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

Sirisha Bhupathi Shu Ying Chen Nick Cirella Luke Verdi Group 1

```r
library(dplyr)
# !diagnostics off
# covers obnoxious unnecessary dplyr warnings
library(ggplot2)
library(tidyverse)
library(GGally)
library(nnet)
library(car)
library(MASS)
library(VGAM)
library(readr)
library(lubridate)
library(forecast)
library(fable)
library(fpp2)
library(fpp3)
library(astsa)
library(imputeTS)
library(urca)
library(gridExtra)
library(caret)
library(stargazer)
library(feather)
library("ggplotify")

model_diagnostic = function(model) {

  par(mfrow = c(2,2))
  plot(model)

  residualPlots(model)

}
```

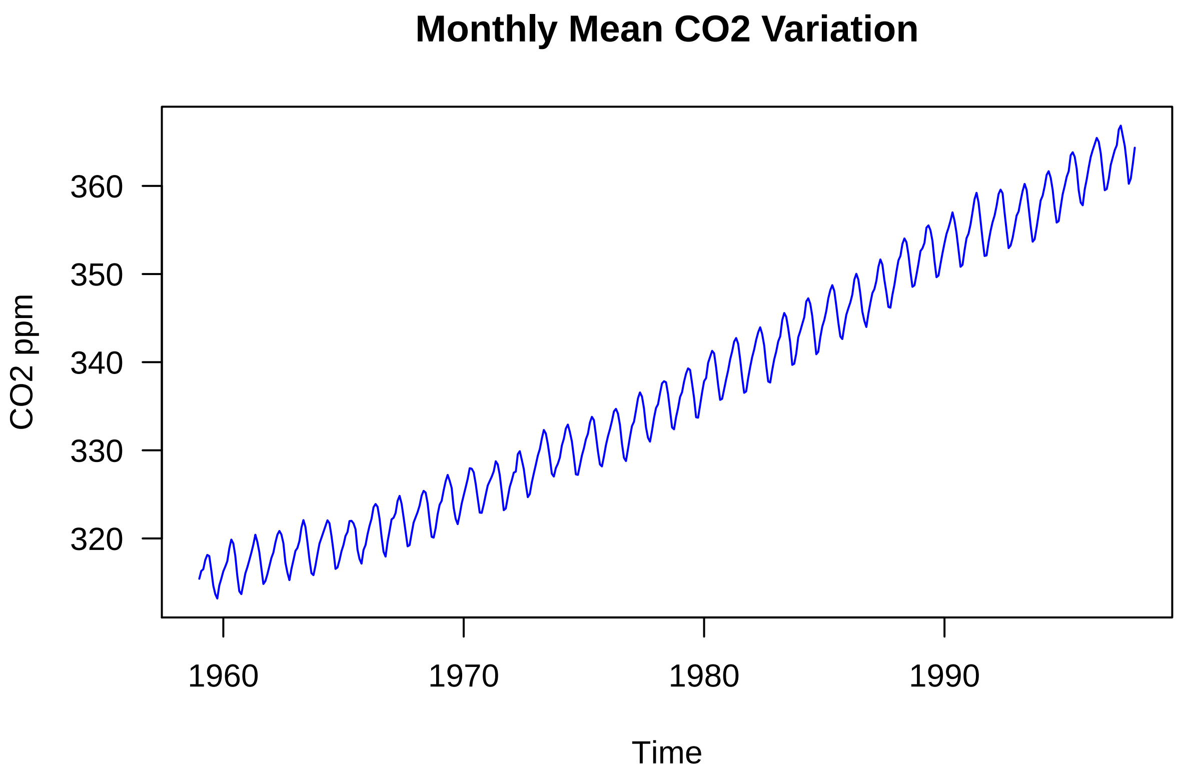
The Keeling Curve

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'.

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



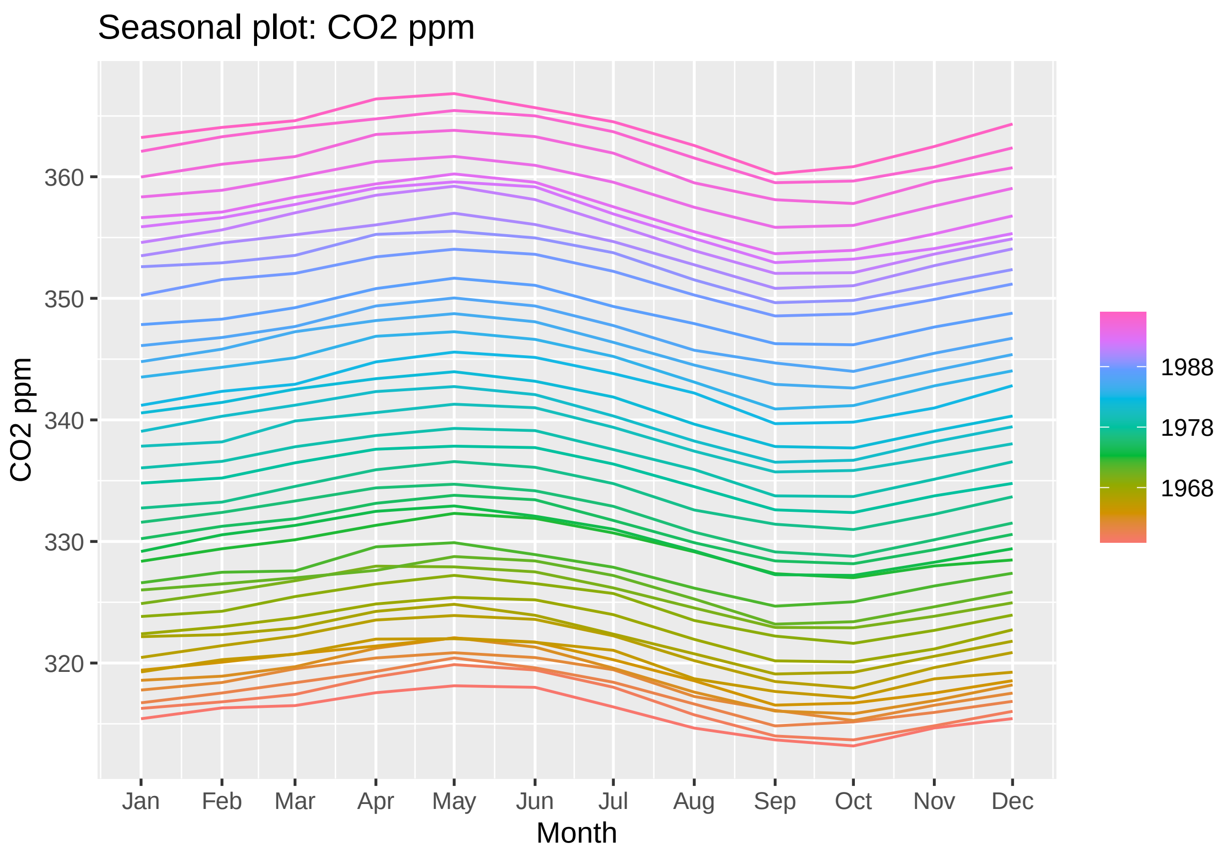
Part 1 (3 points)

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

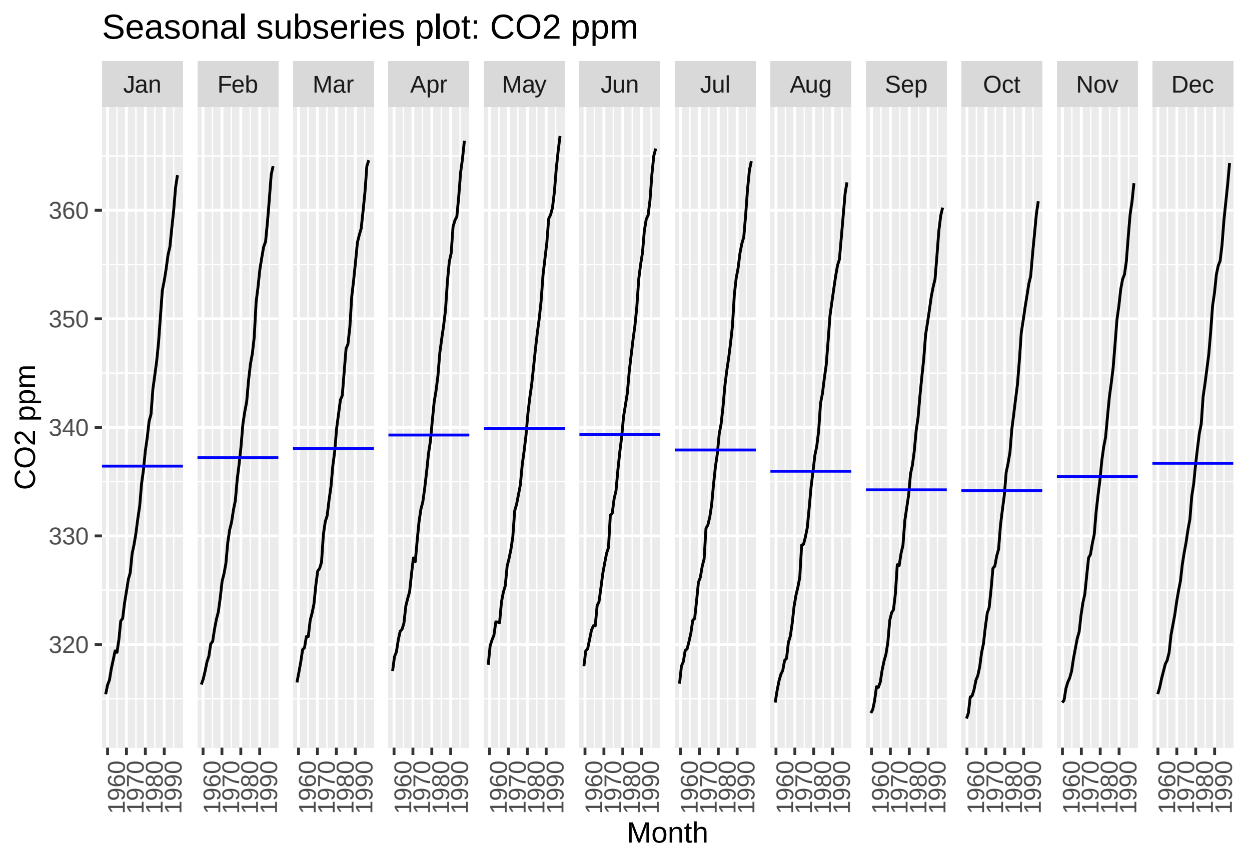
We start our analysis looking at the plot of the data over time above. We see clear general trend upwards, almost linear. Seasonal component is robust, highest in May and lowest in October every year. While the data appears generally linear, there also may be a slight non-linear component that seems apparent around 1970 and again around 1990 with slight shifts in the slope of the average co2 level.

We next look at seasonality trends looking at the data by year and by month.

```
co2_tsibble <- co2 %>% as_tsibble() %>% rename(CO2_ppm = value, Date = index)
co2_tsibble %>%
  gg_season(CO2_ppm) +
  ylab("CO2 ppm") + xlab('Month') +
  ggtitle("Seasonal plot: CO2 ppm")
```



```
co2_tsibble %>%
  gg_subseries(CO2_ppm) +
  ylab("CO2 ppm") + xlab('Month') +
  ggtitle("Seasonal subseries plot: CO2 ppm")
```

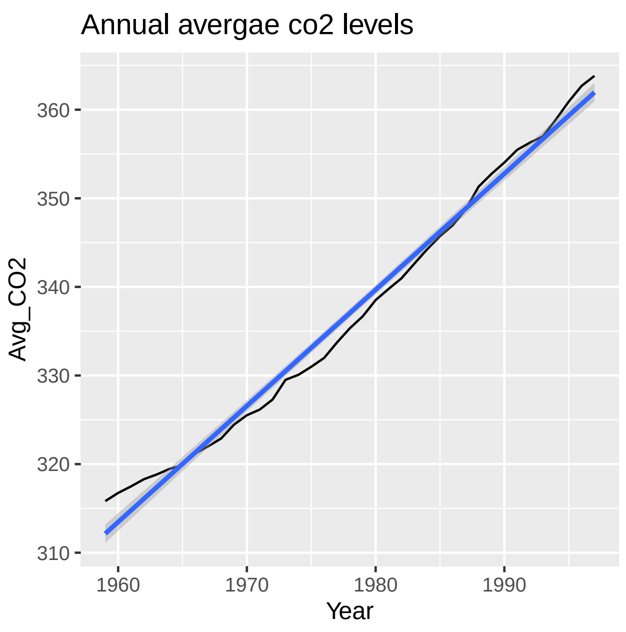


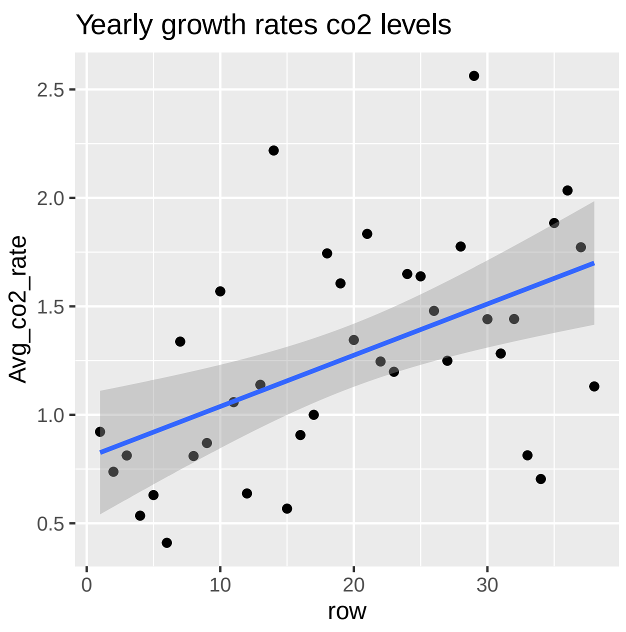
The above charts show clear seasonality with increases in May/June and decreases in September/October. The annual increase is also apparent from the colored graph.

This is further justified by the seaonal and treng graphs below. However the below also shows the anticipated non-linear elements around 1970 and 1990. The irregualr elements further support this thesis. We should and will bear this in mind when evaluating ARIMA models later in our analysis.

```
co2_agg <- co2_tsibble %>%
  mutate(Year = year(Date)) %>%
  as.data.frame() %>%
  group_by(Year) %>%
  summarise(Avg_CO2 = mean(CO2_ppm), .groups = "keep") %>%
  ungroup()
co2_agg_rate = data.frame("Avg_co2_rate" = diff(co2_agg$Avg_CO2), "row" = seq(1,38,1))
p1 = ggplot(co2_agg, aes(x = Year, y = Avg_CO2)) + geom_line() + geom_smooth(method = "lm") + g
p2 = ggplot(co2_agg_rate, aes(x = row, y = Avg_co2_rate)) + geom_point() + geom_smooth(method =
grid.arrange(p1, p2,  ncol=2, nrow=1)
```

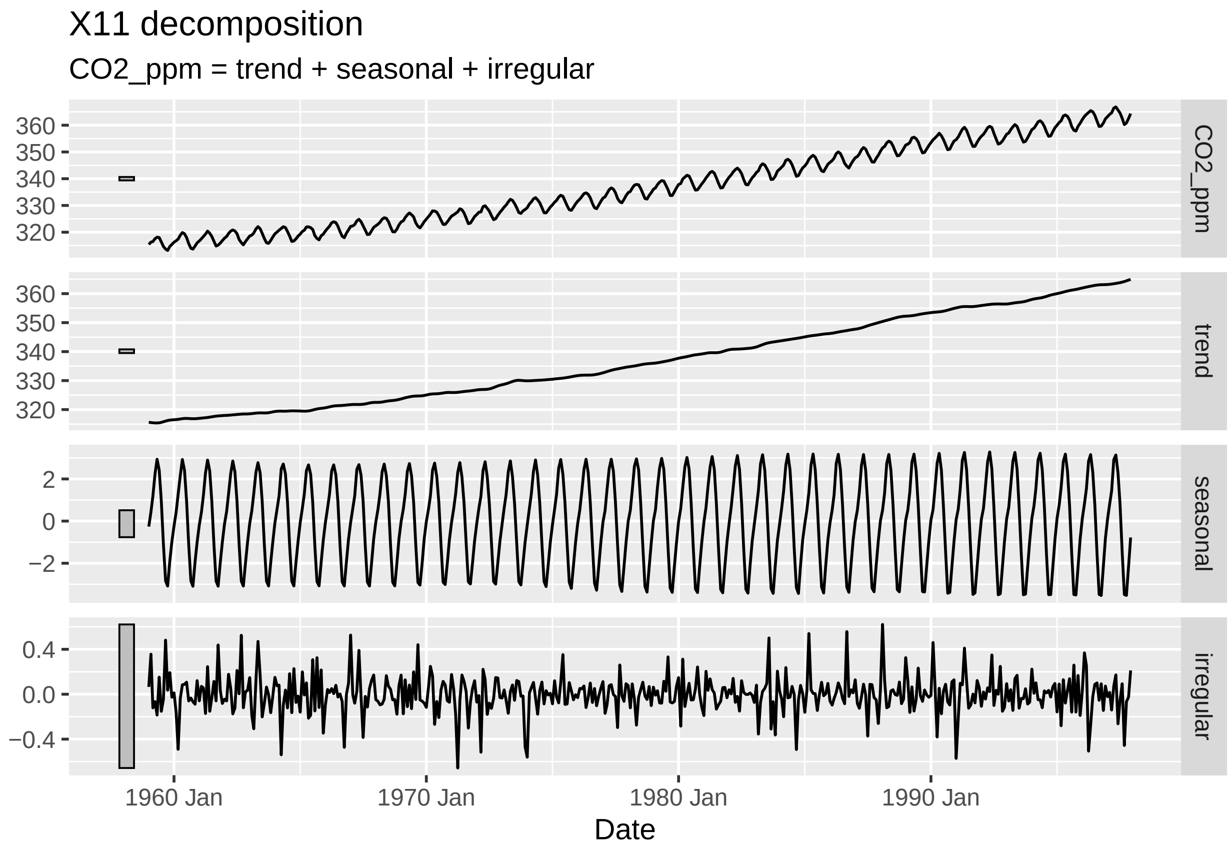
```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```





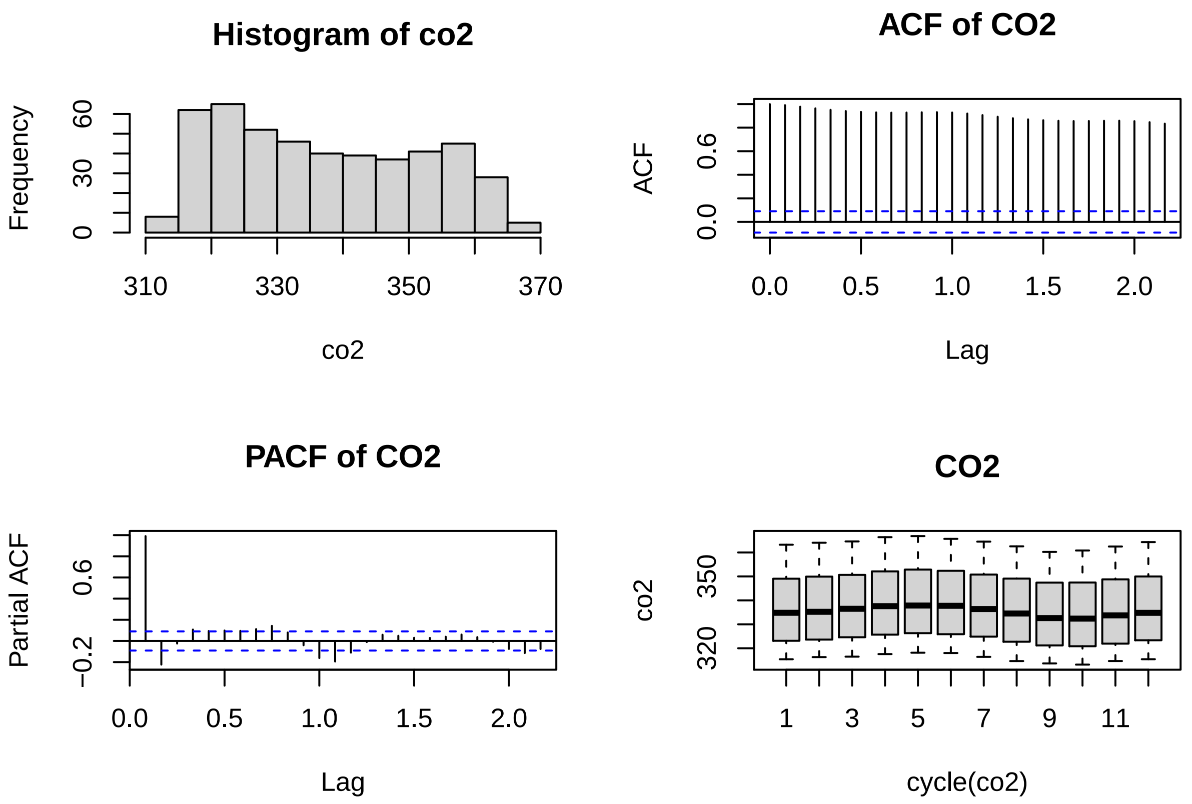
We see that not only the co2 levels have been increasing every year, but no average the year over year growth rate of the co2 levels has also increase over the years. co2 levels are both increasing and are increasing at faster every year.

```
co2_tsibble %>% model(x11 = feasts:::X11(CO2_ppm, type = "additive")) %>% components() %>% auto
```



Finally, we can take a look at ACF and PACF, along with the frequences of various co2 levels to confirm our analysis. This is done through the charts below.

```r
par(mfrow=c(2,2))
hist(co2)
acf(co2, main="ACF of CO2")
pacf(co2, main="PACF of CO2")
boxplot(co2 ~ cycle(co2), main="CO2")
```



Once again we see sesonality and a general upward trend. The undulations in the ACF and PACD also indicate potential higher order AR and MA models, with the irregular elements suggesting we also evaluate an integrated component/random walk. We can say that a an ARIMA model is indicated and may provide valuable insight into the data.

Part 2 (3 points)

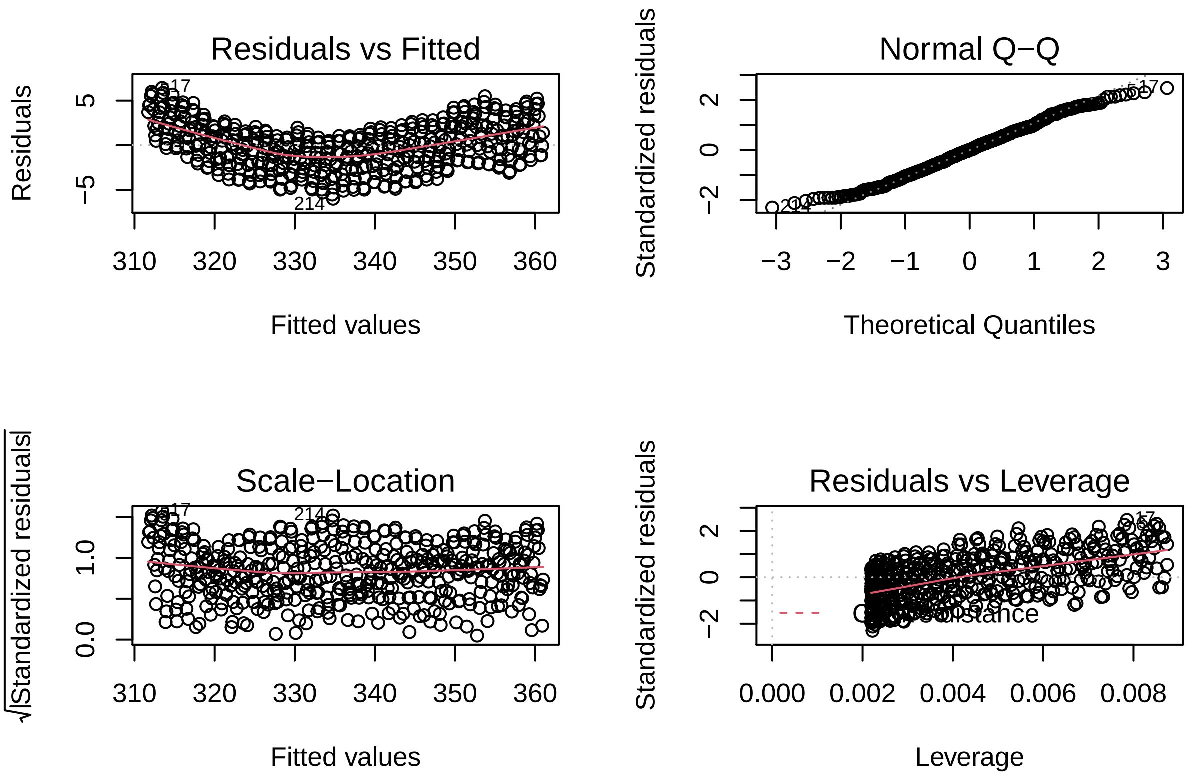
Fit a linear time trend model to the `co2` series, and examine the characteristics of the residuals. Compare this to a higher-order polynomial time trend model. Discuss whether a logarithmic transformation of the data would be appropriate. Fit a polynomial time trend model that incorporates seasonal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

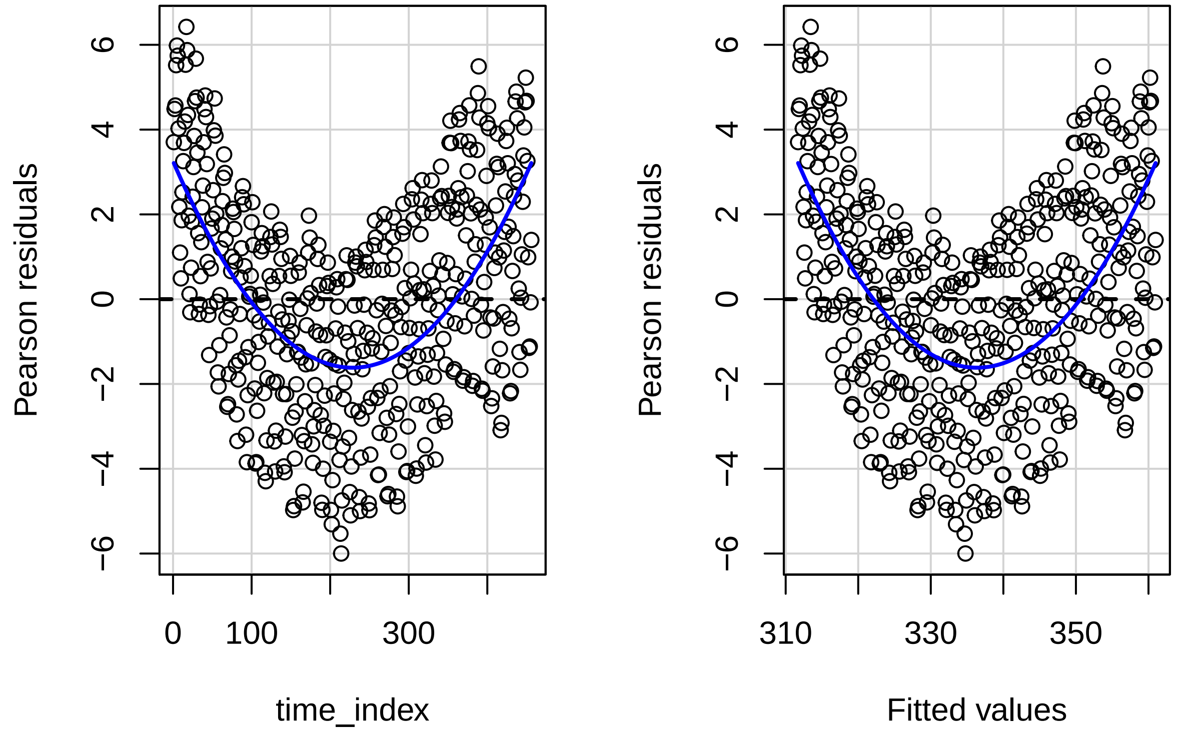
First, owing to the requirement in later questions (predicting between 1997 and 2020) we will consider only timeseries upto 1996. Then we fit a linear model to the data and plot the residuals.

```
co2_1996 = window(co2, end = c(1996, 12))
df = data.frame(co2_1996)
df <- df %>% mutate(time_index = row_number())
colnames(df) <- c("co2", "time_index")
# co2_cust <- co2 %>% as_tsibble()
# co2_cust <- co2_cust %>% mutate(time_index = row_number())
linear.trend.fit <- lm(co2 ~ time_index, data=df)
linear.trend.fit
```

```
##
## Call:
## lm(formula = co2 ~ time_index, data = df)
##
## Coefficients:
## (Intercept)   time_index
##    311.6068       0.1083
```

```
model_diagnostic(linear.trend.fit)
```





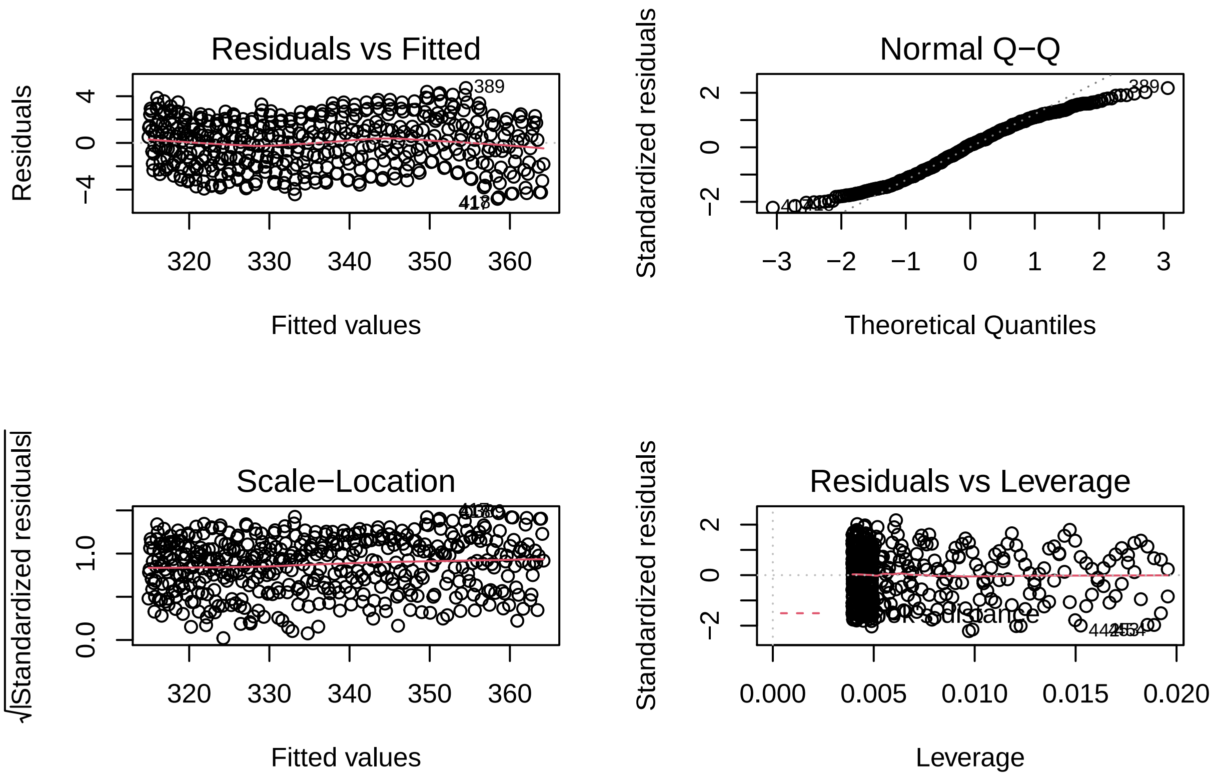
```
##              Test stat Pr(>|Test stat|)
## time_index    14.212        < 2.2e-16 ***
## Tukey test    14.212        < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

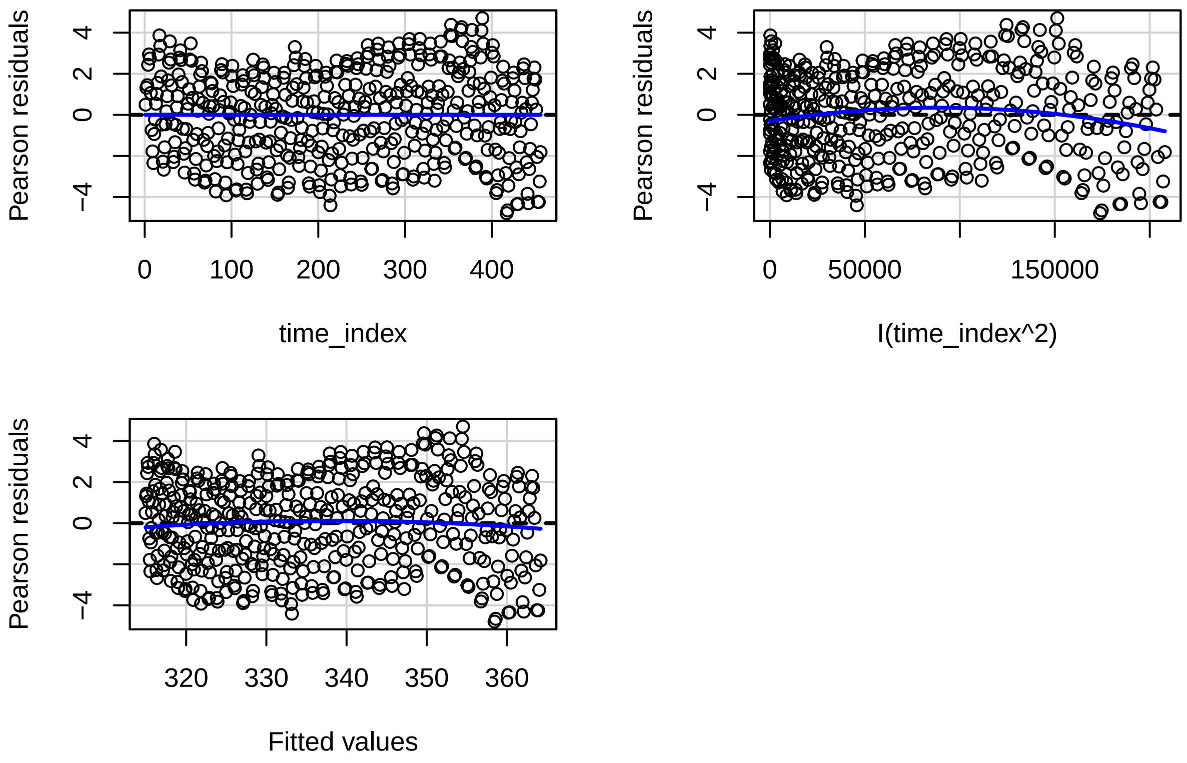
Having applied the liner model, we now evaluate the quadratic model and see th at the trend and residuals flattend out around a 0 mean, which we would have expected given the parabolic trendline from the linear model above.

```r
quadratic.trend.fit <- lm(co2 ~ time_index + I(time_index^2), data=df)
quadratic.trend.fit
```

```
##
## Call:
## lm(formula = co2 ~ time_index + I(time_index^2), data = df)
##
## Coefficients:
##     (Intercept)       time_index  I(time_index^2)
##       3.149e+02        6.566e-02        9.326e-05
```

```r
model_diagnostic(quadratic.trend.fit)
```





```
##                   Test stat Pr(>|Test stat|)
## time_index          -0.4284           0.6686
## I(time_index^2)     -5.4694        7.486e-08 ***
## Tukey test          -5.4032        6.546e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

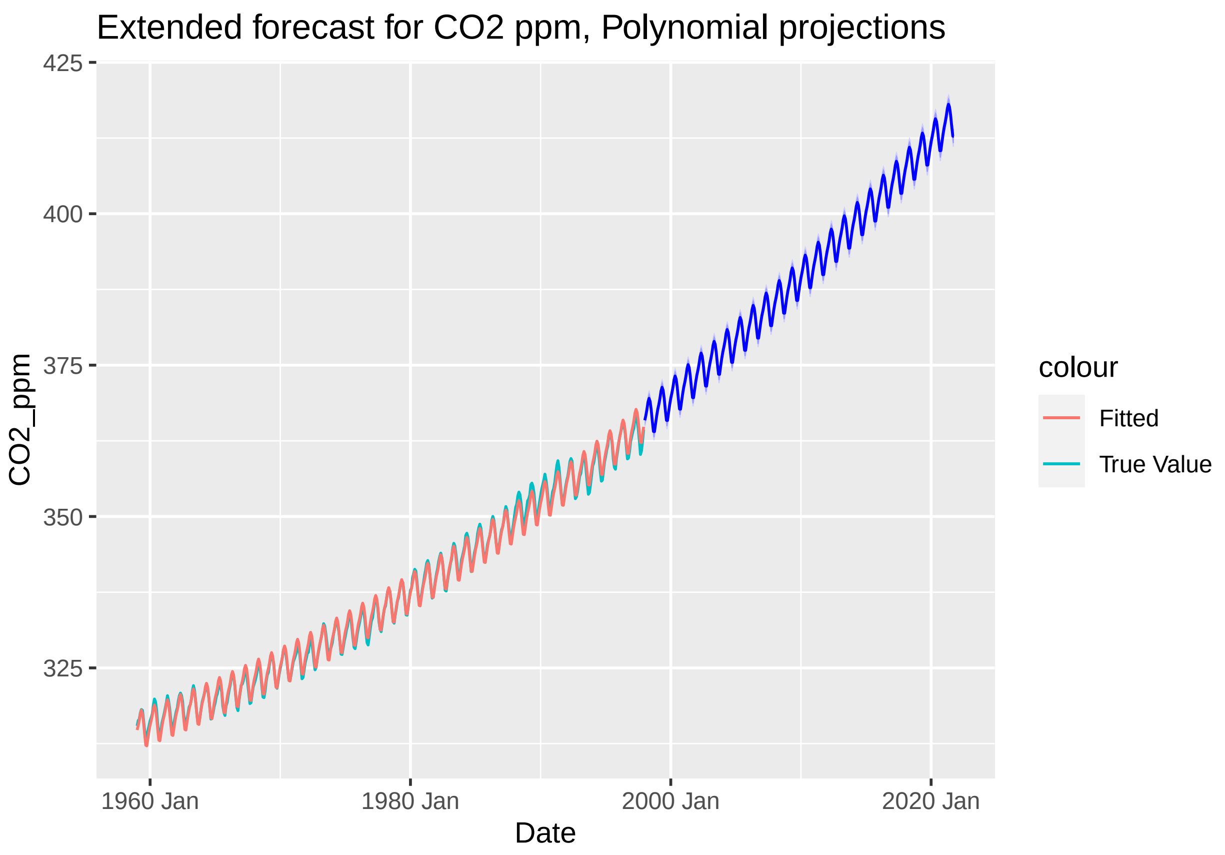
The residual plots look quite reasonable. They are distributed faily equally above and below horizontal axis, with the mean line remaining more or less straight. Based on this, there doesn't seem to be any need to apply a log transformation.

Finally we apply the polynomial model and use it to predict out to 2020.

```
polynomial.tslm.fit <- co2_tsibble %>% model(TSLM(CO2_ppm ~ trend() + I(trend()^2) + season()))
```

Now for predicting up to 2020, we can use the forecast. We will also plot this with the map of the model against the known data:

```
augment(polynomial.tslm.fit) %>%
  ggplot(aes(x = Date)) +
  autolayer(forecast(polynomial.tslm.fit,h=286)) +
  geom_line(aes(y = CO2_ppm, colour = "True Value")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(x = "Date", y = "CO2_ppm") +
  ggtitle('Extended forecast for CO2 ppm, Polynomial projections')
```



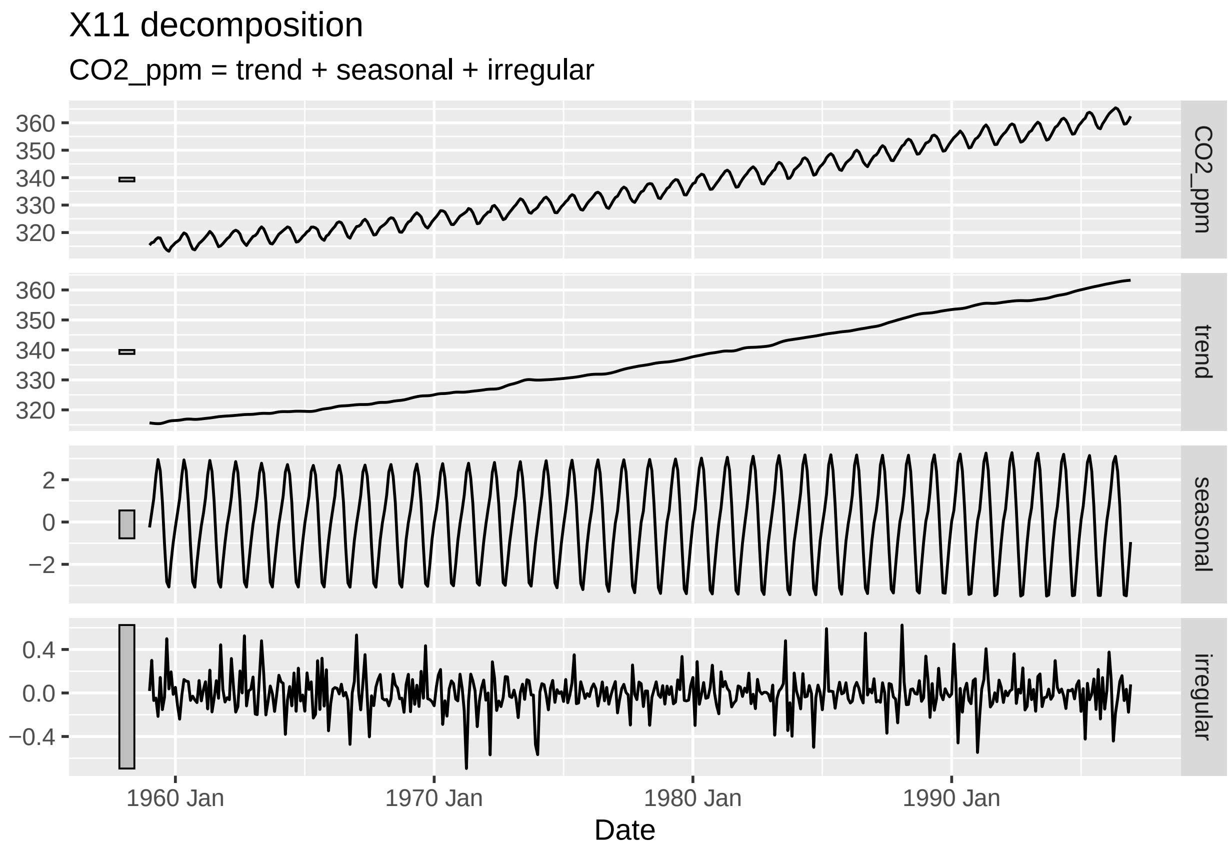
We see that the fitted model doesn't fit exactly, but generally does well, and the future predictions seem reasonable.

Part 3 (3 points)

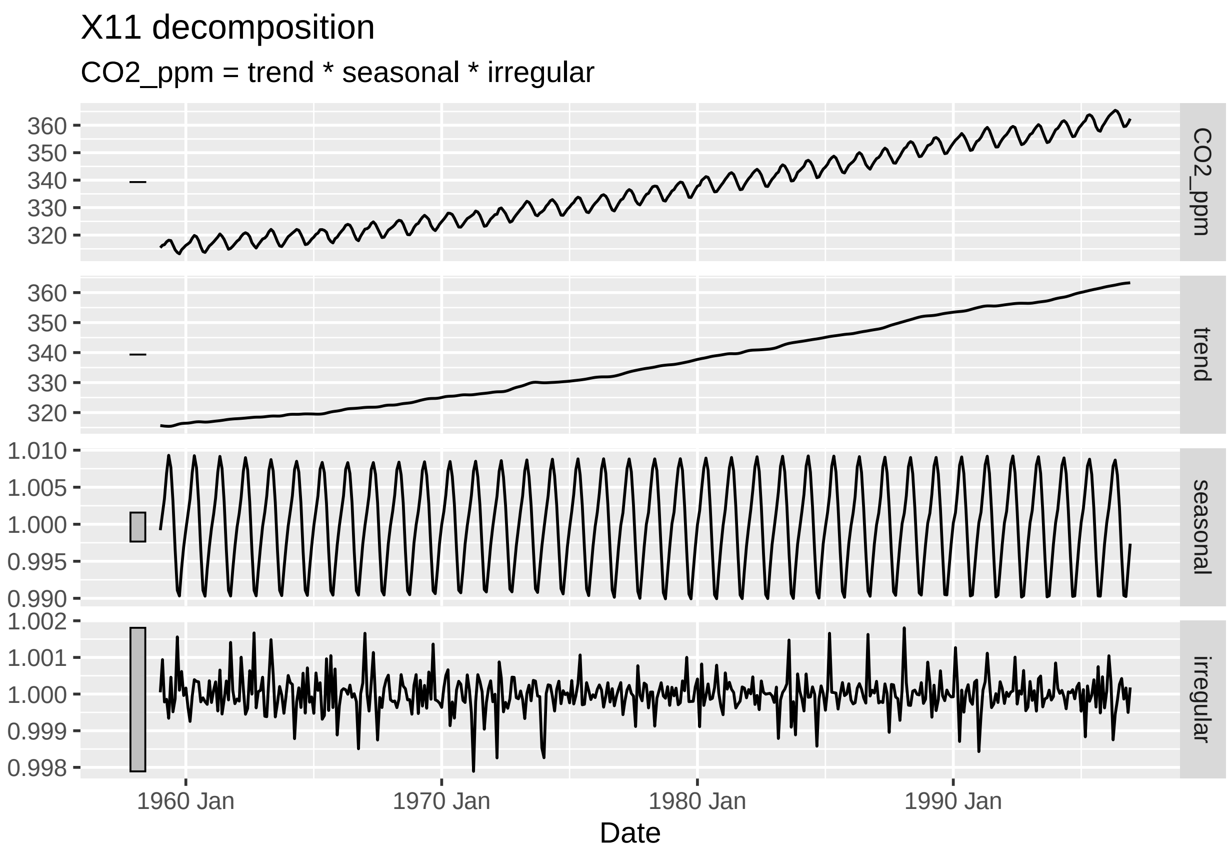
Following all appropriate steps, choose an ARIMA model to fit to the series. Discuss the characteristics of your model and how you selected between alternative ARIMA specifications. Use your model (or models) to generate forecasts to the year 2020.

Before choosing an ARIMA model, we first look back to the decomposition of the data into its trend, seasonal and radom components using the X11 decomposition method that we exained in our EDA above.

```
# Compare additive and multiplicative decompositions
co2_tsibble <- co2_1996 %>% as_tsibble() %>% rename(CO2_ppm = value, Date = index)
co2_tsibble %>% model(x11 = feasts:::X11(CO2_ppm, type = "additive")) %>% components() %>% auto
```

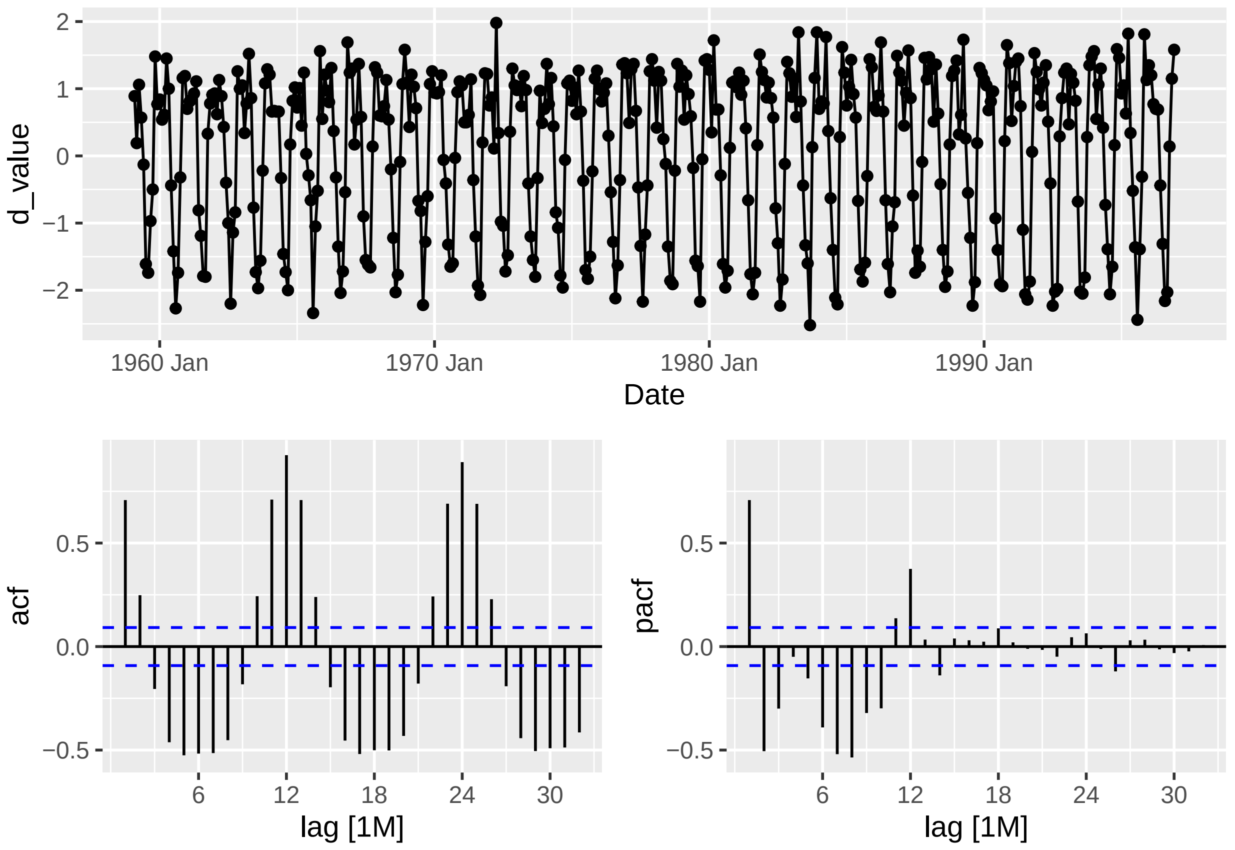


```
co2_tsibble %>% model(x11 = feasts:::X11(CO2_ppm, type = "multiplicative")) %>% components() %
```



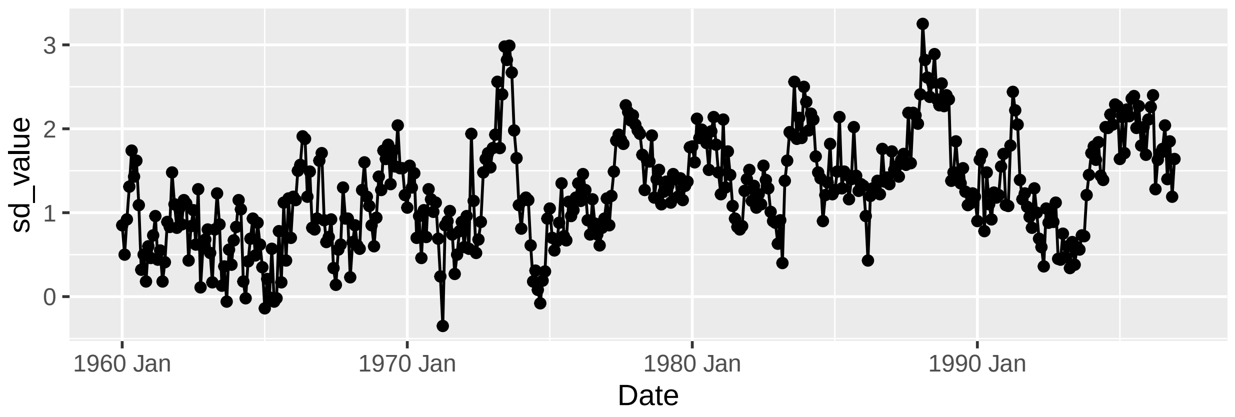
The trend component indicates an increasing mean over time, so we will apply differencing to stablize the mean. The seasonal component indicates stable variance over time. That confirms no need for logarithmic transformation.

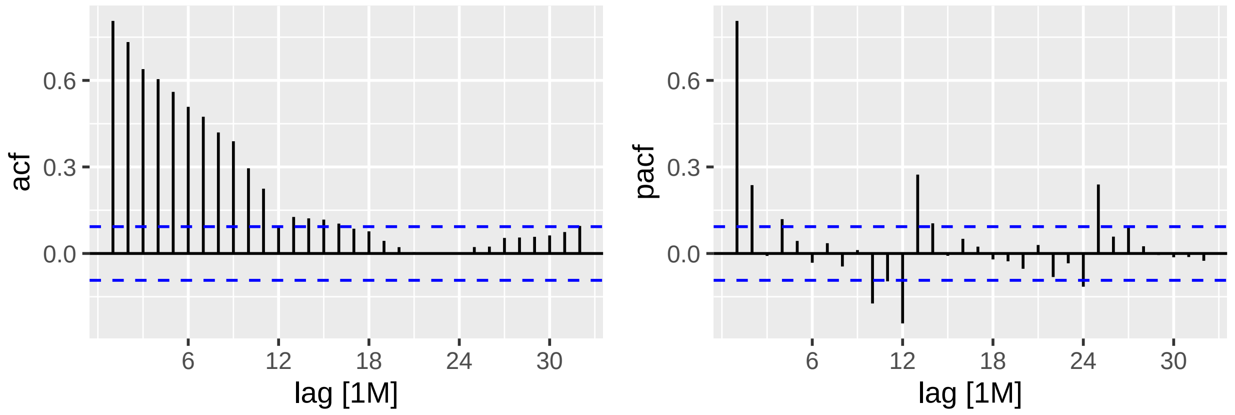
```
# Apply first-differencing to stabilize the mean
co2_tsibble <- co2_tsibble %>% mutate(d_value = difference(CO2_ppm, 1))
co2_tsibble %>%
  filter(!is.na(d_value)) %>%
  gg_tsdisplay(y = d_value, plot = 'partial', lag_max = 32)
```



The first-order differencing removes the increasing pattern for the times series. However, the ACF shows an obvious seasonal autocorrelation still exist after first-differencing is applied. Hence, we next try taking the seasonal differencing of lag 12

```r
# Apply seasonal differencing to stabilize the mean
co2_tsibble <- co2_tsibble %>% mutate(sd_value = difference(CO2_ppm, 12))
co2_tsibble %>%
  filter(!is.na(sd_value)) %>%
  gg_tsdisplay(y = sd_value, plot = 'partial', lag_max = 32)
```





The plot shows the mean of the seasonal differencing does not always stay within a similar range over time. The ACF gradually decays and becomes insignificant after lag 12. The pacf drops sharply after lag 2. These observations suggests that the series still exhibit non-stationarity. Furthermore, under the KPSS unit root test, null hypotheses of stationarity would be rejected for the seasonal-differenced series.

```r
co2_tsibble %>% features(d_value, unitroot_kpss)
```

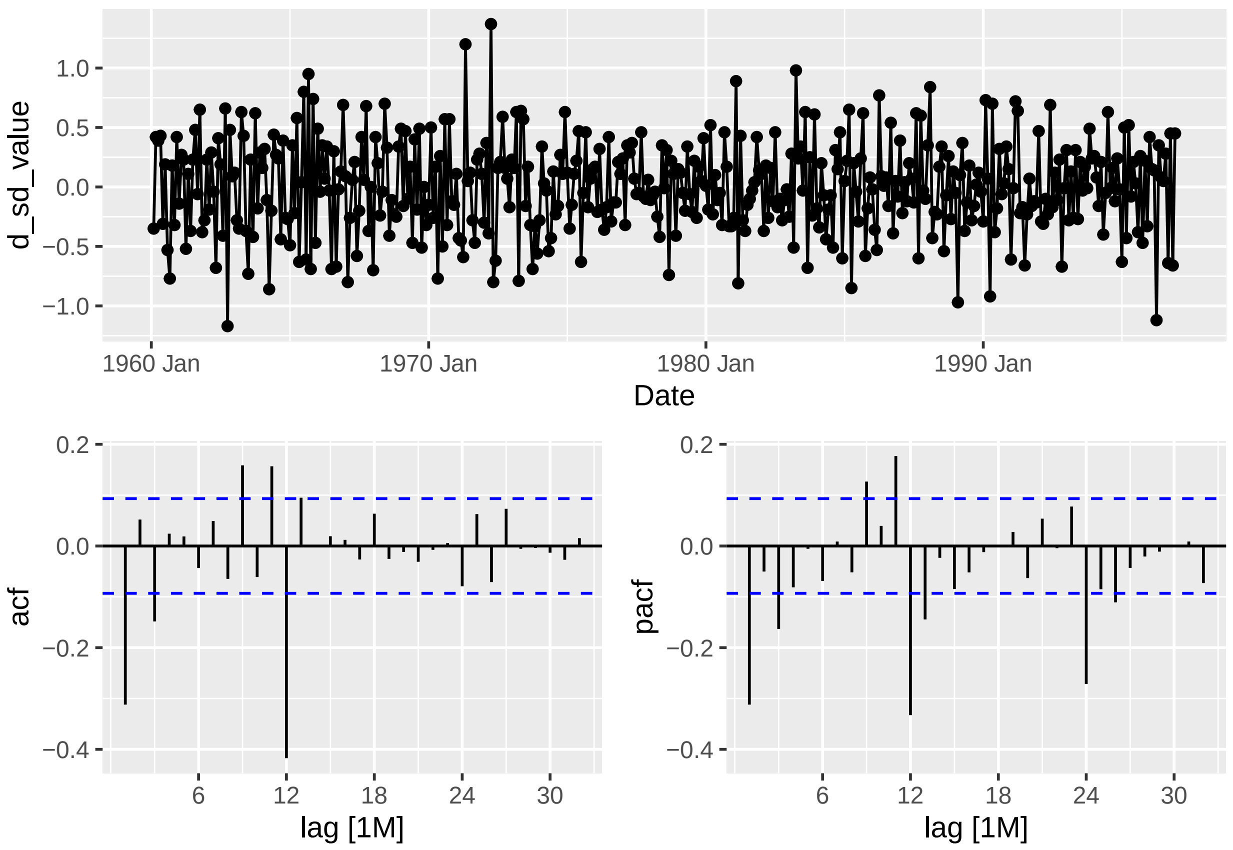
```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0115         0.1
```

```r
co2_tsibble %>% features(sd_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      2.07        0.01
```

Hence, we should try taking the first-order differencing on top of seasonal differencing.

```
# Apply first order differencing on top of seasonal differencing
co2_tsibble <- co2_tsibble %>% mutate(d_sd_value = difference(sd_value, 1))
co2_tsibble %>%
  filter(!is.na(d_sd_value)) %>%
  gg_tsdisplay(y = d_sd_value, plot = 'partial', lag_max = 32)
```



```
co2_tsibble %>% features(d_sd_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0112         0.1
```

```
co2_tsibble %>% features(CO2_ppm, unitroot_ndiffs)
```

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

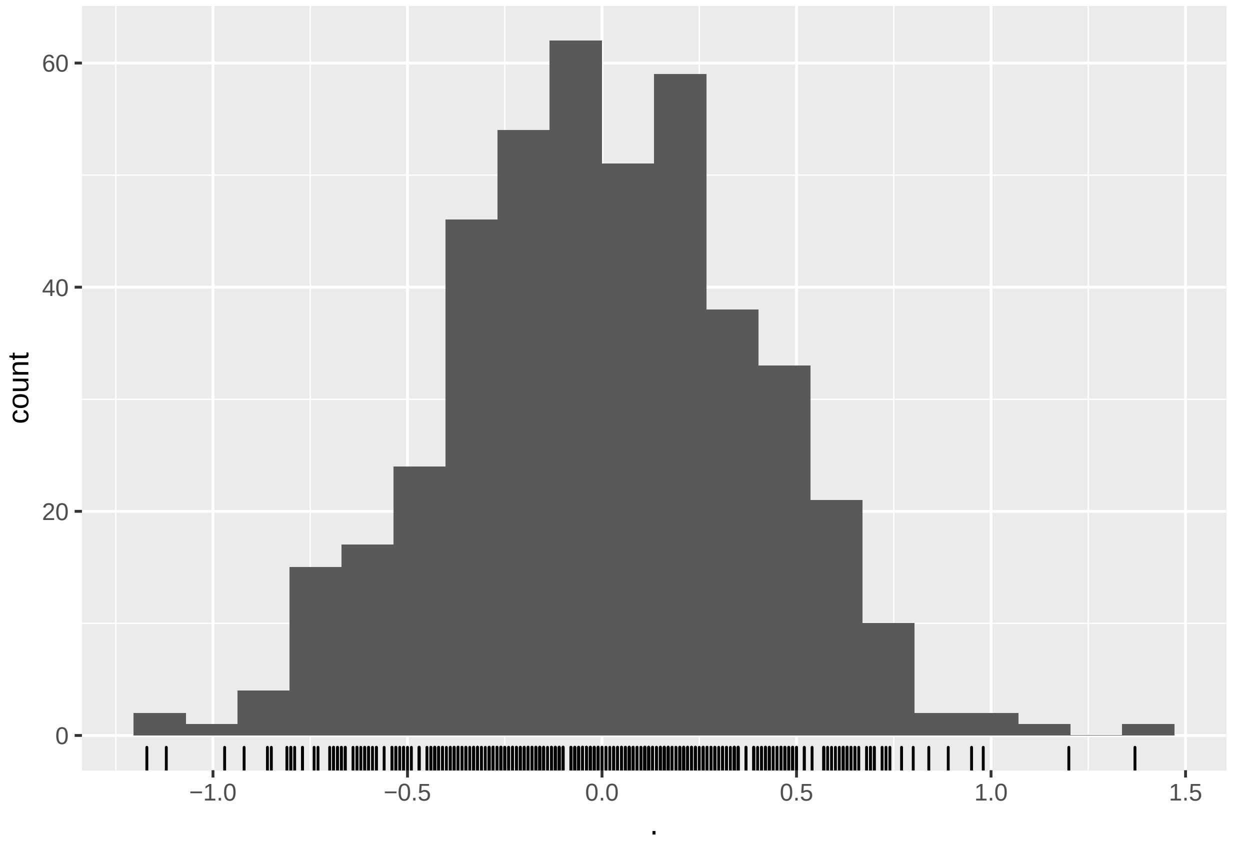
```r
co2_tsibble %>% features(CO2_ppm, unitroot_nsdiffs)
```

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

Per the above, the kpss test statistic is small and within the range we would expect for stationary data, so we can conclude that the series is stationary. Also by running unitroot_ndiff and unitroot_nsdiffs, we deterimine that the appropriate number of first difference is 1 and the number of seasonal difference is 1.

Moreover, the histogram of the differenced series (d=1 and D=1) looks like a Normal distribution.

```r
co2_tsibble %>% filter(!is.na(d_sd_value)) %>% pull(d_sd_value) %>% gghistogram()
```



Now that it has been determined a first difference and seasonal difference is needed, we can study the ACF and the PACF of the properly difference model further (this is the plot above the histogram above). We see some seasonality in the ACF as there is a large spike at period 12 to go along with the additional spikes at 1 and 3. While the ACF is generally trending downwards, there is perhaps some indication that a higher order AR model is appropriate. The PACF shows periodic spikes occuring every 12 periods, indicating that the seasonlity is at 12 period intervals. The general trend is towards 0 for the PCF, but with some increases around 1 and 2, again appearing to indicate a higher level. Therefore, we will experiment with several different models of varying orders.

We will evaluate models by seeking to minimize the AIC or the BIC. We've written a helper function here to perform this loop, trying p, q, P and Q values from 0 to 2, as it will be helpful again in later sections. Running the loop will give us two models to discuss further, before ultimately selecting a desired model. All models we evalute will include the first differencing and seasonal differencing determined above.

Note that we've run this loop in advance and am reading from the saved file, just to expedite the process in creating the markdown file and not have to repeat this computation excessively.

```
#loop for models
test_ARIMA <- function(tsibble, max_params, differences, period) {
  # max_params in form of c(p_max,P_max,q_max,Q_max)

    models <- tibble(model = 1) %>%
      crossing(p = 0:max_params[1], q = 0:max_params[2],
               P = 0:max_params[3], Q = 0:max_params[4]) %>%
      mutate(model = 1:nrow(.),
             AIC = 0,
             BIC = 0)

  for (i in 1:nrow(models)) {

    tryCatch({
        suppressWarnings(
          mod <- tsibble %>% model(ARIMA(data ~  0 +
                            pdq(models$p[i],differences[1],models$q[i]) +
                            PDQ(models$P[i],differences[2],models$Q[i], period=period)))
        )
        if(has_name(glance(mod),'AIC')){

          models$AIC[i] <- glance(mod)$AIC
          models$BIC[i] <- glance(mod)$BIC

        }
      })
  }

  rbind(models %>%
          filter(AIC != 0) %>%
          arrange(AIC) %>%
          head(1),
        models %>%
          filter(AIC != 0) %>%
          arrange(BIC) %>%
          head(1))
}
# part3_arima <- test_ARIMA(dplyr::select(co2_tsibble, data = CO2_ppm), c(3,3,3,3), c(1,1), 12
# write_feather(part3_arima, 'Part 3 ARIMA Results.feather')
```

```
read_feather('Part 3 ARIMA Results.feather')
```

```
## # A tibble: 2 x 7
##   model     p     q     P     Q   AIC   BIC
##   <int> <int> <int> <int> <int> <dbl> <dbl>
## 1    59     0     3     2     2  158.  191.
## 2    18     0     1     0     1  165.  177.
```

We can also use the auto arima function to get a sense of the best model:

```
auto.arima(ts(co2_tsibble$CO2_ppm,frequency = 12,start = c(1959,1)))
```

```
## Series: ts(co2_tsibble$CO2_ppm, frequency = 12, start = c(1959, 1))
## ARIMA(1,1,1)(1,1,2)[12]
##
## Coefficients:
##          ar1      ma1     sar1     sma1     sma2
##       0.2694  -0.5922  -0.5384  -0.2702  -0.5116
## s.e.  0.1393   0.1183   0.6196   0.5988   0.5074
##
## sigma^2 estimated as 0.08406:  log likelihood=-77.64
## AIC=167.28   AICc=167.47   BIC=191.84
```

Based on the above, we can compare across the 3 models (what was recommended by minimizing AIC, minimizing BIC and by auto.arima).

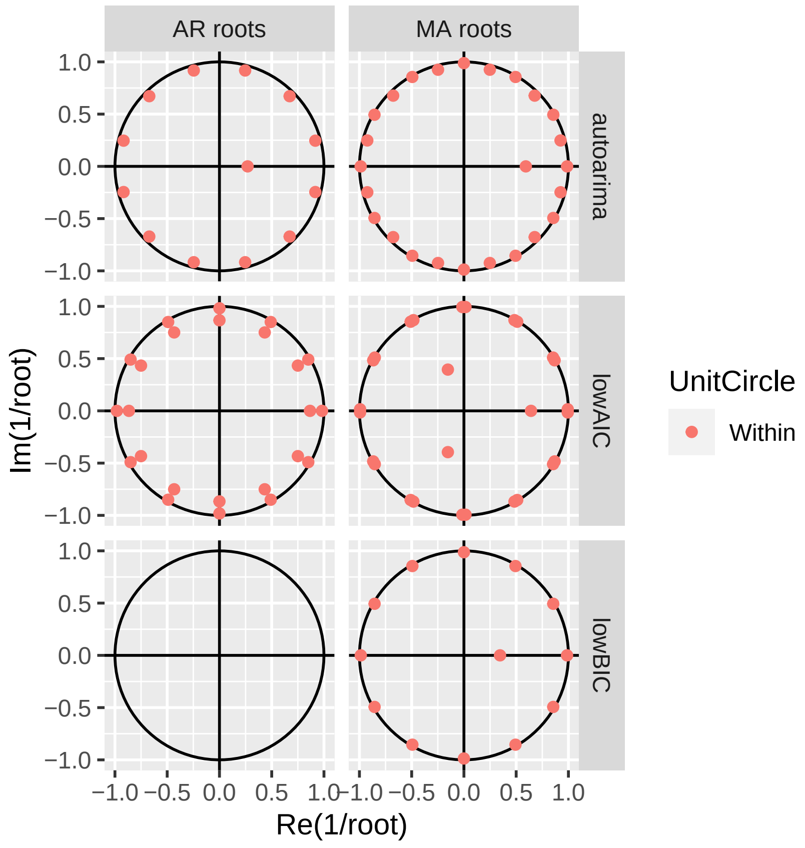
```
models <- co2_tsibble %>% model(lowBIC = ARIMA(CO2_ppm ~ 0 + pdq(0,1,1) + PDQ(0,1,1, period=12
                                lowAIC = ARIMA(CO2_ppm ~ 0 + pdq(0,1,3) + PDQ(2,1,2, period=12
                                autoarima = ARIMA(CO2_ppm ~ 0 + pdq(1,1,1) + PDQ(1,1,2, period

glance(models)
```

```
## # A tibble: 3 x 8
##   .model    sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 lowBIC    0.0842   -79.6  165.  165.  177. <cpl [0]>  <cpl [13]>
## 2 lowAIC    0.0810   -71.1  158.  158.  191. <cpl [24]> <cpl [27]>
## 3 autoarima 0.0841   -77.6  167.  167.  192. <cpl [13]> <cpl [25]>
```

```
models %>% accuracy()
```

```
## # A tibble: 3 x 9
##   .model    .type        ME  RMSE   MAE     MPE   MAPE  MASE     ACF1
##   <chr>     <chr>     <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>
## 1 lowBIC    Training 0.0151 0.285 0.227 0.00441 0.0677 0.179  0.0258
## 2 lowAIC    Training 0.0109 0.278 0.223 0.00329 0.0664 0.176  0.0121
## 3 autoarima Training 0.0177 0.284 0.228 0.00516 0.0678 0.179 -0.00464
```

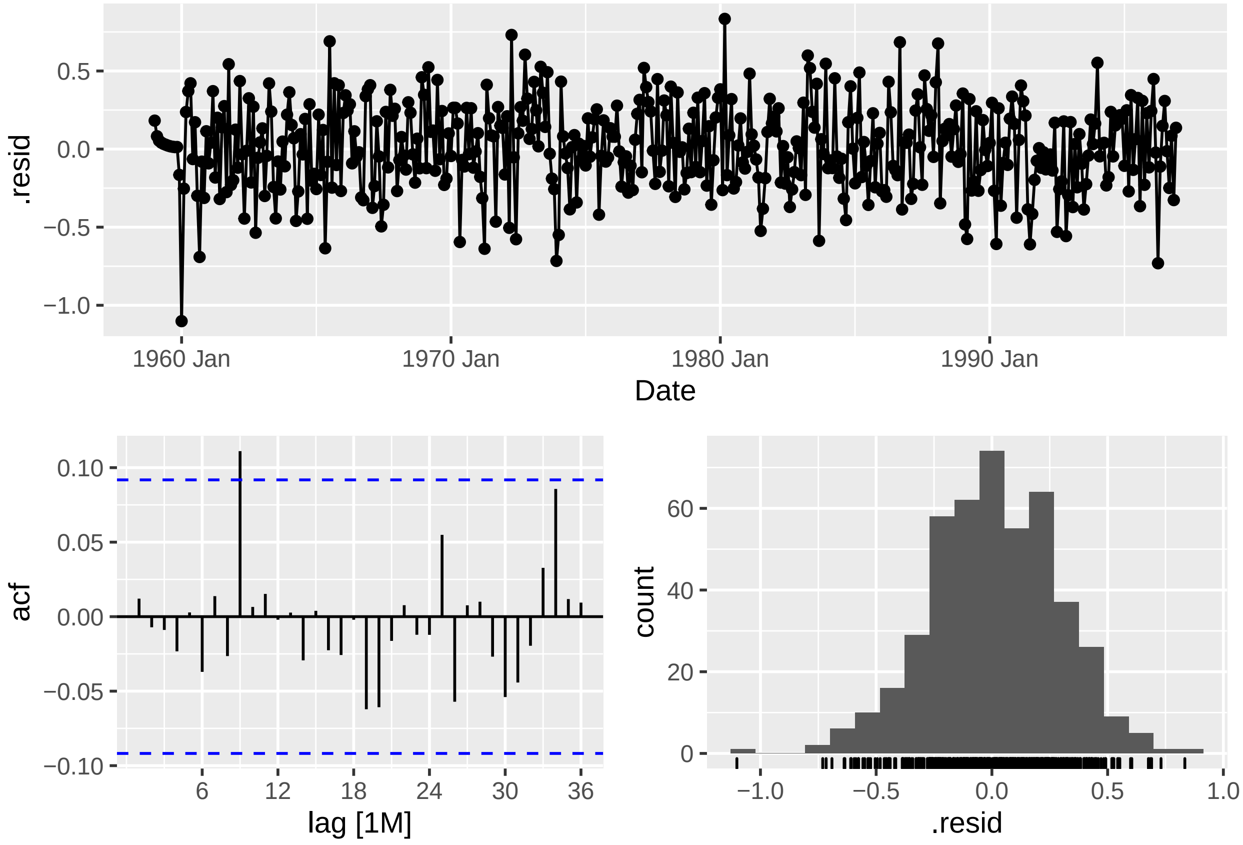
```
gg_arma(models)
```



While each of these models performs well in some aspect, we can see that the lowAIC model produces a more optimal log-likelihood, lowest AIC and 2nd lowest BIC. It also minimizes RMSE, and incorporates both AR and MA terms as we predicted was necessary. Especially since we're able to verify that all roots are within the unit circle, we'll chose this model moving forward.

Now with this model, we'll validate that the residuals look normally distributed and random.

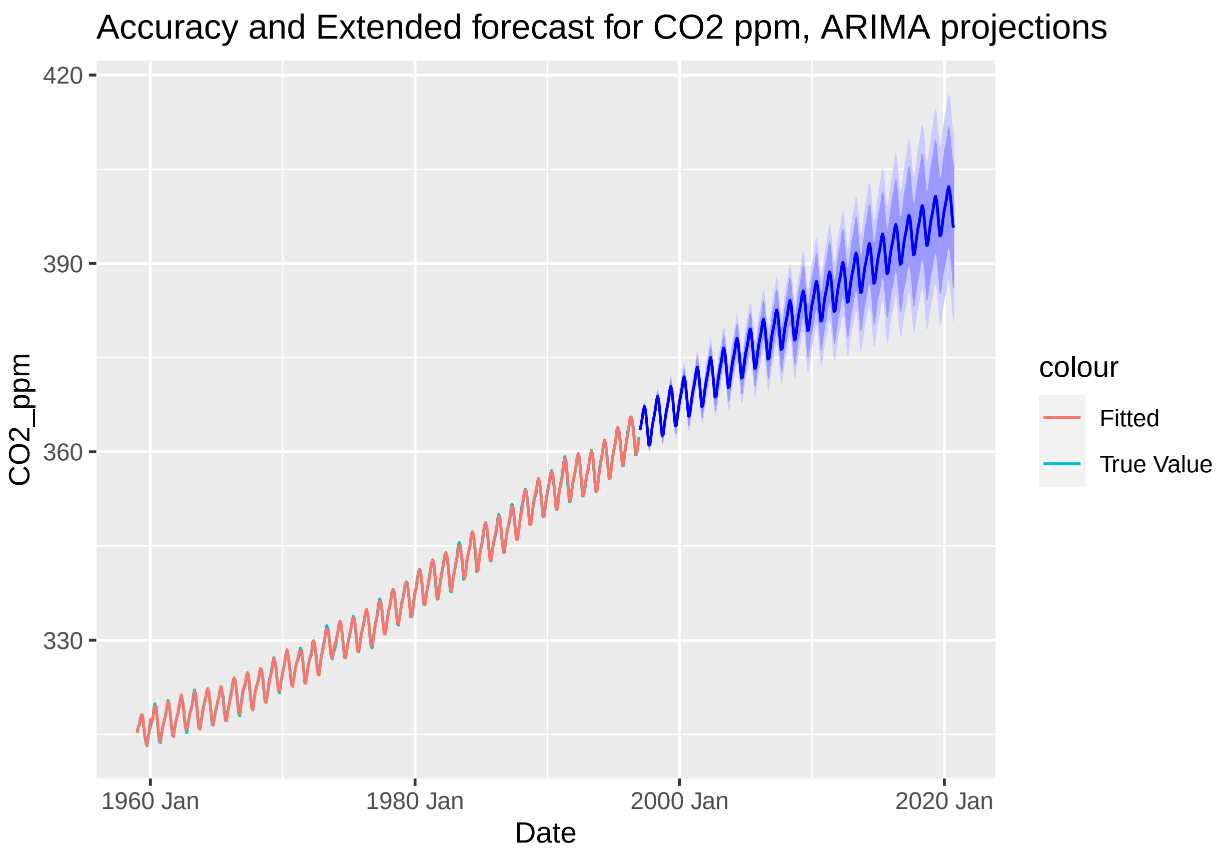
```
selected.arima.mod <- co2_tsibble %>% model(lowAIC_ARIMA = ARIMA(CO2_ppm ~ 0 + pdq(0,1,3) + PD
selected.arima.mod %>% gg_tsresiduals(lag_max = 36)
```



We see above this model generates an approximately normal distribution of residuals, and almost all ACF terms are now insignificant. The one term barely achieving significance at lag 9 doesn't seem to be strongly worth adding to the model.

Below we use this model to forecast out to 2020.

```r
augment(selected.arima.mod) %>%
  ggplot(aes(x = Date)) +
  autolayer(forecast(selected.arima.mod,h=286)) +
  geom_line(aes(y = CO2_ppm, colour = "True Value")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(x = "Date", y = "CO2_ppm") +
  ggtitle('Accuracy and Extended forecast for CO2 ppm, ARIMA projections')
```


Accuracy and Extended forecast for CO2 ppm, ARIMA projections
420
390
360
330
CO2_ppm
1960 Jan
1980 Jan
2000 Jan
2020 Jan
Date
colour
Fitted
True Value

Part 4 (4 points)

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 this to the predictions from your forecasts in Parts 2 and 3. Use the weekly data to generate a month-average series from 1997 to the present, and use this to generate accuracy metrics for the forecasts generated by your models from Parts 2 and 3.

First, we will read in the data from the given .txt file and create a tsibble.

```r
# read data
co2_weekly_mlo = read_delim('co2_weekly_mlo.txt', delim = ' ', skip = 49,
                            col_names = c('Year', 'Month', 'Day', 'Date Decimal', 'CO2_ppm',
                                          'Days', 'CO2 ppm Year ago', 'CO2 ppm 10 Years ago',
                                          'Increase since 1800'),
                            col_types = 'nnnnnnnnn')

# create tsibble
co2_weekly_tsibble <- co2_weekly_mlo %>%
  mutate(Date = as.Date(paste(Year, Month, Day, sep = '/'))) %>%
  dplyr::select(Date, CO2_ppm) %>%
  as_tsibble(index = Date)
```

Now we can begin to conduct a thorough EDA on the tsibble data.

```r
summary(co2_weekly_tsibble)
```

```
##       Date                 CO2_ppm
##  Min.   :1974-05-19   Min.   :-1000.0
##  1st Qu.:1985-12-23   1st Qu.:  346.7
##  Median :1997-07-30   Median :  364.5
##  Mean   :1997-07-30   Mean   :  356.2
##  3rd Qu.:2009-03-06   3rd Qu.:  386.9
##  Max.   :2020-10-11   Max.   :  417.4
```

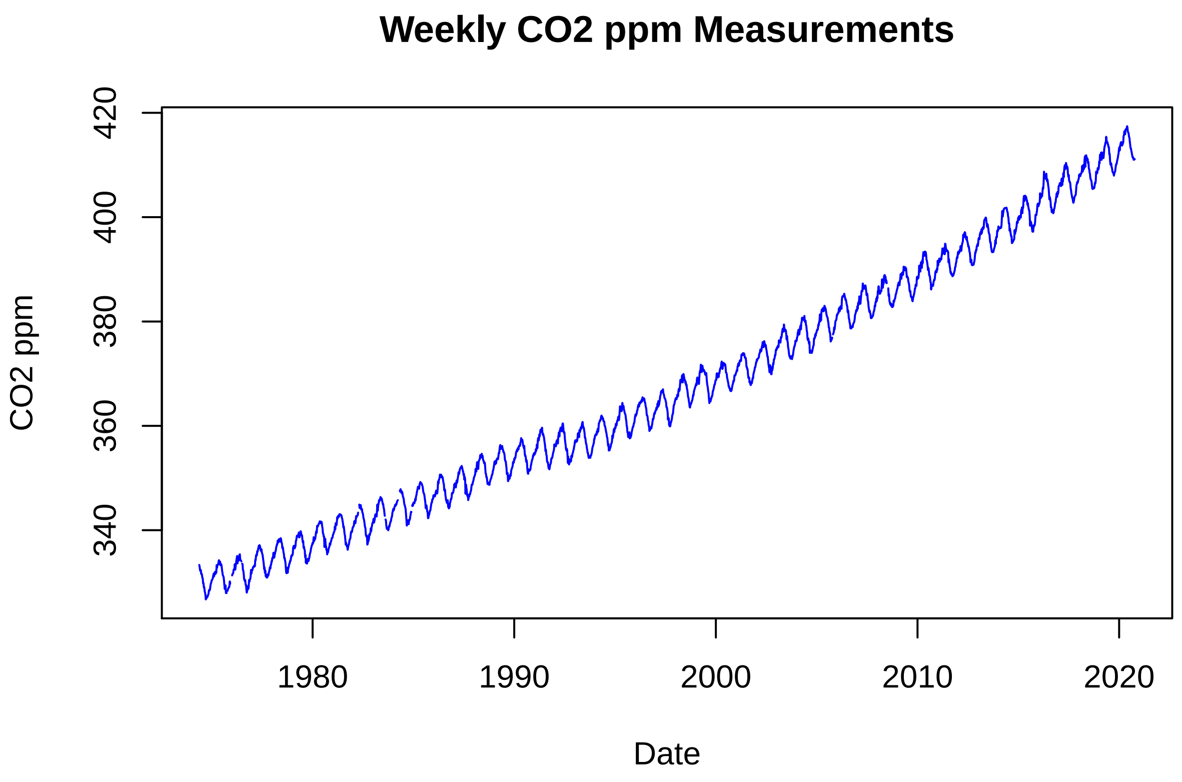
We see the data runs from 1974 to 2020. Most values look reasonable, but there are certainly some -1000 values that need to be further investigated.

```r
co2_weekly_tsibble %>% filter(CO2_ppm == min(CO2_ppm)) %>% head(5)
```

```
## # A tsibble: 5 x 2 [7D]
##   Date       CO2_ppm
##   <date>       <dbl>
## 1 1975-10-05  -1000.
## 2 1975-12-07  -1000.
## 3 1975-12-14  -1000.
## 4 1975-12-21  -1000.
## 5 1975-12-28  -1000.
```

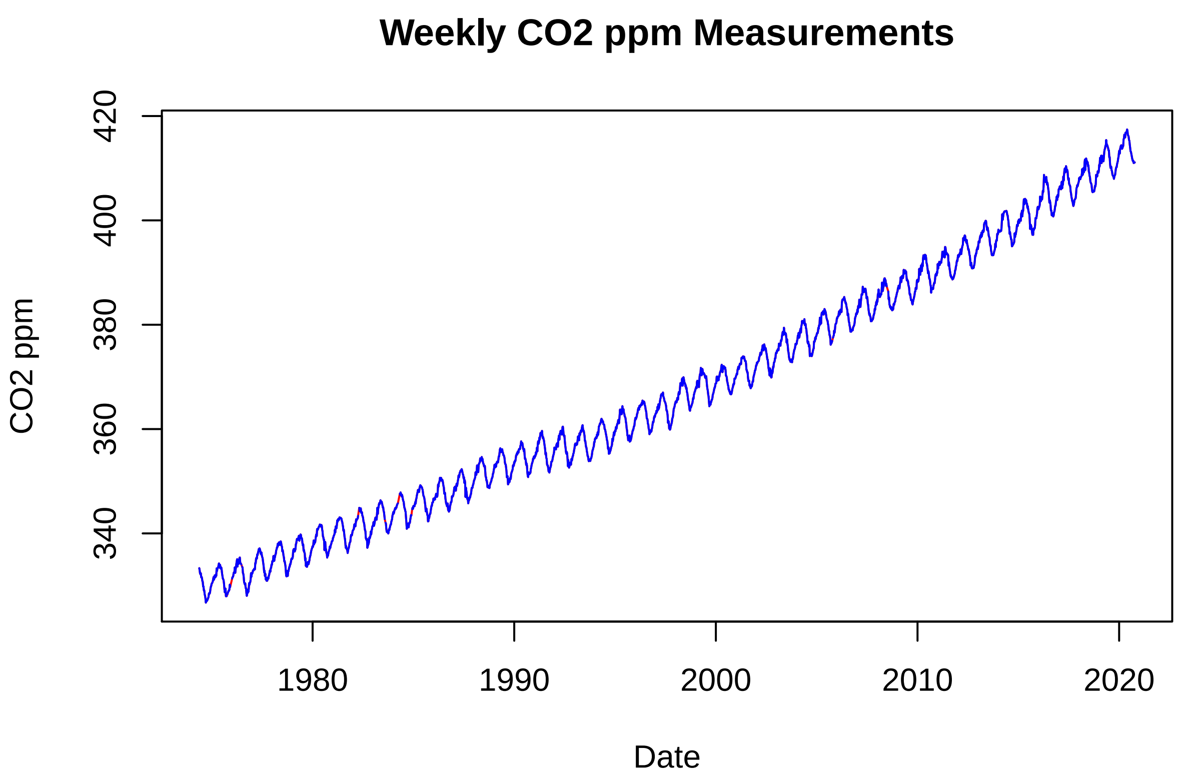
We see 20 weeks that we are missing, and we note that some of these missing values are consecutive, which will effect how we can remedy them. First, let's set the values to NA and construct a plot to understand the trends of the data.

```
co2_weekly_tsibble <- co2_weekly_tsibble %>%
  mutate(CO2_ppm = ifelse(CO2_ppm == min(CO2_ppm), NA, CO2_ppm))

plot(co2_weekly_tsibble, ylab = expression("CO2 ppm"), col = 'blue', type = 'l',
     main = 'Weekly CO2 ppm Measurements')
```



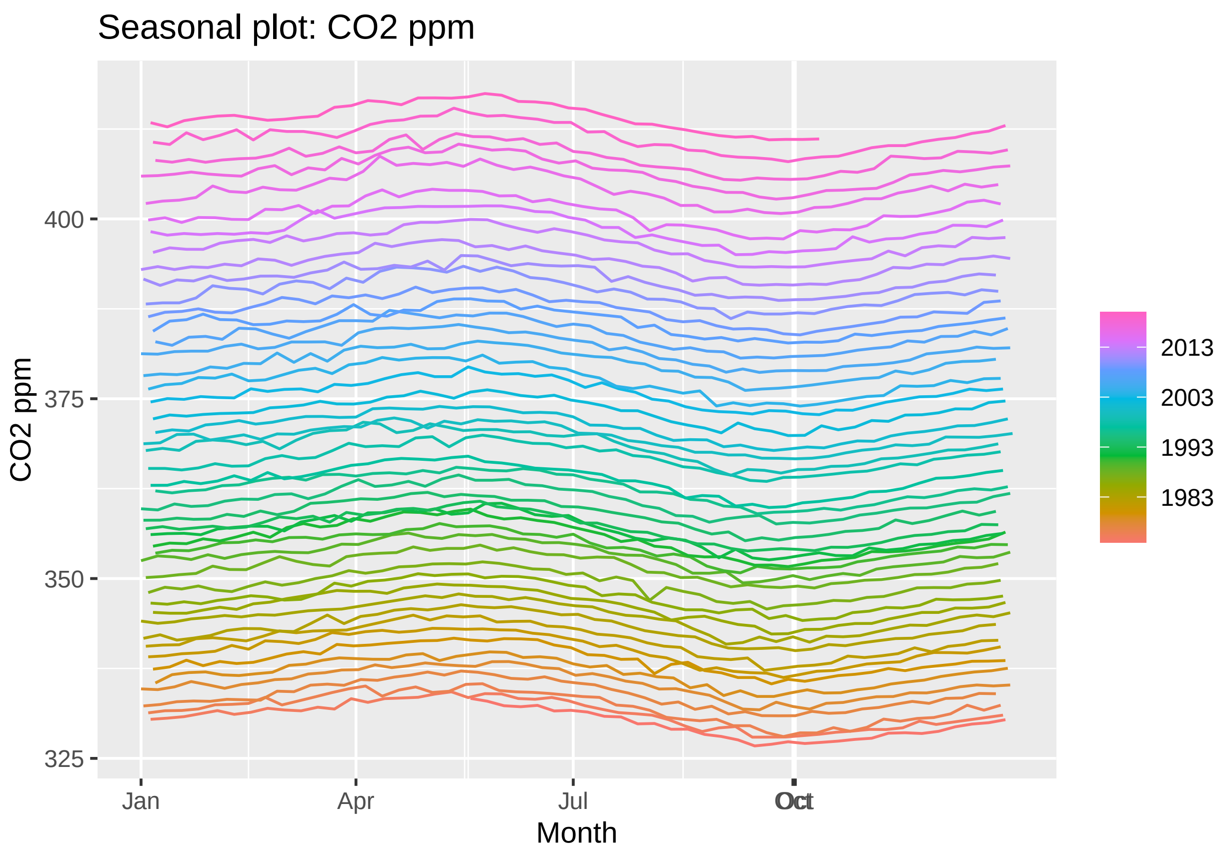
We see the smae trends in the data as above, just with slightly more noise now that the data is being measured weekly. We also see the holes in the line corresponding to the missing values, which we can now attempt to fill in. We will do this by utilizing the na_interpolation function frmo the imputeTS package. This uses linear (default), pline or stineman interpolation to replace missing values. We can then check this interpolation:

```
co2_weekly_tsibble_filled <- na_interpolation(co2_weekly_tsibble)
plot(co2_weekly_tsibble_filled, ylab = expression("CO2 ppm"), col = 'red', type = 'l',
     main = 'Weekly CO2 ppm Measurements')
lines(co2_weekly_tsibble, col="blue")
```



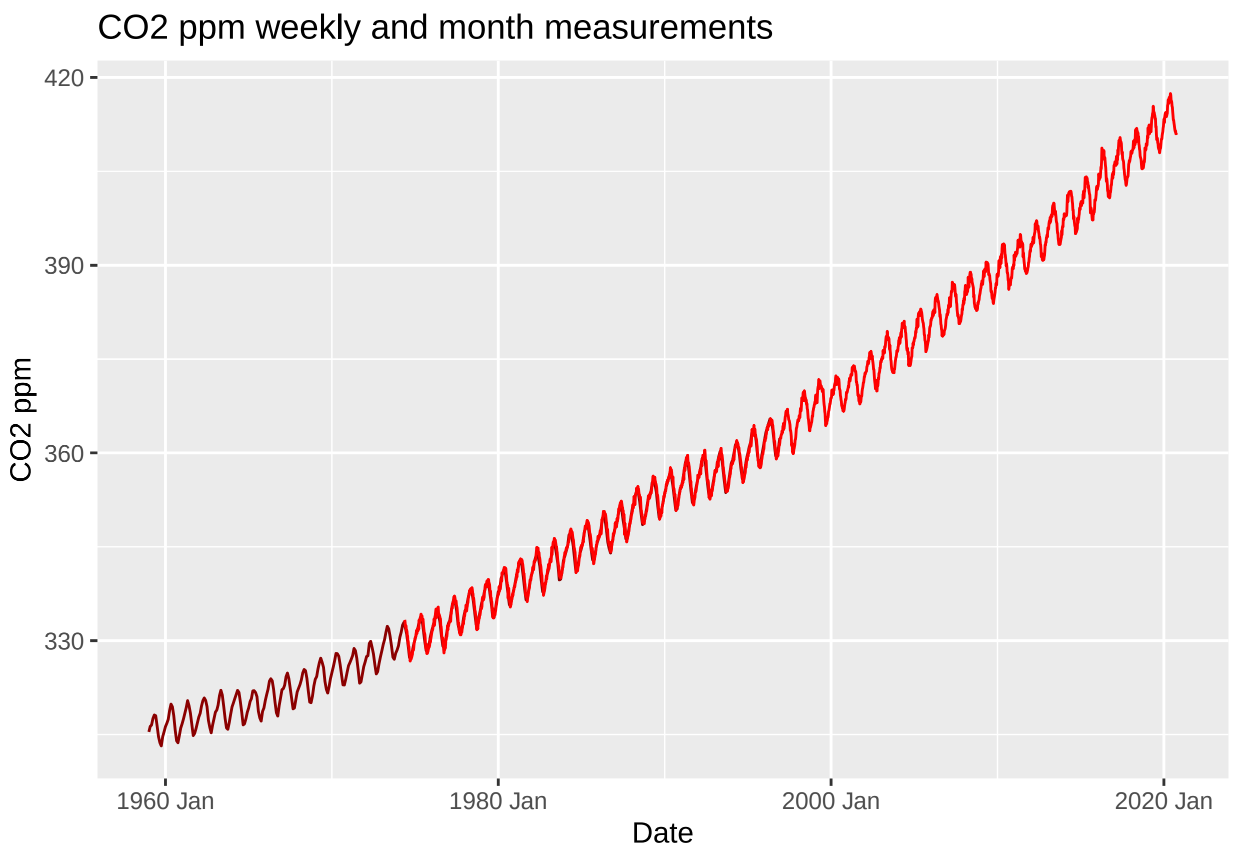
We see from the plot that the imput method has filled in reasonable values (shown in red). So we can now we can assess if we see the same kind of trends of data in each year increasing, and the same seasonal variations:

```
co2_weekly_tsibble_filled %>%
  gg_season(CO2_ppm) +
  ylab("CO2 ppm") + xlab('Month') +
  ggtitle("Seasonal plot: CO2 ppm")
```



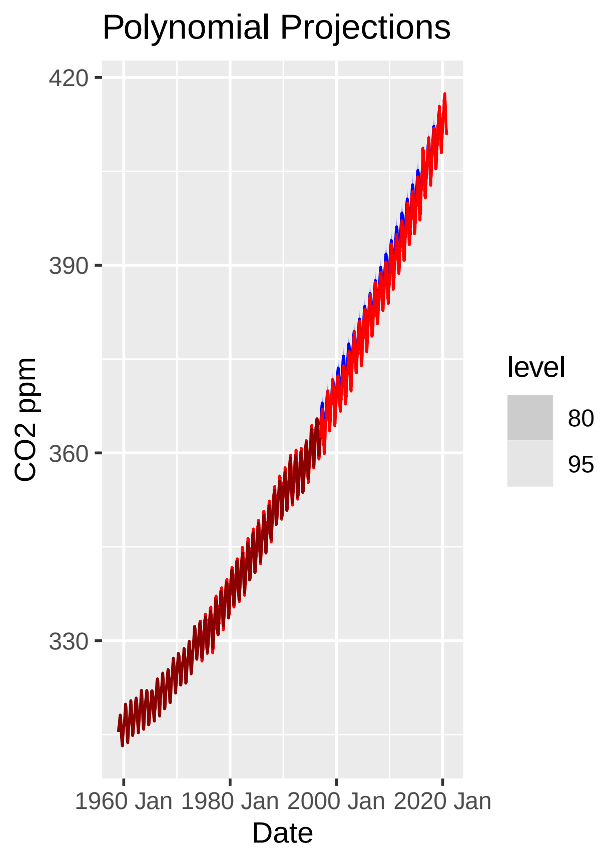
We see the same seasonal variations and trend of increasing data. Now, we can plot the data along with the monthly data above to confirm the data looks very similar.

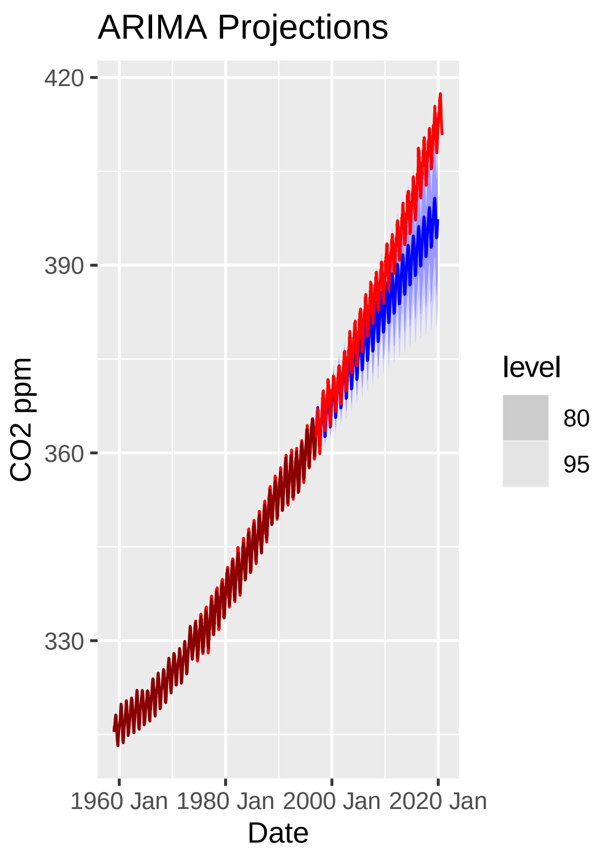
```
co2_tsibble %>% autoplot(.vars = CO2_ppm, color = 'dark red') +
  autolayer(.vars = CO2_ppm, co2_weekly_tsibble_filled, color = 'red') +
  ylab("CO2 ppm") + xlab('Date') +
  ggtitle("CO2 ppm weekly and month measurements")
```



Importantly, we see that the two sets of data seem very similar (the overlap in red and dark red), and that the light red weekly data continues on the same path. Now that we've explored and verified this new data, we can see how it compares to the projections generated in part 2 and 3 from the polynomial model and from the selected ARIMA model.

```
polynomial.tslm.fit <- co2_tsibble %>% model(TSLM(CO2_ppm ~ trend() + I(trend()^2) + season()))
p1 <- forecast(polynomial.tslm.fit, h = 276) %>%
  autoplot() +
  autolayer(co2_weekly_tsibble_filled, color = 'red', .vars = CO2_ppm) +
  autolayer(co2_tsibble, color = 'dark red', .vars = CO2_ppm) +
  ylab("CO2 ppm") + xlab('Date') +
  ggtitle("Polynomial Projections")

p2 <- autoplot(forecast(selected.arima.mod, h = 276)) +
  autolayer(co2_weekly_tsibble_filled, color = 'red', .vars = CO2_ppm) +
  autolayer(co2_tsibble, .vars = CO2_ppm, color = 'dark red') +
  ylab("CO2 ppm") + xlab('Date') +
  ggtitle("ARIMA Projections")
grid.arrange(p1, p2, nrow = 1, ncol = 2)
```





Interestingly, we see that the model from part 2 seemed to do much better in predicting future values, as evidenced by the fact that the blue predictions are almost completely covered up by the red actual values. The ARIMA model from part 3 had much wider confidence intervals, and indeed the true value is perhaps in the 95% interval, but the predictions was not nearly as useful. To get better sense of this difference in fit, we can generate the monthly average data from the weekly data, and use this to assess the accuracy. Specificaly, we will pull the Root Mean Squared Average and Mean Absolue Error from the accuracy function (of the forecast package).

```
co2_agg <- co2_weekly_tsibble_filled %>%
  group_by_key() %>%
  index_by(YearMonth = yearmonth(Date)) %>%
  summarise(CO2_ppm = mean(CO2_ppm)) %>%
  rename(Date = YearMonth)

rbind(forecast(selected.arima.mod, h = 274) %>% accuracy(co2_agg),
      forecast(polynomial.tslm.fit, h = 274) %>% accuracy(co2_agg)) %>%
  dplyr::select(.model, MAE, RMSE)
```

```
## # A tibble: 2 x 3
##   .model                                              MAE  RMSE
##   <chr>                                             <dbl> <dbl>
## 1 lowAIC_ARIMA                                       5.42  6.75
## 2 TSLM(CO2_ppm ~ trend() + I(trend()^2) + season())  1.18  1.32
```

We see that the polynomial seasonal model has a much lower mean absolute error and root mean squared error, signaling a better fit.

Part 5 (3 points)

Seasonally adjust the weekly NOAA data, and split the seasonally-adjusted (SA) series into training and test sets, using the final two years of observations as the test set. Fit an ARIMA model to the SA series following all appropriate steps. This should include measurement and discussion how your model performs in-sample and (psuedo-) out-of-sample, comparing candidate models and explaining your choice. In addition, fit a polynomial time-trend model to the SA series and compare its performance to that of your ARIMA model.

Let's first decompose the weekly NOAA time series into seasonal, trend and random components

```r
# Convert weekly NOAA data into ts object
#co2_weekly_ts <- ts(co2_weekly_tsibble_filled$CO2_ppm, start = c(1974, 20), frequency = 52) %
co2_weekly_ts <-  ts(co2_weekly_tsibble_filled$CO2_ppm, start = c(1974, 20), frequency = 52)
co2_weekly.fit <- stl(co2_weekly_ts, s.window="periodic",robust=TRUE)
autoplot(co2_weekly.fit) + ggtitle("STL decomposition of CO2 level from NOAA data")
```

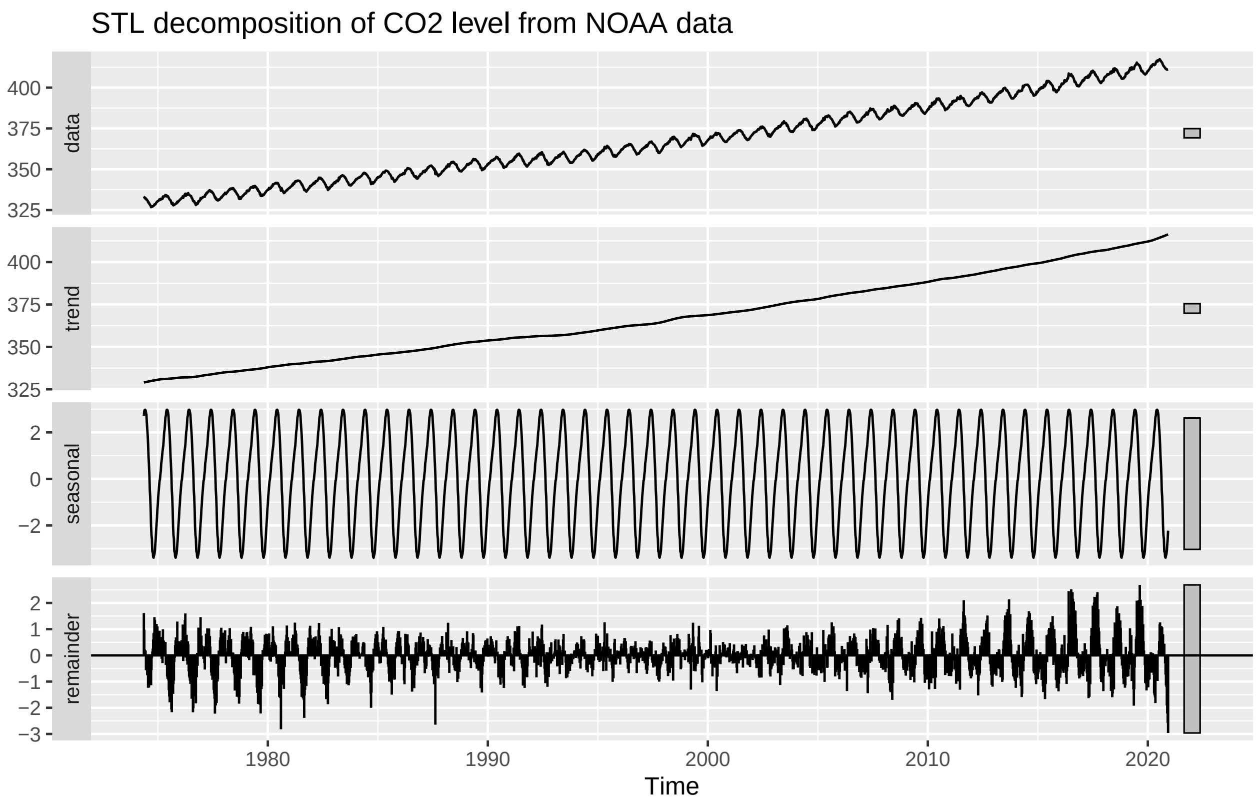


Figure 1: STL decomposition of Weekly CO2 from 1974 May to 2020 Oct

```r
co2_weekly_ts.seas <- seasadj(co2_weekly.fit)
plot(co2_weekly_ts.seas)
```

By plotting the seasonally-adjusted (SA) CO2 data against the original time series, we confirm that the seasonality has been removed.

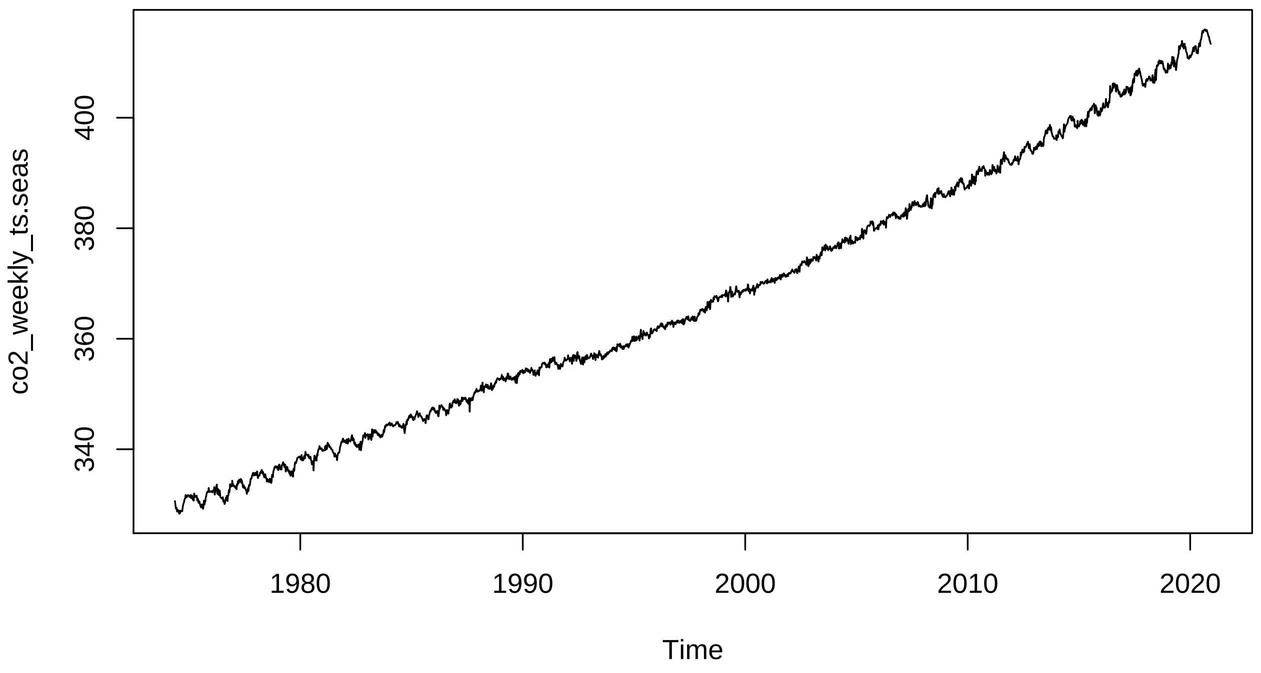


Figure 2: STL decomposition of Weekly CO2 from 1974 May to 2020 Oct

```
autoplot(co2_weekly_ts, series="Data") +
  autolayer(trendcycle(co2_weekly.fit), series="Trend") +
  autolayer(co2_weekly_ts.seas, series="Seasonally Adjusted") +
  xlab("Year") + ylab("New orders index") +
  ggtitle("Seasonally-adjusted weekly CO2 data from 1974 May to 2020 Oct") +
  scale_colour_manual(values=c("gray","blue","red"),
             breaks=c("Data","Seasonally Adjusted","Trend"))
```

Next, we split the SA weekly CO2 into a training set (1974 May - 2018 Sep) and a test set (2018 Oct - 2020 Oct).

```
co2_weekly_tsibble.seas <- co2_weekly_tsibble_filled %>%
  mutate(CO2_ppm = co2_weekly_ts.seas)
# split training and test:
co2_training <- co2_weekly_tsibble.seas %>% filter_index(~'2018-10-11')
co2_test <- co2_weekly_tsibble.seas %>% filter_index('2018-10-11'~.)
```

We apply first-differencing to stablize the mean. The acf and pacf does not show seasonality, and the unit root test also suggests that the number of first differencing should be 1. The kpss test suggests that the null hypothesis of stationarity cannot be rejected at 5% significance level and the historgram shows that the residuals follow a normal distribution with mean at 0. Hence, we proceed with the order of difference 1 to design ARIMA models and then can compare the models with the order of difference 1 vs. 2 in the next step.

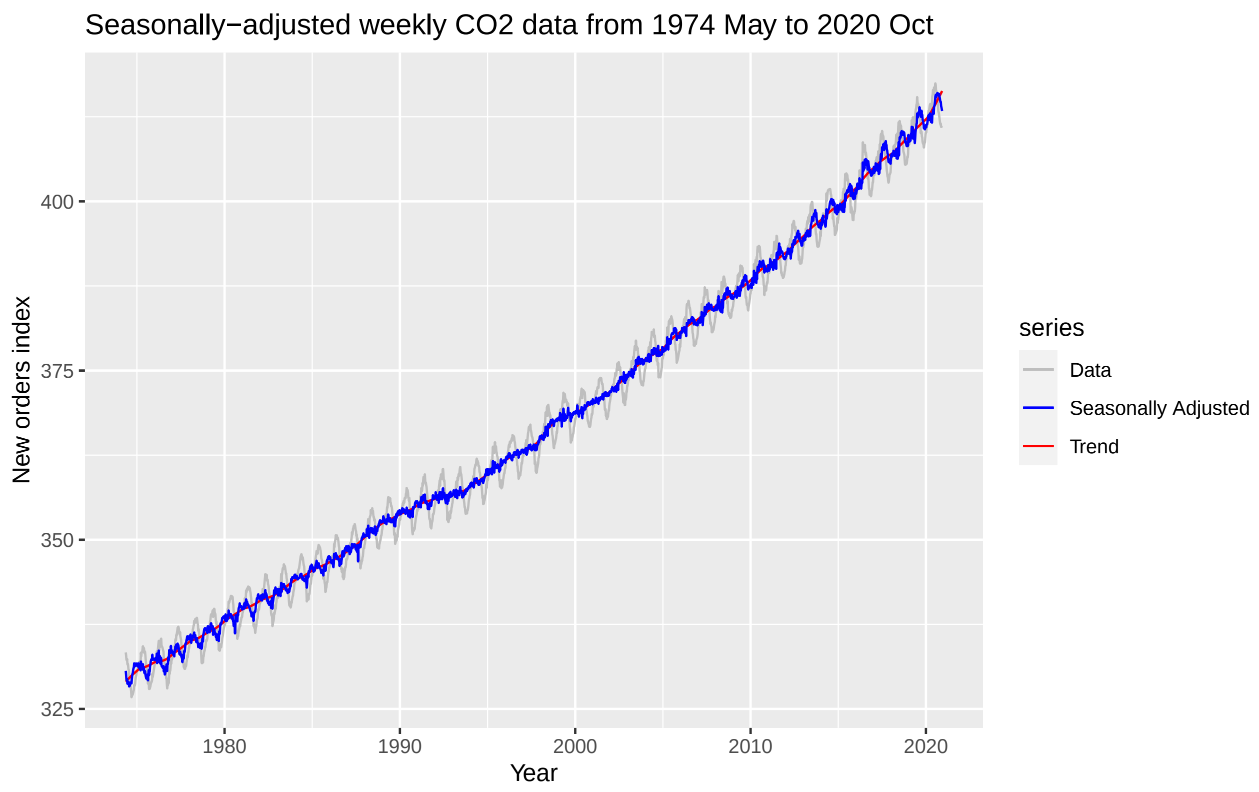


Figure 3: Seasonally-adjusted Weekly CO2 Level from 1974 May to 2020 Oct

```
# Let's perform first differencing to stabilize the mean of the training set:
co2_training <- co2_training %>% mutate(d_CO2 = difference(CO2_ppm, 1))
co2_training %>% filter(!is.na(d_CO2)) %>% gg_tsdisplay(y = d_CO2, plot = 'partial', lag_max =
```

```
co2_training %>% features(CO2_ppm, unitroot_ndiffs)
```

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

```
# now don't see any trends:
co2_training %>% features(d_CO2, unitroot_kpss) # cannot reject stationality
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0919         0.1
```

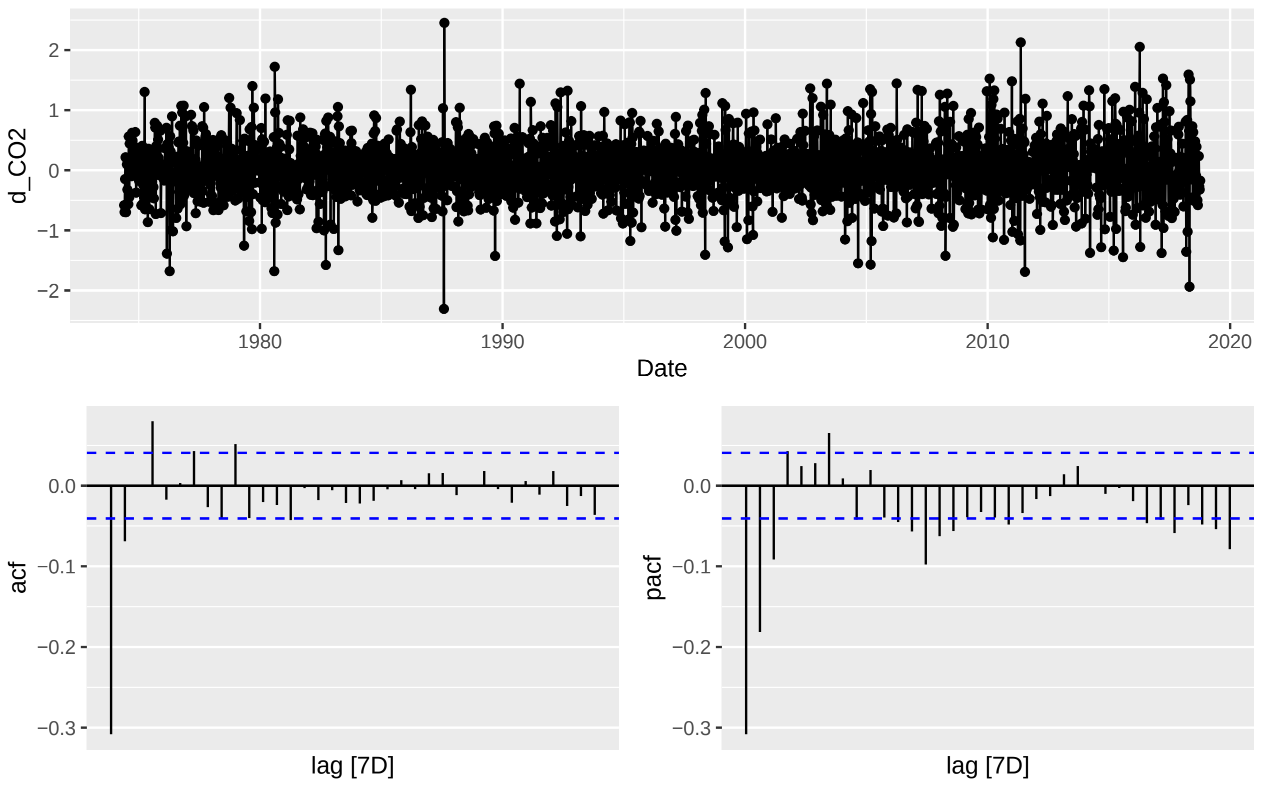


Figure 4: First differencing SA Weekly CO2 training data

```
co2_training%>% filter(!is.na(d_CO2)) %>% pull(d_CO2) %>% gghistogram()
```

We first run the test_ARIMA function to loop through all possible models from AR(0) to AR(3) and from MA(0) to MA(3). Note that we will use the test_ARIMA function defined above, just with edits made to not use a seasonal term.

```
# find arima model using the test_ARIMA function from above
# pdq(2,1,2) has the lowest AICc and BIC.
test_ARIMA(dplyr::select(co2_training, data = CO2_ppm), c(3,3,0,0), c(1,0), 52)
```

```
## # A tibble: 2 x 7
##   model     p     q     P     Q   AIC   BIC
##   <int> <int> <int> <int> <int> <dbl> <dbl>
## 1    11     2     2     0     0 2844. 2873.
## 2    11     2     2     0     0 2844. 2873.
```

We can also check what is produced by auto-arima:

```
co2_training %>% dplyr::select(CO2_ppm) %>% auto.arima()
```

```
## Series: .
```

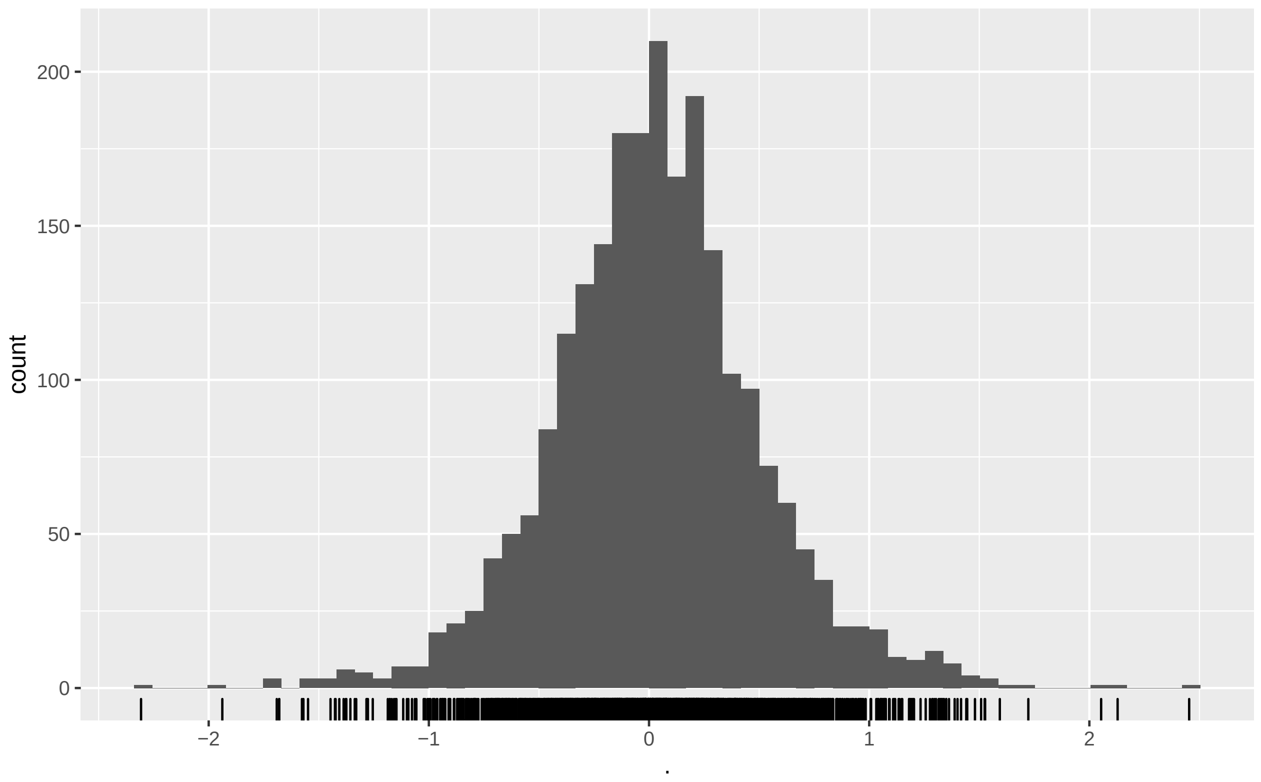


Figure 5: First differencing SA Weekly CO2 training data

```
## ARIMA(2,1,2) with drift
##
## Coefficients:
##          ar1      ar2      ma1     ma2   drift
##       0.9013  -0.2714  -1.2850  0.5515  0.0335
## s.e.  0.0934   0.0444   0.0889  0.0534  0.0066
##
## sigma^2 estimated as 0.1975:  log likelihood=-1405.49
## AIC=2822.98   AICc=2823.02   BIC=2857.47
```

Auto-arima generates the same (2,1,2) model as our testing loop (albeit with drift). Given that this model has the lowest AIC, BIC and was chosen by auto arima, it seems a robust choice. We can include a few other models at this stage for comparison sake, of model accuracy.

```
models_NOAA <- co2_training %>% model(mod1 = ARIMA(CO2_ppm ~ pdq(2,1,2)),
                                      mod2 = ARIMA(CO2_ppm ~ pdq(1,1,1)),
                                      mod3 = ARIMA(CO2_ppm ~ pdq(1,1,0)))
glance(models_NOAA)
```

```
## # A tibble: 3 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 mod1    0.198  -1405. 2823. 2823. 2857. <cpl [2]> <cpl [2]>
## 2 mod2    0.200  -1420. 2848. 2848. 2871. <cpl [1]> <cpl [1]>
## 3 mod3    0.207  -1461. 2927. 2927. 2944. <cpl [1]> <cpl [0]>
```

```
models_NOAA %>% accuracy()
```

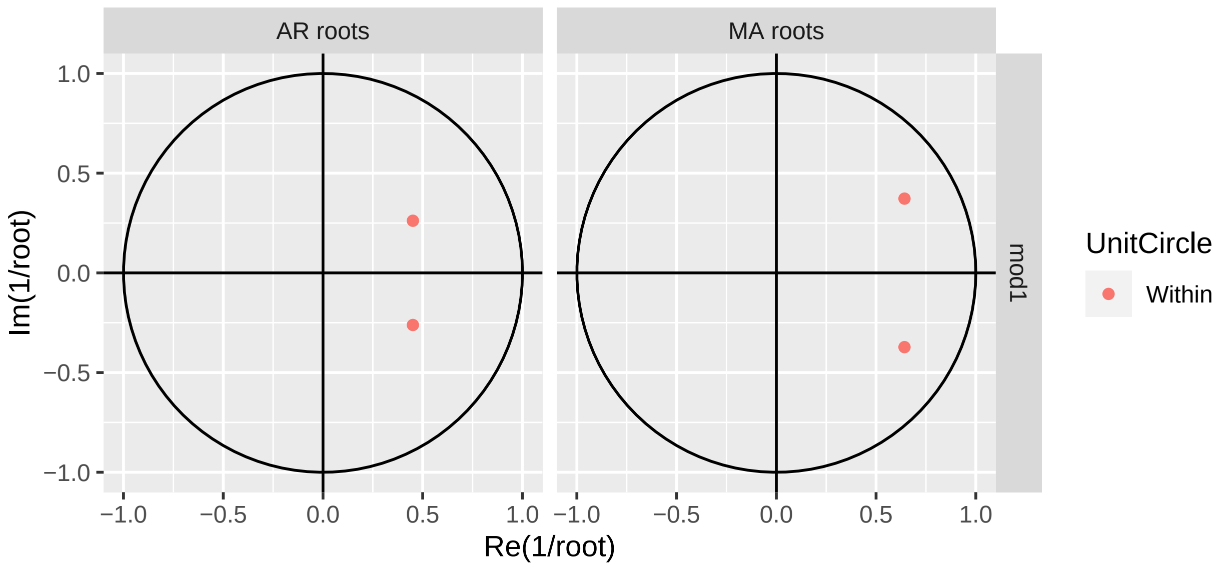
```
## # A tibble: 3 x 9
##   .model .type            ME  RMSE   MAE       MPE   MAPE  MASE      ACF1
##   <chr>  <chr>         <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>     <dbl>
## 1 mod1   Training  0.000127  0.444 0.339 -0.000194 0.0927 0.940 -0.000985
## 2 mod2   Training -0.0000797 0.447 0.342 -0.000297 0.0935 0.949  0.00255
## 3 mod3   Training  0.0000752 0.455 0.346 -0.000204 0.0945 0.959 -0.0563
```

The first model, pdq(2,1,2), is the best model based on RMSE, MAE, AIC and BIC. We can now check the model report and unit roots.

```
# Select pdq(2,1,2) for the seasonally adjusted series:
selected_mod <- models_NOAA %>% dplyr::select(mod1)
selected_mod %>% report()
```

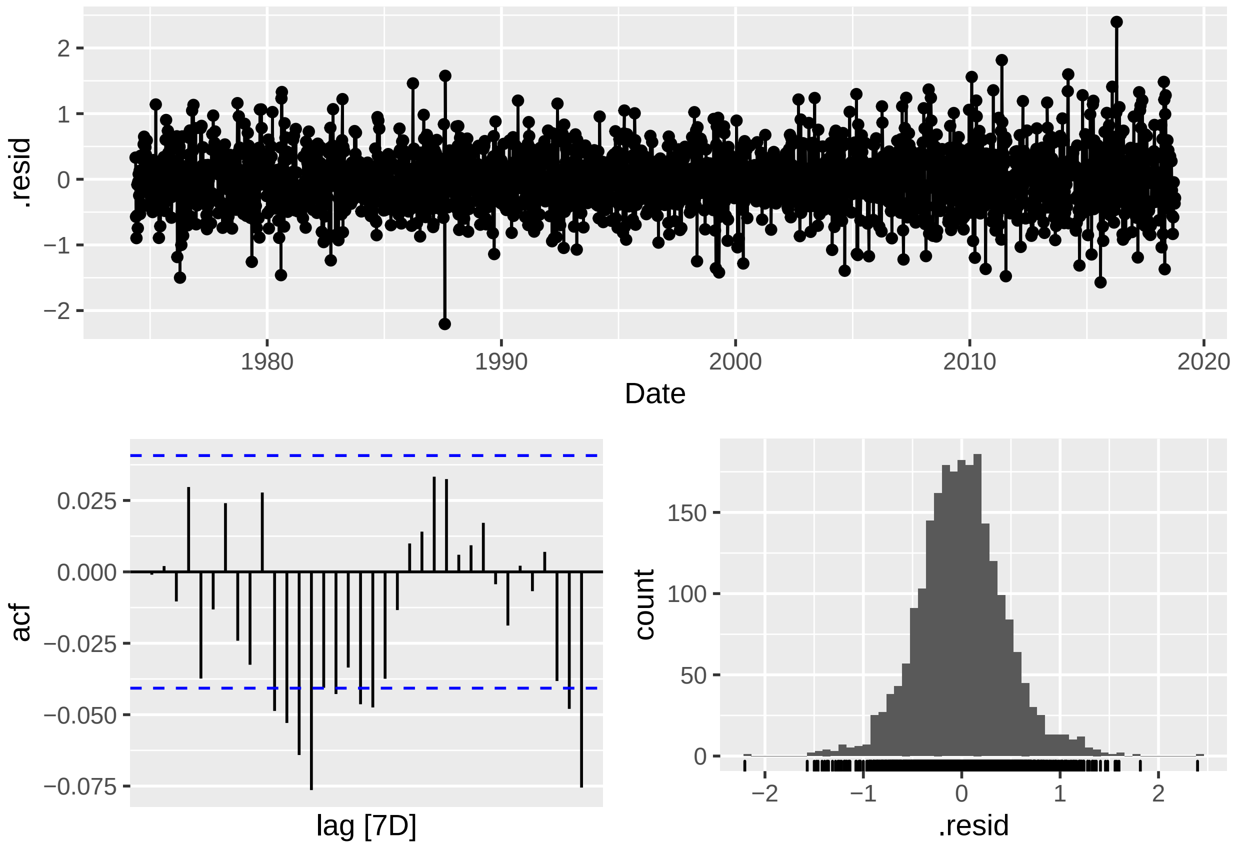
```
## Series: CO2_ppm
## Model: ARIMA(2,1,2) w/ drift
##
## Coefficients:
##          ar1      ar2      ma1     ma2  constant
##       0.9013  -0.2714  -1.2850  0.5515    0.0124
## s.e.  0.0934   0.0444   0.0889  0.0534    0.0025
##
## sigma^2 estimated as 0.1975:  log likelihood=-1405.49
## AIC=2822.98   AICc=2823.02   BIC=2857.47
```

```
gg_arma(selected_mod)
```



Examining our best model more closely, we can see that the residuals stay around 0 across time and follow a normal distribution. The acf does not show a significant autocorrelation.

```r
selected_mod %>% gg_tsresiduals(lag_max = 36)
```



To check psuedo-out-of-sample performance of the model, we plot the two-year ahead forecasted values vs the actual values from October 2018 to October 2020. The forecast is very close to the actual values, but underestimated by about 5ppm. The 95% confidence interval has captured most of the actual values.

```r
selected_mod %>% forecast(h = 104) %>%
  autoplot(co2_test) + ggtitle('CO2 two-year ahead forecasts') +
  theme(legend.position = "right")
```

Next, we estimate a polynomial time-trend model to the SA series. We will evaluate 2 models as follows:

Model 1:

$$CO2 = \beta_0 + \beta_1 TimeIndex + \beta_2 TimeIndex^2$$

Model 2:

$$CO2 = \beta_0 + \beta_1 TimeIndex + \beta_2 TimeIndex^2 + \beta_3 TimeIndex^3$$

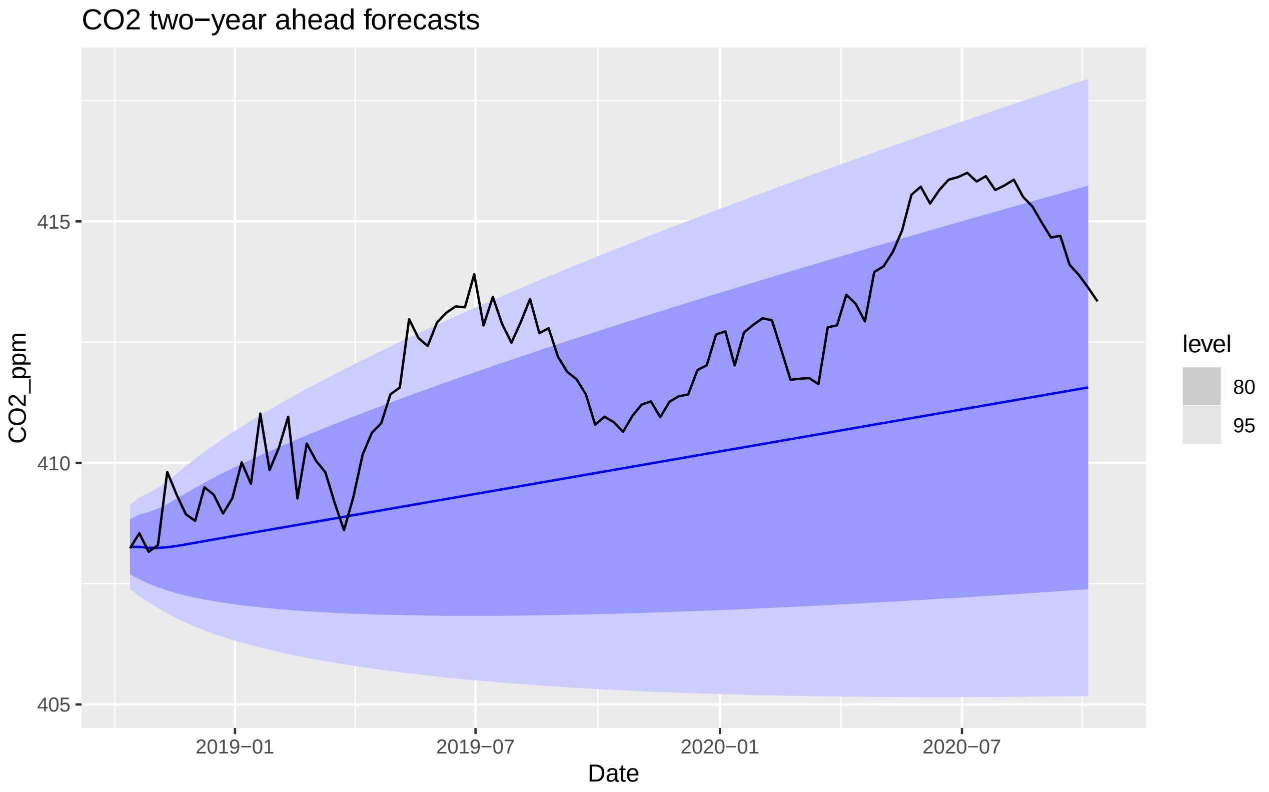


Figure 6: CO2 two-year ahead forecasts

```
# create a time index starting from 0 which is equivalent to 1974 May
co2_training <- co2_training %>% mutate(time_index = row_number())
# Estimate the quadratic linear model
co2_wkly.poly_mod2 <- lm(CO2_ppm ~ time_index + I(time_index^2), data = co2_training)
co2_wkly.poly_mod3 <- lm(CO2_ppm ~ time_index + I(time_index^2) + I(time_index^3), data = co2_t
# summarize models and view the coefficients
suppressWarnings(stargazer(co2_wkly.poly_mod2, co2_wkly.poly_mod3, type='text', summary=F,
          dep.var.labels = c("Weekly CO2 Level"),
          title = "Polynomial Time-trend Regression Model",
          header=F))
```

```
##
## Polynomial Time-trend Regression Model
## ======================================================================
##                                          Dependent variable:
##                      -------------------------------------------------
##                                           Weekly CO2 Level
##                                  (1)                        (2)
## ----------------------------------------------------------------------
## time_index                    0.023***                   0.030***
##                               (0.0001)                   (0.0003)
##
```

```
## I(time_index2)                0.00000***                   -0.00000***
##                                (0.00000)                    (0.00000)
##
## I(time_index3)                                               0.000***
##                                                               (0.000)
##
## Constant                      330.641***                   329.200***
##                                 (0.064)                       (0.072)
##
## ---------------------------------------------------------------------------
## Observations                    2,317                         2,317
## R2                              0.998                         0.999
## Adjusted R2                     0.998                         0.999
## Residual Std. Error       1.024 (df = 2314)             0.868 (df = 2313)
## F Statistic         560,613.300*** (df = 2; 2314) 520,438.600*** (df = 3; 2313)
## ===========================================================================
## Note:                                             *p<0.1; **p<0.05; ***p<0.01
```

All coefficients in both models are significant, but the fit increases very little for the additional complexity. Thus, we'll use the quadratic model moving forward.

```r
summary(co2_wkly.poly_mod2)$coefficients
```

```
##                    Estimate   Std. Error    t value Pr(>|t|)
## (Intercept)    3.306408e+02 6.386975e-02 5176.79829        0
## time_index     2.267382e-02 1.272622e-04  178.16616        0
## I(time_index^2) 4.695439e-06 5.315981e-08   88.32687        0
```

We estimate the models to be: Model 1:

$$CO2 = 330.6 + 0.0277 TimeIndex + 4.70 * 10^{-6} TimeIndex^2$$

We now examine the model visually.

```r
# Plot time series plot of original series and estimated linear time trend
p1 <- co2_training %>% ggplot(aes(x = Date, y = CO2_ppm)) +
  geom_line(size = 1) +
  theme(legend.position = "none") +
  geom_line(aes(y = co2_wkly.poly_mod2$fitted.values), colour = "maroon", size = 1) +
  ggtitle('Weekly CO2 Level - Polynomial Time-trend model 1')
# Plot residuals
p2 <- co2_training %>% ggplot(aes(x = Date, y = co2_wkly.poly_mod2$residuals)) +
  geom_line(colour = 'blue') + geom_smooth(colour = 'red') + geom_hline(yintercept = 0)
# QQ Plot
p3 <- co2_training %>% ggplot(aes(sample = co2_wkly.poly_mod2$residuals)) + stat_qq() + stat_q
grid.arrange(p1, p2, p3, nrow = 1, ncol = 3)
```

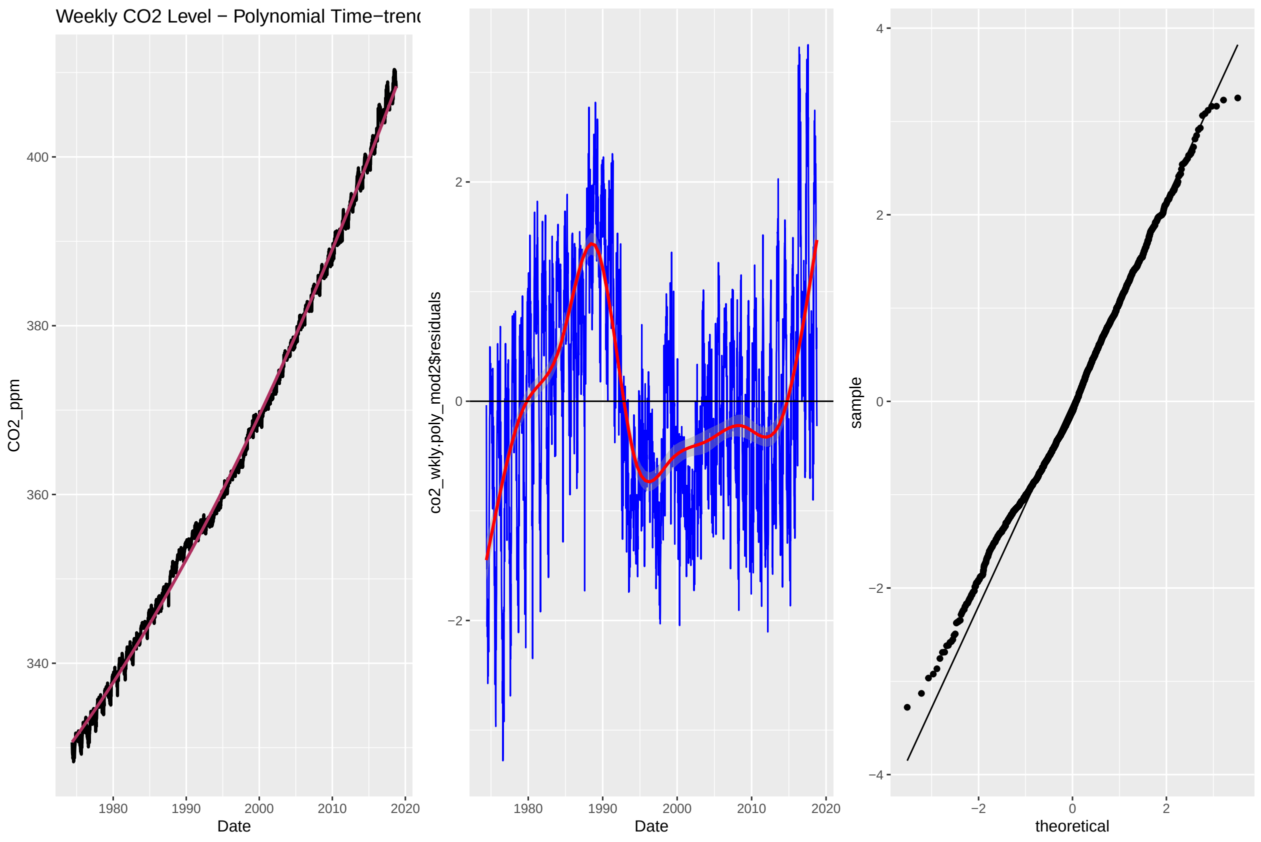
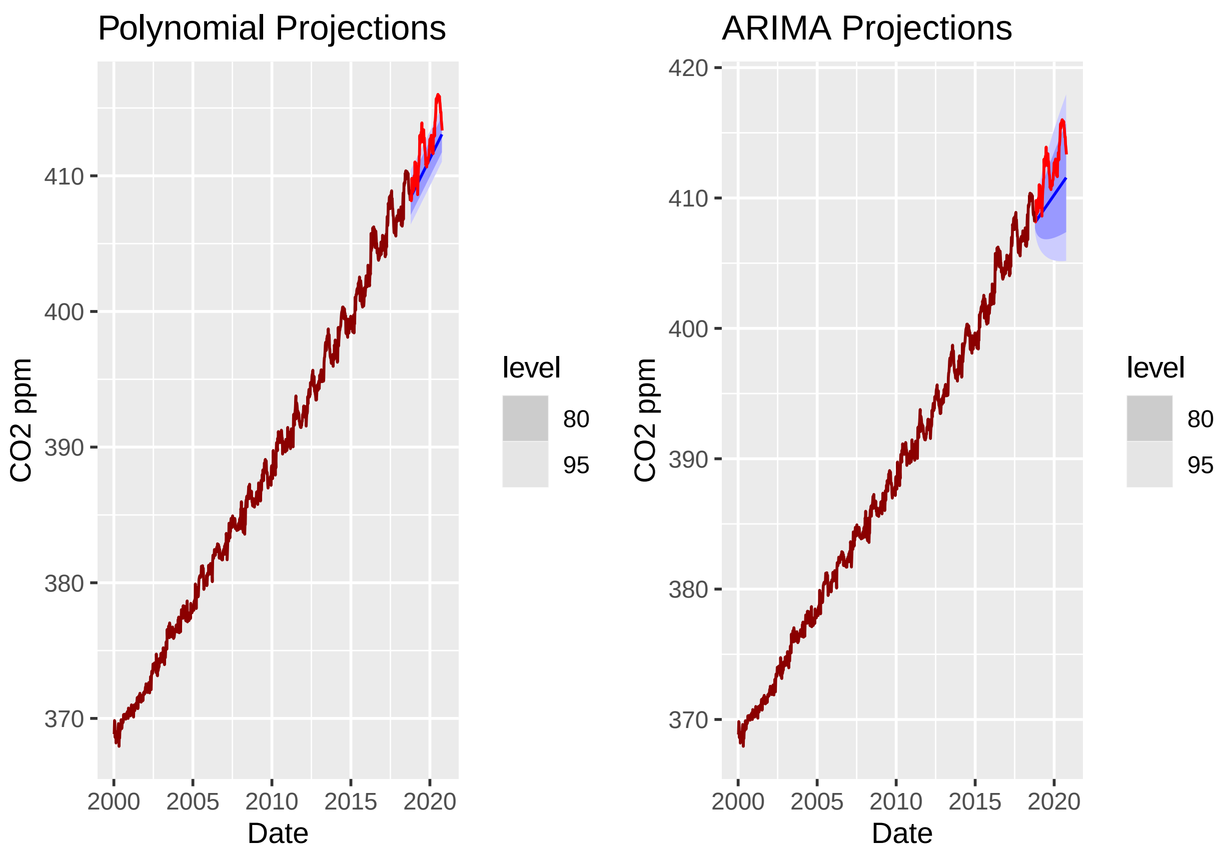


Figure 7: Examining the second polynomial model

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

We can see that the model follows the values very well. It captures the trend of CO2 very well and the mean of residuals stay around 0 across time with little deviation due to imperfect removing of seasonality. The QQ plot shows that the residuals follows the Normal distribution, but with heavy tails. Now, we can finally assess how the model predicts moving forward, and compare against the ARIMA model chosen above.

```
co2_wkly.poly_mod2 <- co2_training %>% model(TSLM(CO2_ppm ~ trend() + I(trend()^2)))
p1 <- forecast(co2_wkly.poly_mod2, h = 104) %>%
  autoplot() +
  autolayer(co2_training %>% filter_index('2000-01-01'~.), color = 'dark red', .vars = CO2_ppm)
  autolayer(co2_test, color = 'red', .vars = CO2_ppm) +
  ylab("CO2 ppm") + xlab('Date') +  ggtitle("Polynomial Projections")

p2 <- autoplot(forecast(selected_mod, h = 104)) +
  autolayer(co2_test, color = 'red', .vars = CO2_ppm) +
  autolayer(co2_training %>% filter_index('2000-01-01'~.), .vars = CO2_ppm, color = 'dark red')
  ylab("CO2 ppm") + xlab('Date') + ggtitle("ARIMA Projections")
grid.arrange(p1, p2, nrow = 1, ncol = 2)
```



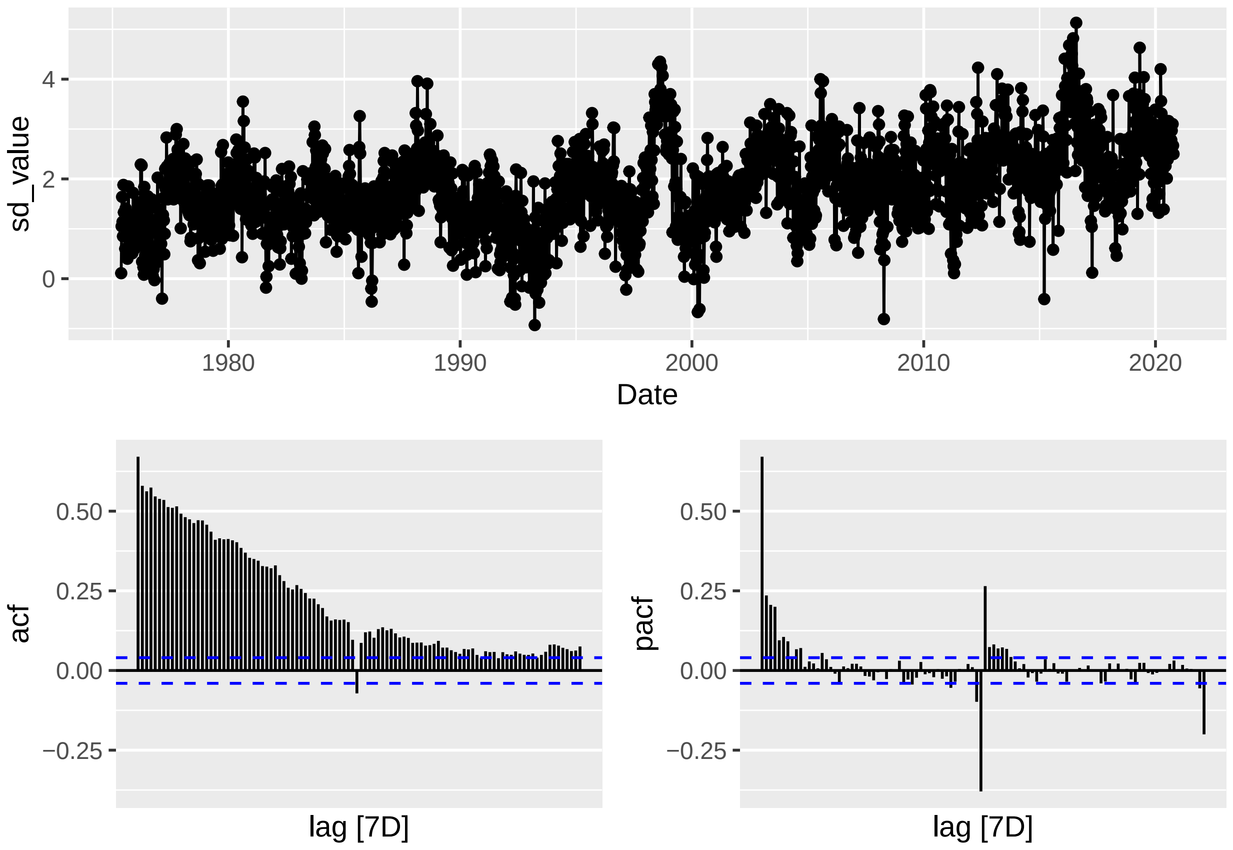
While the test section of the plots is quite small, it is still very clear that the Polynomial model performs better.

Part 6 (4 points)

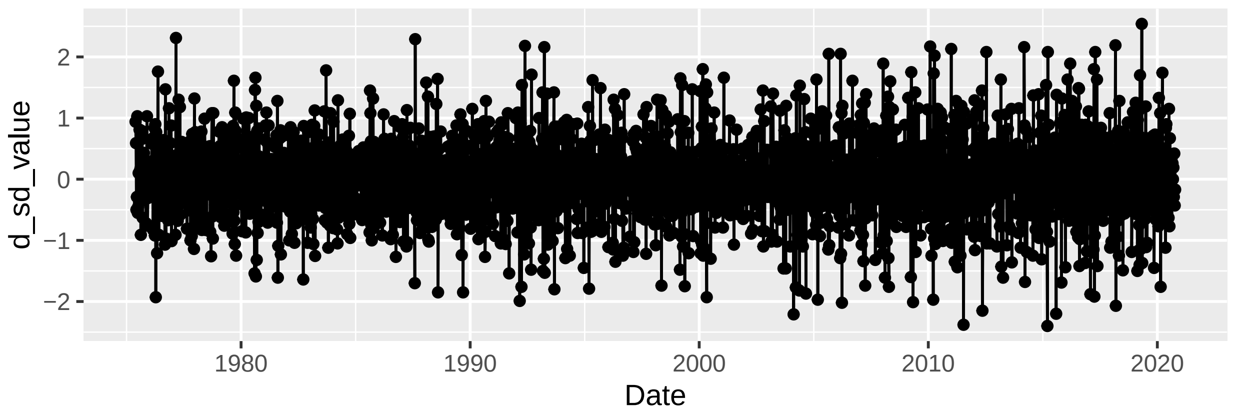
Fit an ARIMA model to the original (NSA) weekly series following all appropriate steps. Generate predictions for when atmospheric CO2 is expected to be at 420 ppm and 500 ppm levels for the first and final times, considering prediction intervals as well as point estimates in your answer. Generate a prediction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

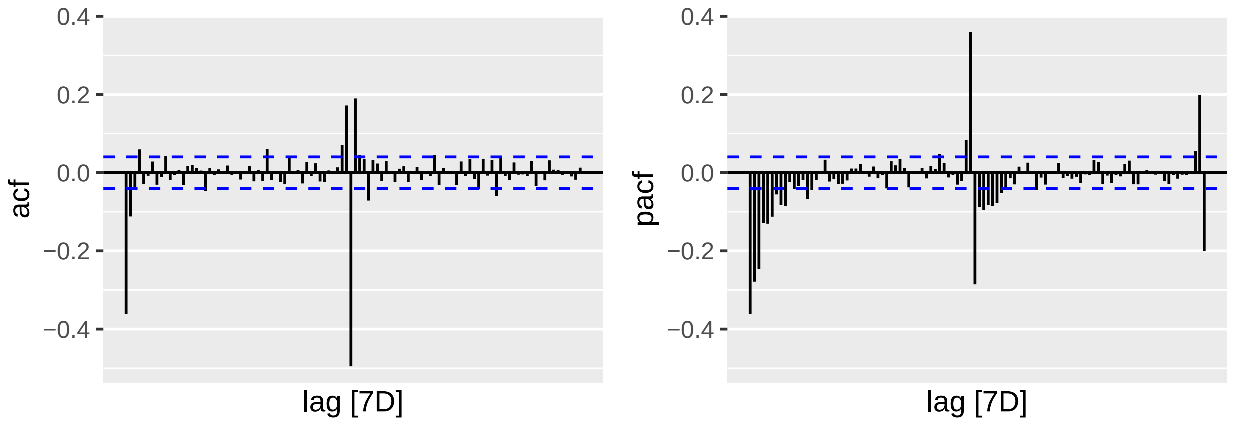
Through the earlier analysis, we already know that the seasonal differencing should be applied. After applying the 52th differencing, we still see a significant autocorrelation in the acf graph. Hence, we apply first differencing on top of seasonal differencing and then the series looks stationary and the acf cut off after lag 2 and the pacf tails off. The acf and pacf patterns indicates an MA(2) model, so we will focus our model specification on q = 2.

```
# Apply seasonal differencing
co2_weekly_tsibble <- co2_weekly_tsibble_filled %>% mutate(sd_value = difference(CO2_ppm, 52))
co2_weekly_tsibble %>%
  filter(!is.na(sd_value)) %>%
  gg_tsdisplay(y = sd_value, plot = 'partial', lag_max = 104)
```



```
# Apply first order differencing on top of seasonal differencing
co2_weekly_tsibble <- co2_weekly_tsibble_filled %>% mutate(d_sd_value = difference(difference(
co2_weekly_tsibble %>%
  filter(!is.na(d_sd_value)) %>%
  gg_tsdisplay(y = d_sd_value, plot = 'partial', lag_max = 104)
```





The unit root test suggests that the number of first differencing should be 1 and the number of seasonal differencing should be 0. However, as we clearly saw the stationarity improved after the seasonal differencing, we will go ahead with d = 1 and D = 1. The kpss test statistic shows that we cannot reject the null hypothesis at 5% significance, so the double-differenced series is stationary.

```
co2_weekly_tsibble %>% features(CO2_ppm, unitroot_ndiffs)
```

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

```
co2_weekly_tsibble %>% features(CO2_ppm, unitroot_nsdiffs)
```

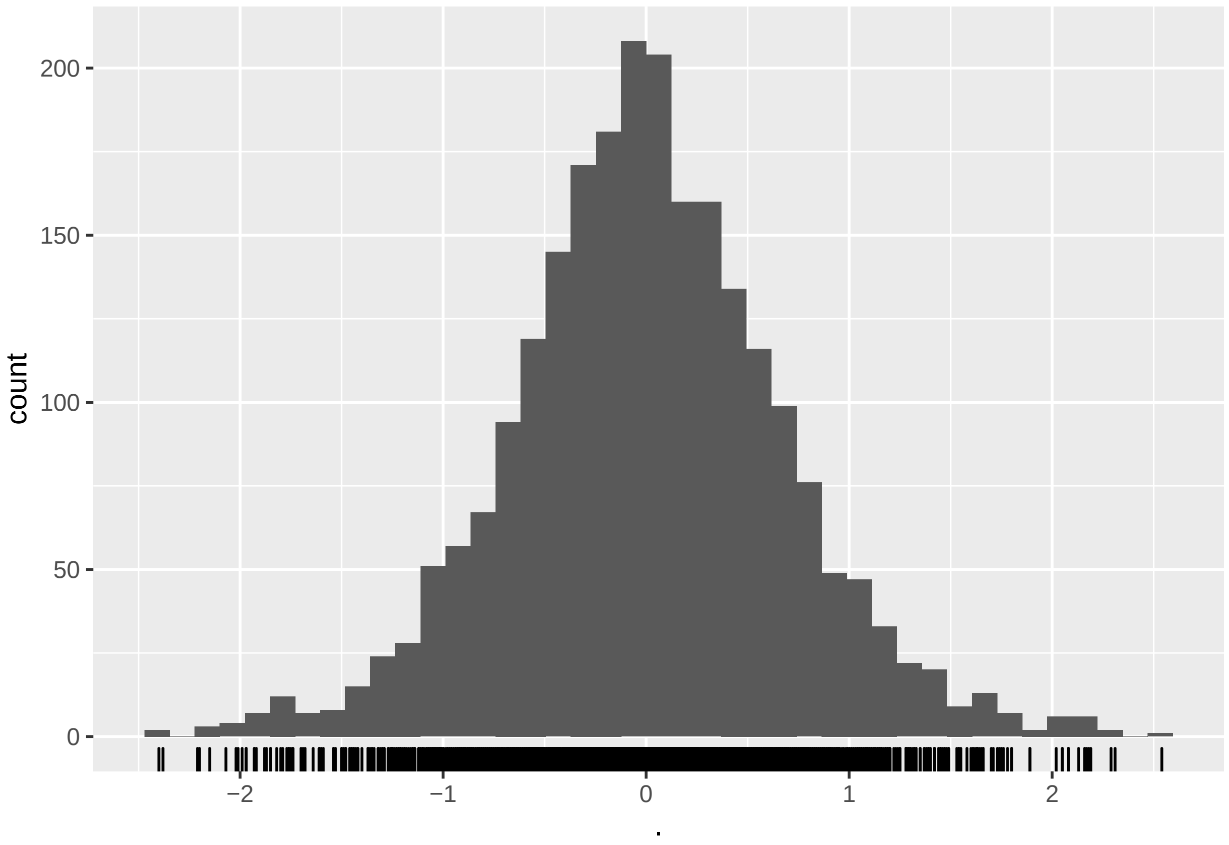
```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       0
```

```
co2_weekly_tsibble %>% features(d_sd_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1   0.00527         0.1
```

The histogram of the differenced series (d=1 and D=1) follows a Normal distribution, but with a high kurtosis. This visually indicates stationarity of the double differenced series.

```
co2_weekly_tsibble %>% filter(!is.na(d_sd_value)) %>% pull(d_sd_value) %>% gghistogram()
```



With that part of the model decided, we can run the test arima function again to assess several different AIC and BIC models. Note that we've once again run this loop in advance and are reading from the results.

```
# find arima model by using the test_ARIMA function to loop through 0 to 2 for all p, q, P and
# part6_arima <- test_ARIMA(select(co2_weekly_tsibble, data = CO2_ppm), c(2,2,2,2), c(1,1), 52
# write_feather(part6_arima, 'Part 6 ARIMA Results.feather')
read_feather('Part 6 ARIMA Results.feather')
```

```
## # A tibble: 2 x 7
##   model     p     q     P     Q   AIC   BIC
##   <int> <int> <int> <int> <int> <dbl> <dbl>
## 1    39     0     2     1     2 2770. 2805.
## 2    39     0     2     1     2 2770. 2805.
```

The test_ARIMA function finds that ARIMA(0,1,2)(1,1,2) is the model with the lowest AICc and BICc. Nonetheless, we should run reports for few other models with a focus on MA(q) to compare the accuracy measures. We examine the following three models:

1. ARIMA(0,1,2)(1,1,2)
2. ARIMA(0,1,2)(0,1,2)
3. ARIMA(1,1,1)(0,1,1)

By examining the accuracy measures, the first model is still the best model regardless of which accuracy measure is used though a few of its MA roots are outside of the unit circle. We further used auto.arima() to check whether there is a better model, but the model (5,1,2) provided by auto.arima() has a higher AICc, BIC and all accuracy measures.

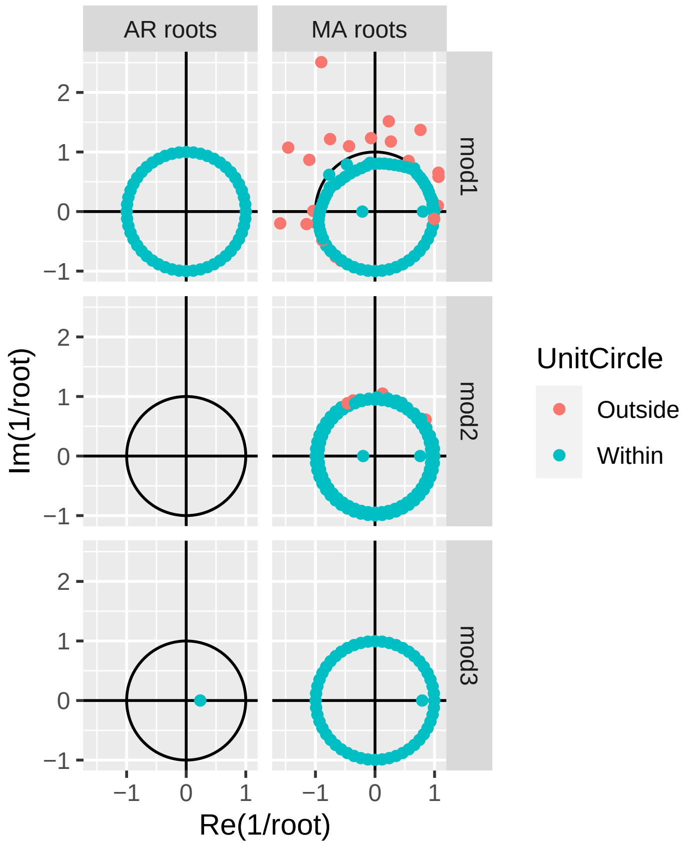
```r
# from the above we chose:
co2_weekly_models <- co2_weekly_tsibble %>% model(mod1 = ARIMA(CO2_ppm ~ pdq(0,1,2) + PDQ(1,1,
                                                  mod2 = ARIMA(CO2_ppm ~ pdq(0,1,2) + PDQ(0,1,2, period=52)),
                                                  mod3 = ARIMA(CO2_ppm ~ pdq(1,1,1) + PDQ(0,1,1, period=52)))

glance(co2_weekly_models)
```

```
## # A tibble: 3 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots    ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>      <list>
## 1 mod1    0.182  -1379. 2770. 2771. 2805. <cpl [52]> <cpl [106]>
## 2 mod2    0.192  -1416. 2841. 2841. 2870. <cpl [0]>  <cpl [106]>
## 3 mod3    0.192  -1418. 2844. 2844. 2867. <cpl [1]>  <cpl [53]>
```

```r
co2_weekly_models %>% accuracy()
```

```
## # A tibble: 3 x 9
##   .model .type         ME  RMSE   MAE     MPE   MAPE  MASE     ACF1
##   <chr>  <chr>      <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>
## 1 mod1   Training 0.00899 0.422 0.319 0.00222 0.0867 0.735 0.00531
## 2 mod2   Training 0.00862 0.433 0.328 0.00208 0.0889 0.755 0.00286
## 3 mod3   Training 0.00928 0.433 0.329 0.00225 0.0891 0.756 0.000339
```

```r
gg_arma(co2_weekly_models)
```



```
# compare against auto arima, again better:
auto_arima <- co2_weekly_tsibble %>%
  dplyr::select(CO2_ppm) %>%
  auto.arima(start.p = 0, start.q = 0, start.P = 0, start.Q = 0)

auto_arima
```

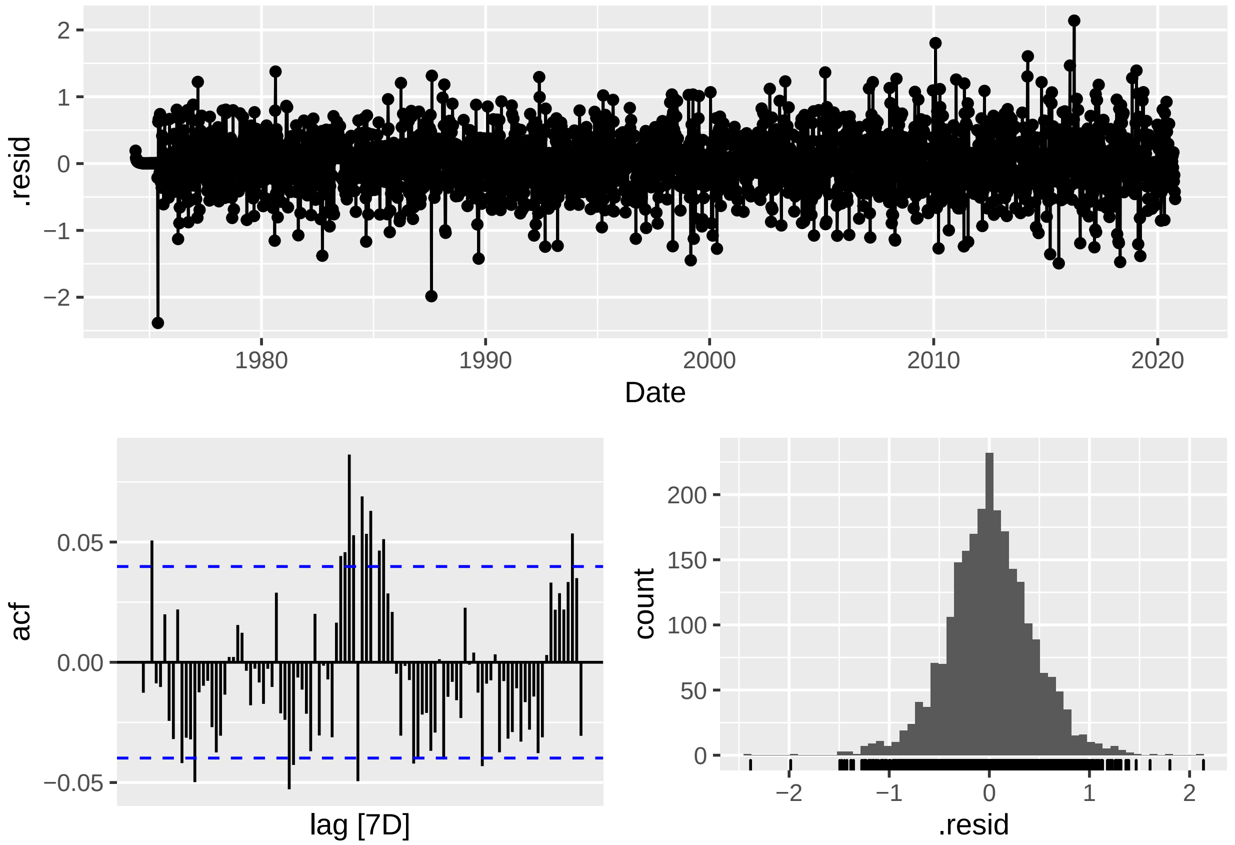
```
## Series: .
## ARIMA(5,1,3) with drift
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1     ma2     ma3
##       1.2508  -0.0173  -0.1946  -0.0258  -0.0949  -1.4458  0.3335  0.2502
## s.e.  0.2323   0.3689   0.1207   0.0380   0.0288   0.2326  0.4130  0.2060
##        drift
##       0.0323
## s.e.  0.0169
##
## sigma^2 estimated as 0.2432:  log likelihood=-1719.57
## AIC=3459.15   AICc=3459.24   BIC=3517.07
```

```r
accuracy(auto_arima)
```

```
##                             ME      RMSE       MAE           MPE      MAPE      MASE
## Training set 0.0003502789 0.4920946 0.3758627 -4.118732e-06 0.1021839 0.8652477
##                           ACF1
## Training set 0.0002079068
```

Given the roots outside of the circle, we will select mod3 as the optimal model. The histogram the residuals of the model follows a Normal distribution with mean = 0. The acf and pacf indicates an insignificant autocorrelation in the residuals. Hence, we proceed to forecast with this model.

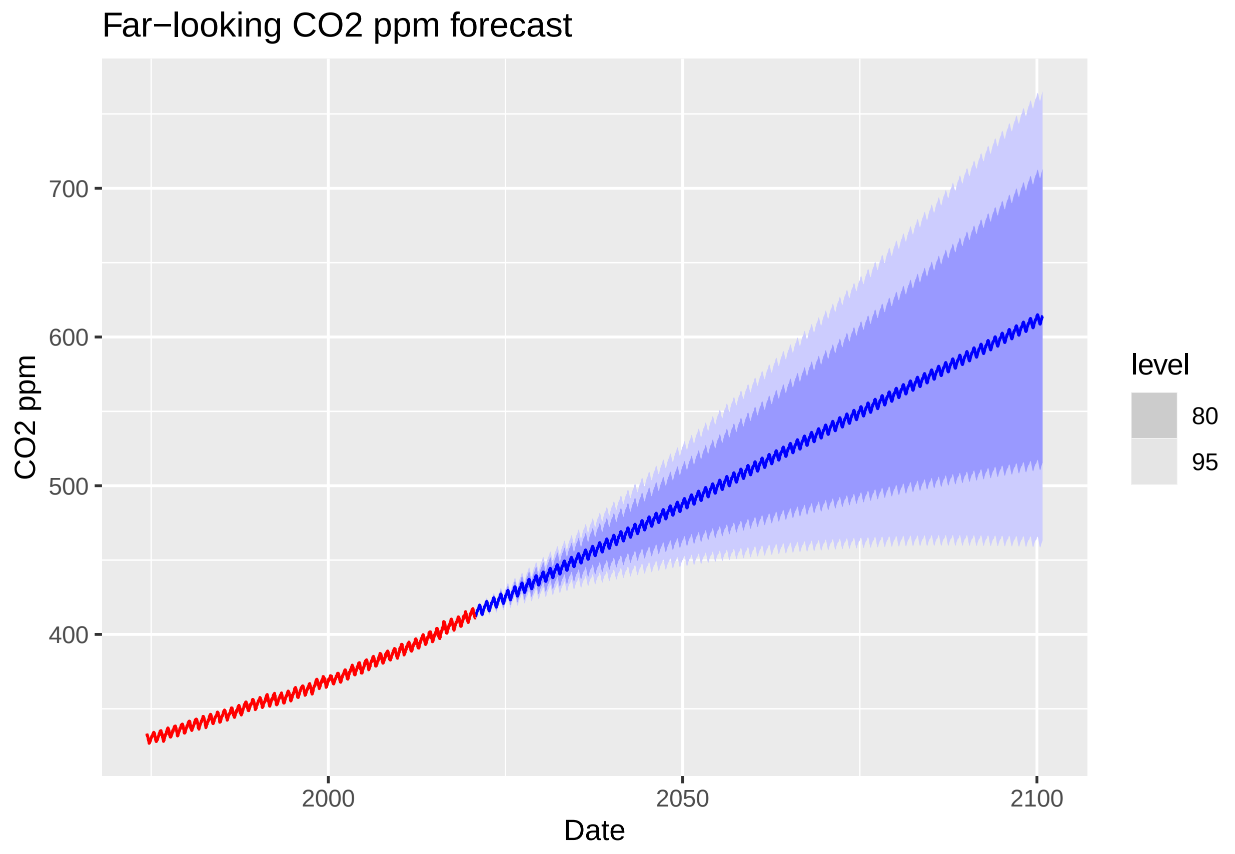
```r
selected_mod <- co2_weekly_models %>% dplyr::select(mod3)
selected_mod %>% gg_tsresiduals(lag_max = 104)
```



The model forecasts atmospheric CO2 level will continue in a linear fashion with same seasonality pattern as previous. The 95% confidence interval of the forecasted values expands wider as time goes to 2100 because there will be more uncertainties that are not captured within the model. As the forecast goes further ahead, we are less confident that the CO2 level would stay around our mean forecasted values.

Considering the 95% confidence intervals, we expect the atmospheric CO2 level to cross the 420ppm threshold for the first time in and for the last time in . We also expect that the atmospheric CO2 level crosses the 500ppm threshold for the first time in , and perhaps never reaches 500ppm level from now to 2100.

```r
#now forecast forward to end of 2100:
co2_forecast <- forecast(selected_mod, h=52*(2100-2020)+15, level = 95)

autoplot(co2_forecast) +
  autolayer(co2_weekly_tsibble, color = 'red', .vars = CO2_ppm) +
  ylab("CO2 ppm") + xlab('Date') +
  ggtitle('Far-looking CO2 ppm forecast')
```



```r
# first mean above 420:
colnames(co2_forecast)[4] = 'Mean'
co2_forecast %>% filter(Mean >= 420) %>%  head(1) %>% pull(Date)
```

```
## [1] "2022-03-20"
```

```r
# first mean above 500:
co2_forecast %>% filter(Mean >= 500) %>%  head(1) %>% pull(Date)
```

```
## [1] "2054-02-22"
```

```r
# loop to work with weird distribution in co2_forecast:
co2_forecast <- co2_forecast %>% mutate(lower_ci = 0, upper_ci = 0)
```

```r
for (i in 1:nrow(co2_forecast)) {

  co2_forecast$lower_ci[i] <- qnorm(0.025, co2_forecast$CO2_ppm[[i]][1]$mu, co2_forecast$CO2_pp
  co2_forecast$upper_ci[i] <- qnorm(0.975, co2_forecast$CO2_ppm[[i]][1]$mu, co2_forecast$CO2_pp

}

# date range for crossing 420 ppm:
first_date <- co2_forecast %>% filter(upper_ci >= 420) %>%  head(1) %>% pull(Date)
last_date <- co2_forecast %>% filter(lower_ci >= 420) %>%  head(1) %>% pull(Date)
print(paste0('CO2 level will cross the 420ppm threshold between ', first_date, ' and ', last_da
```

```
## [1] "CO2 level will cross the 420ppm threshold between 2021-04-04 and 2023-04-02"
```

```r
# date range for crossing 500 ppm:
first_date <- co2_forecast %>% filter(upper_ci >= 500) %>%  head(1) %>% pull(Date)
last_date <- co2_forecast %>% filter(lower_ci >= 500) %>%  head(1) %>% pull(Date)
print(paste0('CO2 level will cross the 500ppm threshold between ', first_date, ' and ', last_da
```

```
## [1] "CO2 level will cross the 500ppm threshold between 2043-03-08 and "
```

```r
# We don't seem likely to ever reach 95% confidence of going above 500 ppm at this point
```