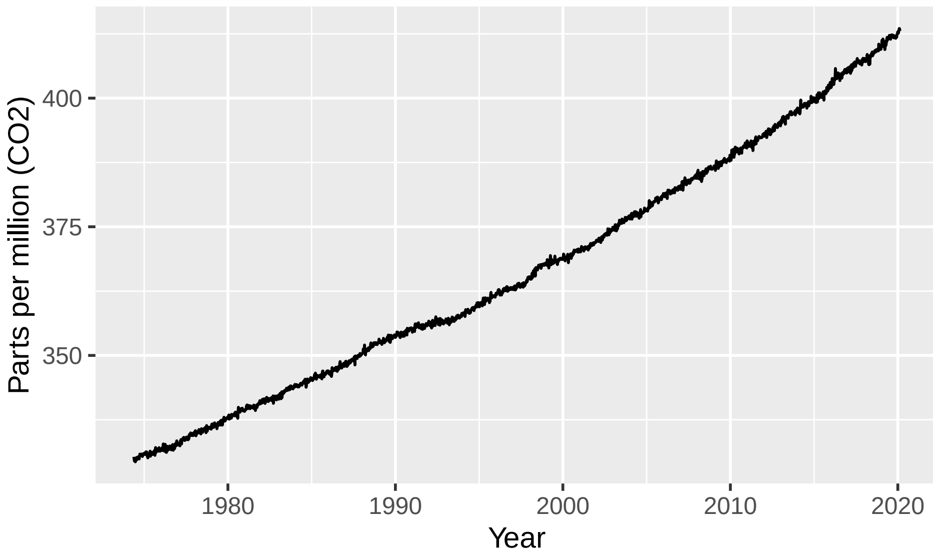


Look at the seasonally adjusted data

```
comps <- components(dcmp)
co2_tsibble_ajd <- as_tsibble(comps %>% dplyr::select(season_adjust),
    index = date)
```

```
## Selecting index: "date"
```

```
co2_tsibble_ajd %>% autoplot(season_adjust) + ylab("Parts per million (CO2)") +
    xlab("Year")
```



Split the datasets

```
season_adjust_test <- co2_tsibble_ajd %>% filter(year(date) >=
    2019)
season_adjust_train <- co2_tsibble_ajd %>% filter(year(date) <
    2019)

nsa_adjust_test <- co2_tsibble %>% filter(year(date) >= 2019)
nsa_adjust_train <- co2_tsibble %>% filter(year(date) < 2019)

paste("Number of samples in train (SA) = ", nrow(season_adjust_train))
```

```
## [1] "Number of samples in train (SA) =  2329"
```

```
paste("Number of samples in test (SA) = ", nrow(season_adjust_test))
```

```
## [1] "Number of samples in test (SA) =  59"
```

```
paste("Number of samples in train (NSA) = ", nrow(nsa_adjust_train))
```

```
## [1] "Number of samples in train (NSA) =  2329"
```

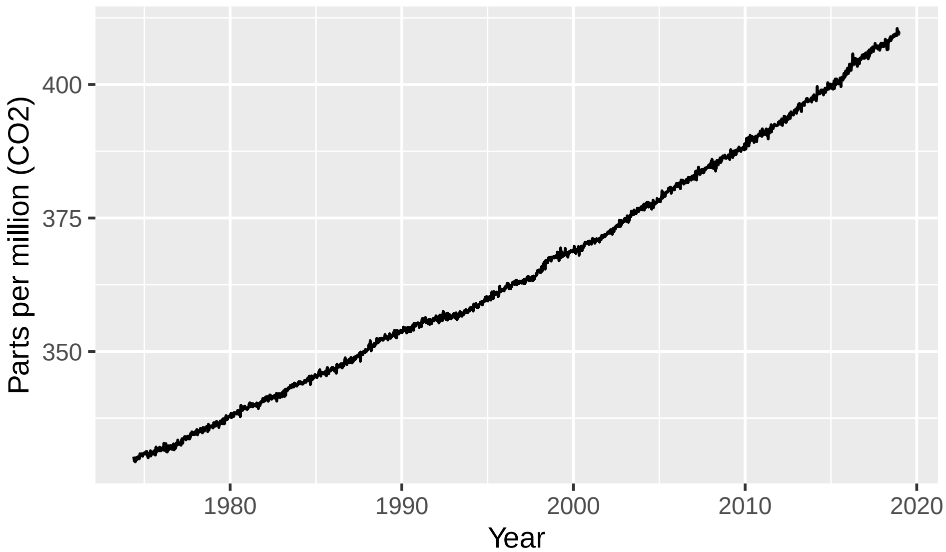
```
paste("Number of samples in test (NSA) = ", nrow(nsa_adjust_test))
```

```
## [1] "Number of samples in test (NSA) =  59"
```

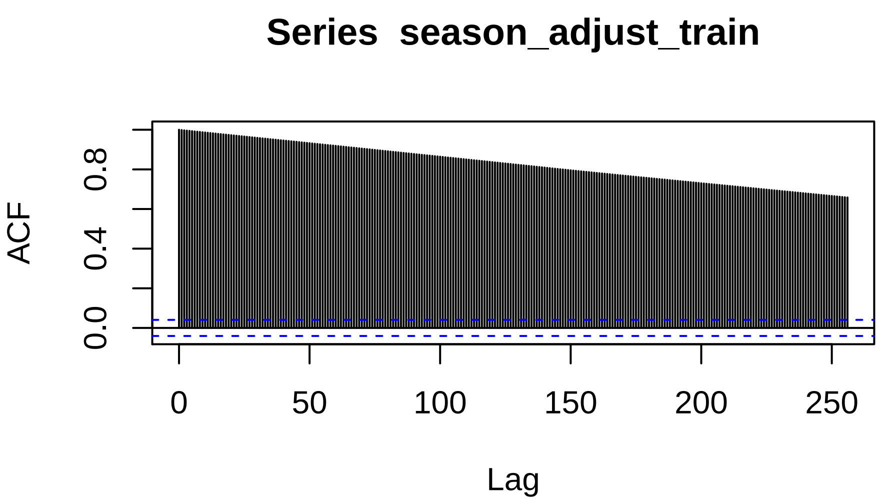
Fit ARIMA models

For seasonally adjusted series

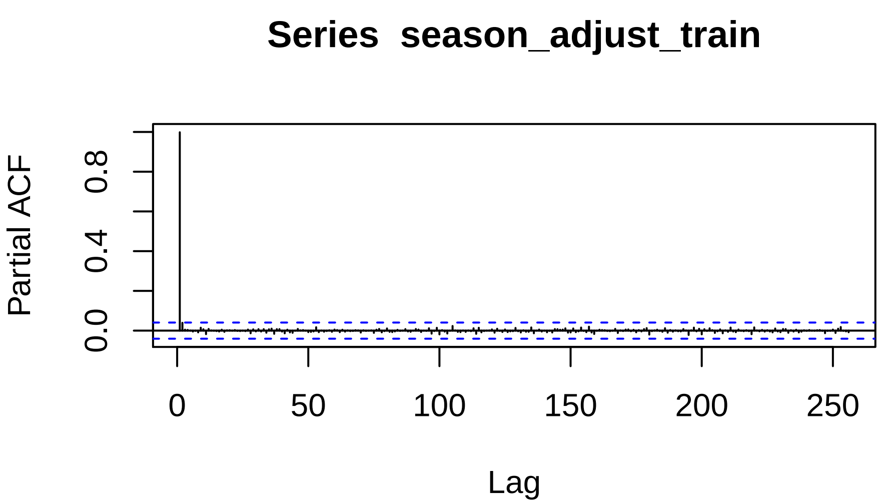
```
season_adjust_train %>% autoplot(season_adjust) + ylab("Parts per million (CO2)") +
    xlab("Year")
```




```r
acf(season_adjust_train, lag.max = 256)
```




```r
pacf(season_adjust_train, lag.max = 256)
```



An AR signature corresponds to a PACF plot displaying a sharp cut-off and a more slowly decaying ACF

Test for stationarity

```
adf.test(season_adjust_train$season_adjust)
```

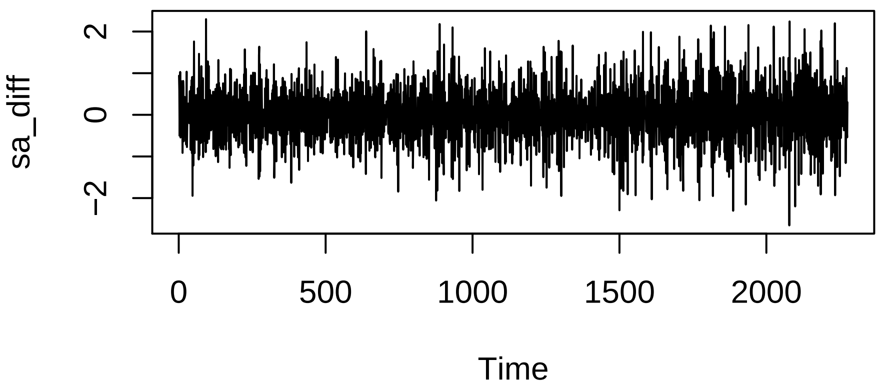
```
## Warning in adf.test(season_adjust_train$season_adjust): p-value greater
## than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  season_adjust_train$season_adjust
## Dickey-Fuller = -0.14557, Lag order = 13, p-value = 0.99
## alternative hypothesis: stationary
```

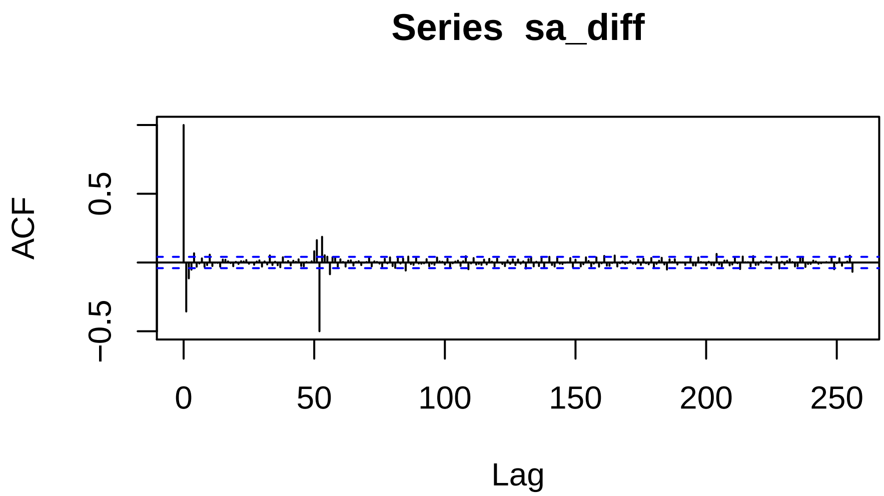
We fail to reject the null hypothesis that the series has a unit root. Therefore the series is **non-stationary**.

First order difference

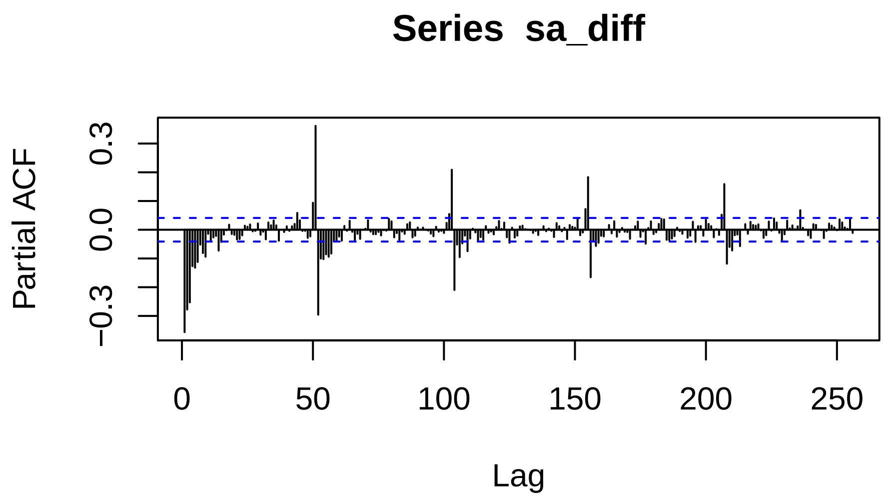
```
sa_diff <- na.omit(difference(difference(season_adjust_train$season_adjust,
    lag = 365.25/7)))
ts.plot(sa_diff, type = "l")
```




```
acf(sa_diff, lag.max = 256)
```




```r
pacf(sa_diff, lag.max = 256)
```




```r
# ADF test
adf.test(sa_diff)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  sa_diff
## Dickey-Fuller = -18.633, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

The ADF test shows that the first order difference of the sesonally adjusted series is **stationary**. Spike all the way upto lags 4 of the ACF shows the presence of a non-seasonal MA component. Again the spike at lag 52 shows a seasonal MA component can be used. Both ACF and PACF show very strong seasonality effects at lags of 52 weeks.

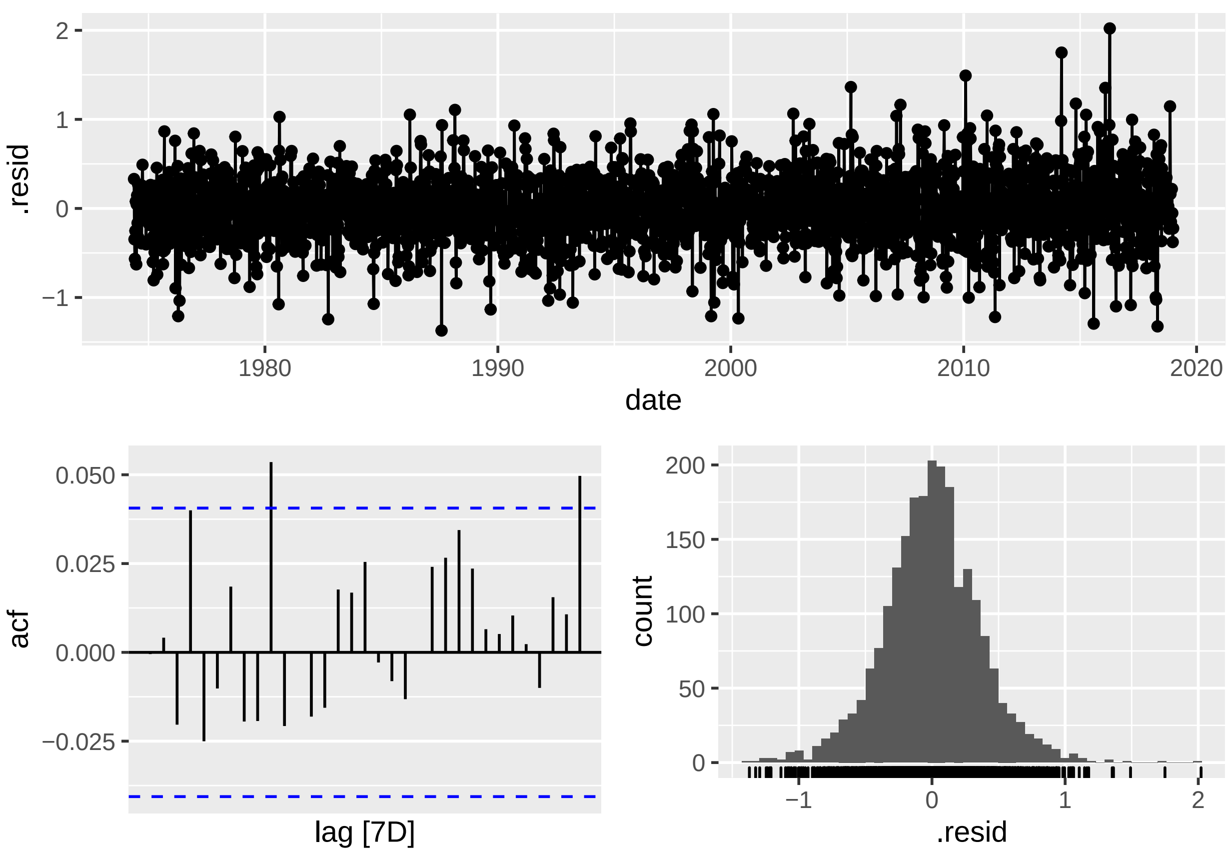
An MA signature corresponds to an ACF plot displaying a sharp cut-off and a PACF plot that decays more slowly

ARIMA

```
# pnd.arima <- arima(season_adjust_train$season_adjust,
# order=c(2,1,1), seasonal = list(order=c(0,1,1)))
pnd.arima <- season_adjust_train %>% model(ARIMA(season_adjust ~
    pdq(2, 1, 1) + PDQ(0, 1, 1)))

report(pnd.arima)
```

```
## Series: season_adjust
## Model: ARIMA(2,1,1) w/ drift
##
## Coefficients:
##          ar1      ar2      ma1  constant
##       0.2064  -0.0425  -0.8194    0.0287
## s.e.  0.0272   0.0245   0.0181    0.0014
##
## sigma^2 estimated as 0.1343:  log likelihood=-964.41
## AIC=1938.82   AICc=1938.85   BIC=1967.58
```

```
pnd.arima %>% gg_tsresiduals()
```




```
# plot(resid(pnd.arima), type='l') acf(resid(pnd.arima),
# lag.max = 256) pacf(resid(pnd.arima), lag.max = 256)
```

We start by estimating an ARIMA model based off the EDA we did above.

Ljung-Box test

```
augment(pnd.arima) %>% features(.resid, ljung_box)
```

```
## # A tibble: 1 x 3
##   .model                                           lb_stat lb_pvalue
##   <chr>                                              <dbl>     <dbl>
## 1 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 0.000544     0.981
```

The p-value is **not significant**, which confirms that this is a **white noise** process.

Verify the accuracy of the model we just trained

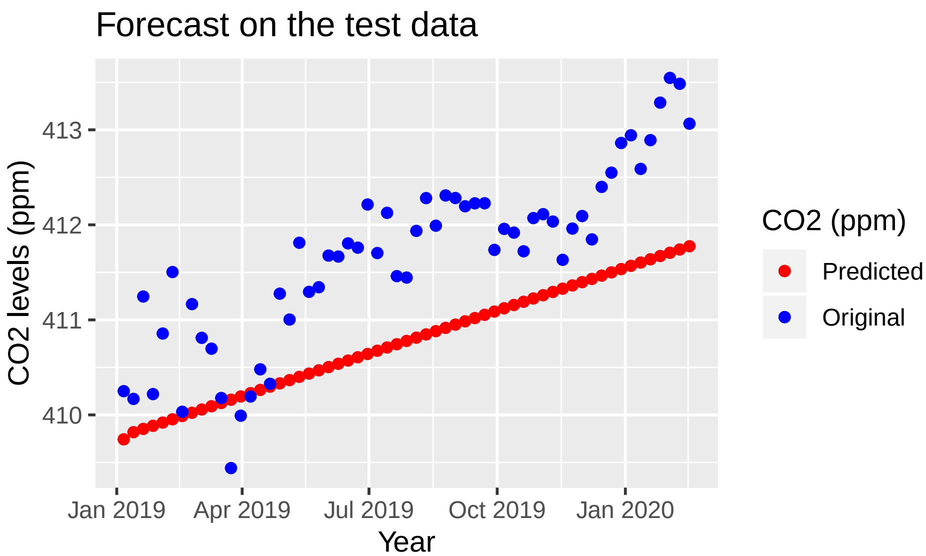
```
library(forecast)
forecast_test <- pnd.arima %>% forecast(h = nrow(season_adjust_test))
accuracy(forecast_test, season_adjust_test)
```

```
## # 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(season_adjust ~ pd~ Test  0.888  1.02 0.921 0.216 0.223   NaN 0.556
```

The RMSE of the trained model at 1.023977 seems to be really good.

Let's review how the model does on the test data via a time series plot

```
# forecast the ranges of the test data
forecast_test <- pnd.arima %>% forecast(h = nrow(season_adjust_test))
# create a dataframe of the forecasted data (for ggplot)
forecast_test_df <- as.data.frame(forecast_test)
# add the original data to the new dataframe (for ggplot)
forecast_test_df$original <- season_adjust_test$season_adjust

ggplot(data = forecast_test_df, mapping = aes(date)) + geom_point(aes(y = season_adjust,
    col = "blue")) + geom_point(aes(y = original, col = "red")) +
    scale_color_manual(name = "CO2 (ppm)", labels = c("Predicted",
        "Original"), values = c("red", "blue")) + xlab("Year") +
    ylab("CO2 levels (ppm)") + ggtitle("Forecast on the test data")
```




```
# calculate RMSE
paste("RMSE on test data =", RMSE(forecast_test_df$original -
    forecast_test_df$season_adjust))
```

```
## [1] "RMSE on test data = 1.02397683622316"
```

We see a linear trend in the predictions.

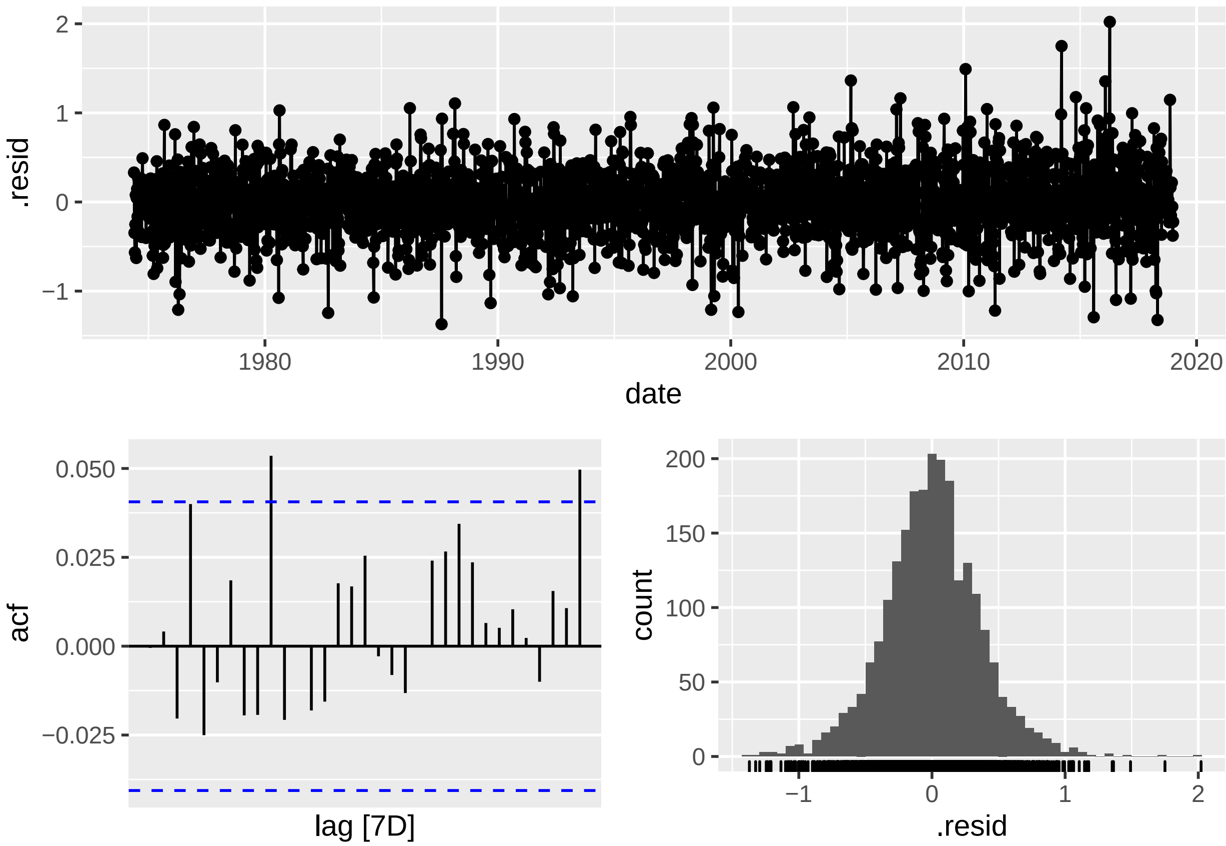
Forecast

We are now ready to forecast the seasonaly adjusted series.

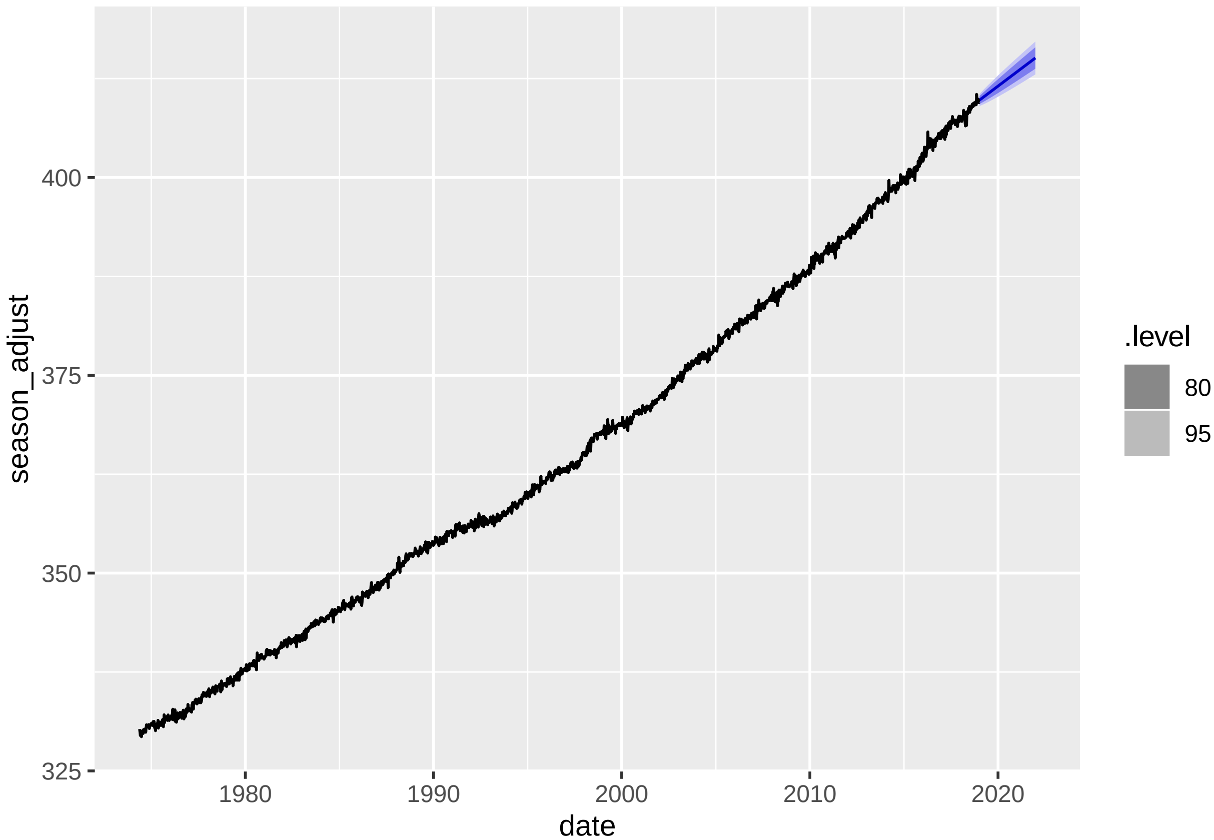
```
fit <- season_adjust_train %>% model(ARIMA(season_adjust ~ pdq(2,
    1, 1) + PDQ(0, 1, 1)))
resid(fit)
```

```
## # A tsibble: 2,329 x 3 [7D]
## # Key:       .model [1]
##    .model                                              date        .resid
##    <chr>                                               <date>       <dbl>
##  1 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-05-19  0.330
##  2 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-05-26 -0.347
##  3 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-06-02 -0.567
##  4 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-06-09 -0.251
##  5 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-06-16  0.0787
##  6 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-06-23 -0.628
##  7 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-06-30  0.0449
##  8 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-07-07  0.147
##  9 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-07-14 -0.164
## 10 ARIMA(season_adjust ~ pdq(2, 1, 1) + PDQ(0, 1, 1)) 1974-07-21  0.272
## # ... with 2,319 more rows
```

```
fit %>% gg_tsresiduals()
```



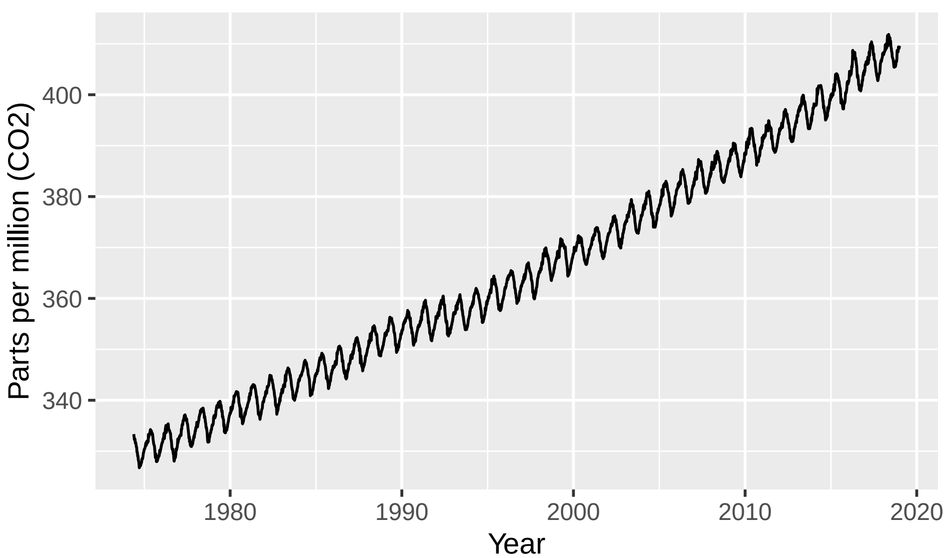

```r
fit %>% forecast(h = 156) %>% autoplot(season_adjust_train)
```



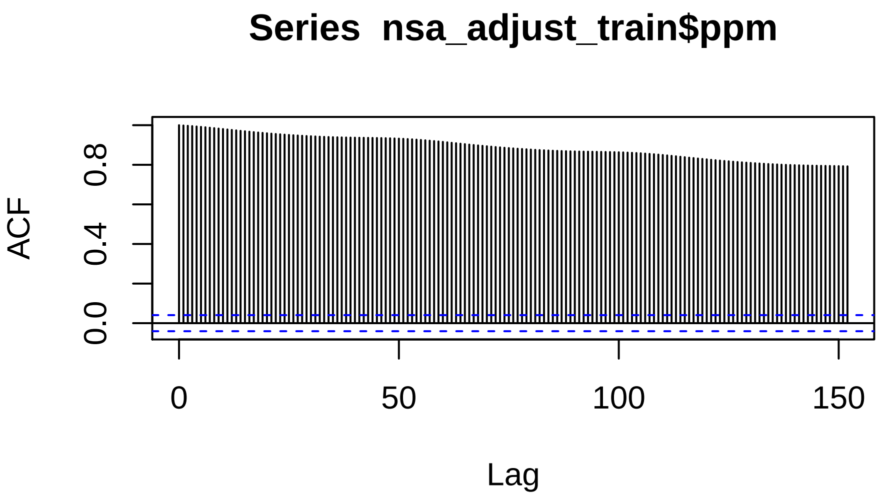
Modelling the non-sesonally adjusted series

Use the training data we prepared to examine the relevant plots

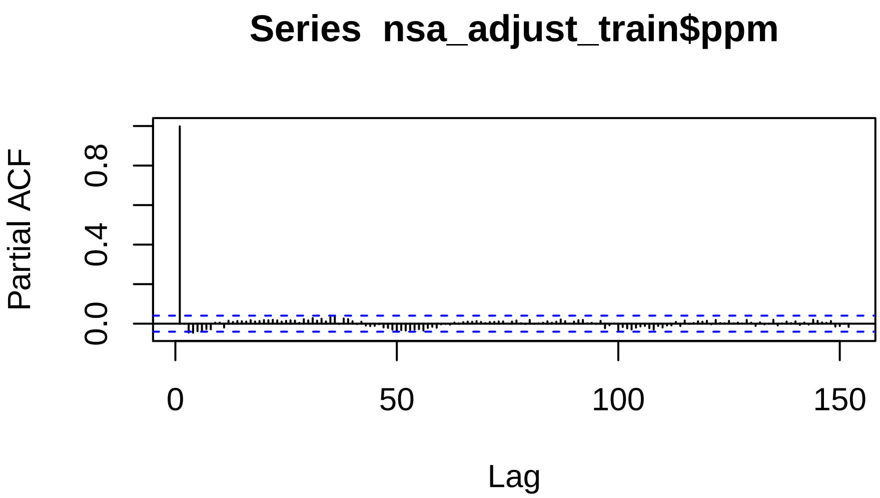
```
nsa_adjust_train %>% autoplot(ppm) + ylab("Parts per million (CO2)") +
    xlab("Year")
```



```r
acf(nsa_adjust_train$ppm, lag.max = 152)
```




```r
pacf(nsa_adjust_train$ppm, lag.max = 152)
```

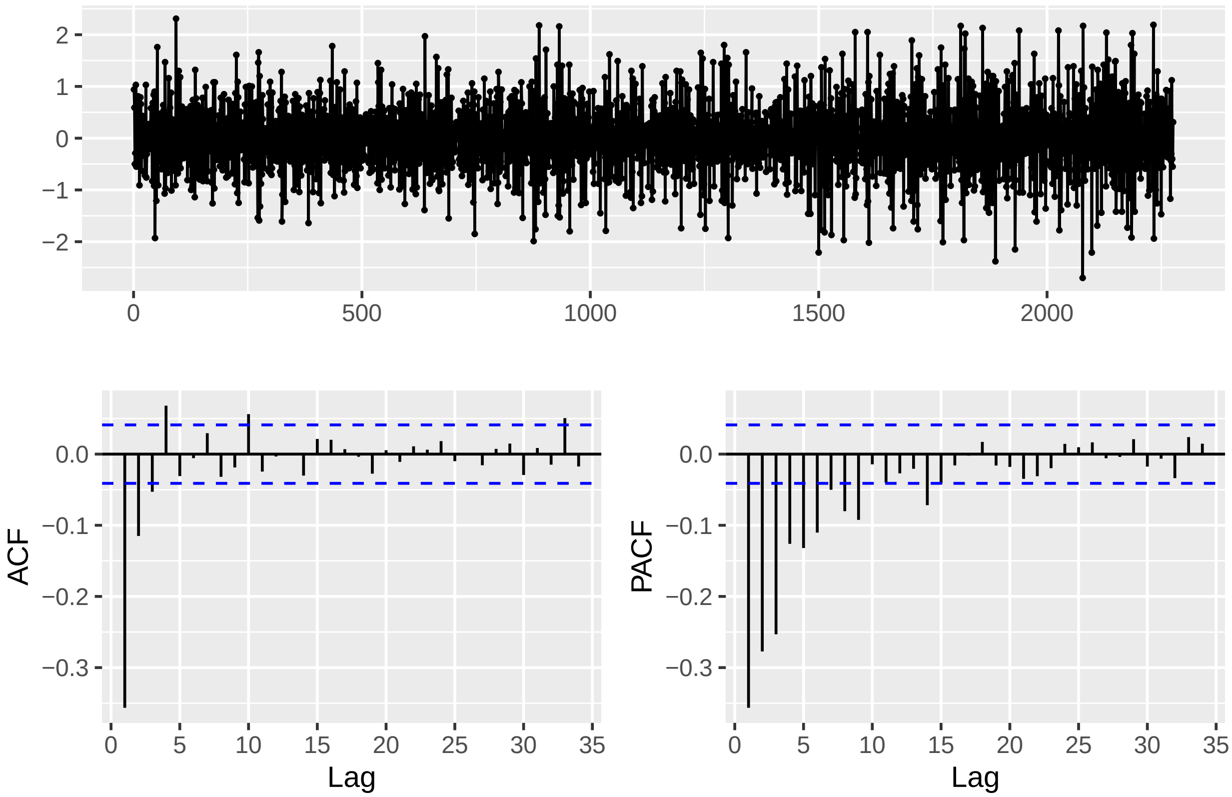



```r
adf.test(nsa_adjust_train$ppm)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  nsa_adjust_train$ppm
## Dickey-Fuller = -7.7612, Lag order = 13, p-value = 0.01
## alternative hypothesis: stationary
```

There is evidence of strong seasonal effects in the ACF plots. The ACF and PACF shows evidence of an AR process. We will need to determine the order of the process and deal with the sesonal effects. The ACF plots decays very slowly and even upto 150 lags the auto-correlation is significant. The series will need differencing. The PACF drops off rapidly after the 1st lag, so we can try with an non-seasonal AR component.

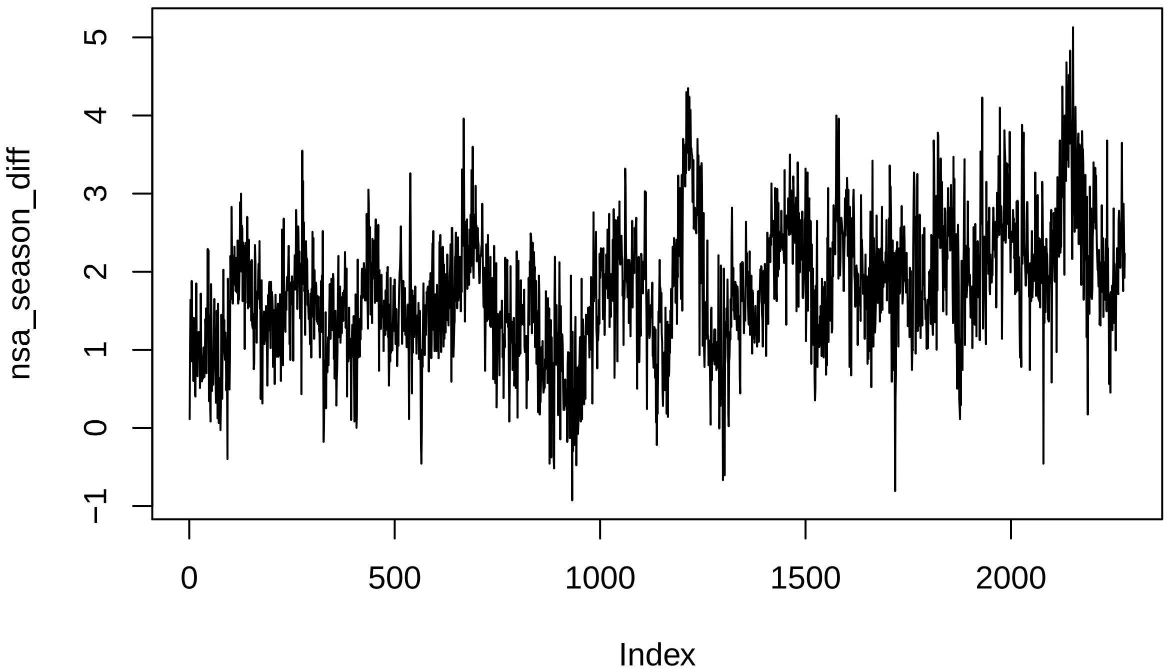
```r
nsa_adjust_train$ppm %>% diff(lag = 52) %>% diff() %>% ggtsdisplay()
```



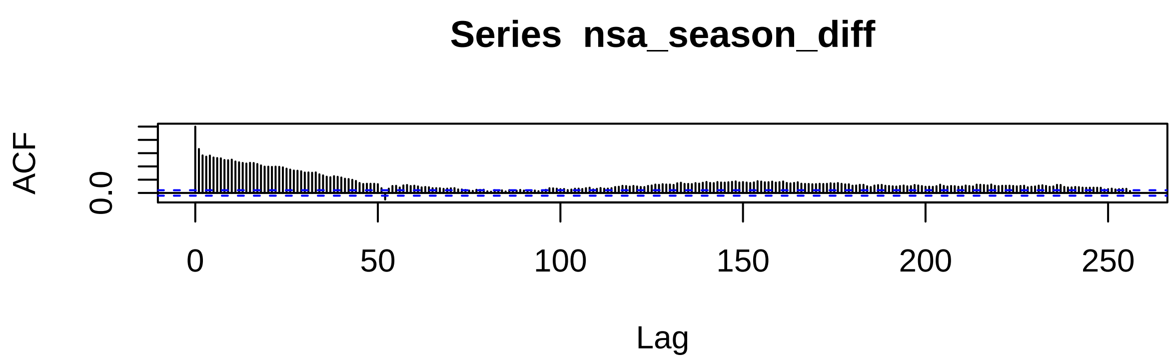
Let's look at the properties of the sesonally differenced series

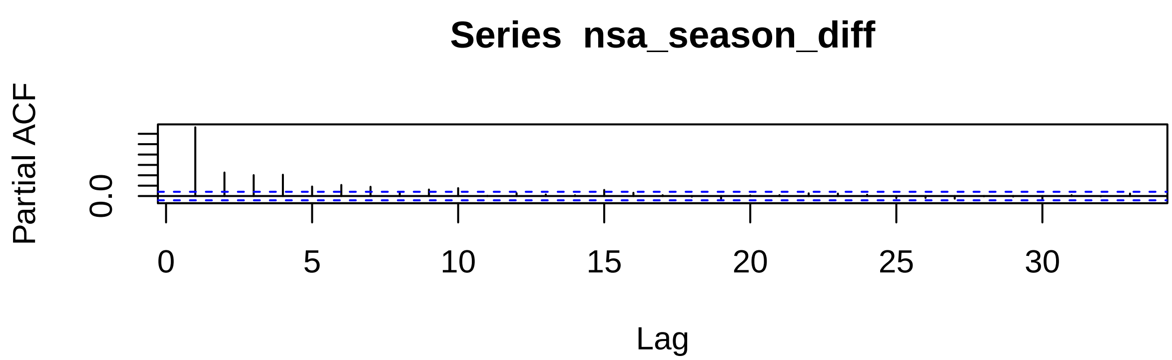
Sesonal differenced

```r
nsa_season_diff <- na.omit(difference(nsa_adjust_train$ppm, lag = 52))
plot(nsa_season_diff, type = "l")
```




```r
par(mfrow = c(2, 1))
acf(nsa_season_diff, lag = 256)
pacf(nsa_season_diff)
```

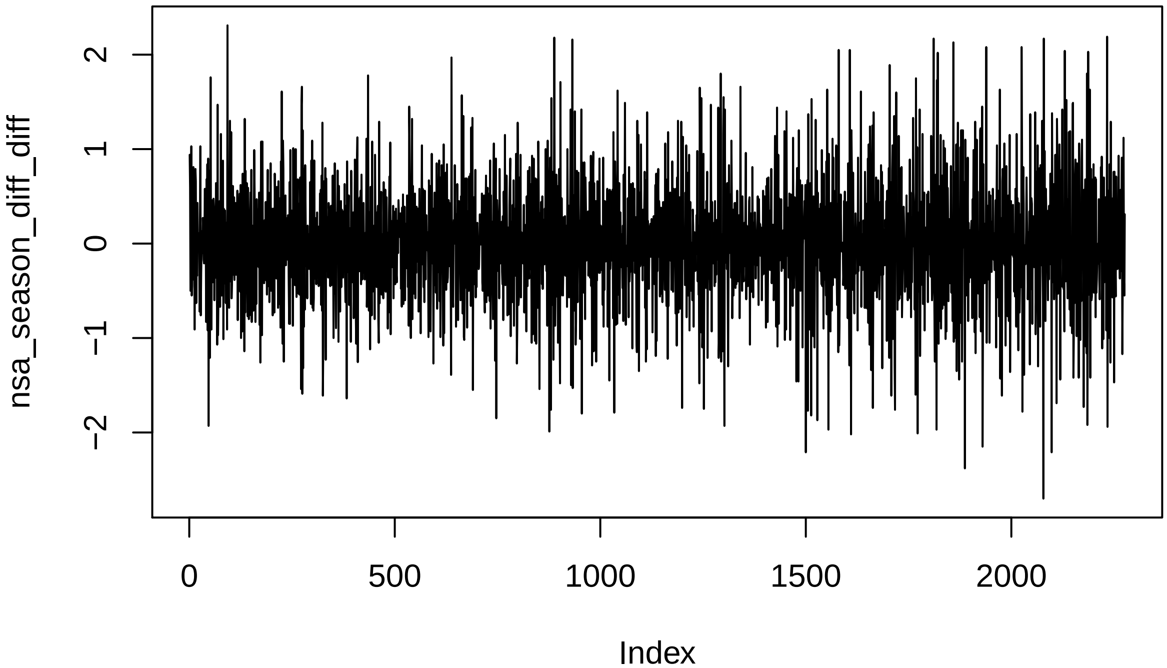




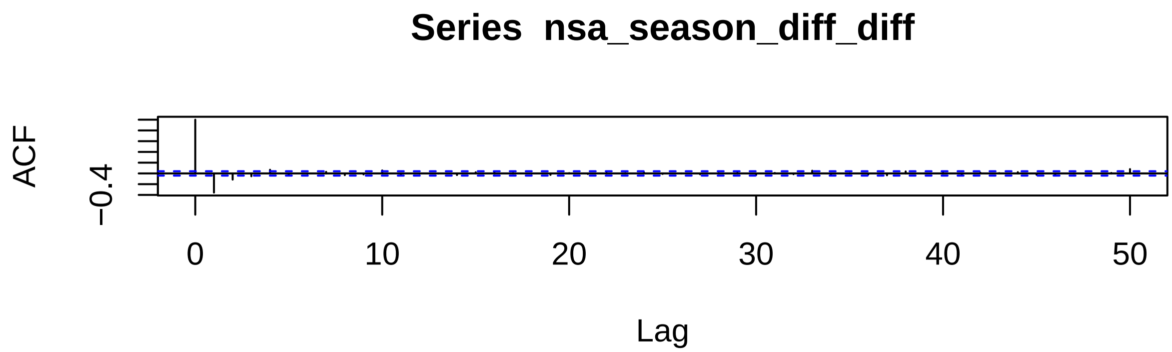
The plot of the series clearly shows that we still have trend in the sesonaly differenced series. Again, both ACF and PACF has significant auto-correlations. The ACF has auto-correlation even upto a lag of 52. The series may need an ordinary differencing over the sesonal differencing.

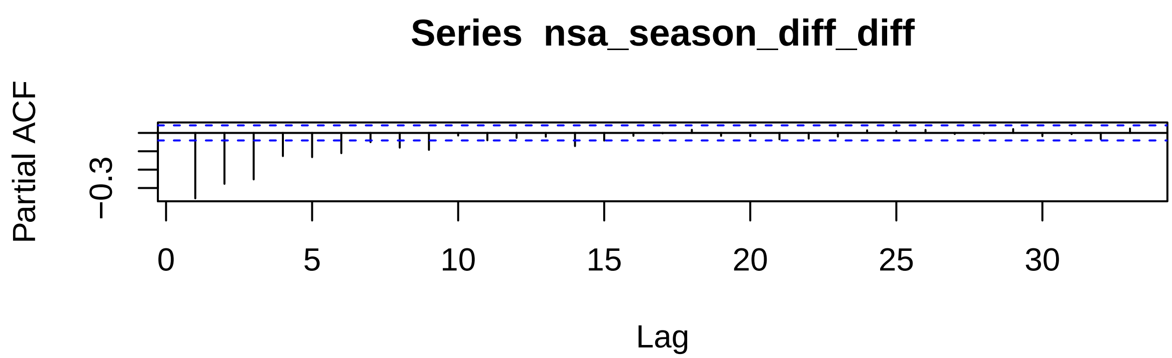
Differencing the sesonally differenced series

```
nsa_season_diff_diff <- na.omit(diff(diff(nsa_adjust_train$ppm,
    lag = 52)))
plot(nsa_season_diff_diff, type = "l")
```




```
par(mfrow = c(2, 1))
acf(nsa_season_diff_diff, lag = 50)
pacf(nsa_season_diff_diff)
```





We seem to have removed the trend. The ACF plot shows a spike at lag 1 and another at lag 2. This shows that we will need a non-seasonal AR component, possibly an AR(2). We will try with both $p = 1$ and $p = 2$. The PACF plot shows significant auto-correlations upto lag of 9 which shows we will need a non-seasonal MA component here as well.

Again, the seasonal difference shows spikes at lag 1 and 2 in the ACF and significant correlations upto lag 9 in PACF. So, we may need seasonal AR and MA components as well.

We will start with the following orders:

$$d = 1, D = 1, p = 2, q = 1, P = 0, Q = 0$$

ARIMA (NSA)

We will iterate over a few parameter values and examine the AIC. This allows us to isolate into a model. # this takes a really long time and we might want to look at a different approach

```
find_aic <- function(p, d, q, P, D, Q) {
    pnd.arima <- arima(nsa_adjust_train$ppm, order = c(p, d,
        q), seasonal = list(order = c(P, D, Q), period = 52))
    return(pnd.arima$aic)
}
# initialize the parameters vectors
vec1 <- c(2, 1, 1, 0, 1, 0)
vec2 <- c(2, 1, 1, 0, 1, 1)
vec3 <- c(1, 1, 1, 0, 1, 1)
vec4 <- c(1, 1, 1, 0, 1, 0)
vec5 <- c(1, 1, 1, 0, 1, 2)
vec6 <- c(1, 1, 3, 0, 1, 2)
```

```r
params <- t(array(c(vec1, vec2, vec3, vec4, vec5, vec6), dim = c(6,
    6)))
df <- data.frame()
for (i in 1:nrow(params)) {
    aic <- find_aic(params[i, 1], params[i, 2], params[i, 3],
        params[i, 4], params[i, 5], params[i, 6])
    df <- rbind(df, cbind(params[i, 1], params[i, 3], params[i,
        4], params[i, 6], aic))
}
df
```

```
##   V1 V2 V3 V4      aic
## 1  2  1  0  0 3853.238
## 2  2  1  0  1 2708.534
## 3  1  1  0  1 2707.541
## 4  1  1  0  0 3854.396
## 5  1  1  0  2 2701.366
## 6  1  3  0  2 2704.691
```

The **best** AIC is for the parameters $ARIMA(p = 1, d = 1, q = 1, P = 0, D = 1, Q = 2)_{52}$

Train the ARIMA model for the best AIC

```r
# The Arima model reformats the date to numerical values
# which throws the plots off. Even though both give the same
# result, we would use the ARIMA here so that the date
# formats are plot friendly nsa.arima <- nsa_adjust_train$ppm
# %>% Arima(order=c(1,1,1), seasonal = list(order=c(0,1,2),
# period = 52))
nsa.ARIMA <- nsa_adjust_train %>% model(ARIMA(ppm ~ 0 + pdq(1,
    1, 1) + PDQ(0, 1, 2, period = 52)))
```

```r
report(nsa.ARIMA)
```

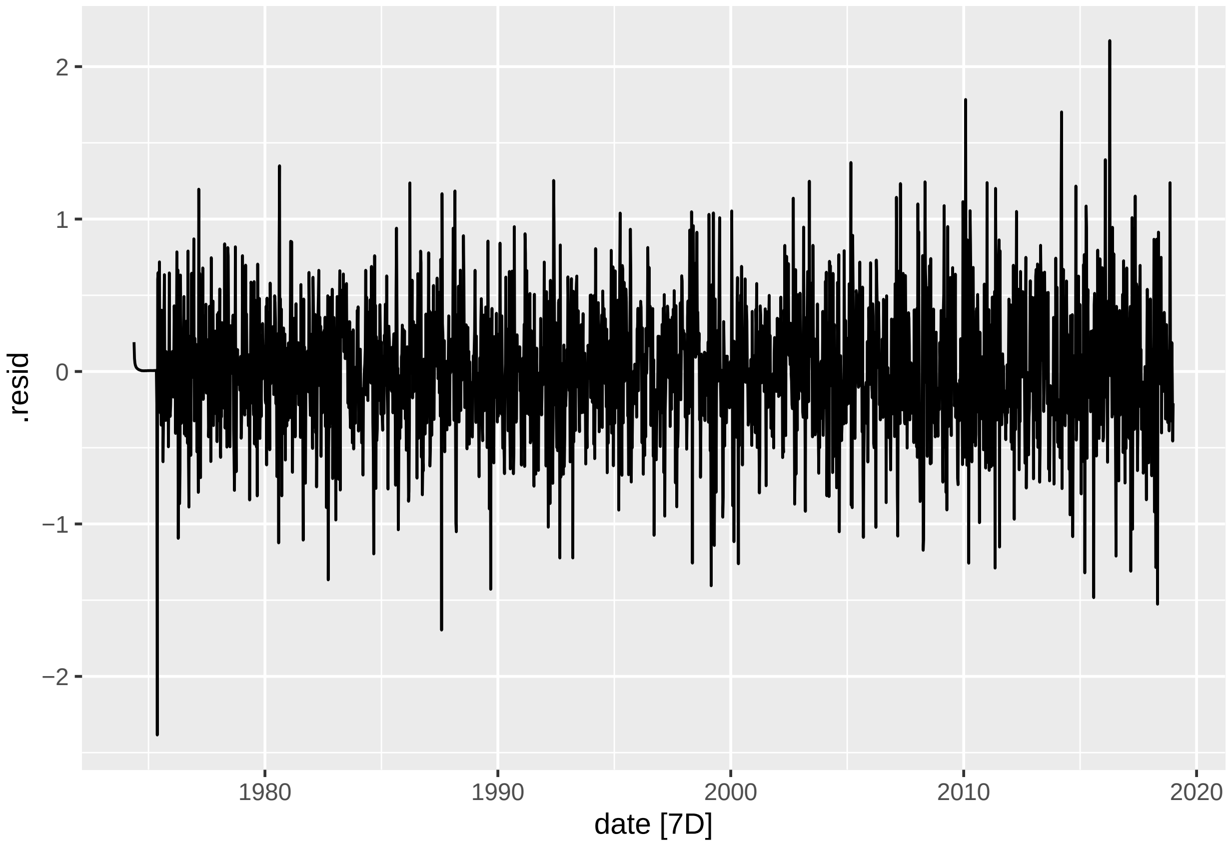
```
## Series: ppm
## Model: ARIMA(1,1,1)(0,1,2)[52]
##
## Coefficients:
##          ar1      ma1     sma1    sma2
##       0.2316  -0.7936  -0.8562  0.0619
## s.e.  0.0319   0.0226   0.0232  0.0216
##
## sigma^2 estimated as 0.1893:  log likelihood=-1345.68
## AIC=2701.37   AICc=2701.39   BIC=2730.02
```

All the values are significant.

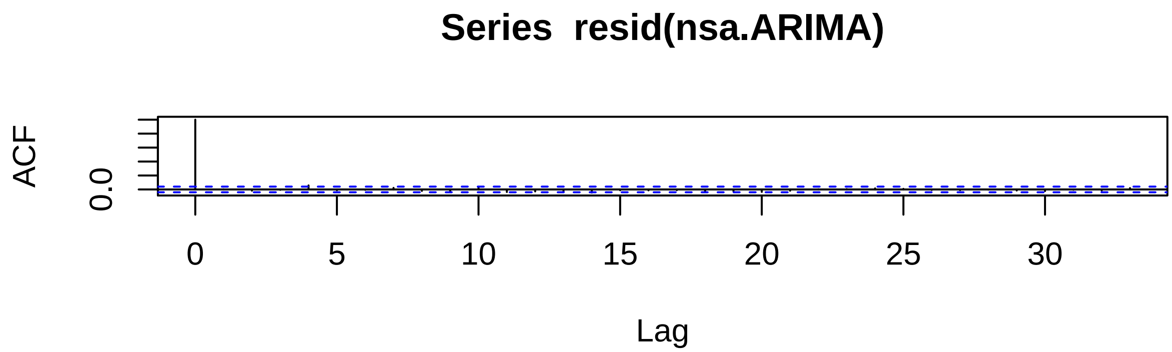
Plot the residuals

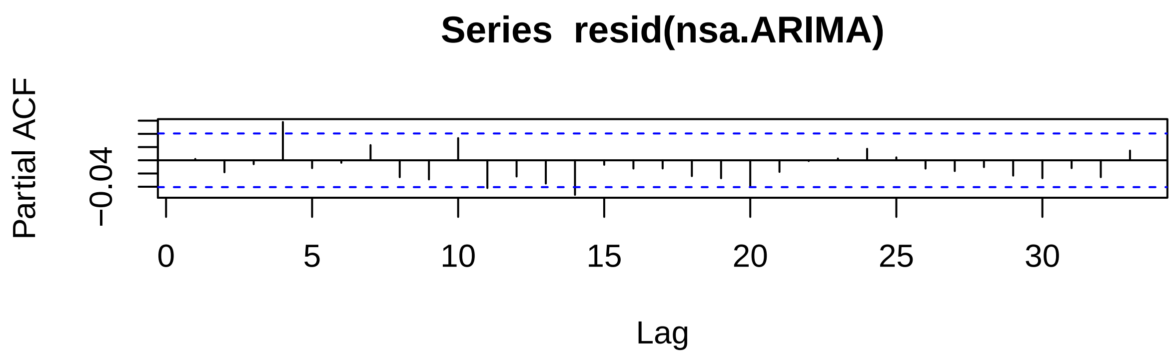
```r
resid(nsa.ARIMA) %>% autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = .resid`
```




```
par(mfrow = c(2, 1))
acf(resid(nsa.ARIMA))
pacf(resid(nsa.ARIMA))
```





The ACF and PACF shows that the residuals look like a white noise process.

Ljung-Box test

```r
model_list <- nsa.ARIMA %>% residuals()
Box.test(model_list[, 3], type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  model_list[, 3]
## X-squared = 0.0094113, df = 1, p-value = 0.9227
```

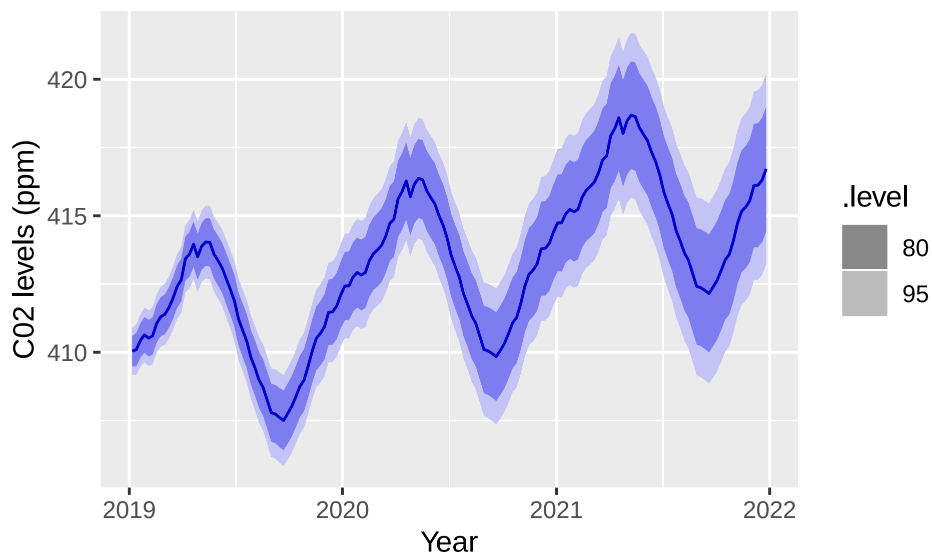
The p-value is **not significant**, which confirms that this is a **white noise** process.

We are now read to forecast with this model.

Forecast

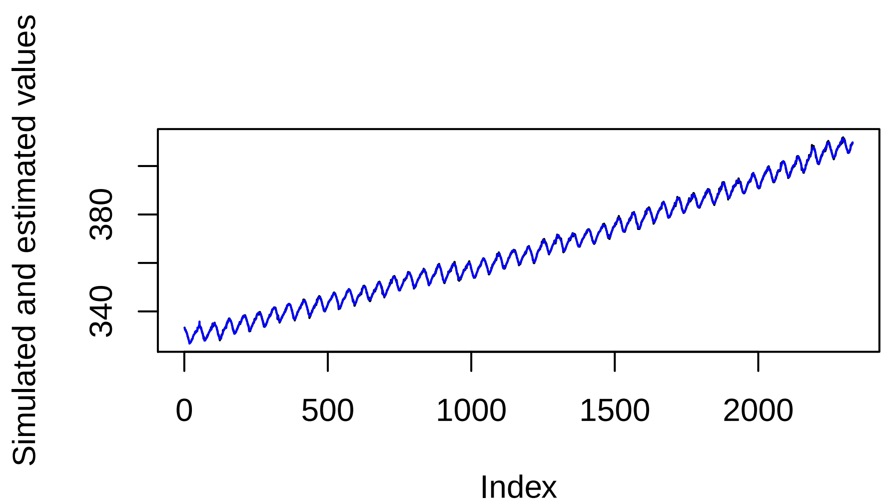
A 3 year ahead forecast is as follows

```r
nsa.ARIMA %>% forecast(h = 156) %>% autoplot() + ylab("CO2 levels (ppm)") +
    xlab("Year")
```



Plot in-sample fits

```
plot(nsa_adjust_train$ppm, type = "l", col = "black", ylab = "Simulated and estimated values")
lines(fitted(nsa.ARIMA)[, c(3)], col = "blue", lty = 1)
legend(10, 17, legend = c("Actual", "Estimated"), col = c("black",
    "blue"), lty = 1:2, cex = 0.8)
```



Accuracy

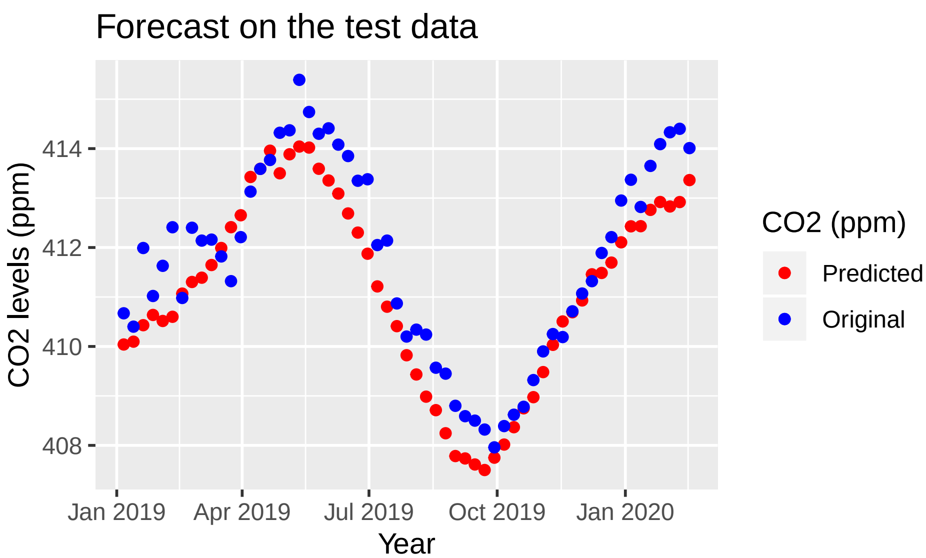
```
accuracy(nsa.ARIMA)
```

```
## # 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(ppm ~ 0 + p~ Trai~ 0.00825 0.430 0.327 0.00201 0.0890 0.753 0.00201
```

The RMSE of the model seems good at 0.4297062.

Let's see how the model does on the test data

```
nsa_adjust_test_df <- as.data.frame(nsa_adjust_test)

fcast <- nsa.ARIMA %>% forecast(h = 59)
df_cast <- as.data.frame(fcast)
df_cast$original <- nsa_adjust_test_df$ppm

ggplot(data = df_cast, mapping = aes(date)) + geom_point(aes(y = ppm,
    col = "blue")) + geom_point(aes(y = original, col = "red")) +
    scale_color_manual(name = "CO2 (ppm)", labels = c("Predicted",
        "Original"), values = c("red", "blue")) + xlab("Year") +
    ylab("CO2 levels (ppm)") + ggtitle("Forecast on the test data")
```



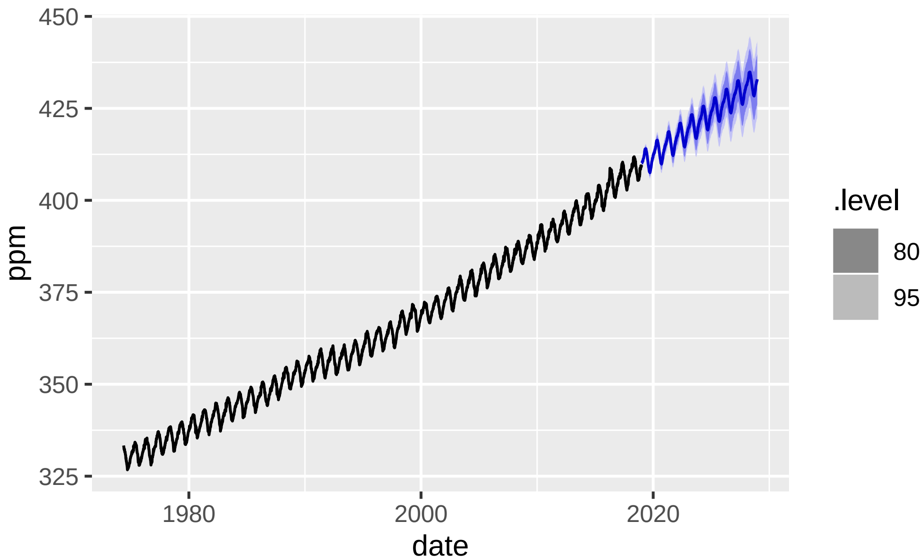

```
# calculate RMSE
paste("RMSE on test data =", RMSE(df_cast$original - df_cast$ppm))
```

```
## [1] "RMSE on test data = 0.846579938086959"
```

The RMSE 0.8465 on the test set is also very good. We are now ready to use this series to do year ahead forecasts.

The forecasted value along with the original series, a 10 year ahead forecast

```
nsa.ARIMA %>% forecast(h = 520) %>% autoplot(nsa_adjust_train)
```



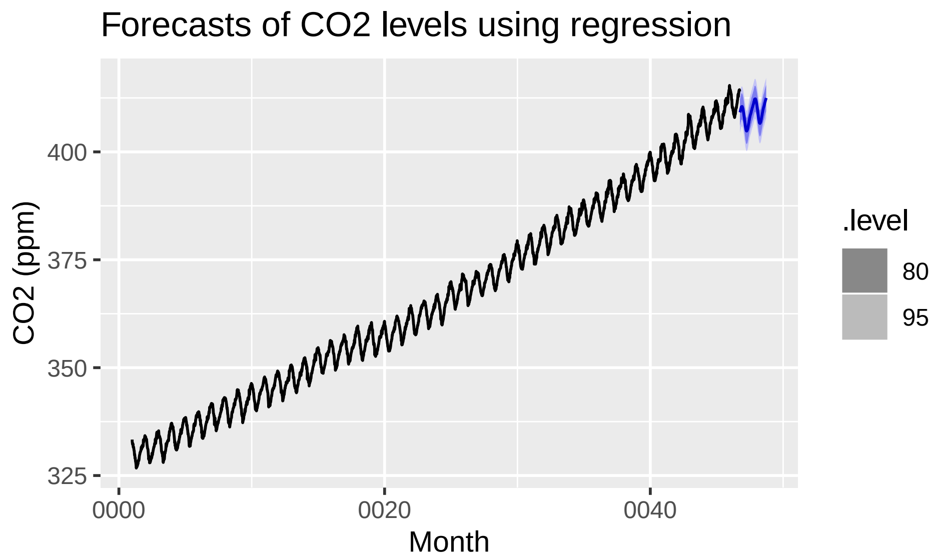
We can compare this to our answer in part 2, looking at the difference in predictions.

Fit a linear time trend model using TSLM

```
library(fable)
all_tsib <- as_tsibble(ts(co2_withdate_mut$ppm, frequency = 365.25/7),
    index = date)
fit_ppm <- all_tsib %>% model(TSLM(value ~ trend() + season()))
```

Forecast

```
fc_ppm <- forecast(fit_ppm)
fc_ppm %>% autoplot(all_tsib) + ggtitle("Forecasts of CO2 levels using regression") +
    xlab("Month") + ylab("CO2 (ppm)")
```



Plot the fitted values

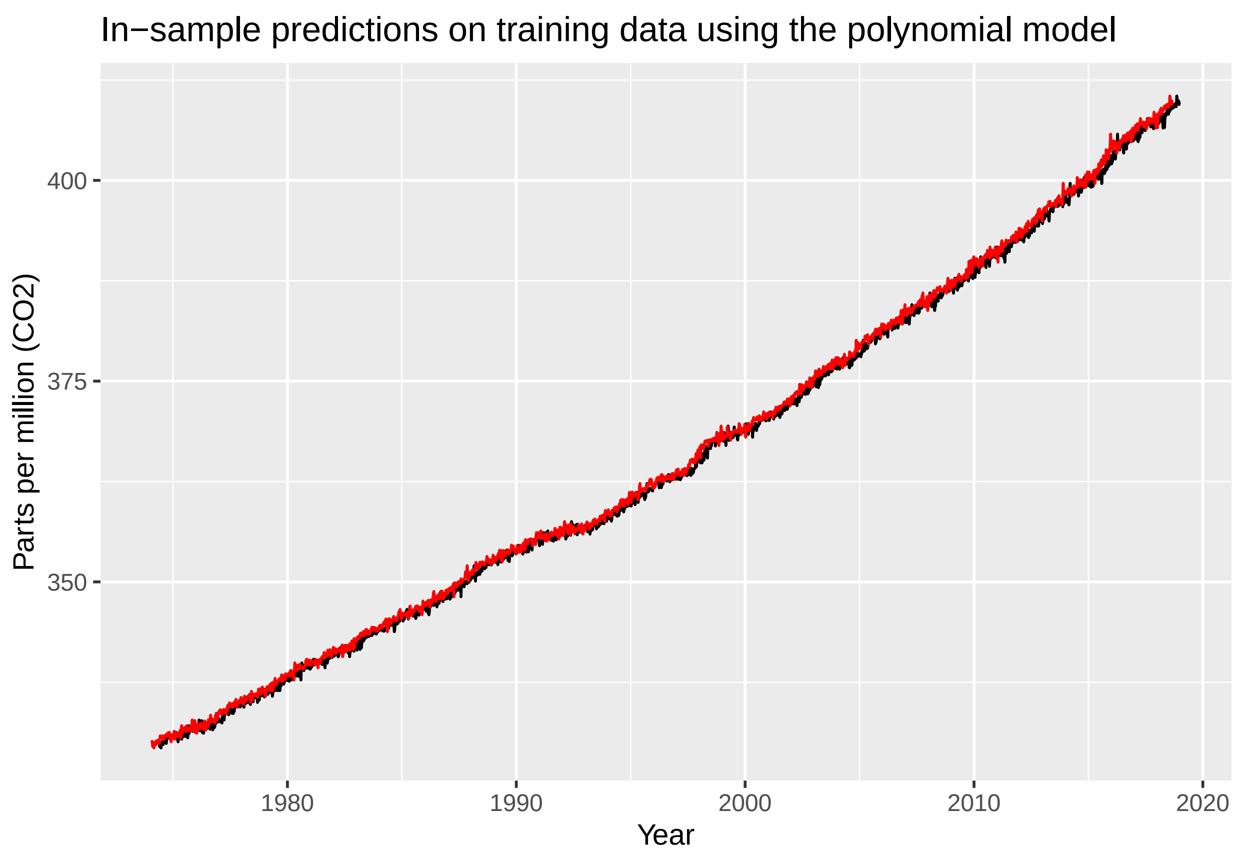
Fit a polynomial time-trend model

```
wk = time(season_adjust_train$season_adjust) - mean(time(season_adjust_train$season_adjust))
wk2 = wk^2
wk3 = wk^3
co2_poly_reg = lm(season_adjust_train$season_adjust ~ wk + wk2 +
    wk3, na.action = NULL)
# plot(season_adjust_train$season_adjust, type='l',
# col='blue', main='Regression smoothing using a 3rd degree
# polynomial', ylab='CO2 (ppm)') lines(fitted(co2_poly_reg),
# col='black')
paste("RMSE of polynomial model on training data =", sqrt(mean((fitted(co2_poly_reg) -
    season_adjust_test$season_adjust)^2)))
```

```
## Warning in fitted(co2_poly_reg) - season_adjust_test$season_adjust: longer
## object length is not a multiple of shorter object length

## [1] "RMSE of polynomial model on training data = 51.389859341136"
```

```
predicted.ts <- as_tsibble(ts(fitted(co2_poly_reg), frequency = 365.25/7,
    start = c(1974, 5)), index = date)
original.ts <- as_tsibble(ts(season_adjust_train$season_adjust,
    frequency = 365.25/7, start = c(1974, 5)), index = date)
season_adjust_train %>% autoplot(season_adjust) + autolayer(predicted.ts,
    series = "Predicted", color = "red") + ylab("Parts per million (CO2)") +
    xlab("Year") + ggtitle("In-sample predictions on training data using the polynomial model")
```

```
## Plot variable not specified, automatically selected `.vars = value`

## Warning: Ignoring unknown parameters: series
```



Clearly, the RMSE of this model is much worse when compared to the ARIMA model on the seasonally adjusted series.

Let's review how this model did on the test data

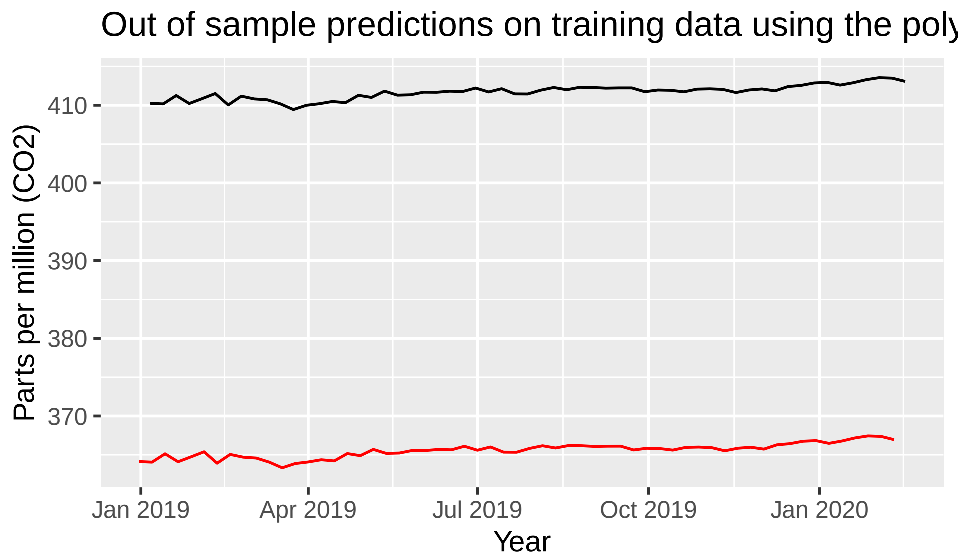
```
wk = time(season_adjust_test$season_adjust) - mean(time(season_adjust_test$season_adjust))
wk2 = wk^2
wk3 = wk^3
forecast_test <- predict(co2_poly_reg, newdata = data.frame(wk,
    wk2, wk3))
paste("RMSE of polynomial model on test data =", sqrt(mean((forecast_test -
    season_adjust_test$season_adjust)^2)))
```

```
## [1] "RMSE of polynomial model on test data = 46.1088081726043"
```

```
predicted.ts <- as_tsibble(ts(forecast_test, frequency = 365.25/7,
    start = c(2019, 1)), index = date)
original.ts <- as_tsibble(ts(season_adjust_test$season_adjust,
    frequency = 365.25/7, start = c(2019, 1)), index = date)

season_adjust_test %>% autoplot(season_adjust) + autolayer(predicted.ts,
    series = "Predicted", color = "red") + ylab("Parts per million (CO2)") +
    xlab("Year") + ggtitle("Out of sample predictions on training data using the polynomial mod
```

```
## Plot variable not specified, automatically selected `.vars = value`
```

```
## Warning: Ignoring unknown parameters: series
```



The RMSE of the trained model at 046.1088 is **much worse than the ARIMA model**.

Part 6 (3 points): Predict 420 and 500ppm

Generate predictions for when atmospheric CO2 is expected to be at 420 ppm and 500 ppm levels for the first and final times (consider prediction intervals as well as point estimates in your answer). Generate a prediction for atmospheric C02 levels in the year 2100. How confident are you that these are accurate predictions?

Forecast the values using ARIMA model for NSA (not-seasonaly adjusted series)

Prediction intervals

```r
# get the sigma2 of the residuals
df <- glance(nsa.ARIMA) %>% dplyr::select(sigma2)
sigma2 <- df[[1]]
# predict future values
d <- forecast(nsa.ARIMA, h = 520)
pred_confint <- data.frame(d[[2]], d[[3]], d[[3]] - 1.96 * sqrt(sigma2),
    d[[3]] + 1.96 * sqrt(sigma2))
colnames(pred_confint) <- c("date", "ppm", "low", "high")
head(pred_confint)
```

```
##         date      ppm      low     high
## 1 2019-01-06 410.0393 409.1866 410.8920
## 2 2019-01-13 410.0963 409.2436 410.9490
## 3 2019-01-20 410.4298 409.5770 411.2825
## 4 2019-01-27 410.6367 409.7840 411.4895
## 5 2019-02-03 410.5149 409.6621 411.3676
## 6 2019-02-10 410.5996 409.7469 411.4523
```

```r
pred_confint <- pred_confint %>% mutate(lev420 = ifelse(ppm >=
    420 & ppm <= 421, 1, 0))
pred_confint <- pred_confint %>% mutate(high420 = ifelse(high >=
    420 & high <= 421, 1, 0))
pred_confint <- pred_confint %>% mutate(low420 = ifelse(low >=
    420 & low <= 421, 1, 0))
pred_confint <- pred_confint %>% mutate(conf = ifelse(lev420 ==
    1 | high420 == 1 | low420 == 1, 1, 0))

pred_confint <- pred_confint %>% mutate(colr = ifelse(lev420 ==
    1 & high420 == 0 & low420 == 0, 1, ifelse(lev420 == 0 & high420 ==
    1 & low420 == 0, 2, ifelse(lev420 == 0 & high420 == 0 & low420 ==
    0, 3, 0))))

dt_st <- pred_confint %>% filter(high420 == 1) %>% dplyr::select(date)
paste("First time CO2 reaches 420 ppm is on", dt_st[[1]][1])
```

```
## [1] "First time CO2 reaches 420 ppm is on 2022-03-20"
```

```r
dt_end <- pred_confint %>% filter(low420 == 1) %>% dplyr::select(date)
paste("Last time CO2 reaches 420 ppm is on", dt_end[[1]][nrow(dt_end)])
```

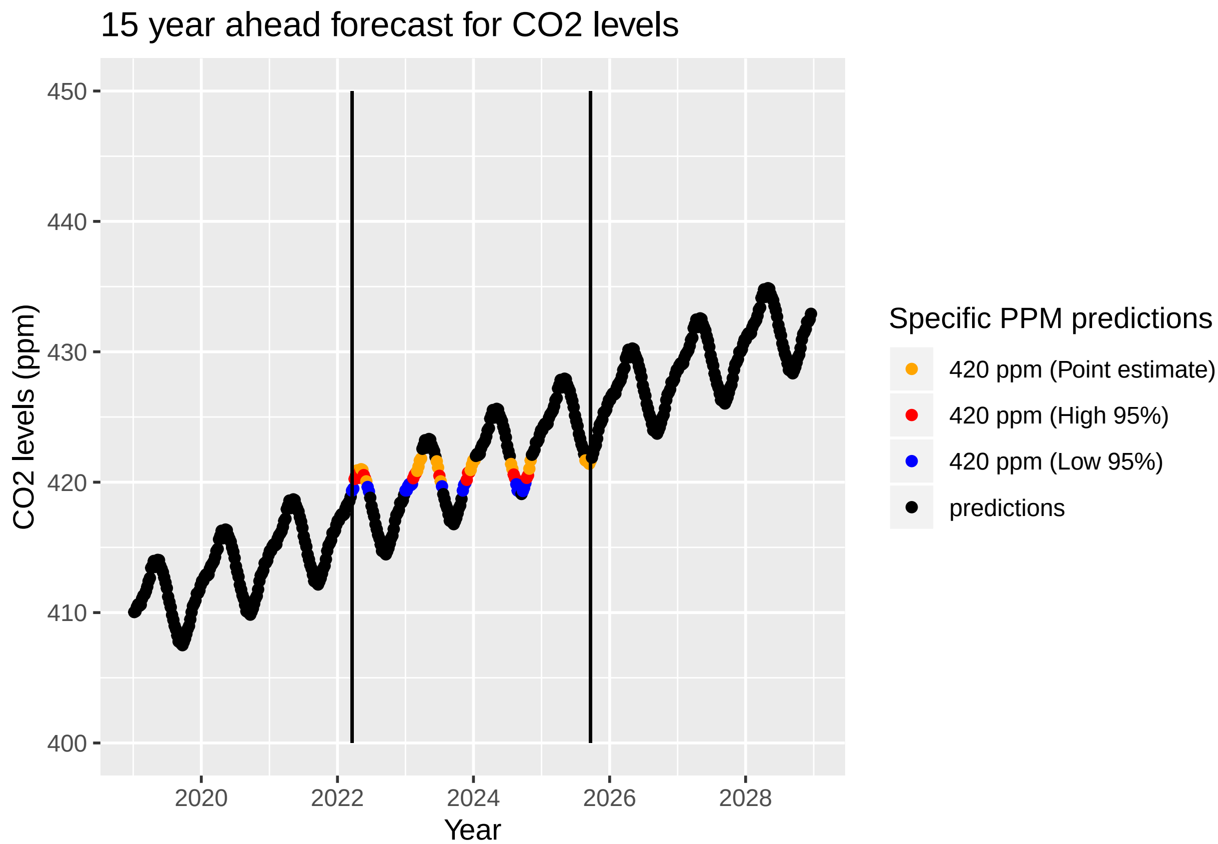
```
## [1] "Last time CO2 reaches 420 ppm is on 2025-09-21"
```

The PPM value is expected to reach 420 on $2022-04-03$ for the first time and on $2024-10-20$ for the final time. Within 95% confidence interval, the CO2 level will reach 420 on $2022-03-20$ for the first time and on $2024-10-06$ for the final time, which is about a day earlier than when the actual prediction is going to hit the same level.

From the current model, it is not possible to estimate when the PPM value would reach 500.

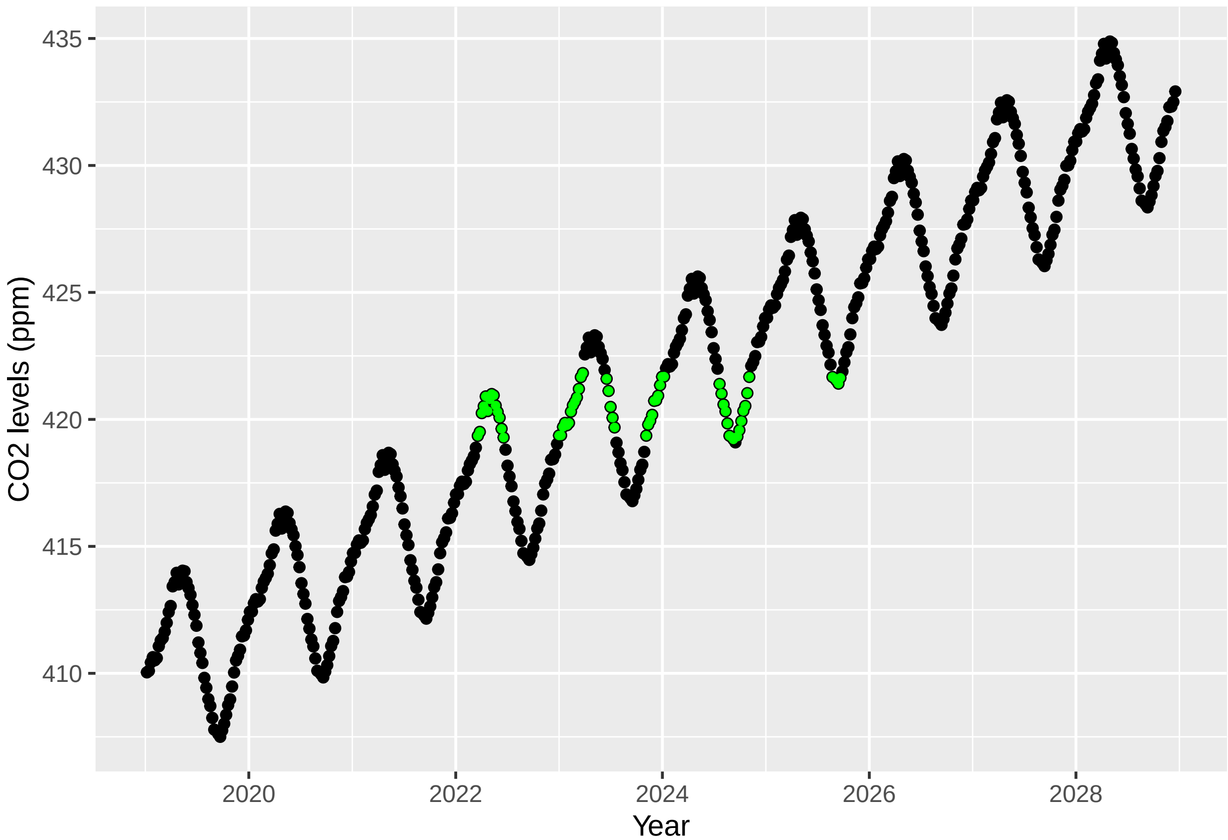
Plots for the timelines of the desired level

```r
library(ggplot2)
ggplot(pred_confint, aes(date, ppm, color = factor(colr))) +
    geom_point(na.rm = TRUE) + geom_segment(aes(x = ymd("2022-03-20"),
    y = 400, xend = ymd("2022-03-20"), yend = 450), color = "black") +
    geom_segment(aes(x = ymd("2025-09-21"), y = 400, xend = ymd("2025-09-21"),
        yend = 450), color = "black") + scale_color_manual(name = "Specific PPM predictions",
    labels = c("420 ppm (Point estimate)", "420 ppm (High 95%)",
        "420 ppm (Low 95%)", "predictions"), values = c("orange",
        "red", "blue", "black")) + xlab("Year") + ylab("CO2 levels (ppm)") +
    ggtitle("15 year ahead forecast for CO2 levels")
```



The 95% confidence interval for a interval estimate of PPM to be 420 is shows in the plot below:

```
ggplot(pred_confint, aes(date, ppm)) + geom_point(na.rm = TRUE) +
    xlab("Year") + ylab("CO2 levels (ppm)") + geom_point(data = pred_confint[pred_confint$conf
    1, ], color = "green", size = 1)
```

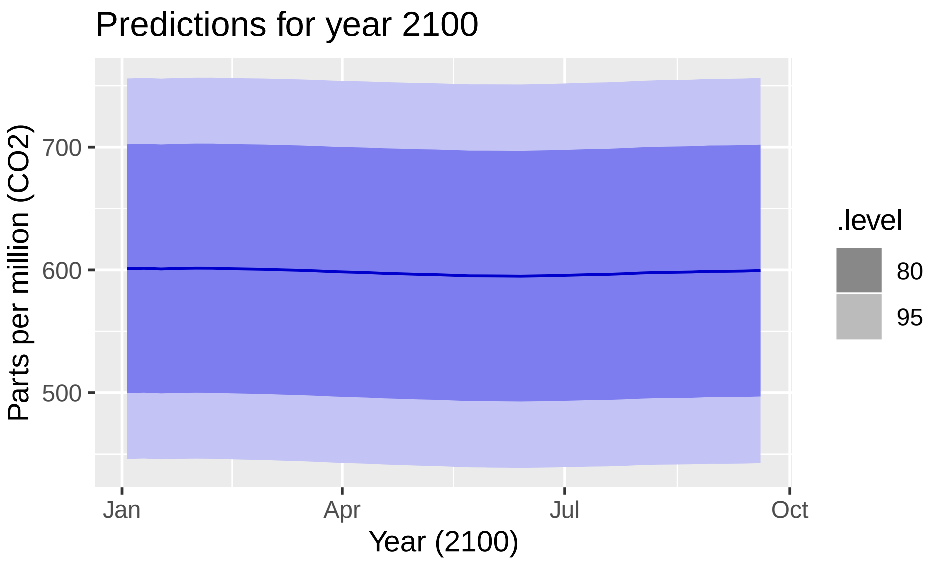


Predict CO2 levels in 2100

Year 2100 begins at 4212 weeks from the end date of the available data. We predict the CO2 levels 4212 weeks ahead until 4264 weeks ahead.

```
predictions_2100 <- forecast(nsa.ARIMA, h = 4264, level = c(95))
```

```
# get the predictions only for the year 2100
predictions_2100 <- predictions_2100 %>% filter(date >= as.Date("2100-01-01"))
predictions_2100 %>% autoplot() + ggtitle("Predictions for year 2100") +
    ylab("Parts per million (CO2)") + xlab("Year (2100)")
```



In the year 2100, the CO_2 levels are expected to be around 600 ppm. The 95% confidence interval estimate for the CO_2 levels for the year 2100 is between 450 ppm and 750 ppm.