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

James Darmody, Maria Vasiliadis and Gaurav Desai

```r
library(knitr)
opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)

# Start with a clean R environment
rm(list = ls())

# Set Fixed random seed to replicate the results
set.seed(28740)

library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```r
library(gridExtra)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following objects are masked from 'package:Hmisc':
##
##     src, summarize

## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(purrr)
library(imputeTS)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tsibble)
```

```
##
## Attaching package: 'tsibble'

## The following object is masked from 'package:dplyr':
##
##     id
```

```
library(fabletools)
library(feasts)
library(fable)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:tsibble':
##
##     interval, new_interval

## The following object is masked from 'package:base':
##
##     date
```

```
library(forecast)
```

```
##
## Attaching package: 'forecast'

## The following objects are masked from 'package:fabletools':
##
##     accuracy, forecast, GeomForecast, StatForecast
```

```
library(seasonal)
library(tsbox)
library(tibble)
```

```
##
## Attaching package: 'tibble'
## The following object is masked from 'package:seasonal':
##
##     view
```

```r
library(svMisc)
```

```
##
## Attaching package: 'svMisc'
```

```
## The following object is masked from 'package:utils':
##
##     ?
```

```r
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

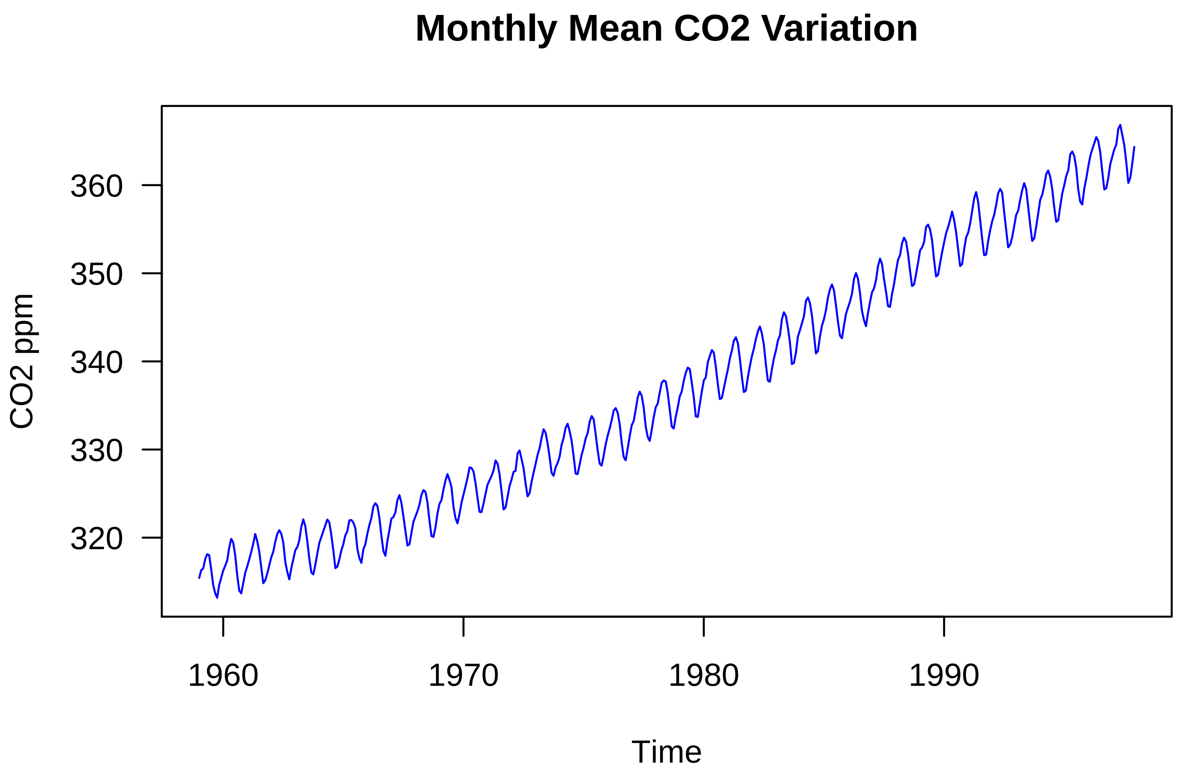
The Keeling Curve

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 was able to attribute this pattern to the difference in the amount of land area and vegetation cover between the northern and southern hemispheres, and the resulting variation in global rates of photosynthesis as the hemispheres' seasons alternated throughout the year.

In 1958 Keeling began continuous monitoring of atmospheric carbon dioxide concentrations from the Mauna Loa Observatory in Hawaii and soon observed a trend increase carbon dioxide levels in addition to the seasonal cycle. He was able to attribute this trend increase to growth in global rates of fossil fuel combustion. This trend 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")
```


Monthly Mean CO2 Variation
CO2 ppm
320
330
340
350
360
1960
1970
1980
1990
Time

Part 1 (4 points)

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.

Answer Lets first look at the data

```
head(co2)
```

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

```
tail(co2)
```

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

```
str(co2)
```

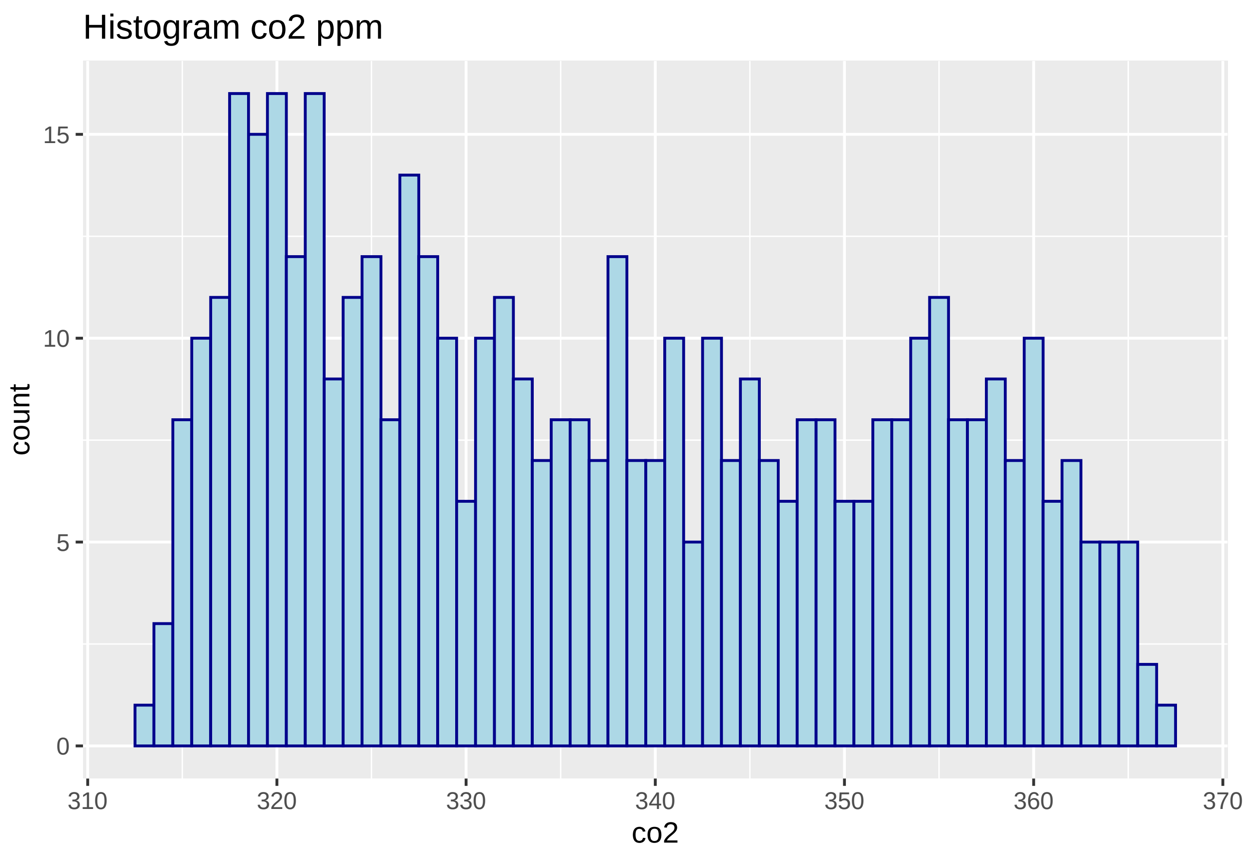
```
##  Time-Series [1:468] from 1959 to 1998: 315 316 316 318 318 ...
```

```
summary(co2)
```

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

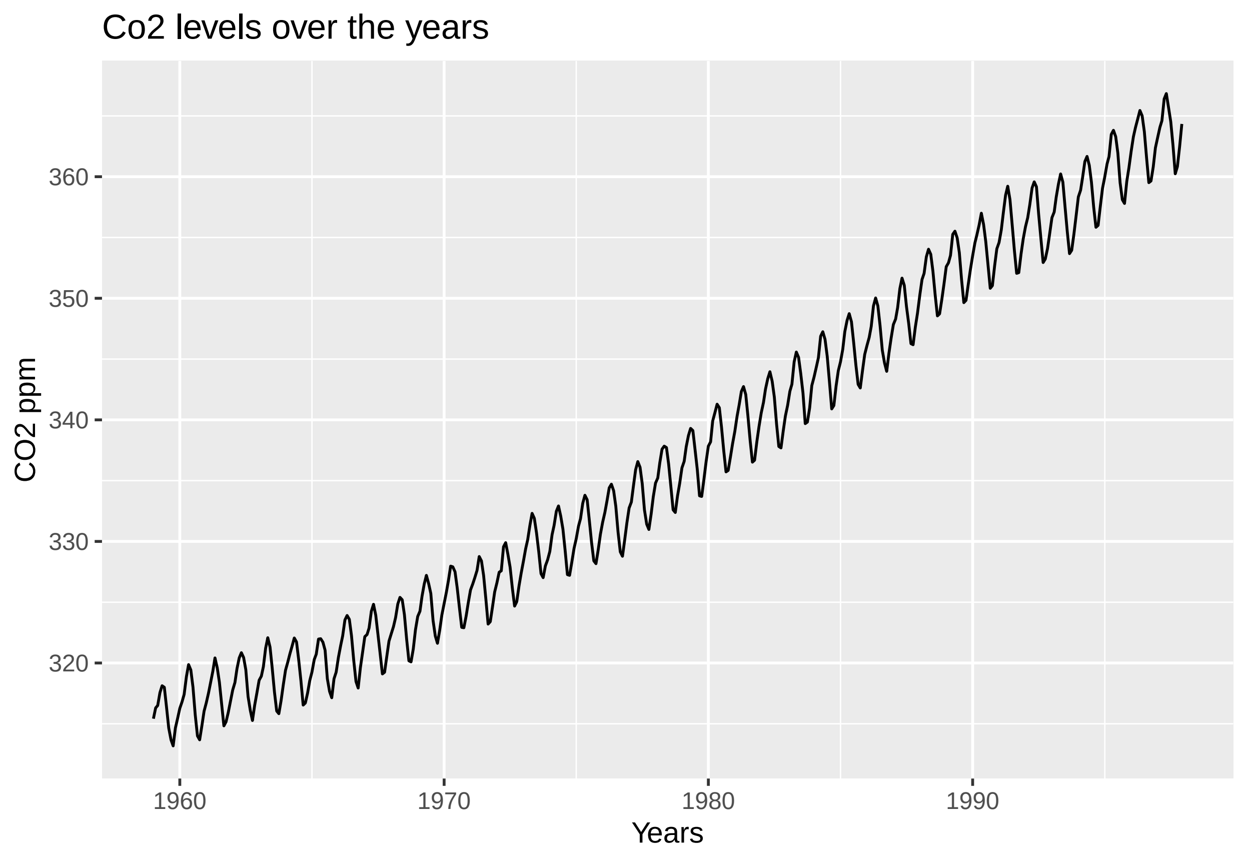
```
ggplot(co2, aes(x = co2)) + geom_histogram(color = "darkblue",
    fill = "lightblue", binwidth = 1) + labs(title = "Histogram co2 ppm")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



The co2 dataset is comprised of 468 observations of data from 1959 to 1998 where each datapoint corresponds to an observation at a month. The co2 levels in the dataset range from 313.2 to 366.8. The histogram shows us that the most common co2 levels are between 320 and 325, and generally the frequency of co2 values decreases as co2 goes up. There is a small spike of frequencies between co2 level 355 and 360. There are no missing values in the dataset. Histogram of value is left skewed but that may not be correct representation of the co2 emission levels over time. So lets look at it using time dimension.

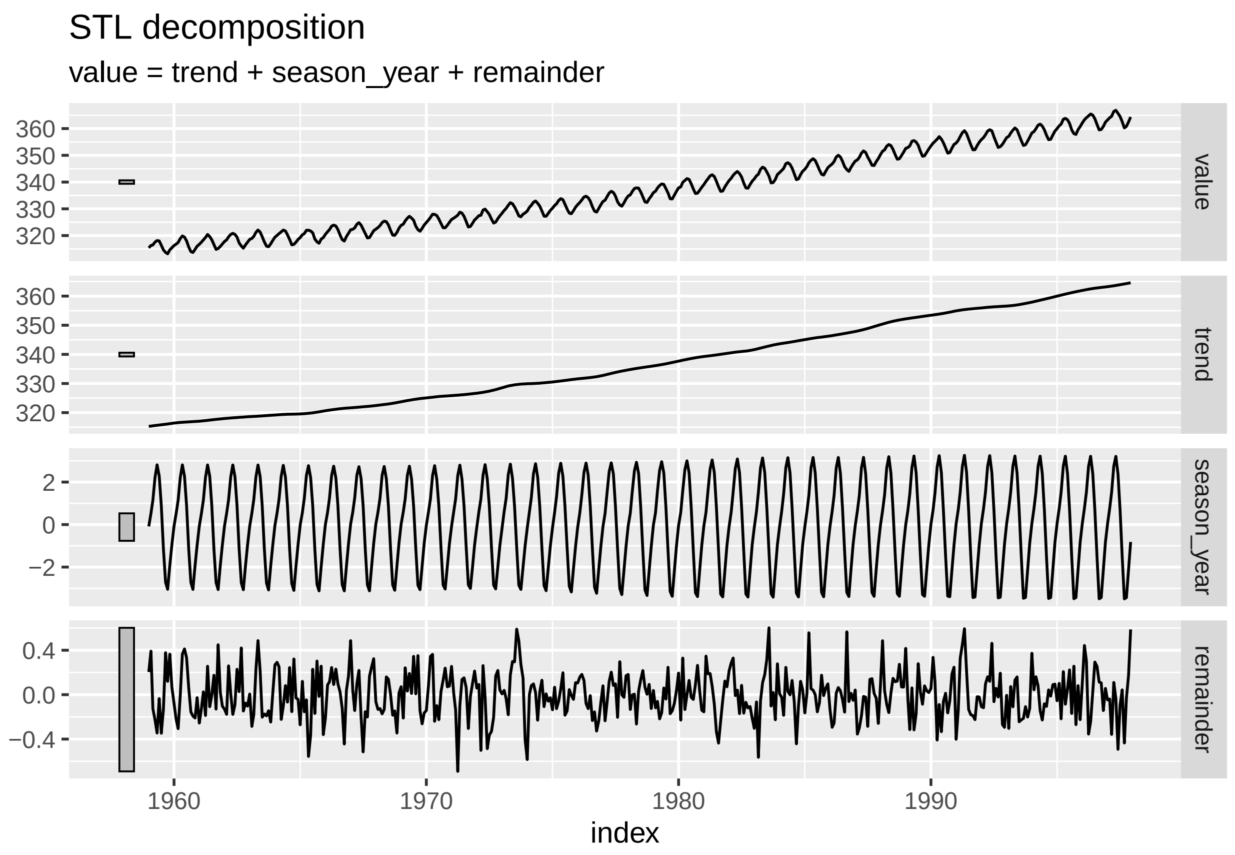
```
co2.ts <- as_tsibble(co2)
co2.ts %>% autoplot(.vars = value) + labs(title = "Co2 levels over the years") +
    xlab("Years") + ylab("CO2 ppm")
```



We observe two things 1. There is upward trend year on year in CO2 emissions 2. There is a seasonal fluctuation within a year

Lets try to decompose and see the individual components in more detail

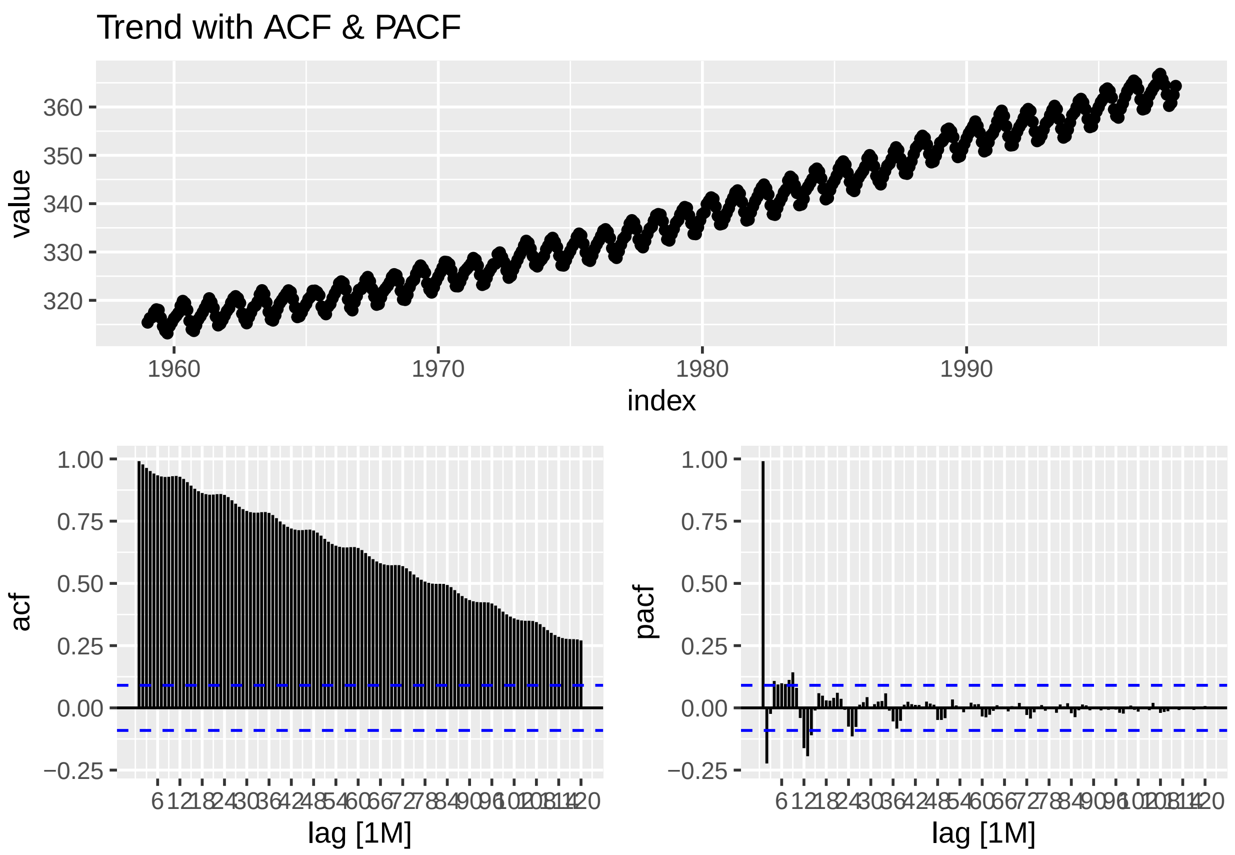
```
co2.ts %>% model(STL(value)) %>% components() %>% autoplot()
```



When we decompose using STL model, we can see the upward trend separated from yearly seasonal cycles.

We note that seasonal cycle ups and downs are growing with time, so we may need to stabilize the seasons for better predictions using box-cox transformation.

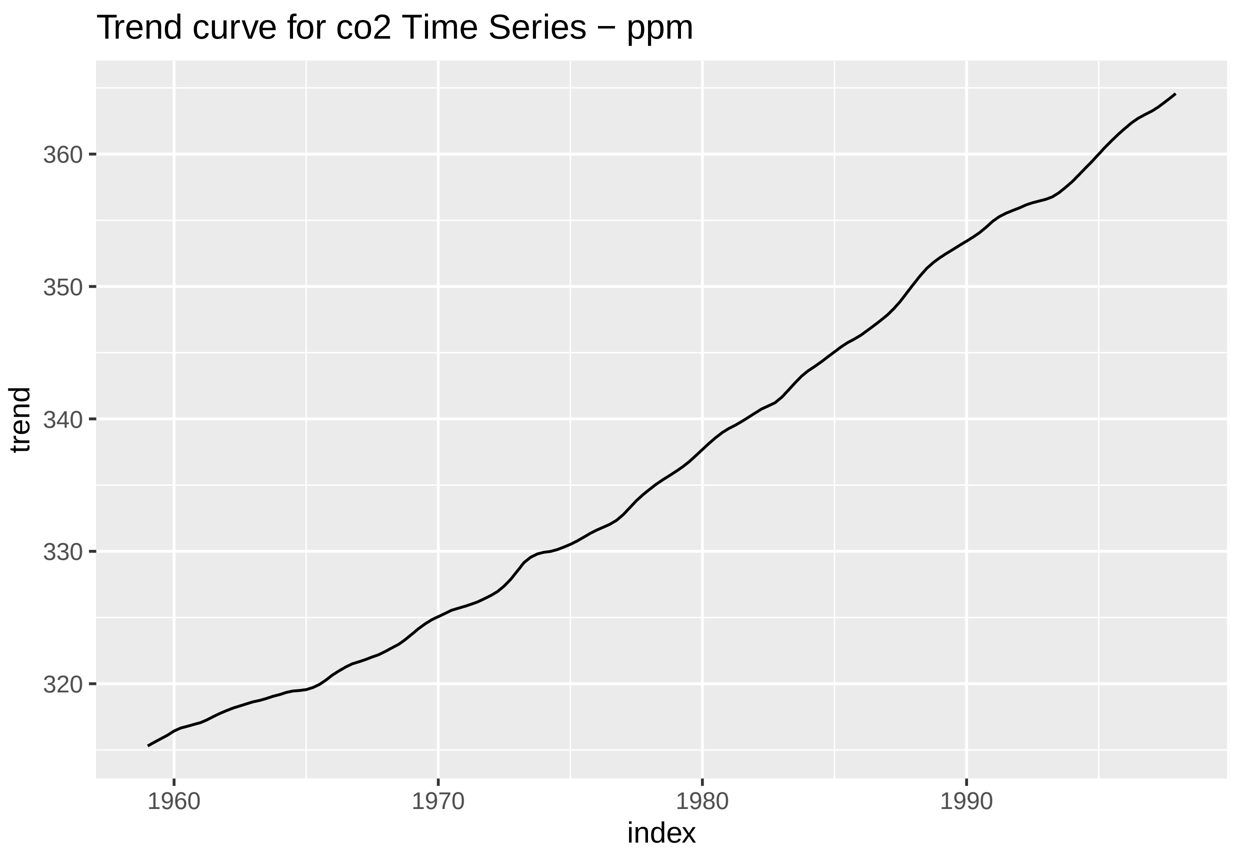
```r
co2.ts %>% gg_tsdisplay(lag_max = 120, plot_type = "partial",
    y = value) + ggtitle("Trend with ACF & PACF")
```



The ACF tells us that there is autocorrelation even after 24 months. On the other hand PACF drops after 2 but again pops up at multiples of 12 months. This indicates that the model could be AR model with 12 months seasonal cycles.

Now lets look at each component one by one starting with trend:

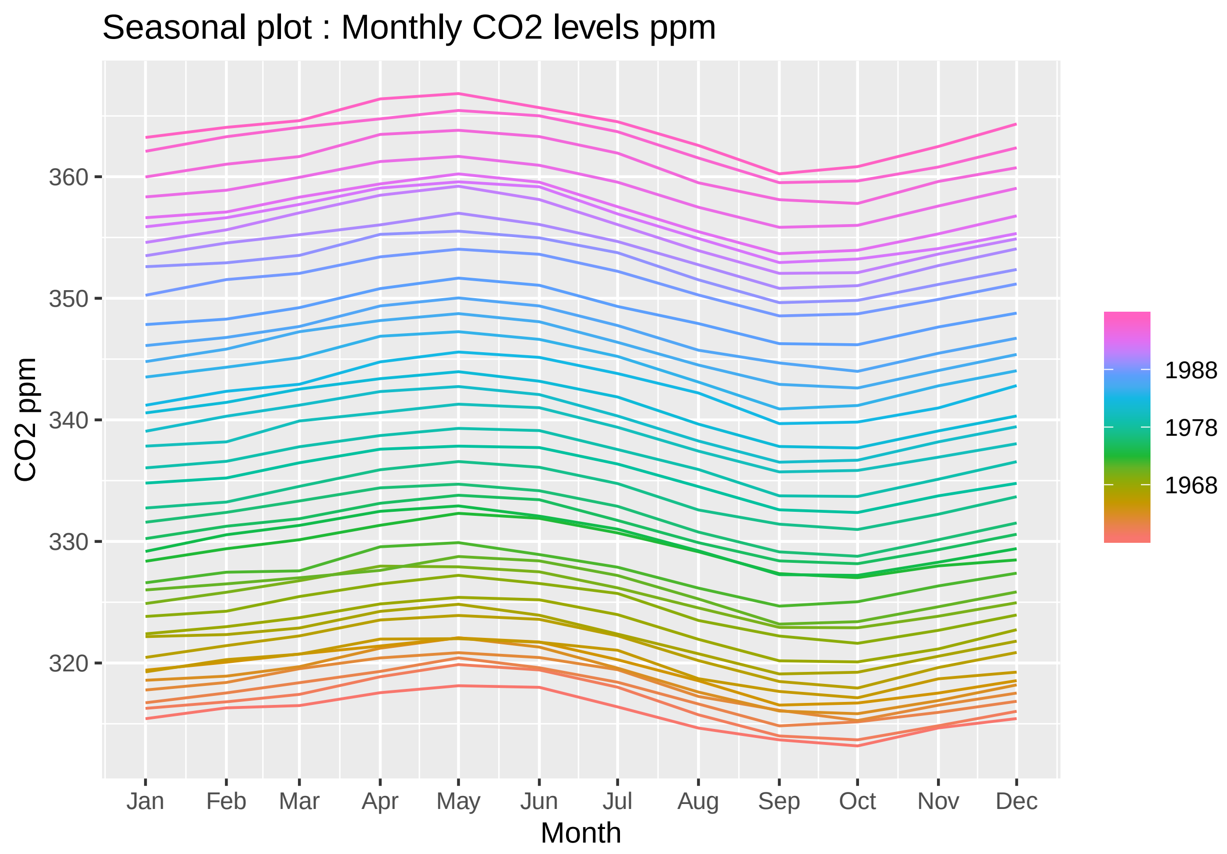
```
co2.ts.components <- co2.ts %>% model(STL(value)) %>% components()
ggplot(data = co2.ts.components) + geom_line(aes(x = index, y = trend)) +
    ggtitle("Trend curve for co2 Time Series - ppm")
```



There are minor bumps and dips but there is no major shock or sudden movement in trend that we need to take care while building a model.

Now lets look at seasonal cycles.

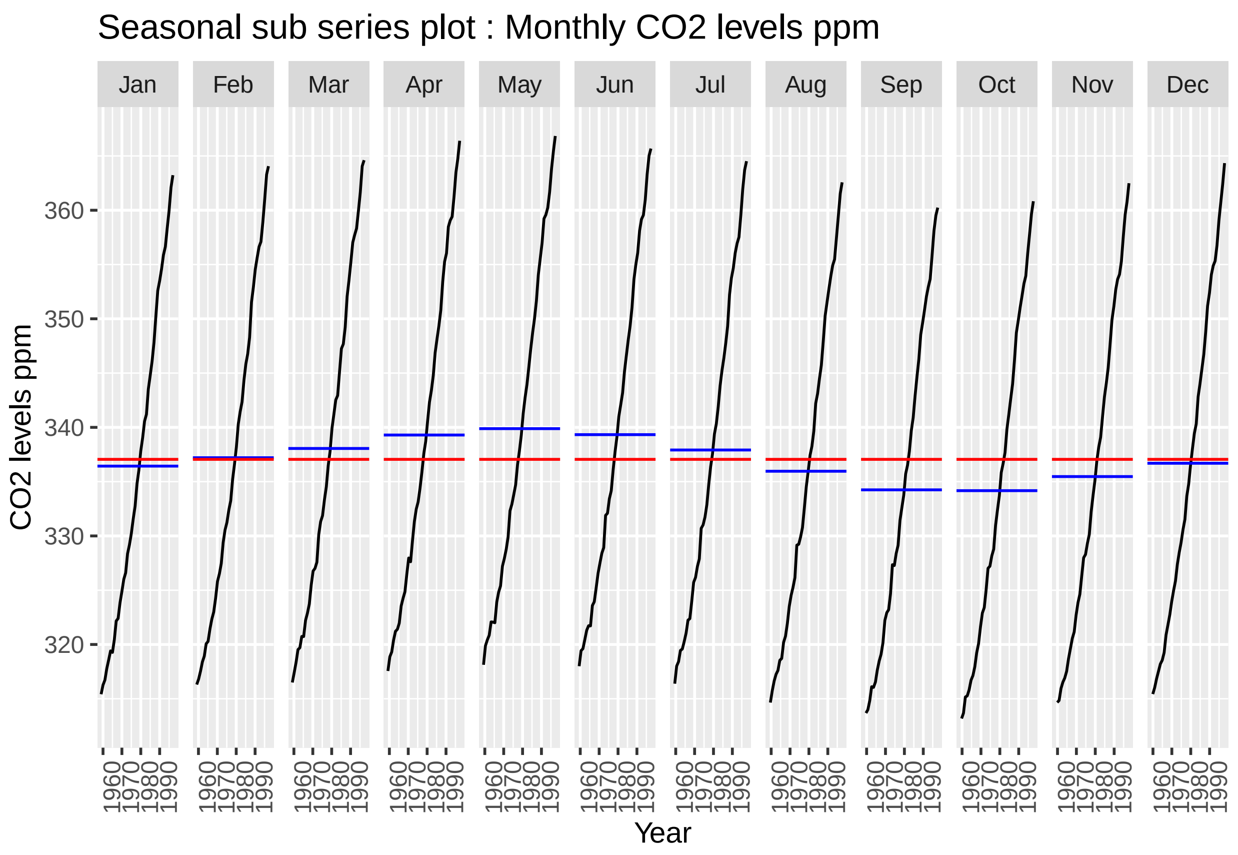
```
co2.ts %>% gg_season(y = value, period = "year") + ylab("CO2 ppm") +
    xlab("Month") + ggtitle("Seasonal plot : Monthly CO2 levels ppm")
```



We can clearly see that CO2 emission is higher around April-May and goes down around September-October every year. This could be due to summer in Northern hemisphere vs southern hemisphere. Population, vegetation and other attributes of these two halves of earth are quite different from each other and could potentially cause this seasonal effect.

Now lets look at same seasonal data from another perspective

```
co2.ts %>% gg_subseries(y = value, period = "year") + geom_hline(aes(yintercept = mean(co2.ts$
    colour = "red") + ylab("CO2 levels ppm") + xlab("Year") +
    ggtitle("Seasonal sub series plot : Monthly CO2 levels ppm")
```

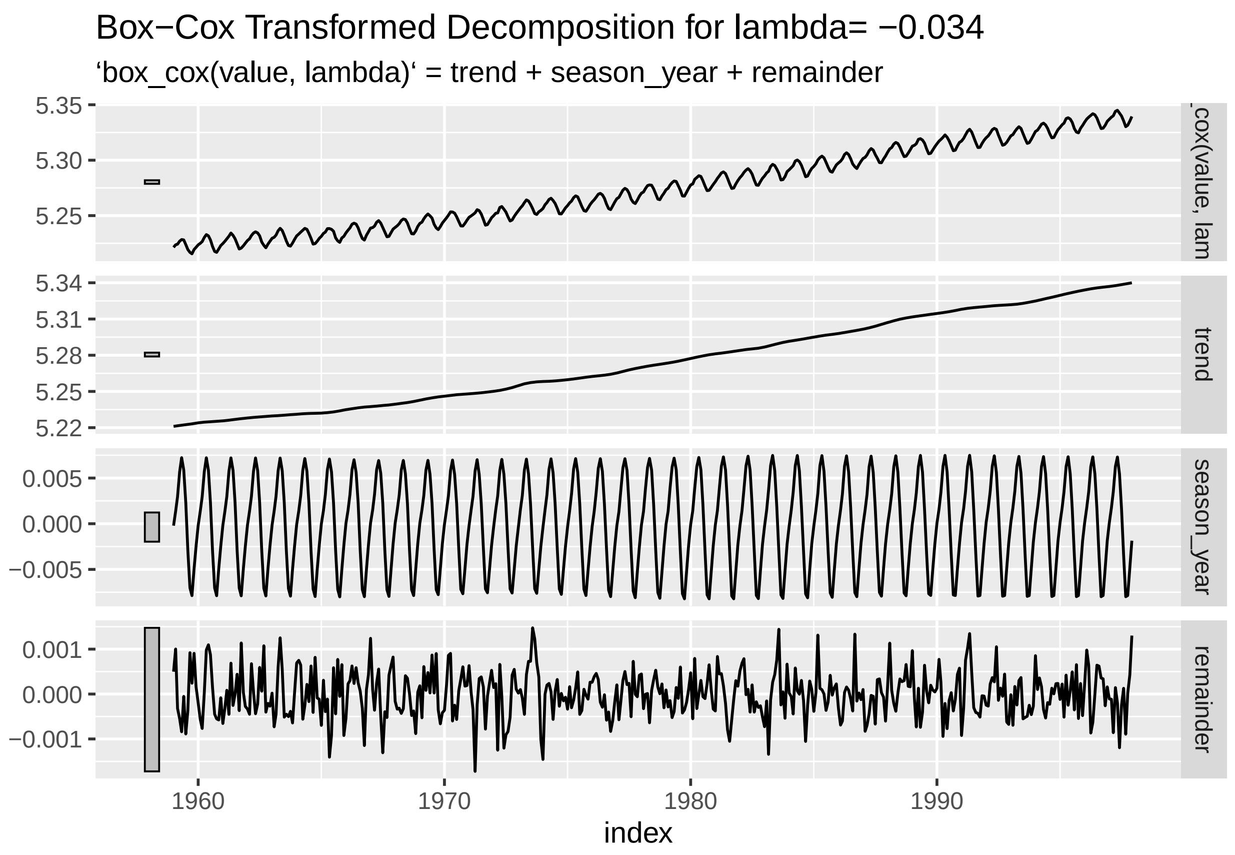


Following the difference between blue and red lines tells us more or less same story that northern summer increases CO2 levels and southern summer decreases CO2 levels within a year. But we can clearly see that CO2 emission is on the rise year on year for each and every month.

Since we saw earlier that seasonal patterns are increasing in variance, we can try to transform it so the seasonal changes are uniform across time series. This is desirable property for finding a good model fit.

Lets try Box-Cox transformation

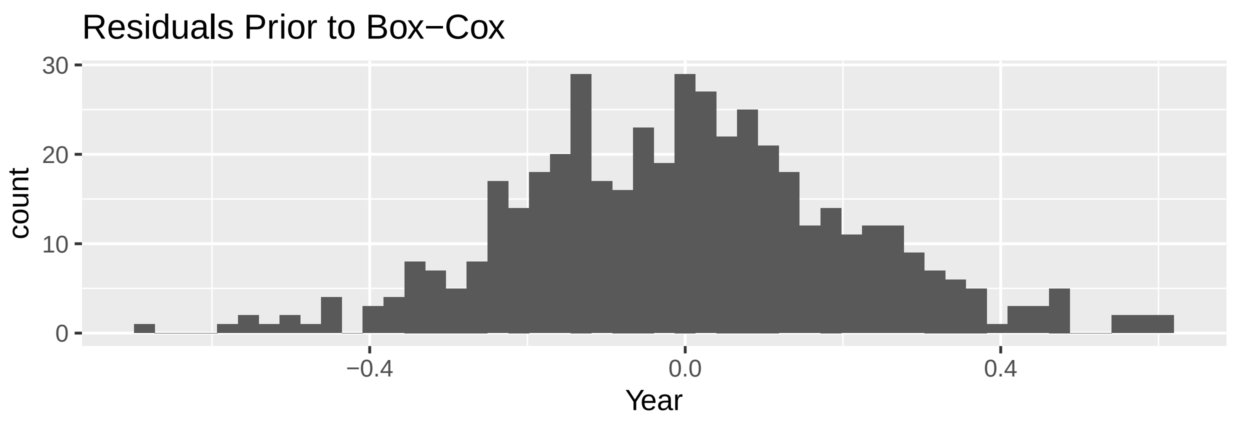
```
lambda <- co2.ts %>% features(value, features = guerrero) %>%
    pull(lambda_guerrero)

co2.ts.trans <- as_tsibble(ts(box_cox(co2.ts$value, lambda),
    start = c(1959, 1)))
co2.ts.trans.comp <- co2.ts %>% model(STL(box_cox(value, lambda)))

co2.ts.trans.comp %>% components() %>% autoplot() + ggtitle(paste("Box-Cox Transformed Decompos
    round(lambda, 3)))
```

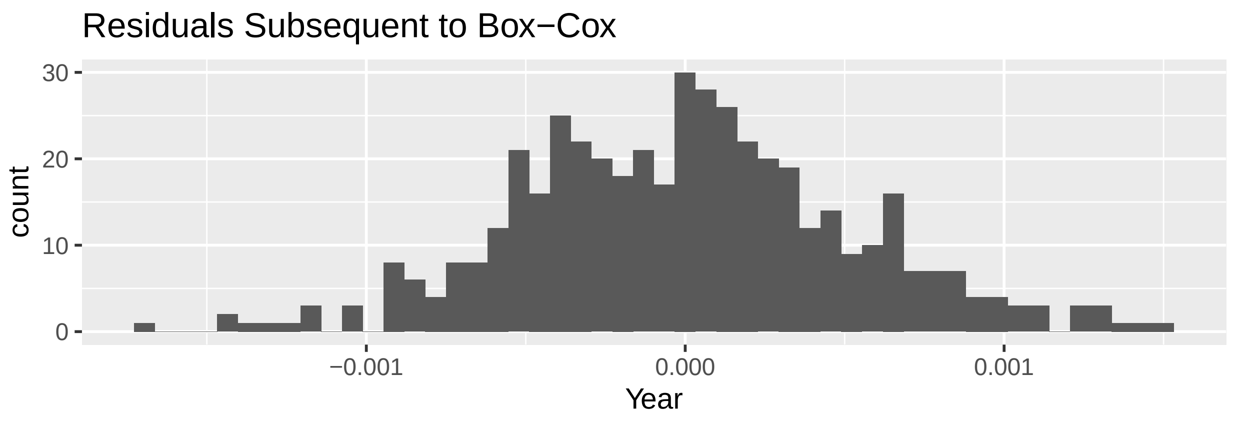


After transformation we can see the seasonal pattern variance remains constant throughout.

Now lets look at residuals before and after transformation:

```r
# Before box-cox transformation
plot.co2.ts.components <- co2.ts.components %>% select("remainder",
    index = "index") %>% ggplot(aes(x = remainder)) + geom_histogram(bins = 50) +
    xlab("Year") + ggtitle("Residuals Prior to Box-Cox")

# After box-cox transformation
plot.co2.ts.trans.comp <- co2.ts.trans.comp %>% components() %>%
    select("remainder", index = "index") %>% ggplot(aes(x = remainder)) +
    geom_histogram(bins = 50) + xlab("Year") + ggtitle("Residuals Subsequent to Box-Cox")

grid.arrange(plot.co2.ts.components, plot.co2.ts.trans.comp,
    nrow = 2)
```

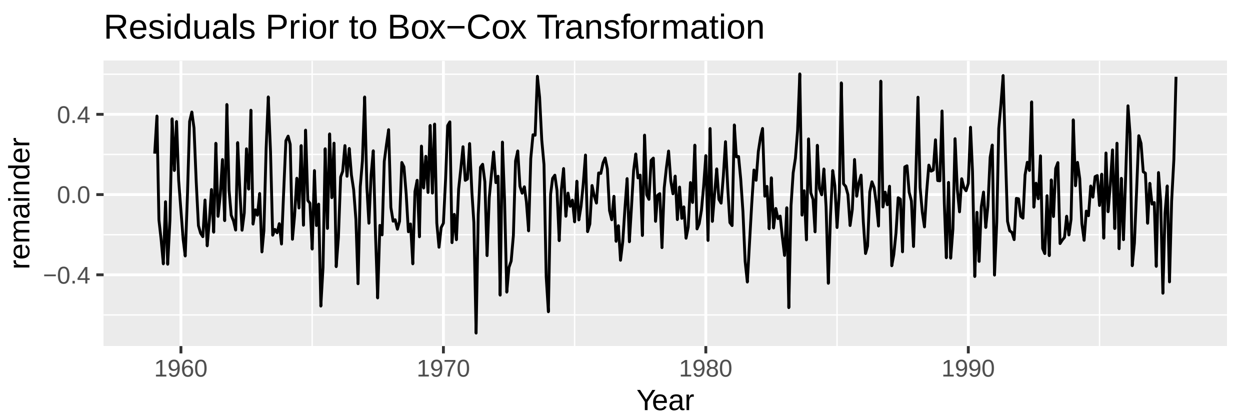


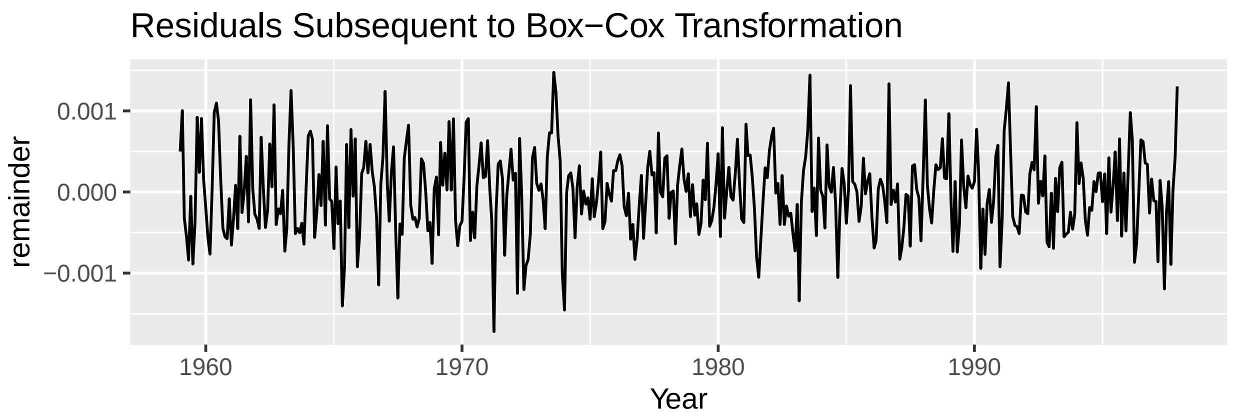


Both before and after box-cox transformation residuals look fairly normally distributed.

Now lets see the residuals over time

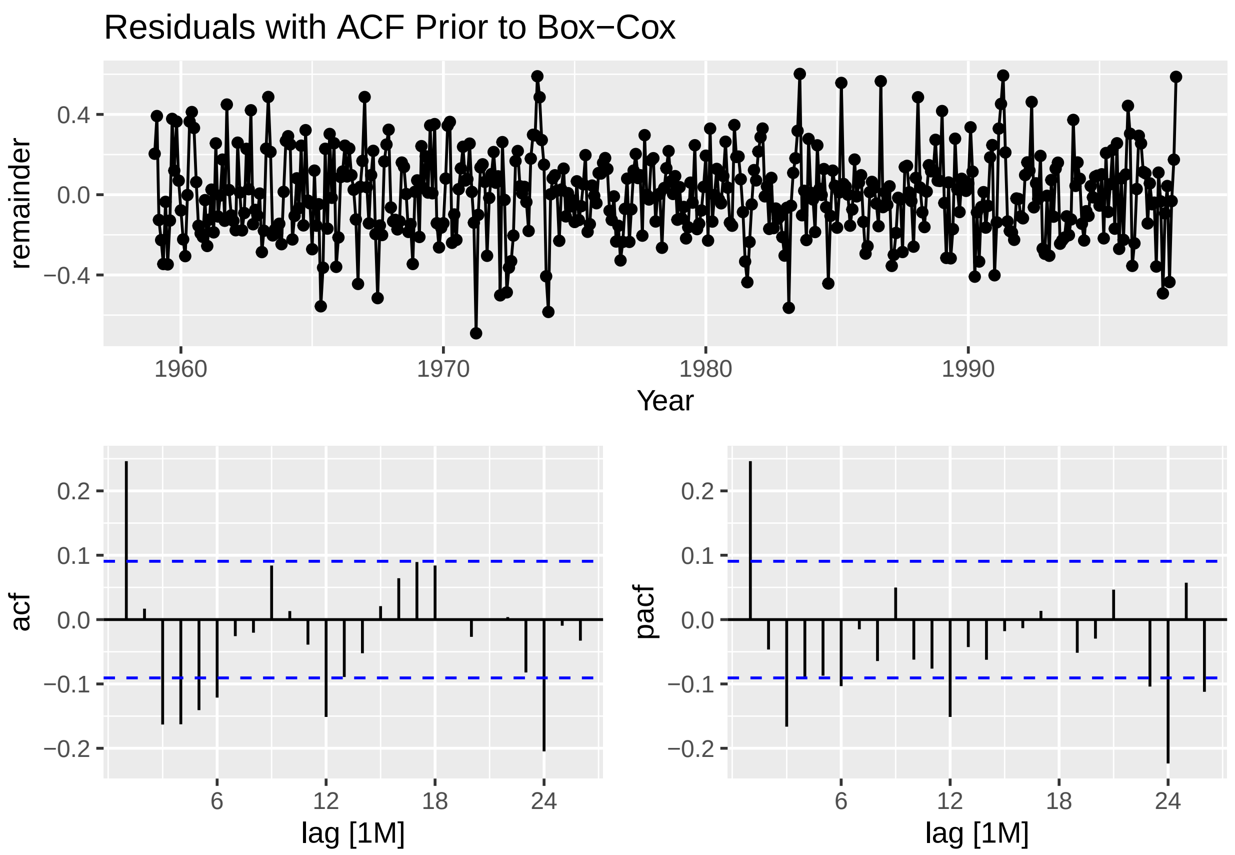
```r
plot.resid.co2.ts.components <- co2.ts.components %>% select("remainder",
    index = "index") %>% autoplot(.vars = remainder) + ggtitle("Residuals Prior to Box-Cox Tra
    xlab("Year")

plot.resid.co2.ts.trans.comp <- co2.ts.trans.comp %>% components() %>%
    select("remainder", index = "index") %>% autoplot(.vars = remainder) +
    ggtitle("Residuals Subsequent to Box-Cox Transformation") +
    xlab("Year")

grid.arrange(plot.resid.co2.ts.components, plot.resid.co2.ts.trans.comp,
    nrow = 2)
```



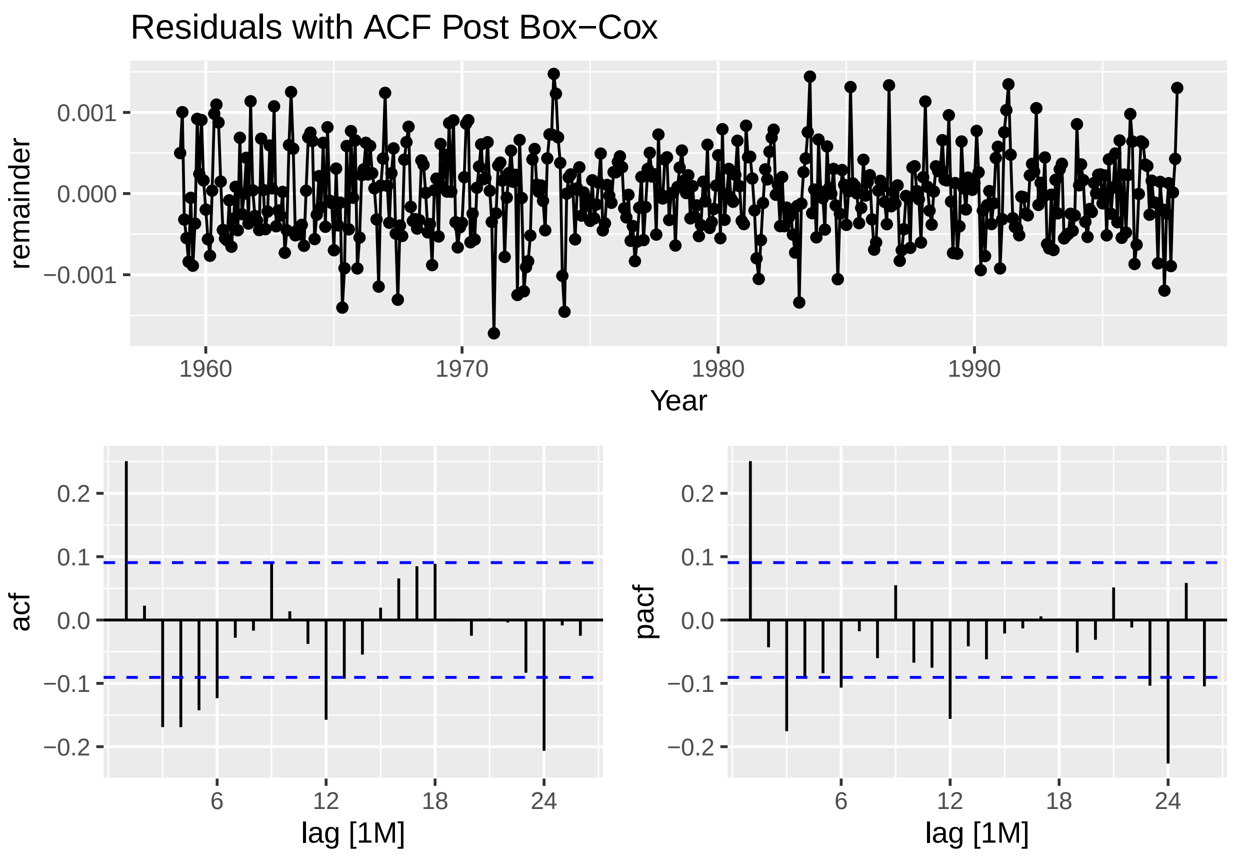


Residuals appear to be stationary and appear to be white noise with no trend or seasonal patterns. Residual variance has decreased post transformation.

```
co2.ts.components %>% select("remainder", index = "index") %>%
    gg_tsdisplay(plot_type = "partial", y = remainder) + ggtitle("Residuals with ACF Prior to 
    xlab("Year")
```



```
co2.ts.trans.comp %>% components() %>% select("remainder", index = "index") %>%
    gg_tsdisplay(plot_type = "partial", y = remainder) + ggtitle("Residuals with ACF Post Box-(
    xlab("Year")
```



Part 2 (3 points)

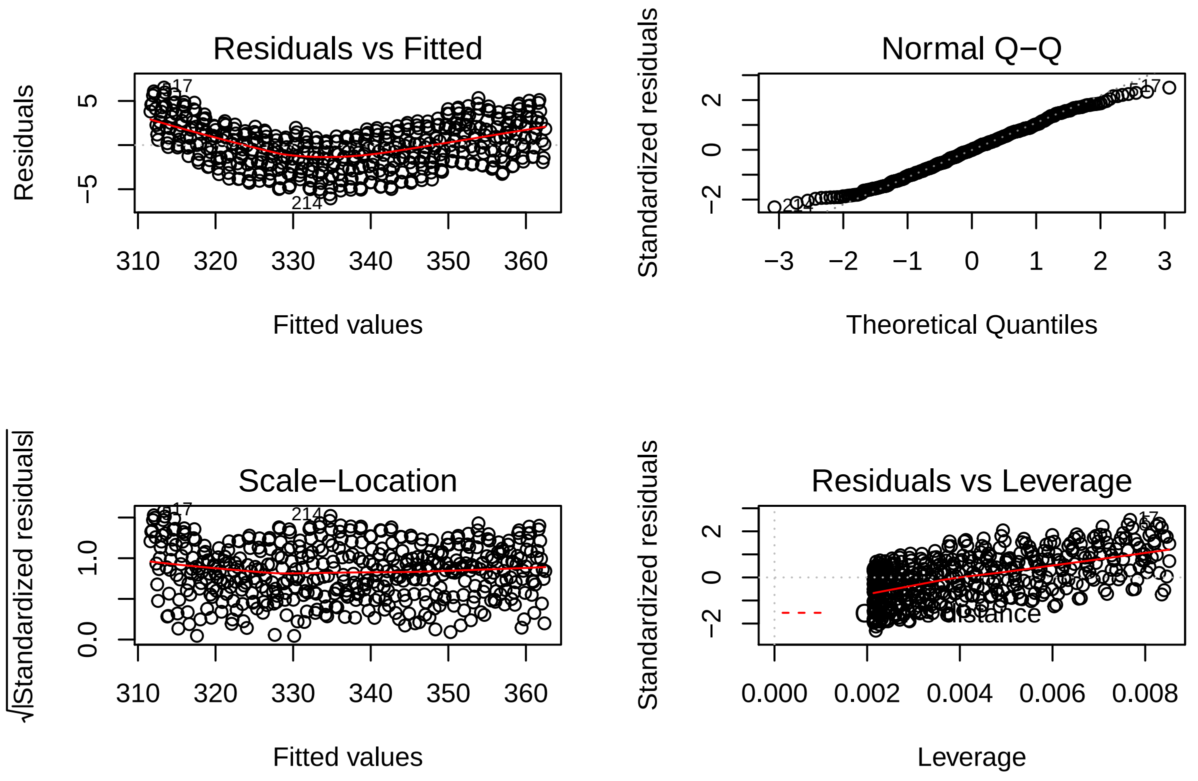
Fit a linear time trend model to the `co2` series, and examine the characteristics of the residuals. Compare this to a quadratic time trend model. Discuss whether a logarithmic transformation of the data would be appropriate. Fit a suitable polynomial time trend model that incorporates seasonal dummy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

Lets first fit a linear time trend model:

```r
time.index <- 1:length(co2.ts$index)
mod.ln.1 <- lm(formula = value ~ time.index, data = co2.ts)
summary(mod.ln.1)
```

```
##
## Call:
## lm(formula = value ~ time.index, data = co2.ts)
##
## 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
```

```r
par(mfrow = c(2, 2))
plot(mod.ln.1)
```

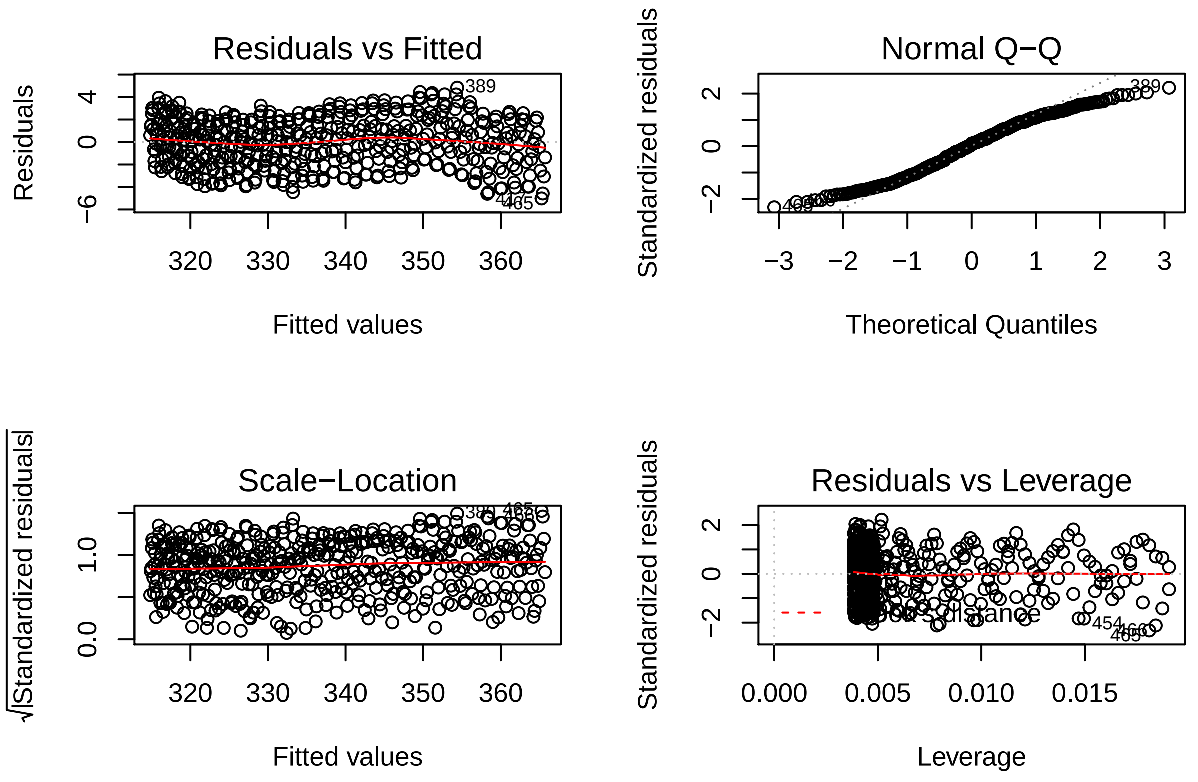


The residuals show a pattern indicating linear model is not able to capture the curvature of the trend. Lets try quadratic term:

```r
mod.ln.2 <- lm(formula = value ~ time.index + I(time.index^2),
    data = co2.ts)
summary(mod.ln.2)
```

```
##
## Call:
## lm(formula = value ~ time.index + I(time.index^2), data = co2.ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.148e+02  3.039e-01 1035.65   <2e-16 ***
## time.index       6.739e-02  2.993e-03   22.52   <2e-16 ***
## I(time.index^2) 8.862e-05  6.179e-06   14.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.182 on 465 degrees of freedom
## Multiple R-squared:  0.9788, Adjusted R-squared:  0.9787
## F-statistic: 1.075e+04 on 2 and 465 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2, 2))
plot(mod.ln.2)
```

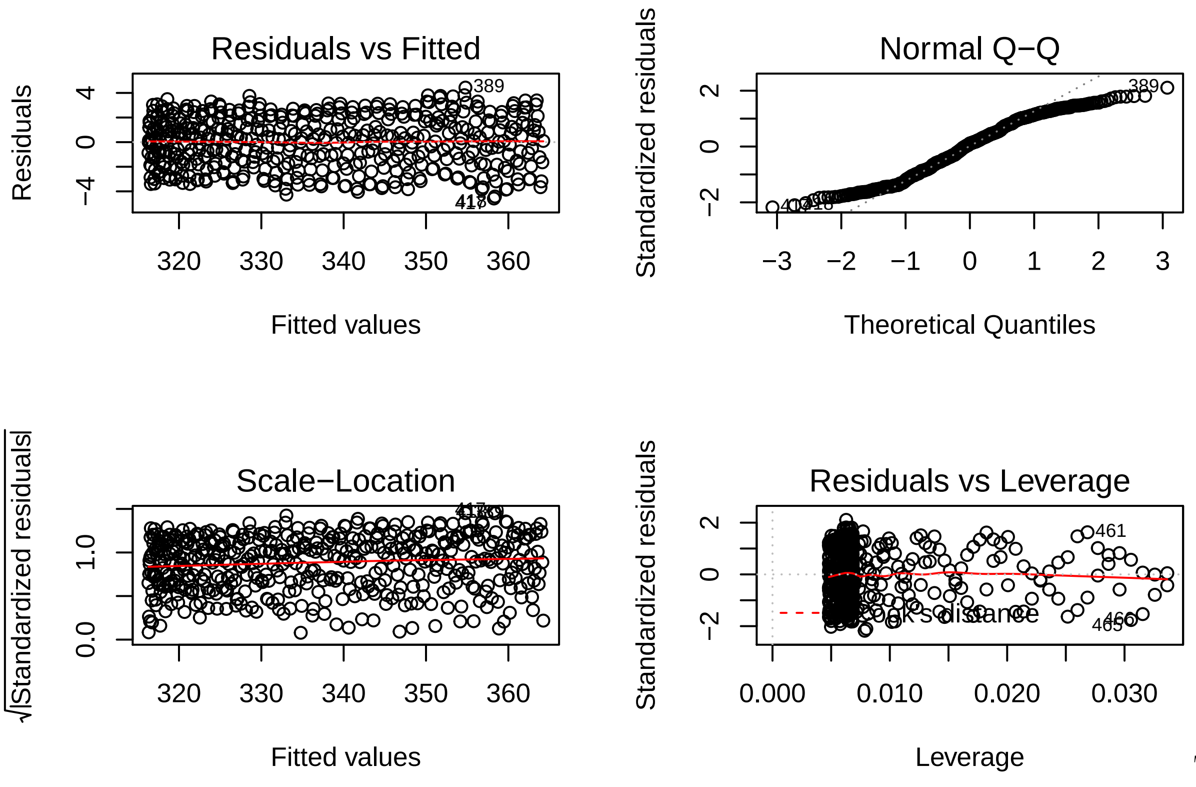


The residuals are showing a much better fit. Now lets try cubic term and see if we get any more improvements.

```
mod.ln.3 <- lm(formula = value ~ time.index + I(time.index^2) +
    I(time.index^3), data = co2.ts)
summary(mod.ln.3)
```

```
##
## Call:
## lm(formula = value ~ time.index + I(time.index^2) + I(time.index^3),
##     data = co2.ts)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5786 -1.7299  0.2279  1.8073  4.4318
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.163e+02  3.934e-01 804.008  < 2e-16 ***
## time.index          2.905e-02  7.256e-03   4.004 7.25e-05 ***
## I(time.index^2)     2.928e-04  3.593e-05   8.149 3.44e-15 ***
## I(time.index^3)    -2.902e-07  5.036e-08  -5.763 1.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 464 degrees of freedom
## Multiple R-squared:  0.9802, Adjusted R-squared:  0.9801
## F-statistic:  7674 on 3 and 464 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2, 2))
plot(mod.ln.3)
```

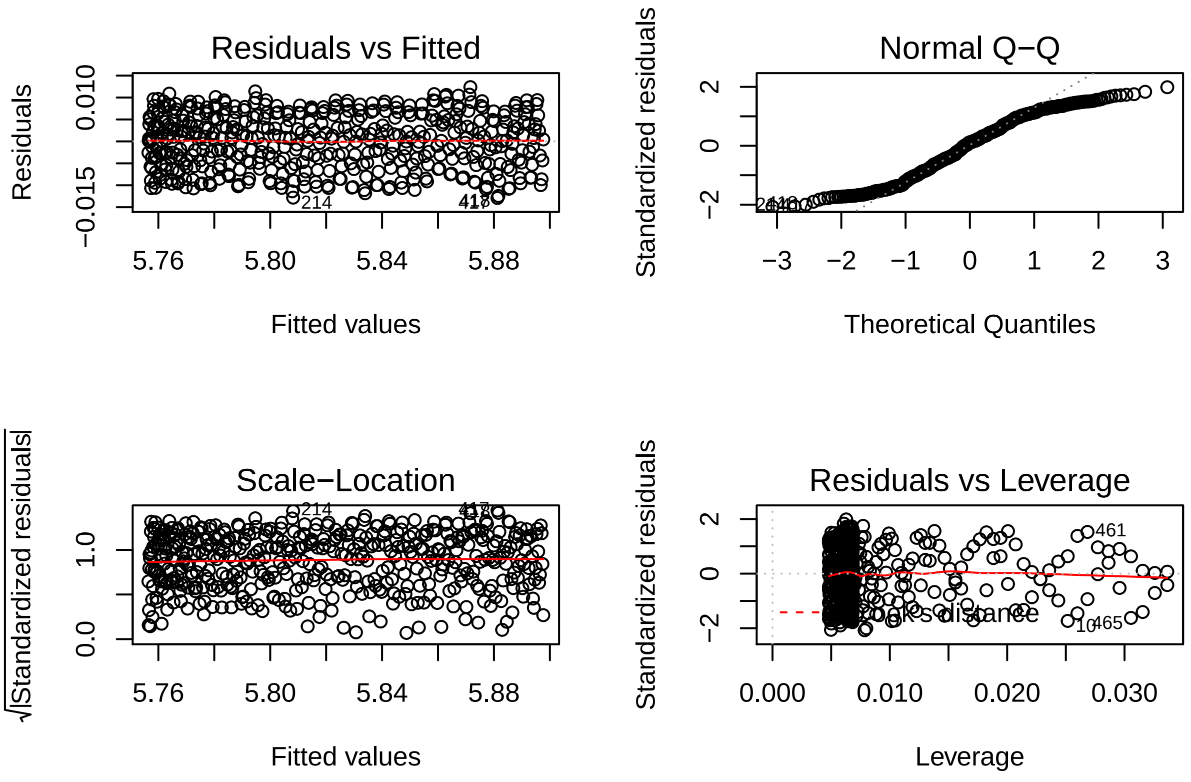


This model is even better fit, but Q-Q plot shows deviation at start and end. So lets try taking log of value and see if that scale fit any better.

```r
mod.ln.4 <- lm(formula = log(value) ~ time.index + I(time.index^2) +
    I(time.index^3), data = co2.ts)
summary(mod.ln.4)
```

```
##
## Call:
## lm(formula = log(value) ~ time.index + I(time.index^2) + I(time.index^3),
##     data = co2.ts)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0128986 -0.0051482  0.0007055  0.0054841  0.0123625
##
## Coefficients:
##                    Estimate Std. Error  t value Pr(>|t|)
## (Intercept)       5.756e+00  1.163e-03 4948.350  < 2e-16 ***
## time.index        9.899e-05  2.146e-05    4.613 5.13e-06 ***
## I(time.index^2)   8.679e-07  1.063e-07    8.168 2.99e-15 ***
## I(time.index^3)  -9.283e-10  1.489e-10   -6.233 1.03e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.006241 on 464 degrees of freedom
## Multiple R-squared:  0.9802, Adjusted R-squared:  0.9801
## F-statistic:  7661 on 3 and 464 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2, 2))
plot(mod.ln.4)
```



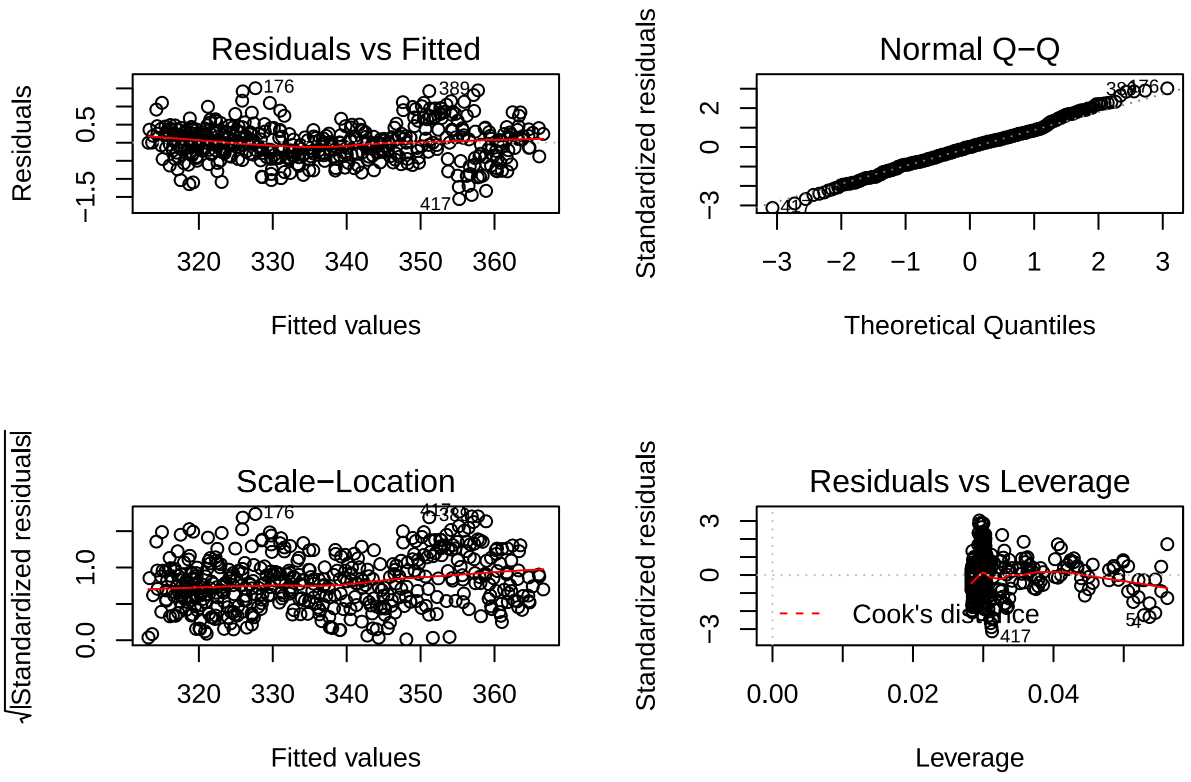
Taking log doesn't improve much.

So finally lets choose the best of the models, which so far is cubic model, and add seasonal dummy variables for January to December:

```
co2.df <- data.frame(value = co2.ts$value, time.index = time.index,
    season = factor(time.index%%12, ordered = F))
mod.ln.5 <- lm(formula = value ~ 0 + time.index + I(time.index^2) +
    I(time.index^3) + season, data = co2.df)
par(mfrow = c(2, 2))
summary(mod.ln.5)
```

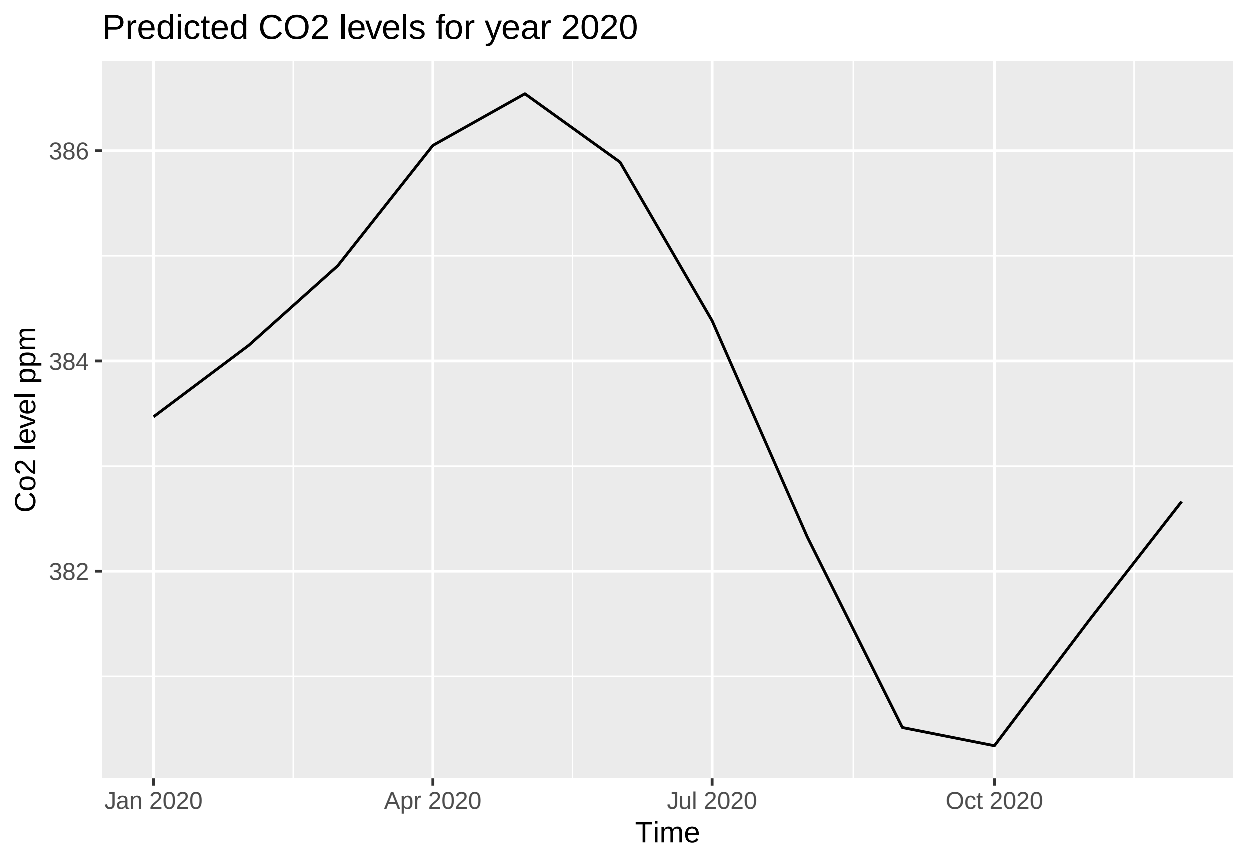
```
##
## Call:
## lm(formula = value ~ 0 + time.index + I(time.index^2) + I(time.index^3) +
##     season, data = co2.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5573 -0.3312  0.0008  0.2880  1.5040
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## time.index       3.275e-02  1.740e-03   18.83   <2e-16 ***
## I(time.index^2)  2.744e-04  8.614e-06   31.85   <2e-16 ***
## I(time.index^3) -2.640e-07  1.207e-08  -21.86   <2e-16 ***
## season0          3.151e+02  1.231e-01 2559.39   <2e-16 ***
## season1          3.160e+02  1.210e-01 2611.63   <2e-16 ***
## season2          3.167e+02  1.212e-01 2612.90   <2e-16 ***
## season3          3.174e+02  1.214e-01 2614.84   <2e-16 ***
## season4          3.186e+02  1.216e-01 2619.99   <2e-16 ***
## season5          3.191e+02  1.218e-01 2619.77   <2e-16 ***
## season6          3.184e+02  1.220e-01 2610.22   <2e-16 ***
## season7          3.169e+02  1.222e-01 2593.69   <2e-16 ***
## season8          3.148e+02  1.224e-01 2572.76   <2e-16 ***
## season9          3.130e+02  1.226e-01 2553.89   <2e-16 ***
## season10         3.128e+02  1.228e-01 2548.46   <2e-16 ***
## season11         3.140e+02  1.229e-01 2554.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5056 on 453 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.389e+07 on 15 and 453 DF,  p-value: < 2.2e-16
```

```
plot(mod.ln.5)
```



This model has better R-square than any of the previous models. Lets use this model to forecast for year 2020:

```
time.index.2020 = seq(from = (2020 - 1959) * 12 + 1, length.out = 12)
co2.2020.df <- data.frame(time.index = time.index.2020, season = factor(time.index.2020%%12,
    ordered = F))

co2.2020.pred <- predict(object = mod.ln.5, newdata = co2.2020.df)

co2.2020.pred.ts <- as_tsibble(ts(data = co2.2020.pred, start = c(2020,
    1), frequency = 12))

co2.2020.pred.ts %>% autoplot(.vars = value) + labs(title = "Predicted CO2 levels for year 202
    xlab("Time") + ylab("Co2 level ppm")
```



Unsurprisingly, the 2020 prediction follows the same seasonal trend as the other years, with a max co2 level of ~386.5 and a minimum value of ~380.5. **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. Write your model (or models) using backshift notation. Use your model (or models) to generate forecasts to the year 2020.

To fit an ARIMA model, we write a function that will evaluate different ARIMA combinations of p, d, q, P, D, and Q by calculating aic, aicc, and bic values for each combination. Because of the large number of models that can be fit, we will have maximum values of 2 for p, q, P, and Q, and a maximum value of 1 for d and D.

```
no.p = no.P = 2
no.d = no.D = 1
no.q = no.Q = 2
no.of.models <- (no.p + 1) * (no.d + 1) * (no.q + 1) * (no.P +
    1) * (no.D + 1) * (no.Q + 1)

print(paste("Number of models to fit =", no.of.models))
```

```
## [1] "Number of models to fit = 324"
```

```
params.df <- expand.grid(p = 0:no.p, d = 0:no.d, q = 0:no.q,
    P = 0:no.P, D = 0:no.D, Q = 0:no.Q)
i <- 1
```

```r
funFitModel <- function(param.row) {
    progress(value = i, max.value = no.of.models, console = TRUE,
        progress.bar = TRUE)
    i <- i + 1
    p = param.row["p"]
    d = param.row["d"]
    q = param.row["q"]
    P = param.row["P"]
    D = param.row["D"]
    Q = param.row["Q"]

    # print(paste(p,q,d,P,Q,D))
    tryCatch({
        model.fit = Arima(y = as.ts(co2.ts), order = c(p, d,
            q), seasonal = c(P, D, Q), lambda = lambda, include.drift = FALSE)

        model.info = data.frame(p, d, q, P, D, Q, model.fit$aic,
            model.fit$aicc, model.fit$bic)
        return(model.info)

    }, error = function(e) {
        return(data.frame())  # Empty DF
    })

}

model.fit.info.df <- do.call("rbind", (apply(params.df, 1, funFitModel)))
```

```
##           0--------------------------------------------------------------324
## Progress:
```

```r
colnames(model.fit.info.df) <- c("p", "d", "q", "P", "D", "Q",
    "aic", "aicc", "bic")
print("Top 6 models by BIC")
```

```
## [1] "Top 6 models by BIC"
```

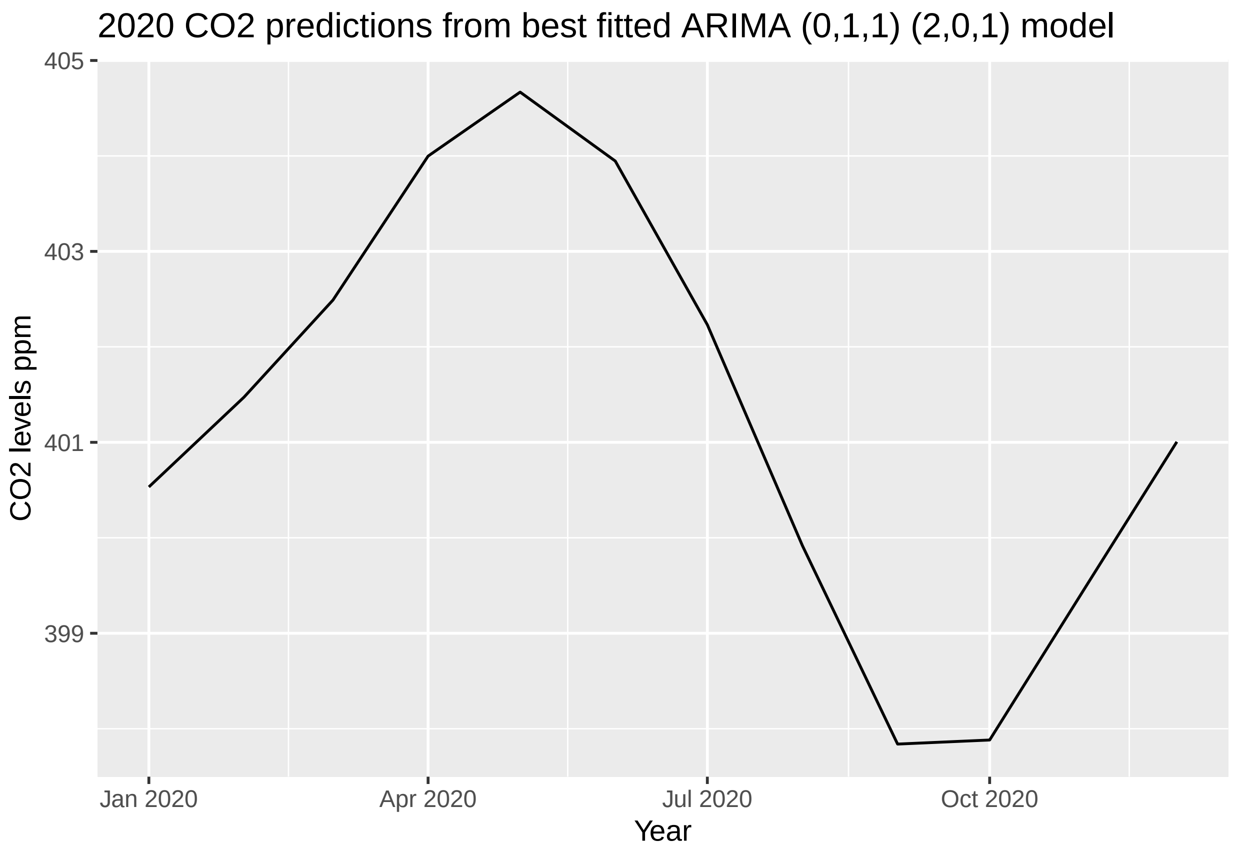
```r
model.fit.info.df %>% arrange(bic) %>% head()
```

```
##   p d q P D Q       aic      aicc       bic
## 1 0 1 1 2 0 1 -5410.497 -5410.367 -5389.765
## 2 1 1 1 2 0 1 -5410.653 -5410.471 -5385.775
## 3 1 1 0 2 0 1 -5405.987 -5405.857 -5385.255
## 4 0 1 2 2 0 1 -5409.346 -5409.164 -5384.468
## 5 2 1 1 2 0 1 -5411.773 -5411.529 -5382.748
## 6 1 1 2 2 0 1 -5410.656 -5410.412 -5381.632
```

Our function evaluated 324 models. Using BIC as criteria we can see that ARIMA (0,1,1) with seasonality (2,0,1) as best model for CO2 time series. BIC for this model is least -5389.765. Note that this model uses lambda= -0.03432107 and no drift.

Lets save the model to forecast for year 2020:

```r
model.arima.best.q3 = Arima(y = as.ts(co2.ts), order = c(0, 1,
    1), seasonal = c(2, 0, 1), lambda = lambda, include.drift = FALSE)

co2.2020.arima.pred.ts <- forecast(model.arima.best.q3, h = 276) %>%
    as_tibble() %>% dplyr::select("Point Forecast") %>% ts(start = c(1998,
    1), frequency = 12) %>% as_tsibble() %>% filter_index("2019-12-31" ~
    .)

co2.2020.arima.pred.ts %>% autoplot(.vars = value) + labs(title = "2020 CO2 predictions from be
    xlab("Year") + ylab("CO2 levels ppm")
```



Our 2020 prediction using the ARIMA model follows the seasonal pattern previously recognized, and has a maximum CO2 level of 404.5 ppm and a minimum level of 397.5 ppm

Below, we inscribe the model using Latex and Backshift notation.

$$x_t(1 - B) - \alpha B^{24} x_t(1 - B) = B w_t$$

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 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.

First, we read in and examine the dataset:

```r
co2.noaa.weekly.df <- read.table("co2_weekly_mlo.txt", header = FALSE,
    comment.char = "#", na.strings = "-999.99")
colnames(co2.noaa.weekly.df) <- c("year", "month", "day", "decimal.day",
    "record.value", "#days", "one.year.ago", "ten.years.ago",
    "since.1800.diff")
```

```r
head(co2.noaa.weekly.df)
```

```
##   year month day decimal.day record.value #days one.year.ago ten.years.ago
## 1 1974     5  19    1974.380       333.34     6           NA            NA
## 2 1974     5  26    1974.399       332.95     6           NA            NA
## 3 1974     6   2    1974.418       332.32     5           NA            NA
## 4 1974     6   9    1974.437       332.18     7           NA            NA
## 5 1974     6  16    1974.456       332.37     7           NA            NA
## 6 1974     6  23    1974.475       331.59     6           NA            NA
##   since.1800.diff
## 1           50.36
## 2           50.06
## 3           49.57
## 4           49.63
## 5           50.07
## 6           49.60
```

```r
tail(co2.noaa.weekly.df)
```

```
##      year month day decimal.day record.value #days one.year.ago ten.years.ago
## 2383 2020     1  12    2020.031       412.82     6       410.66        388.41
## 2384 2020     1  19    2020.051       413.65     7       412.19        388.27
## 2385 2020     1  26    2020.070       414.09     7       411.06        389.37
## 2386 2020     2   2    2020.089       414.33     7       411.11        390.67
## 2387 2020     2   9    2020.108       414.40     6       412.70        390.32
## 2388 2020     2  16    2020.127       414.01     7       411.22        390.45
##      since.1800.diff
## 2383          132.51
## 2384          133.17
## 2385          133.47
## 2386          133.61
## 2387          133.58
## 2388          133.10
```

```r
summary(co2.noaa.weekly.df)  # 20 readings are missing
```

```
##       year          month             day          decimal.day    record.value  
##  Min.   :1974   Min.   : 1.000   Min.   : 1.00   Min.   :1974   Min.   :326.7  
##  1st Qu.:1985   1st Qu.: 4.000   1st Qu.: 8.00   1st Qu.:1986   1st Qu.:346.9  
##  Median :1997   Median : 7.000   Median :16.00   Median :1997   Median :364.3  
##  Mean   :1997   Mean   : 6.539   Mean   :15.71   Mean   :1997   Mean   :366.8  
##  3rd Qu.:2008   3rd Qu.:10.000   3rd Qu.:23.00   3rd Qu.:2009   3rd Qu.:386.2  
##  Max.   :2020   Max.   :12.000   Max.   :31.00   Max.   :2020   Max.   :415.4  
##                                                                 NA's   :20     
##      #days        one.year.ago   ten.years.ago   since.1800.diff 
##  Min.   :0.000   Min.   :326.8   Min.   :326.6   Min.   : 49.57  
##  1st Qu.:5.000   1st Qu.:346.4   1st Qu.:342.9   1st Qu.: 66.72  
##  Median :6.000   Median :363.3   Median :356.2   Median : 83.49  
##  Mean   :5.858   Mean   :365.8   Mean   :357.3   Mean   : 86.86  
##  3rd Qu.:7.000   3rd Qu.:384.8   3rd Qu.:370.8   3rd Qu.:106.31  
##  Max.   :7.000   Max.   :412.7   Max.   :390.7   Max.   :133.61  
##                  NA's   :69      NA's   :542     NA's   :20      
```

```r
describe(co2.noaa.weekly.df)
```

```
## co2.noaa.weekly.df 
## 
##  9  Variables      2388  Observations
## --------------------------------------------------------------------------------
## year 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2388        0       47        1     1997    15.26     1976     1978 
##      .25      .50      .75      .90      .95 
##     1985     1997     2008     2015     2017 
## 
## lowest : 1974 1975 1976 1977 1978, highest: 2016 2017 2018 2019 2020
## --------------------------------------------------------------------------------
## month 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     2388        0       12    0.993    6.539    3.971        1        2 
##      .25      .50      .75      .90      .95 
##        4        7       10       11       12 
## 
## lowest :  1  2  3  4  5, highest:  8  9 10 11 12
##                                                                             
## Value          1     2     3     4     5     6     7     8     9    10    11
## Frequency    203   185   199   193   201   198   204   203   198   203   197
## Proportion 0.085 0.077 0.083 0.081 0.084 0.083 0.085 0.085 0.083 0.085 0.082
##                 
## Value         12
## Frequency    204
## Proportion 0.085
## --------------------------------------------------------------------------------
## day
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2388        0       31    0.999    15.71    10.16        2        4
##      .25      .50      .75      .90      .95
##        8       16       23       28       29
##
## lowest :  1  2  3  4  5, highest: 27 28 29 30 31
## --------------------------------------------------------------------------------
## decimal.day
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2388        0     2388        1     1997    15.26     1977     1979
##      .25      .50      .75      .90      .95
##     1986     1997     2009     2016     2018
##
## lowest : 1974.380 1974.399 1974.418 1974.437 1974.456
## highest: 2020.051 2020.070 2020.089 2020.108 2020.127
## --------------------------------------------------------------------------------
## record.value
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2368       20     2111        1    366.8    27.14    332.8    336.4
##      .25      .50      .75      .90      .95
##    346.9    364.3    386.2    401.6    407.4
##
## lowest : 326.73 326.96 327.07 327.23 327.31, highest: 414.37 414.40 414.41 414.74 415.39
## --------------------------------------------------------------------------------
## #days
##        n  missing distinct     Info     Mean      Gmd
##     2388        0        8    0.899    5.858    1.384
##
## lowest : 0 1 2 3 4, highest: 3 4 5 6 7
##
## Value          0     1     2     3     4     5     6     7
## Frequency     20    12    39    92   178   382   653  1012
## Proportion 0.008 0.005 0.016 0.039 0.075 0.160 0.273 0.424
## --------------------------------------------------------------------------------
## one.year.ago
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2319       69     2028        1    365.8     26.3    332.5    336.3
##      .25      .50      .75      .90      .95
##    346.4    363.3    384.8    399.0    404.7
##
## lowest : 326.77 326.85 326.98 327.21 327.38, highest: 411.58 411.70 411.84 412.19 412.70
## --------------------------------------------------------------------------------
## ten.years.ago
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1846      542     1608        1    357.3    19.49    332.0    334.6
##      .25      .50      .75      .90      .95
##    342.9    356.2    370.8    382.0    385.2
##
## lowest : 326.64 327.06 327.10 327.26 327.27, highest: 390.32 390.36 390.45 390.53 390.67
## --------------------------------------------------------------------------------
## since.1800.diff
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2368       20     2011        1    86.86    27.02    52.39    56.09
##      .25      .50      .75      .90      .95
##    66.72    83.49   106.31   120.98   127.12
##
## lowest :  49.57  49.60  49.63  49.93  49.99, highest: 133.17 133.26 133.47 133.58 133.61
## --------------------------------------------------------------------------------
```

```
co2.noaa.weekly.df <- co2.noaa.weekly.df %>% mutate(record.date = ymd(paste(year,
    month, day)))
```

We can see in the dataset there are 2388 observations. The make the time series easier to work with, we create an index using all of the time columns and convert the data to a tsibble object.

```
week.frequency = 365.25/7
co2.noaa.weekly.ts.raw <- ts(data = co2.noaa.weekly.df$record.value,
    start = 1974.3795, frequency = week.frequency)
co2.noaa.weekly.ts <- as_tsibble(ts(data = co2.noaa.weekly.df$record.value,
    start = 1974.3795, frequency = week.frequency), class = "matrix")
cbind(head(co2.noaa.weekly.ts), tail(co2.noaa.weekly.ts))
```

```
## # A tsibble: 6 x 4 [1W]
##      index value   index1 value1
##     <week> <dbl>   <week>  <dbl>
## 1 1974 W20  333. 2020 W02   413.
## 2 1974 W21  333. 2020 W03   414.
## 3 1974 W22  332. 2020 W04   414.
## 4 1974 W23  332. 2020 W05   414.
## 5 1974 W24  332. 2020 W06   414.
## 6 1974 W25  332. 2020 W07   414.
```

```
summary(co2.noaa.weekly.ts)
```

```
##   index              value
##  NULL:1974 W20   Min.   :326.7
##  NULL:1985 W42   1st Qu.:346.9
##  NULL:1997 W13   Median :364.3
##  NULL:1997 W13   Mean   :366.8
##  NULL:2008 W36   3rd Qu.:386.2
##  NULL:2020 W07   Max.   :415.4
##                  NA's   :20
```

In the summary of the tsibble object, we can see that there are 20 missing values. In the available data, the lowest CO2 value is 326.7 ppm and the greatest is 415.4 ppm.

To fill in the missing data, we fill in the values with interpolation from neighbor data points. After the interpolation we can see that there are 0 rows with missing values.

```
co2.noaa.weekly.ts %>% filter(is.na(value))
```

```
## # A tsibble: 20 x 2 [1W]
##       index value
##      <week> <dbl>
##  1 1975 W40    NA
##  2 1975 W49    NA
##  3 1975 W50    NA
##  4 1975 W51    NA
##  5 1975 W52    NA
##  6 1976 W26    NA
##  7 1979 W20    NA
##  8 1982 W11    NA
##  9 1982 W14    NA
## 10 1982 W15    NA
## 11 1983 W31    NA
## 12 1984 W13    NA
## 13 1984 W14    NA
## 14 1984 W15    NA
## 15 1984 W16    NA
## 16 1984 W48    NA
## 17 2005 W41    NA
## 18 2008 W26    NA
## 19 2008 W27    NA
## 20 2008 W28    NA
```

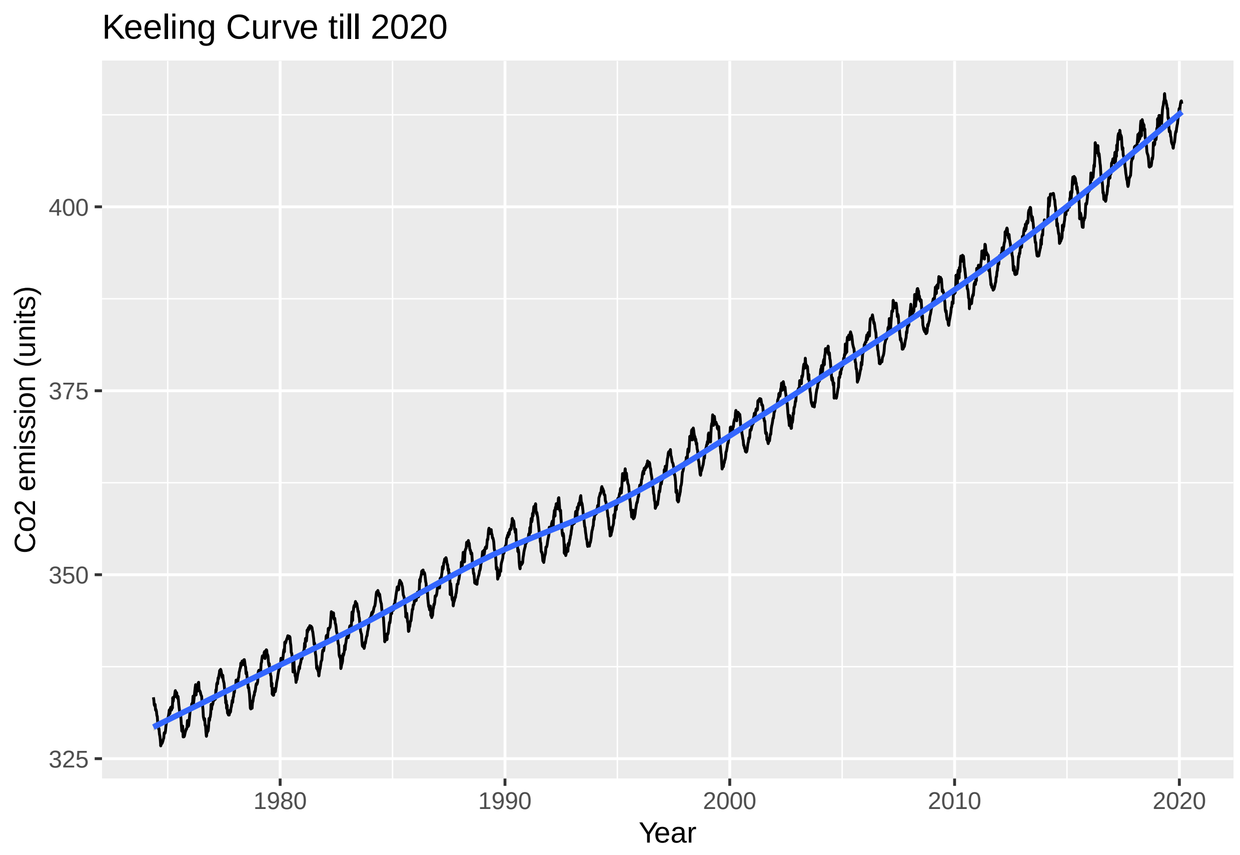
```
co2.noaa.weekly.ts <- na_interpolation(co2.noaa.weekly.ts, option = "spline")

co2.noaa.weekly.ts %>% filter(is.na(value))
```

```
## # A tsibble: 0 x 2 [?]
## # ... with 2 variables: index <week>, value <dbl>
```

Now that we have a complete dataset, we can plot the Keeling curve and continue with EDA.

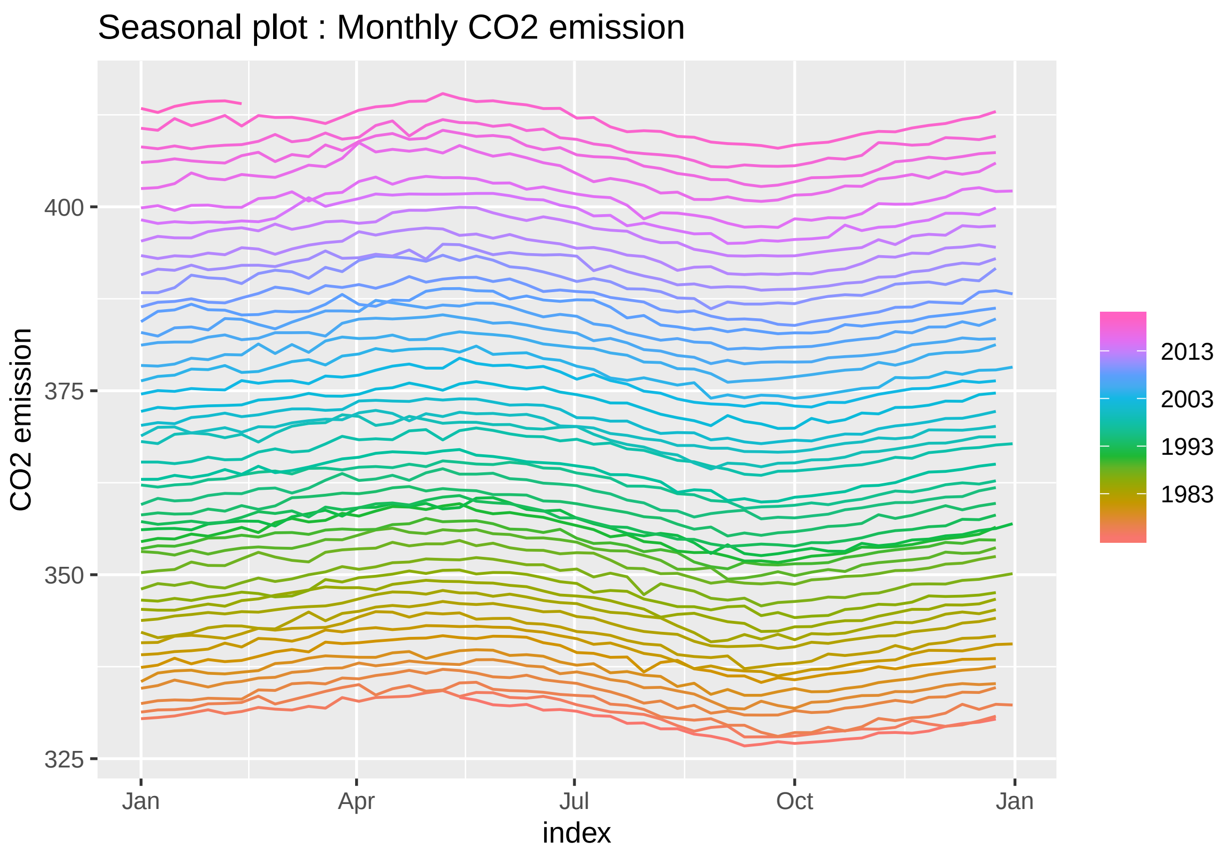
```
co2.noaa.weekly.ts %>% autoplot(.vars = value) + geom_smooth() +
    labs(title = "Keeling Curve till 2020", y = "Co2 emission (units)",
        x = "Year")
```

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



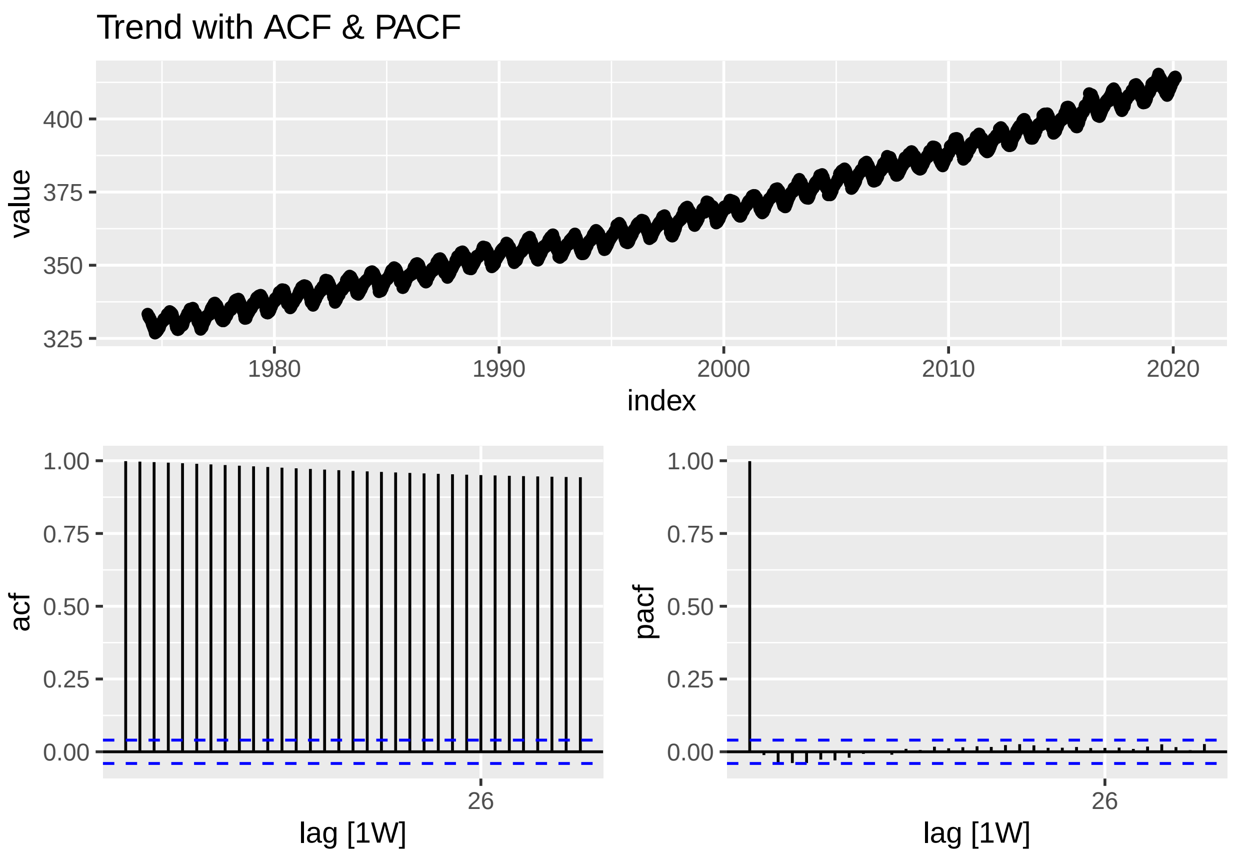
In the Keeling curve, we saw a constant increasing trend from the start of the series to about 1990. After a few years of a slower rate of increase, we can see that from about 1995 onward the trend has picked up and increasing faster than pre 1995 trend. In the curve we see the clear seasonal pattern in CO2 levels as every year the CO2 levels experience a spike and low point.

```r
# Additional EDA
co2.noaa.weekly.ts %>% gg_season(y = value, period = "year") +
    ylab("CO2 emission") + ggtitle("Seasonal plot : Monthly CO2 emission")
```



The seasonal plot looks similar to the monthly CO2 data evaluated earlier, but we can now see the effects of the weekly data points by the variation in each line versus the smoother line in the monthly data. The seasonal monthly trends, however, are clearly the same, with a high in May and low in October.

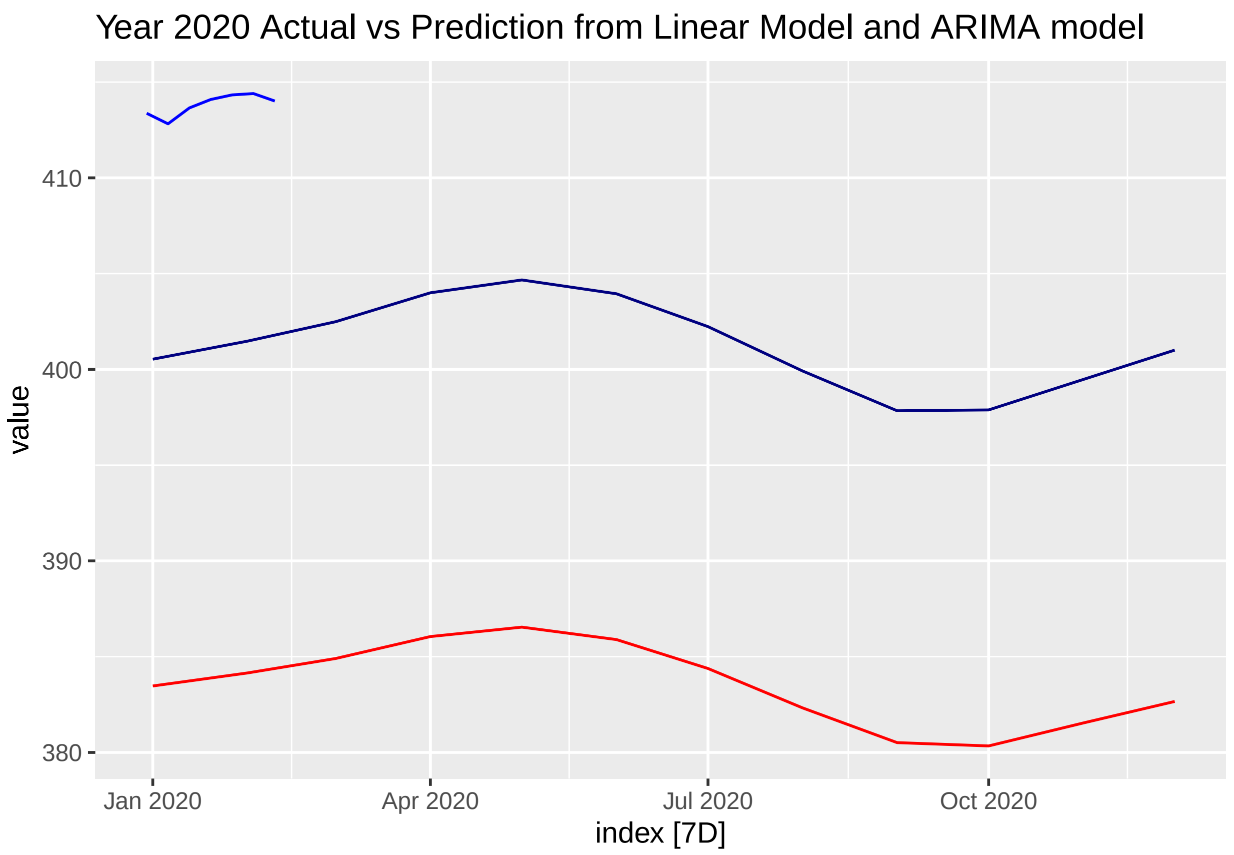
```
co2.noaa.weekly.ts %>% gg_tsdisplay(plot_type = "partial", y = value) +
    ggtitle("Trend with ACF & PACF")
```



The ACF tells us that there is autocorrelation even after 24 months. On the other hand PACF is not indicative of any partial auto correlation. This could have something to do with the fact that the series is a weekly series.

Now lets compare the current CO2 levels to the predicted levels from Q2 and Q3. In the chart below, the blue line represents CO2 actuals, red represents forecasts from the cubic model with monthly dummy variables, and navy represents the forecast from the ARIMA model:

```
co2.noaa.weekly.ts %>%
  filter_index("2019-12-31" ~ .) %>%
  mutate(index = as.Date(index)) %>%
  autoplot(.vars = value, color = "blue") + #Weekly actual time series
  autolayer(co2.2020.pred.ts, .vars = value, color = "red") + # Prediction from polynomial mod
  autolayer(co2.2020.arima.pred.ts, .vars = value, color = "navy") + # Prediction from Arima mo
  labs(title = "Year 2020 Actual vs Prediction from Linear Model and ARIMA model")
```



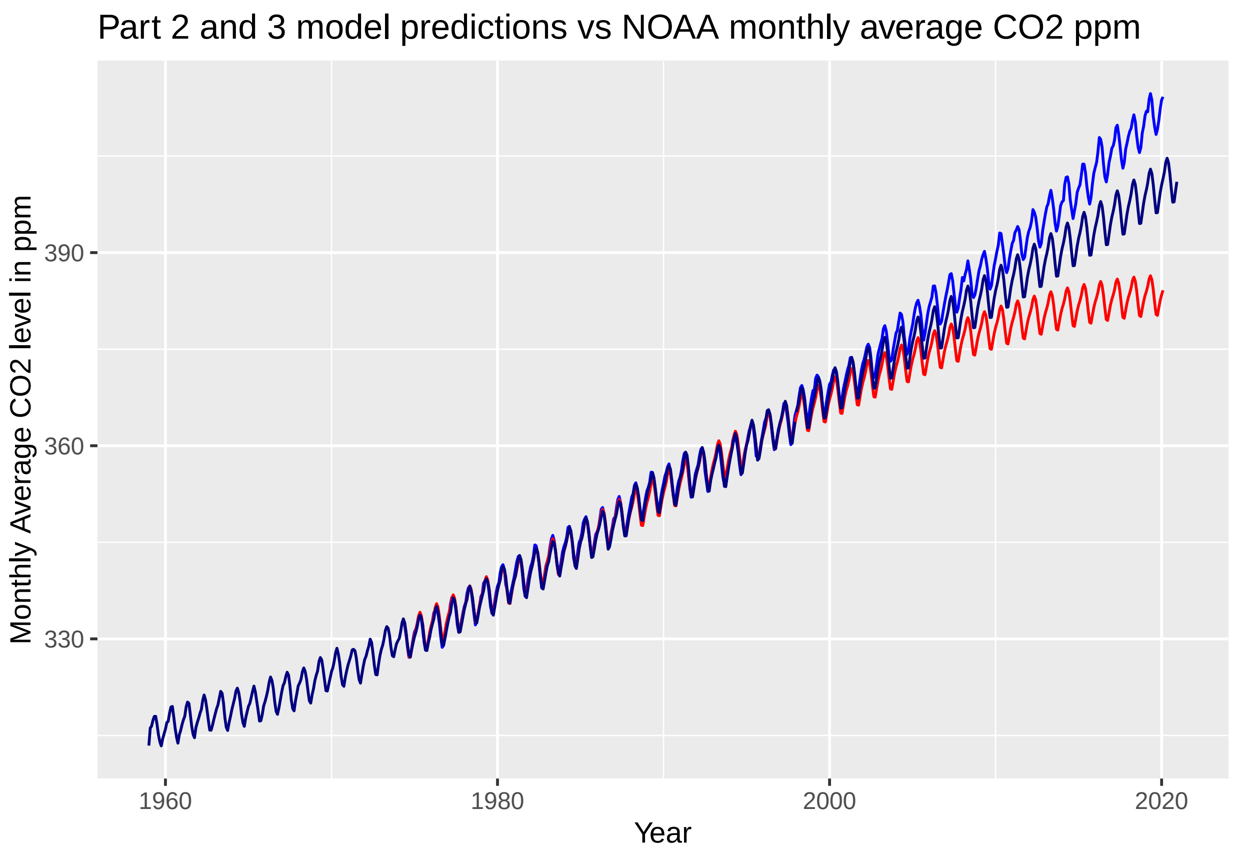
For the first few actual observations in 2020, we see that the actual CO2 levels (blue) are higher than the values predicted from both models. The actuals are around 10-15 units higher than the ARIMA model forecast (navy). The ARIMA model and the cubic plus dummy model follow the same general pattern, but the ARIMA model has forecasted values around 15-20 units greater than the cubic model (red).

Now lets compare monthly average curve with model forecasts. In the chart below, the blue line represents CO2 actuals using monthly averages, red represents forecasts from the cubic model with monthly dummy variables, and navy represents the forecast from the ARIMA model:

```
#First lets convert to monthly time series
co2.noaa.monthly.ts <- co2.noaa.weekly.ts %>%
  as_tibble() %>%
  mutate(yearmonth = yearmonth(yearweek(index))) %>%
  group_by(yearmonth) %>%
  summarise(avg_value = mean(value)) %>%
  as_tsibble(index = yearmonth, value = avg_value)

summary(co2.noaa.monthly.ts) #> 1974 May to 2020 Feb
```

```
##     yearmonth          avg_value
##  Min.   :1974 May   Min.   :327.1
##  1st Qu.:1985 Oct   1st Qu.:346.6
##  Median :1997 Mar   Median :364.0
##  Mean   :1997 Mar   Mean   :366.7
##  3rd Qu.:2008 Aug   3rd Qu.:386.1
##  Max.   :2020 Feb   Max.   :414.7
```

```
# Now predict using linear model for same time frame
time.index.1974.may = seq(from = (1974 - 1959) * 12 + 5, length.out = 550)
co2.noaa.like.df <-
  data.frame(time.index = time.index.1974.may,
             season = factor(time.index.1974.may %% 12, ordered = F))
co2.noaa.like.pred <-
  predict(object = mod.ln.5, newdata = co2.noaa.like.df)
co2.noaa.like.pred.ts <-
  as_tsibble(ts(
    data = co2.noaa.like.pred,
    start = c(1974, 5),
    frequency = 12
  ))

# now predict using Arima model selected in Q3
co2.noaa.arima.pred.ts <- forecast(model.arima.best.q3, h = 276) %>%
  as_tibble() %>%
  dplyr::select('Point Forecast') %>%
  ts(start = c(1998, 1), frequency = 12) %>%
  as_tsibble()

model.arima.best.q3.fitted.ts <- model.arima.best.q3$fitted %>%
  as_tsibble()

co2.noaa.monthly.ts %>%
  autoplot(.vars = avg_value, color = "blue") + # Actual NOAA series averaged to monthly
  autolayer(co2.noaa.like.pred.ts, .vars = value, color = "red") + # Polynomial model predicti
  autolayer(co2.noaa.arima.pred.ts, .vars = value, color = "navy") + # Arima model prediction
  autolayer(model.arima.best.q3.fitted.ts,
            .vars = value,
            color = "navy") + xlab("Year") + ylab("Monthly Average CO2 level in ppm") +
  labs(title = "Part 2 and 3 model predictions vs NOAA monthly average CO2 ppm")
```



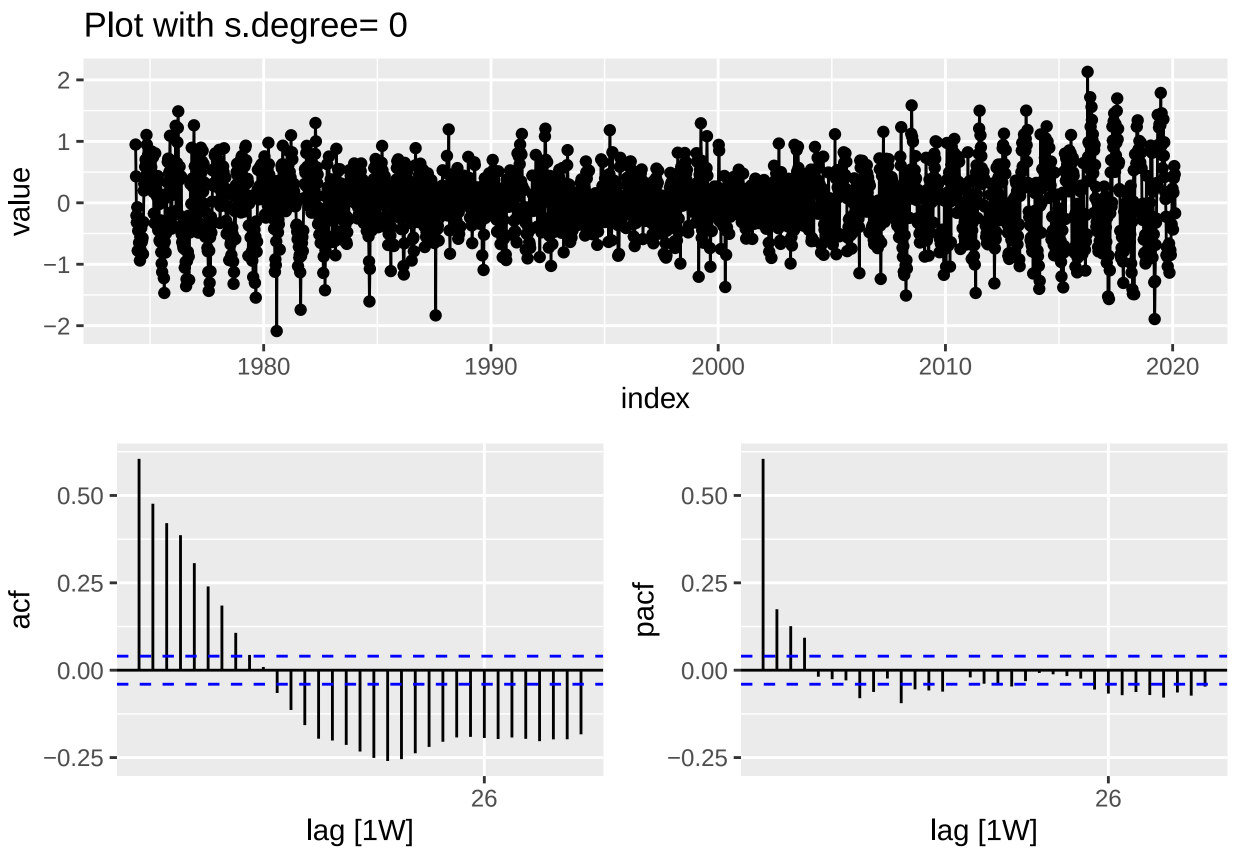
We see that up until about the year 2000, both models follow the actual data closely. After, the models separate from the actuals with the cubic model showing the lowest values (red), and the ARIMA model (navy) also showing lower values than the actual values (blue). The actual CO2 levels are rising faster than either model forecasted. The cubic model forecasts that the CO2 level would stop rising and would flatten. Clearly the ARIMA model is closer to modeling the actual CO2 values and the cubic model is not a good fit.

Part 5 (3 points)

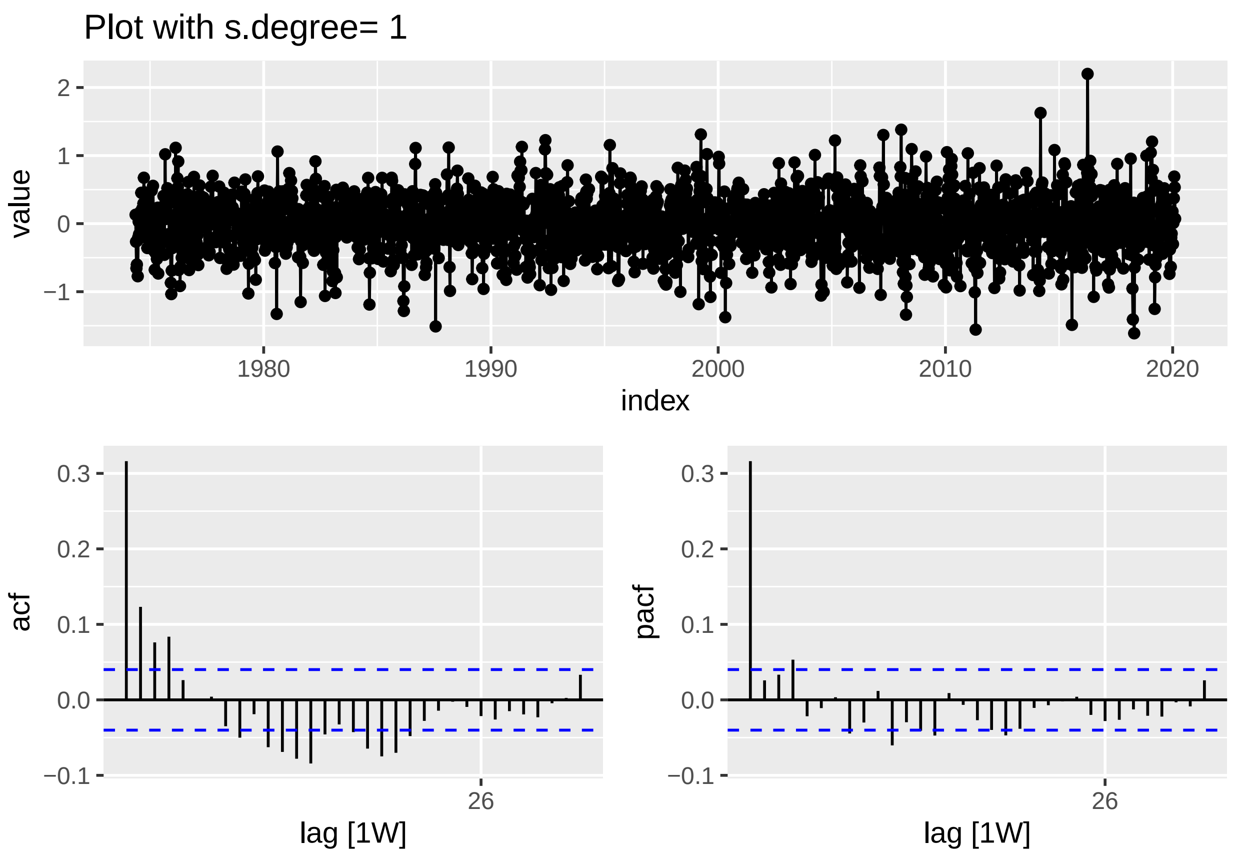
Seasonally adjust the weekly NOAA data, and split both seasonally-adjusted (SA) and non-seasonally-adjusted (NSA) series into training and test sets, using the last two years of observations as the test sets. For both SA and NSA series, fit ARIMA models using all appropriate steps. Measure and discuss how your models perform in-sample and (pseudo-) 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.

First lets try couple of ways in which we can decompose seasonality by analyzing residuals:

```r
plotRemainderFromSTL <- function(s.degree = 0) {
    stl.val <- co2.noaa.weekly.ts %>% stl(s.window = week.frequency,
        s.degree = s.degree)
    ts.obj <- as_tsibble(ts(stl.val$time.series[, "remainder"],
        start = 1974.3795, frequency = week.frequency))

    par(mfrow = c(1, 2))
    ts.obj %>% gg_tsdisplay(y = value, plot_type = "partial") +
        labs(title = paste("Plot with s.degree=", s.degree))

}

plotRemainderFromSTL(s.degree = 0)
```



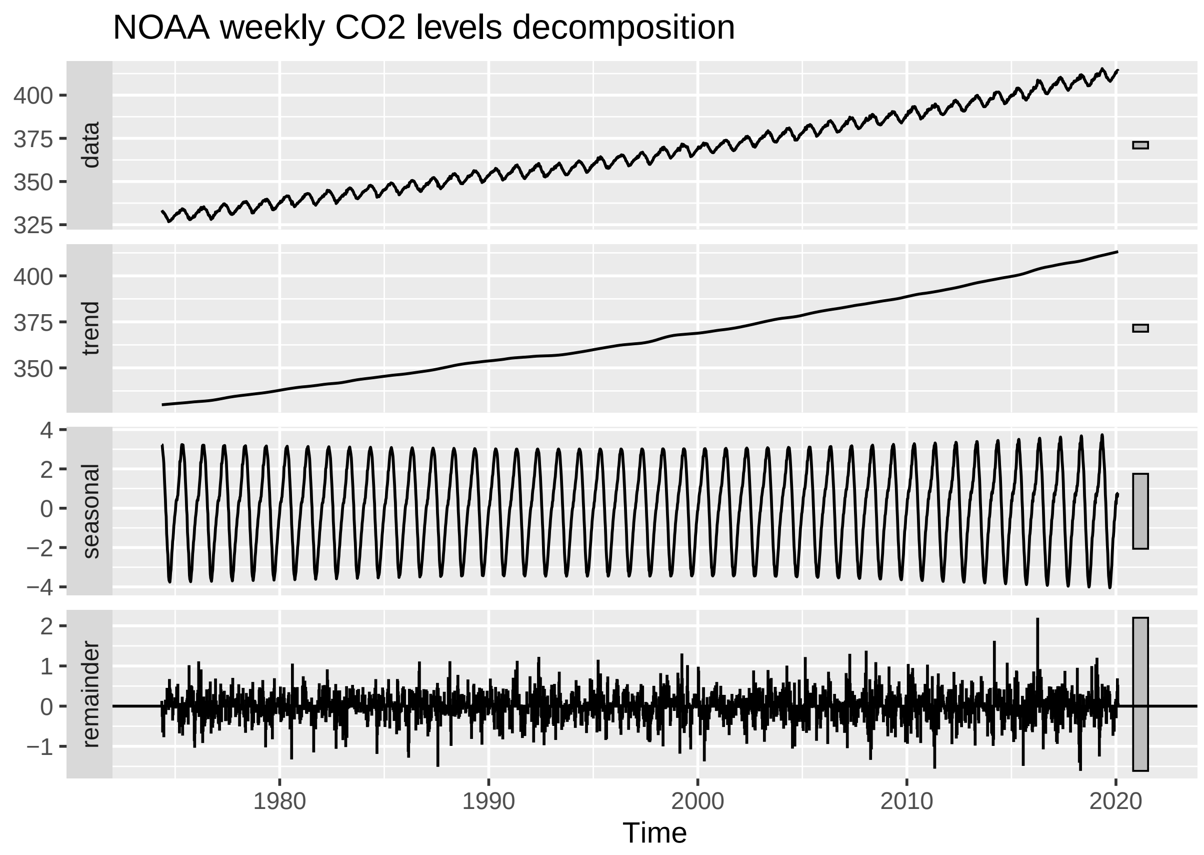
```
plotRemainderFromSTL(s.degree = 1)
```



We can see that with degree of locality 1 in seasonal extraction, we get much better residuals with dampening ACF and sharp decrease in PACF at lag 1. There is still some seasonality left, but it is better than degree 0.

Lets see how seasonal decomposition looks:

```
co2.noaa.weekly.ts %>% stl(s.window = week.frequency, s.degree = 1) %>%
    autoplot() + labs(title = "NOAA weekly CO2 levels decomposition")
```



In the decomposed series we see the clear increasing trend line and the seasonal pattern with somewhat consistent variance.

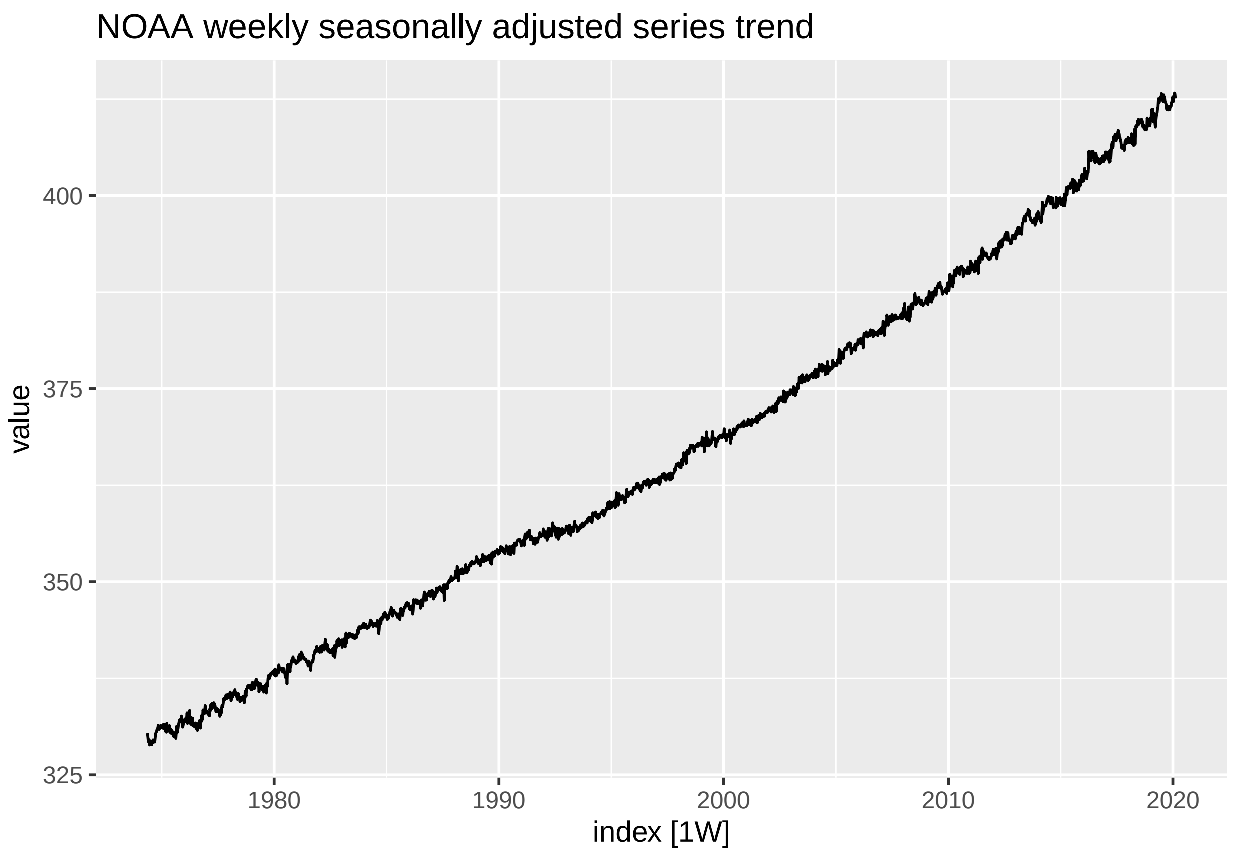
Now lets subtract seasonal data from time series to get seasonally adjusted time series:

```
co2.noaa.weekly.ts.components <- stl(co2.noaa.weekly.ts, s.window = week.frequency)$time.serie
co2.noaa.weekly.ts.components.seasonal <- co2.noaa.weekly.ts.components[,
    "seasonal"]
co2.noaa.weekly.ts.SA <- as_tsibble(ts(co2.noaa.weekly.ts$value -
    co2.noaa.weekly.ts.components.seasonal, start = 1974.3795,
    frequency = week.frequency))
cbind(NSA = head(co2.noaa.weekly.ts), NSA = tail(co2.noaa.weekly.ts),
    SA = head(co2.noaa.weekly.ts.SA), SA = tail(co2.noaa.weekly.ts.SA))
```

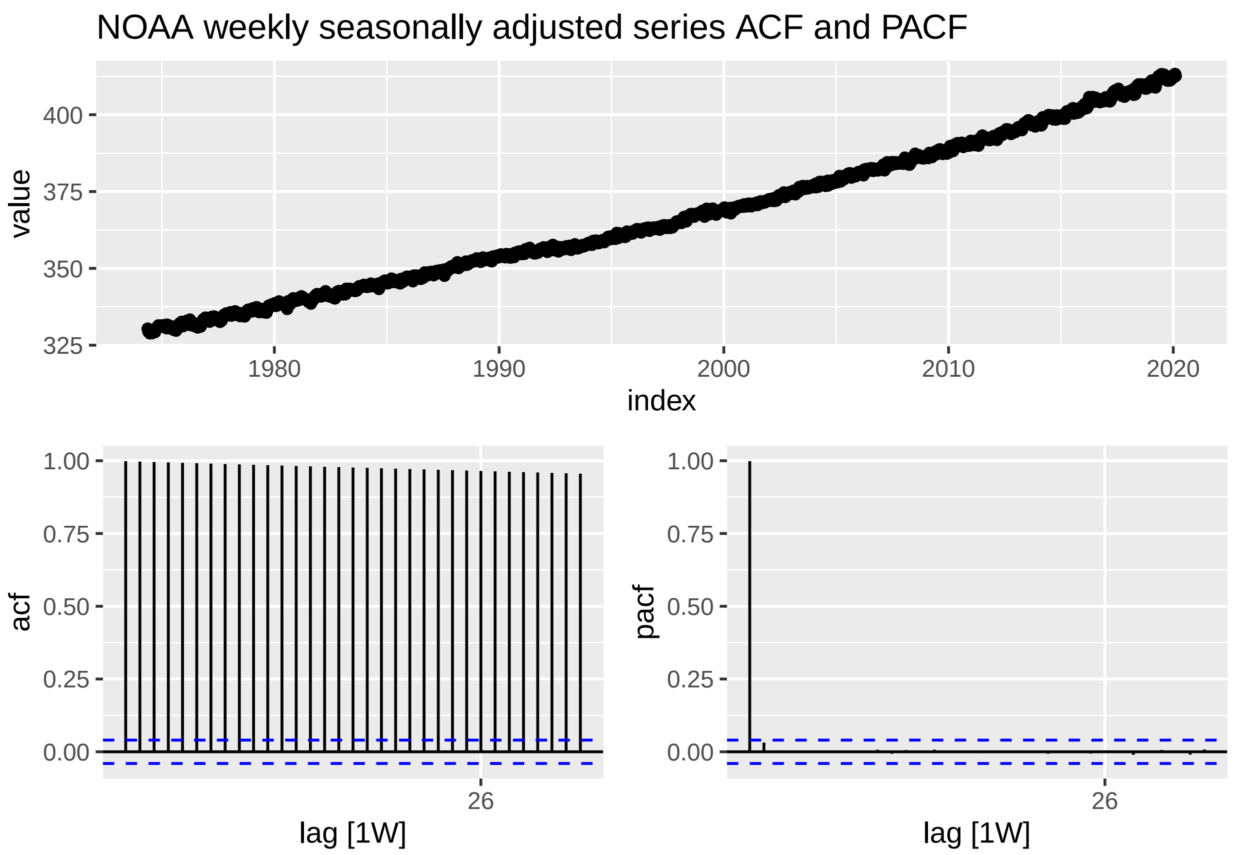
```
## # A tsibble: 6 x 8 [1W]
##      index value   index1 value1   index2 value2   index3 value3
##     <week> <dbl>   <week>  <dbl>   <week>  <dbl>   <week>  <dbl>
## 1 1974 W20  333. 2020 W02   413. 1974 W20   330. 2020 W02   412.
## 2 1974 W21  333. 2020 W03   414. 1974 W21   330. 2020 W03   413.
## 3 1974 W22  332. 2020 W04   414. 1974 W22   329. 2020 W04   413.
## 4 1974 W23  332. 2020 W05   414. 1974 W23   329. 2020 W05   413.
## 5 1974 W24  332. 2020 W06   414. 1974 W24   330. 2020 W06   413.
## 6 1974 W25  332. 2020 W07   414. 1974 W25   329. 2020 W07   413.
```

Lets check the SA time series

```
co2.noaa.weekly.ts.SA %>% autoplot(.vars = value) + labs(title = "NOAA weekly seasonally adjus
```



```
co2.noaa.weekly.ts.SA %>% gg_tsdisplay(plot_type = "partial",
    y = value) + labs(title = "NOAA weekly seasonally adjusted series ACF and PACF")
```



We can see a smoother trend line with the seasonally adjusted series. The acf chart shows a very slow decreasing lag for every 30+ lags. The pacf chart suggests a non-stationary random walk process.

Now we split the seasonally adjusted and non-seasonally adjusted time series into training and test sets by taking the last 2 years of data as the test set.

```
trainTestSplit <- function(time.series) {
    time.series.test <- time.series %>% filter_index("2018-01-01" ~
        .)

    time.series.train <- time.series %>% filter_index(. ~ "2017-12-31")

    return(list(train = time.series.train, test = time.series.test))
}

# Split SA data into training and test

co2.noaa.weekly.ts.SA.train.test <- trainTestSplit(co2.noaa.weekly.ts.SA)
co2.noaa.weekly.ts.SA.test <- co2.noaa.weekly.ts.SA.train.test$test
co2.noaa.weekly.ts.SA.training <- co2.noaa.weekly.ts.SA.train.test$train

# Split NSA data into training and test
co2.noaa.weekly.ts.train.test <- trainTestSplit(co2.noaa.weekly.ts)
co2.noaa.weekly.ts.test <- co2.noaa.weekly.ts.train.test$test
```

```
co2.noaa.weekly.ts.training <- co2.noaa.weekly.ts.train.test$train
```

Now lets fit Arima model for SA and non SA time series using a function that evaluates different combinations of p, d, q, P, D, and Q. For the seasonally adjusted model, we set P, D, and Q to 0, and we can test greater values of p, d, and q because there are less combinations to attempt given there is no seasonal component. Therefore, we evaluate different combinations of models using a max value of 3 for p and q and a max value of 2 for d.

```
# Fit ARIMA model for SA data
no.p = 3
no.d = 2
no.q = 3

# P,D, Q are set to 0 because training data is already
# seasonally adjusted
no.P = 0
no.D = 0
no.Q = 0

no.of.models <- (no.p + 1) * (no.d + 1) * (no.q + 1) * (no.P +
    1) * (no.D + 1) * (no.Q + 1)

print(paste("Number of models to fit =", no.of.models))
```

```
## [1] "Number of models to fit = 48"
```

```
params.df <- expand.grid(p = 0:no.p, d = 0:no.d, q = 0:no.q,
    P = 0:no.P, D = 0:no.D, Q = 0:no.Q)
i <- 1

funFitModel <- function(param.row) {
    progress(value = i, max.value = no.of.models, console = TRUE,
        progress.bar = TRUE)
    i <- i + 1
    p = param.row["p"]
    d = param.row["d"]
    q = param.row["q"]
    P = param.row["P"]
    D = param.row["D"]
    Q = param.row["Q"]

    # print(paste(p,d,q,P,D,Q))
    tryCatch({
        model.fit = Arima(y = as.ts(co2.noaa.weekly.ts.SA.training),
            order = c(p, d, q), seasonal = c(0, 0, 0), include.drift = FALSE)

        model.info = data.frame(p, d, q, P, D, Q, model.fit$aic,
            model.fit$aicc, model.fit$bic)
        return(model.info)

    }, error = function(e) {
        return(data.frame())
    })

}

model.fit.info.df.SA <- do.call("rbind", (apply(params.df, 1,
    funFitModel)))
colnames(model.fit.info.df.SA) <- c("p", "d", "q", "P", "D",
    "Q", "aic", "aicc", "bic")
```

```
print("Top 6 models by BIC")
```

```
## [1] "Top 6 models by BIC"
```

```
model.fit.info.df.SA %>% arrange(bic) %>% head()
```

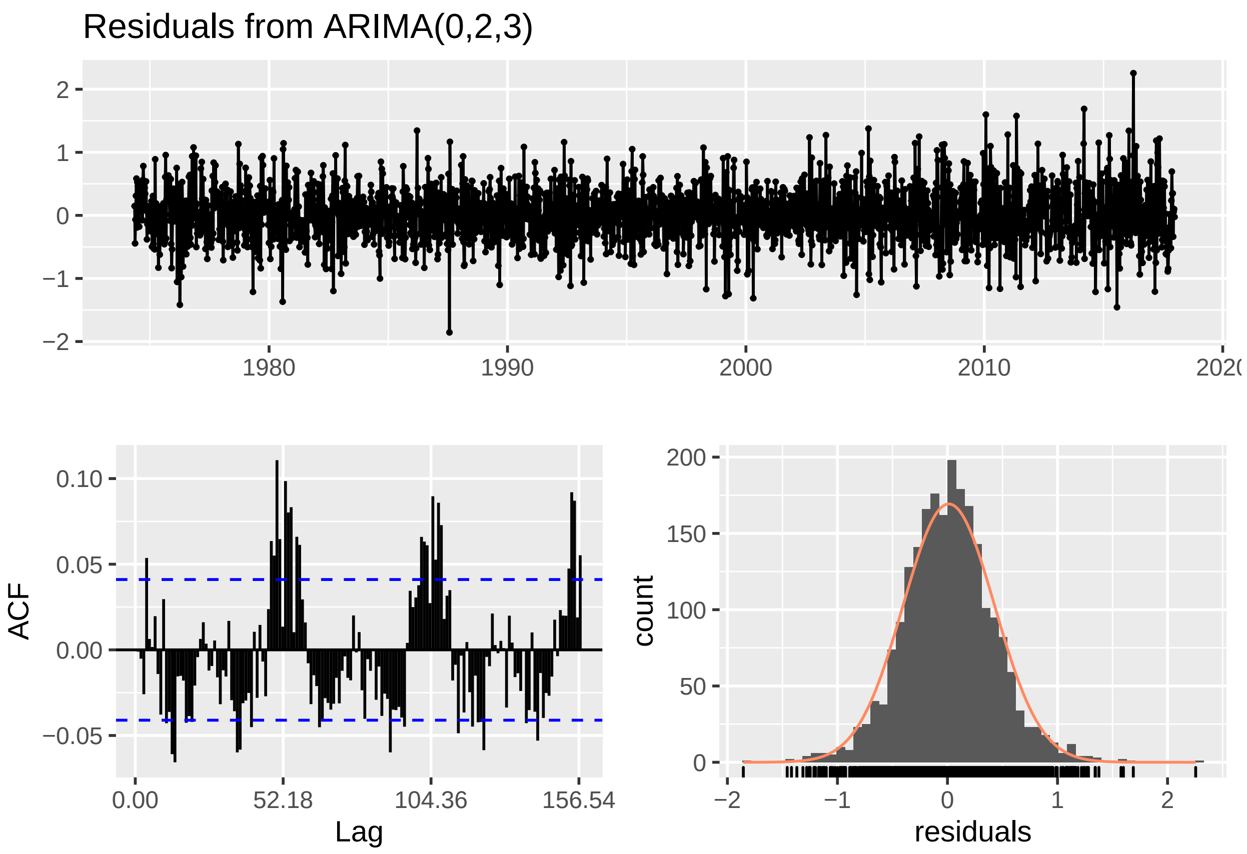
```
##   p d q P D Q      aic     aicc      bic
## 1 0 2 3 0 0 0 2485.557 2485.575 2508.476
## 2 1 2 2 0 0 0 2487.160 2487.178 2510.079
## 3 2 2 2 0 0 0 2486.372 2486.399 2515.021
## 4 3 2 2 0 0 0 2483.124 2483.161 2517.502
## 5 1 2 3 0 0 0 2491.976 2492.002 2520.624
## 6 3 2 1 0 0 0 2492.165 2492.191 2520.813
```

Arima(0,2,3) is in top 3 models by all 3 measures AIC, AICc and BIC. We pick this model as best model as this has least number of parameters compared to other top models.

```
model.arima.best.SA = Arima(y = as.ts(co2.noaa.weekly.ts.SA.training),
    order = c(0, 2, 3), seasonal = c(0, 0, 0), include.drift = FALSE)

summary(model.arima.best.SA)
```

```
## Series: as.ts(co2.noaa.weekly.ts.SA.training)
## ARIMA(0,2,3)
##
## Coefficients:
##           ma1     ma2     ma3
##       -1.4632  0.3590  0.1046
## s.e.   0.0209  0.0358  0.0204
##
## sigma^2 estimated as 0.1737:  log likelihood=-1238.78
## AIC=2485.56   AICc=2485.57   BIC=2508.48
##
## Training set error measures:
##                      ME      RMSE       MAE         MPE       MAPE      MASE
## Training set 0.01205627 0.4162734 0.3200364 0.003176143 0.08773688 0.1791174
##                          ACF1
## Training set -0.001043418
```

```
checkresiduals(model.arima.best.SA)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,3)
## Q* = 338, df = 101.36, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 104.36
```

We can conclude from the residuals chart that the model is an ok fit. The mean appears to be 0, but there may be some uncaptured seasonality as shown in the acf chart.

Now lets fit Arima model to non-SA time series. Because of the number of models to fit and the time taken to evaluate all models, we set a max value of 1 for all parameters p, d, q, P, D, and Q.

```
# Fit ARIMA model for NSA data
no.p = no.P = 1
no.q = no.Q = 1
no.d = no.D = 1
no.of.models <- (no.p + 1) * (no.d + 1) * (no.q + 1) * (no.P +
    1) * (no.D + 1) * (no.Q + 1)
```

```r
print(paste("Number of models to fit =", no.of.models))
```

```
## [1] "Number of models to fit = 64"
```

```r
params.df <- expand.grid(p = 0:no.p, d = 0:no.d, q = 0:no.q,
    P = 0:no.P, D = 0:no.D, Q = 0:no.Q)
i <- 1

funFitModel <- function(param.row) {
    progress(value = i, max.value = no.of.models, console = TRUE,
        progress.bar = TRUE)
    i <- i + 1
    p = param.row["p"]
    d = param.row["d"]
    q = param.row["q"]
    P = param.row["P"]
    D = param.row["D"]
    Q = param.row["Q"]

    # print(paste(p,q,d,P,Q,D))
    tryCatch({
        model.fit = Arima(y = as.ts(co2.noaa.weekly.ts.training),
            order = c(p, d, q), seasonal = c(P, D, Q), include.drift = FALSE)

        model.info = data.frame(p, d, q, P, D, Q, model.fit$aic,
            model.fit$aicc, model.fit$bic)


        return(model.info)

    }, error = function(e) {
        return(data.frame())
    })

}

model.fit.info.df.NSA <- do.call("rbind", (apply(params.df, 1,
    funFitModel)))
colnames(model.fit.info.df.NSA) <- c("p", "d", "q", "P", "D",
    "Q", "aic", "aicc", "bic")
```

```r
print("Top 6 models by BIC")
```

```
## [1] "Top 6 models by BIC"
```

```r
model.fit.info.df.NSA %>% arrange(bic) %>% head()
```

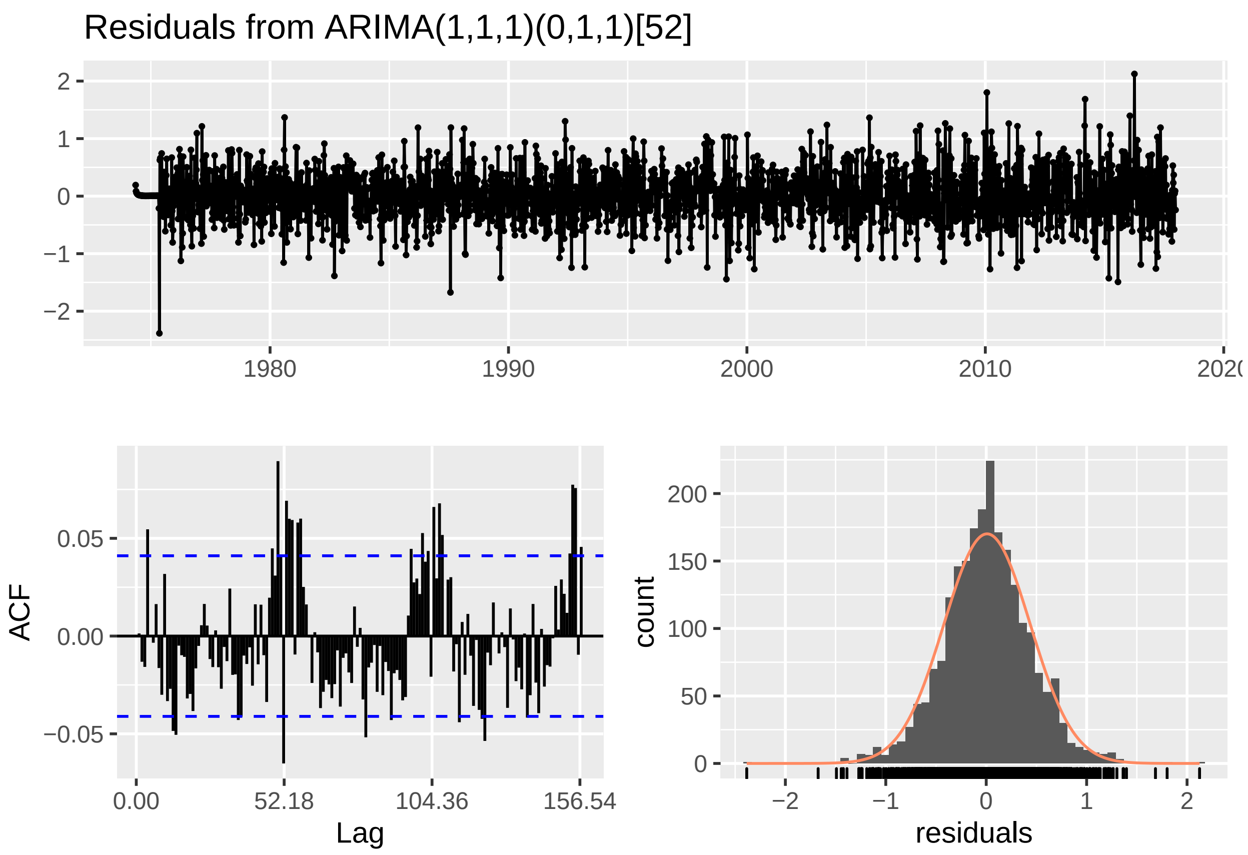
```
##   p d q P D Q      aic     aicc      bic
## 1 1 1 1 0 1 1 2641.192 2641.210 2664.020
## 2 0 1 1 1 1 1 2686.552 2686.570 2709.380
## 3 0 1 1 0 1 1 2694.043 2694.054 2711.164
## 4 0 1 1 1 0 1 2803.372 2803.390 2826.293
## 5 1 1 0 1 1 1 2910.521 2910.539 2933.349
## 6 1 1 0 0 1 1 2916.375 2916.386 2933.496
```

Best model for NSA time series is Arima(1,1,1)(0,1,1). We selected this model because has lowest AIC, AICc and BIC so this is natural selection. Ideally, we could try with higher values of p,d and q if but it is going to take much longer to estimate the models as number of parameters quickly increase exponentially.

```r
model.arima.best.NSA = Arima(y = as.ts(co2.noaa.weekly.ts.training),
    order = c(1, 1, 1), seasonal = c(0, 1, 1), include.drift = FALSE)

summary(model.arima.best.NSA)
```

```
## Series: as.ts(co2.noaa.weekly.ts.training)
## ARIMA(1,1,1)(0,1,1)[52]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.2490  -0.7955  -0.8049
## s.e.  0.0326   0.0232   0.0130
##
## sigma^2 estimated as 0.1895:  log likelihood=-1316.6
## AIC=2641.19   AICc=2641.21   BIC=2664.02
##
## Training set error measures:
##                      ME      RMSE      MAE         MPE       MAPE      MASE
## Training set 0.00850045 0.4299258 0.327362 0.002071709 0.08946474 0.1832073
##                     ACF1
## Training set 0.001344668
```

```r
checkresiduals(model.arima.best.NSA)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,1,1)[52]
## Q* = 219.5, df = 101.36, p-value = 1.051e-10
##
## Model df: 3.   Total lags used: 104.36
```

We can conclude from the residuals chart that the model is an ok fit. The mean appears to be 0, but there may be some uncaptured seasonality as shown in the acf chart.

Now lets compare how these selected model perform in-sample and pseudo out-of-sample data:

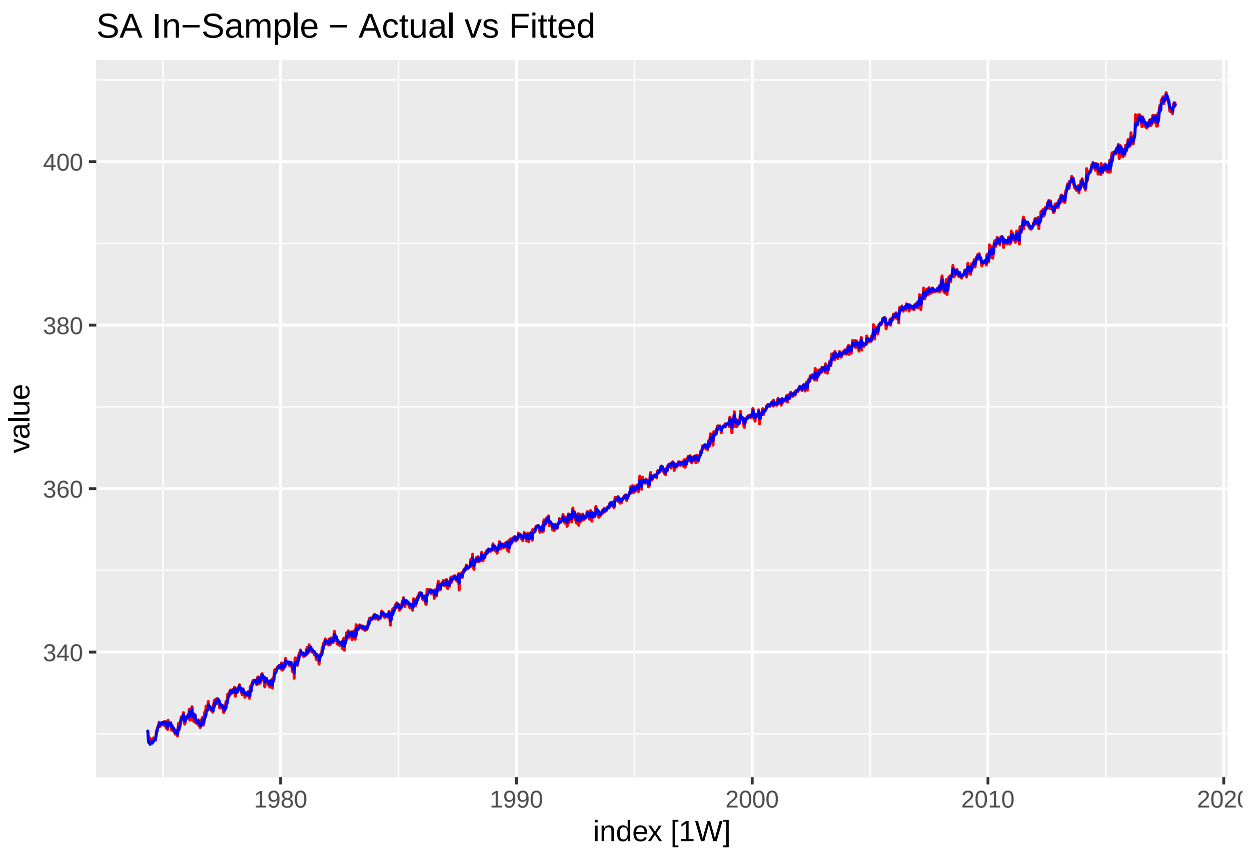
```
# in-sample
in.sample.accuracy = data.frame(rbind(accuracy(model.arima.best.SA),
    accuracy(model.arima.best.NSA)), row.names = c("SA", "NSA"))
in.sample.accuracy
```

```
##              ME      RMSE       MAE         MPE       MAPE      MASE
## SA  0.01205627 0.4162734 0.3200364 0.003176143 0.08773688 0.1791174
## NSA 0.00850045 0.4299258 0.3273620 0.002071709 0.08946474 0.1832073
##              ACF1
## SA  -0.001043418
## NSA  0.001344668
```

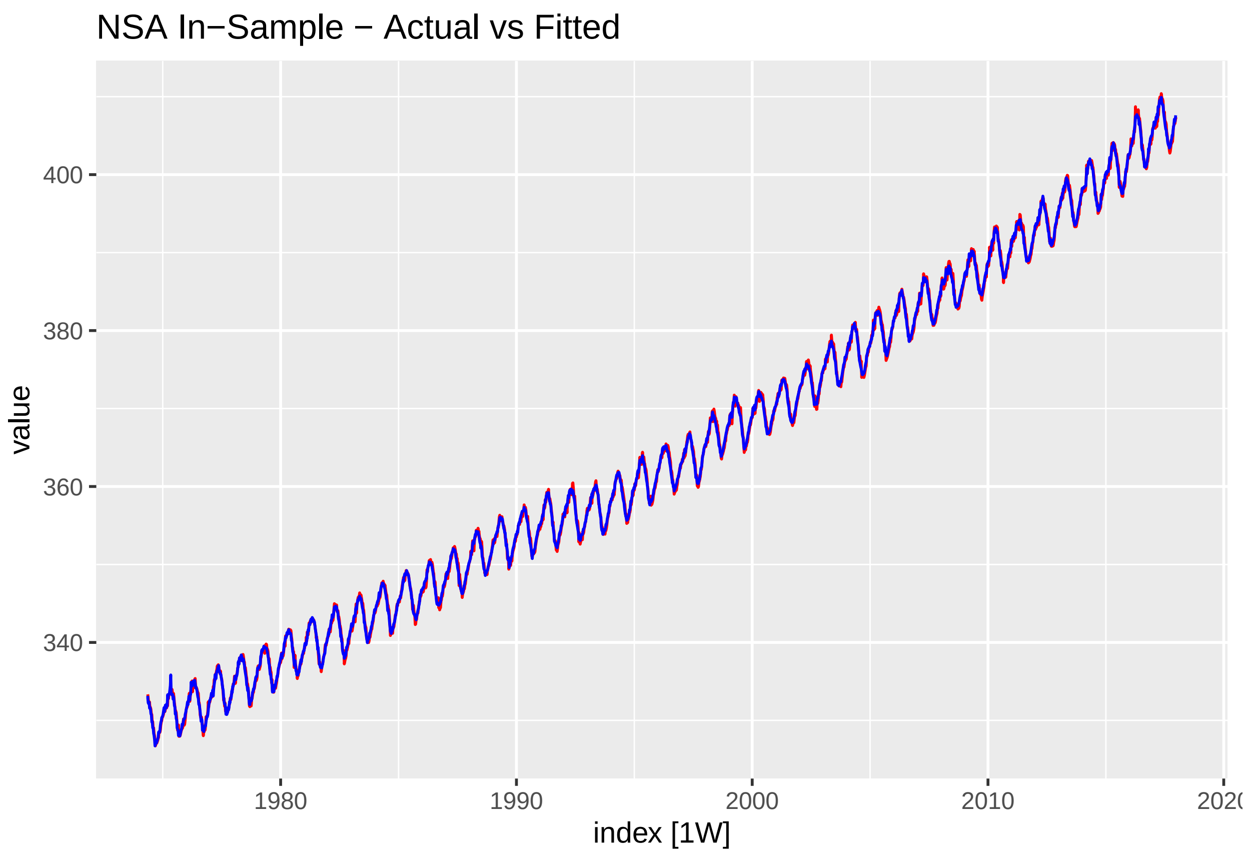
If we look at the in-sample accuracy of our SA and NSA best models, the SA model marginally does better than NSA, but they are very close.

Lets look at actual vs fitted values in chart.

```
co2.noaa.weekly.ts.SA.training %>%
  autoplot(.vars = value, color = "red") + #original data
  autolayer(as_tsibble(model.arima.best.SA$fitted),
            .vars = value,
            color = "blue") + # in sample prediction
  labs(title = "SA In-Sample - Actual vs Fitted ")
```



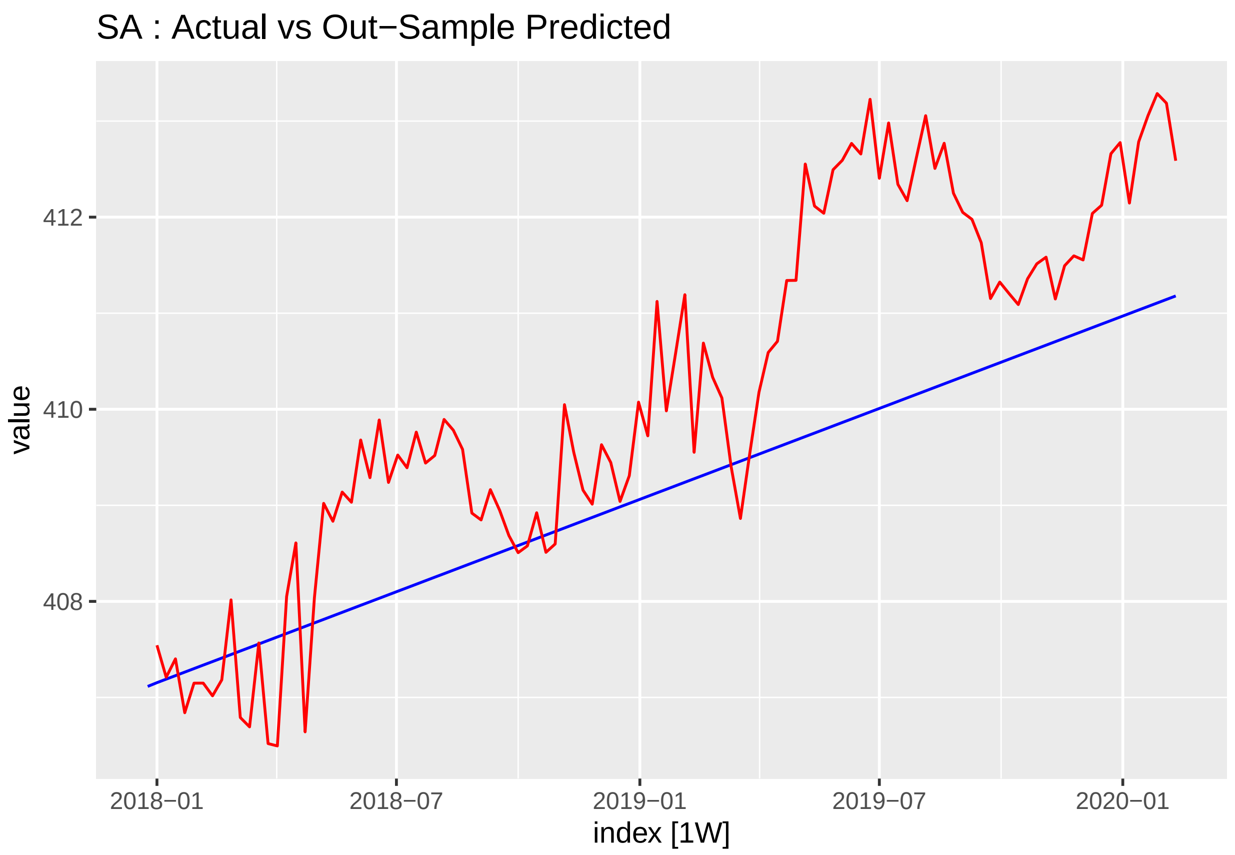
```
co2.noaa.weekly.ts.training %>%
  autoplot(.vars = value, color = "red") + #original data
  autolayer(as_tsibble(model.arima.best.NSA$fitted),
            .vars = value,
            color = "blue") + #in sample data
  labs(title = "NSA In-Sample - Actual vs Fitted ")
```



In the SA and NSA charts, we can see that that our modeled values seems to fit very closely to the actual values.

Now lets look at out-of-sample accuracy for SA model:

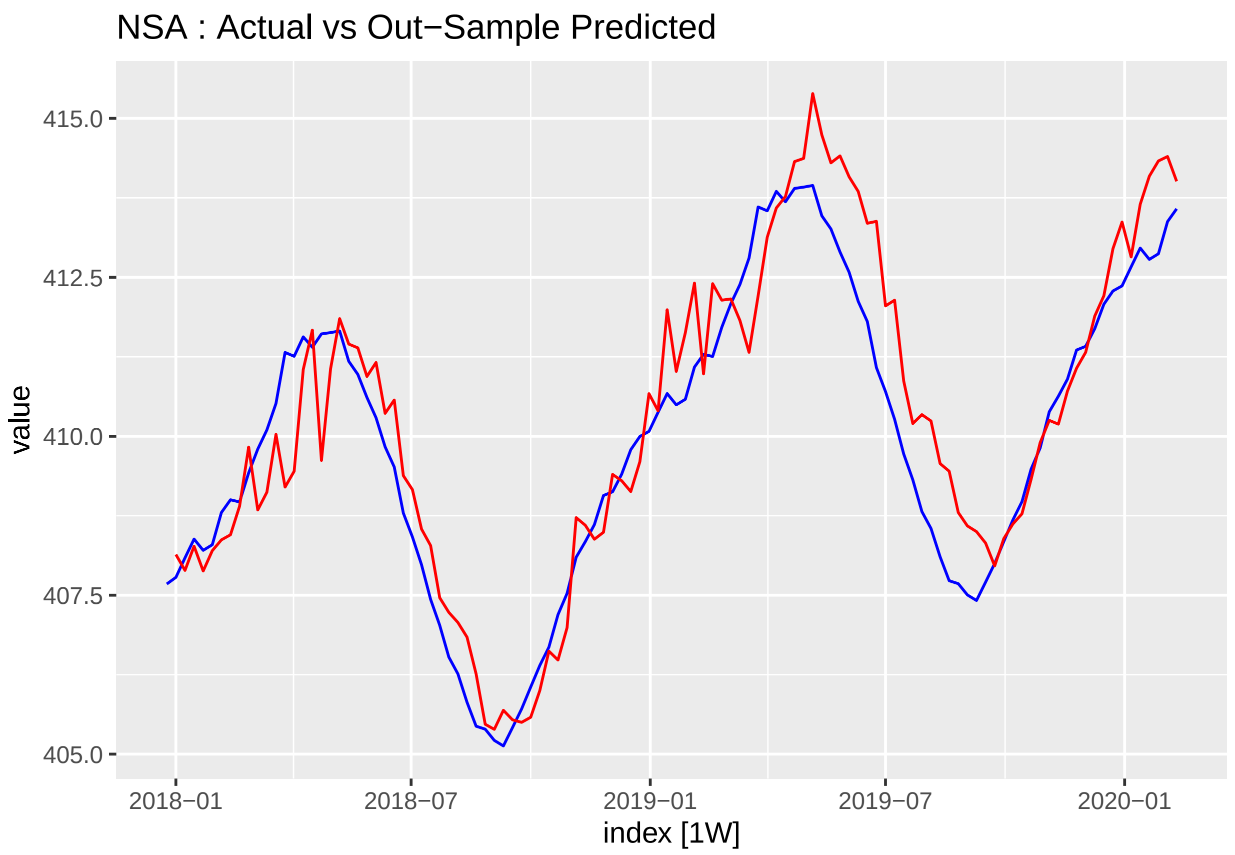
```r
#SA
forecast(model.arima.best.SA, h = length(co2.noaa.weekly.ts.SA.test$index) +
           1)$mean %>%
  as_tsibble() %>%
  autoplot(.vars = value, color = "blue") + # out sample prediction
  autolayer(co2.noaa.weekly.ts.SA.test,
            .vars = value,
            color = "red") + # out sample original data
  labs(title = "SA : Actual vs Out-Sample Predicted")
```



Seasonally adjusted model predicts the point estimates as a linear model and predicts the pseudo out-sample as a trend line. This is expected as model is based on seasonally adjusted training dataset.

Now lets look at out-of-sample accuracy for NSA model:

```
#NSA
forecast(model.arima.best.NSA, h = length(co2.noaa.weekly.ts.test$index) +
           1)$mean %>%
  as_tsibble() %>%
  autoplot(.vars = value, color = "blue") + # out sample predicted
  autolayer(co2.noaa.weekly.ts.test, .vars = value, color = "red") + # out sample original dat
  labs(title = "NSA : Actual vs Out-Sample Predicted")
```



Non SA model follows the pseudo out-sample dataset much more closely than SA model. The predictions are smooth over weekly fluctuations but still follows the original time series curve quite closely.

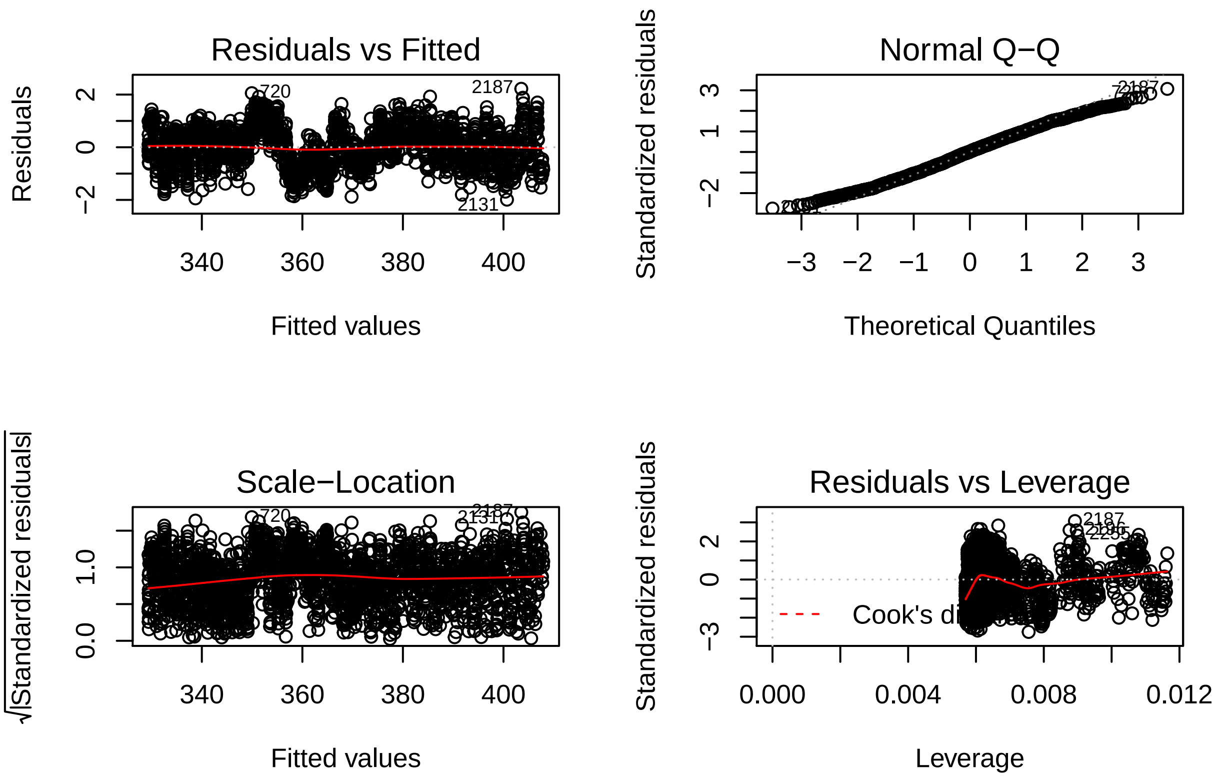
For our polynomial model, we will use a cubic polynomial with dummy variables for each month since this is the polynomial model variation we previously had the most success with.

```
# Fit cubic model with month dummy variable
co2.noaa.weekly.ts.SA.training.df <- data.frame(value = co2.noaa.weekly.ts.SA.training$value,
    time.index = 1:length(co2.noaa.weekly.ts.SA.training$index),
    season = factor(month(co2.noaa.weekly.ts.SA.training$index),
        ordered = F))
mod.ln.noaa.sa <- lm(formula = value ~ 0 + time.index + I(time.index^2) +
    I(time.index^3) + season, data = co2.noaa.weekly.ts.SA.training.df)
summary(mod.ln.noaa.sa)
```

```
##
## Call:
## lm(formula = value ~ 0 + time.index + I(time.index^2) + I(time.index^3) +
##     season, data = co2.noaa.weekly.ts.SA.training.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9898 -0.5391  0.0125  0.5333  2.2192
##
```

```
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## time.index         3.028e-02  2.319e-04  130.54   <2e-16 ***
## I(time.index^2) -3.561e-06  2.366e-07  -15.05   <2e-16 ***
## I(time.index^3)  2.390e-09  6.826e-11   35.01   <2e-16 ***
## season1            3.292e+02  7.924e-02 4154.45   <2e-16 ***
## season2            3.291e+02  8.104e-02 4061.02   <2e-16 ***
## season3            3.292e+02  7.917e-02 4158.17   <2e-16 ***
## season4            3.294e+02  8.035e-02 4099.21   <2e-16 ***
## season5            3.294e+02  7.855e-02 4192.95   <2e-16 ***
## season6            3.293e+02  7.887e-02 4174.85   <2e-16 ***
## season7            3.292e+02  7.856e-02 4190.34   <2e-16 ***
## season8            3.290e+02  7.864e-02 4183.98   <2e-16 ***
## season9            3.289e+02  7.901e-02 4163.52   <2e-16 ***
## season10           3.291e+02  7.894e-02 4169.13   <2e-16 ***
## season11           3.292e+02  7.958e-02 4136.85   <2e-16 ***
## season12           3.292e+02  7.884e-02 4175.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7266 on 2262 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.835e+07 on 15 and 2262 DF,  p-value: < 2.2e-16
```

```r
par(mfrow = c(2, 2))
plot(mod.ln.noaa.sa)
```



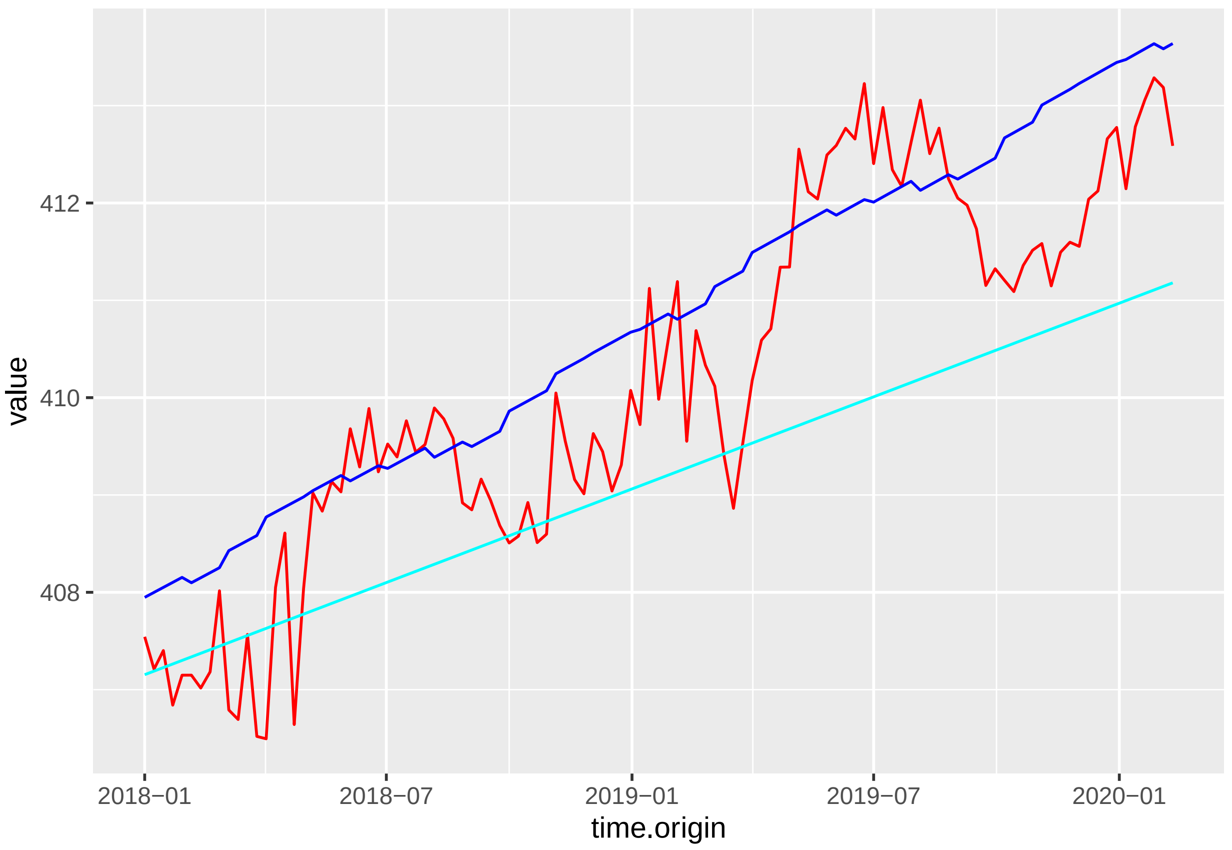
All of the features are significant in the cubic model. The residual charts suggest a decent fit. Lets check the accuracy of this model:

```
in.sample.accuracy.df = data.frame(RMSE = rbind(accuracy(model.arima.best.SA)[,
    "RMSE"], accuracy(mod.ln.noaa.sa)[, "RMSE"]), row.names = c("Arima-SA",
    "Linear-SA"))
in.sample.accuracy.df
```

```
##                RMSE
## Arima-SA  0.4162734
## Linear-SA 0.7242178
```

Accuracy of polynomial model is much worse than Arima model. Lets see how the cubic model performs to pseudo out-sample data:

```
co2.noaa.weekly.ts.SA.test.df <-
  data.frame(
    value = co2.noaa.weekly.ts.SA.test$value,
    time.index = length(co2.noaa.weekly.ts.SA.training$index) + 1:length(co2.noaa.weekly.ts.SA
    season = factor(month(co2.noaa.weekly.ts.SA.test$index), ordered = F),
    time.origin = co2.noaa.weekly.ts.SA.test$index
  )

co2.noaa.weekly.ts.SA.test.df$predicted = predict(mod.ln.noaa.sa, newdata = co2.noaa.weekly.ts
ggplot(co2.noaa.weekly.ts.SA.test.df, aes(x = time.origin)) +
  geom_line(aes(y = value), color = "red") + #original
  geom_line(aes(y = predicted), color = "blue") + #predicted by polynomial model
  geom_line(aes(y = forecast(
    model.arima.best.SA, h = length(co2.noaa.weekly.ts.SA.test$index) + 1
  )$mean[-1]), color = "cyan") # predicted by SA Arima model
```



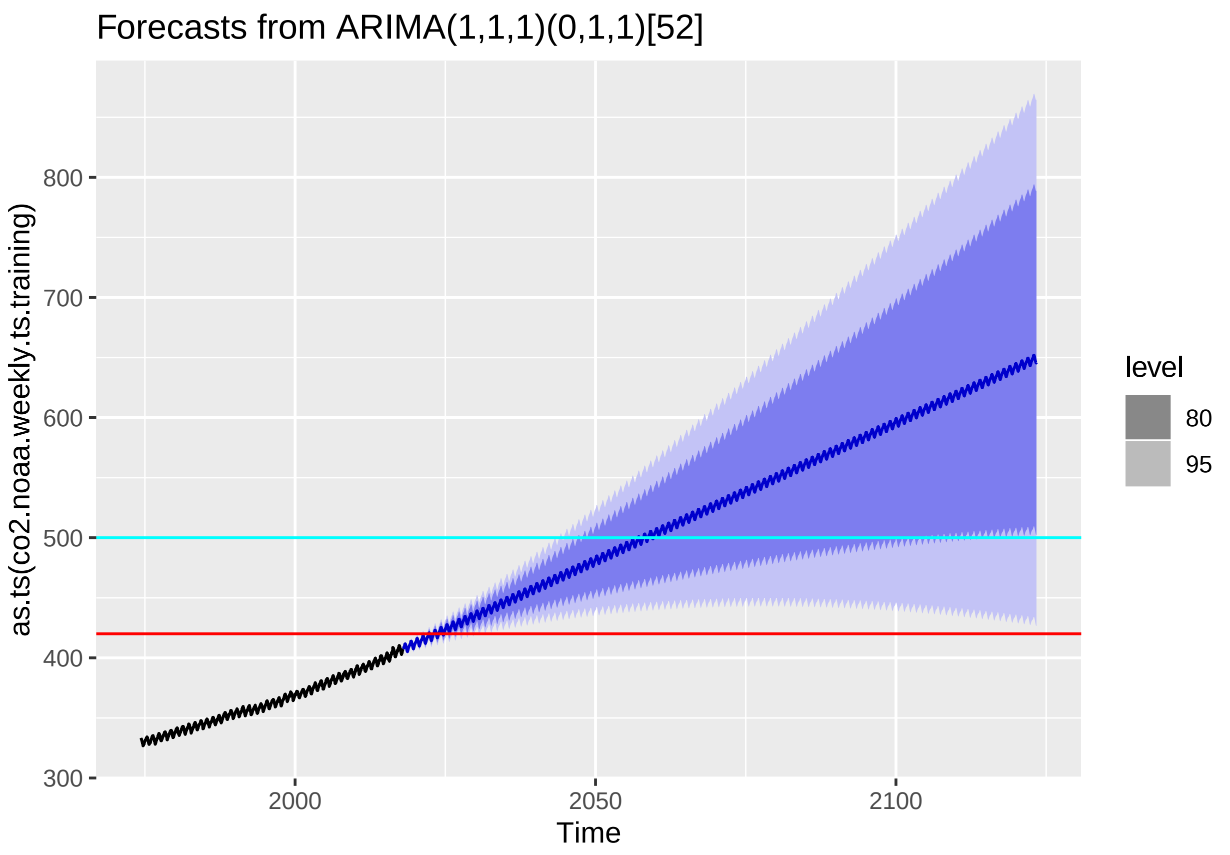
In the chart above the red line is the actual data, the blue line is the cubic model, and the cyan line is the Arima model. The polynomial model follows the seasonally adjusted out sample data at the highs of seasons whereas Arima SA model passes through the mean. By simply looking at the chart, it appears the cubic model is a better fit than the Arima SA model.

Part 6 (3 points)

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?

Lets first plot forecast for 5500 future intervals which would indicate approximately where 420 and 500 ppm levels were first and last met at 80% and 95% confidence intervals.

```
forecast(model.arima.best.NSA, h = 5500) %>% autoplot() + geom_hline(yintercept = 420,
    color = "red") + geom_hline(yintercept = 500, color = "cyan")
```



It appears that using a point estimate, value 420 is first reached in the year 225 and value 500 is first reached at around 2060. We will calculate these years programatically below:

```
# Get forcast for next 5500 weeks, sufficient for lower 95%
# CI to pass 500 mark
model.arima.best.NSA.predict.ft <- forecast(model.arima.best.NSA,
    h = 5500)

# convert to ts object to find time points in future
model.arima.best.NSA.predict.ts <- as_tsibble(ts(model.arima.best.NSA.predict.ft$mean,
    start = 2017.99905280457, frequency = 52.18))

# find all data point which cross 420 or 500 ppm values
model.arima.best.NSA.predict.tbl <- model.arima.best.NSA.predict.ft %>%
    as_tibble() %>% mutate(PF.420 = `Point Forecast` >= 420,
    Lo95.420 = `Lo 95` >= 420, Hi95.420 = `Hi 95` >= 420, Lo80.420 = `Lo 80` >=
        420, Hi80.420 = `Hi 80` >= 420, PF.500 = `Point Forecast` >=
        500, Lo95.500 = `Lo 95` >= 500, Hi95.500 = `Hi 95` >=
        500, Lo80.500 = `Lo 80` >= 500, Hi80.500 = `Hi 80` >=
        500)

findForecastFirst <- function(val) {
    cnt <- model.arima.best.NSA.predict.tbl %>% mutate(count = row_number(),
        first.val = val & !lag(val)) %>% filter(first.val) %>%
        head(1) %>% select(count)
    idx <- model.arima.best.NSA.predict.ts %>% select(index) %>%
        filter(row_number() == cnt$count)

    idx$index
}

findForecastLast <- function(val) {
    cnt <- model.arima.best.NSA.predict.tbl %>% mutate(count = row_number(),
        first.val = val & !lead(val)) %>% filter(first.val) %>%
        tail(1) %>% select(count)

    idx <- model.arima.best.NSA.predict.ts %>% select(index) %>%
        filter(row_number() == cnt$count)

    idx$index
}

# 420
PF.420.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$PF.420)
PF.420.last <- findForecastLast(model.arima.best.NSA.predict.tbl$PF.420)

Lo95.420.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Lo95.420)
Lo95.420.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Lo95.420)

Hi95.420.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Hi95.420)
Hi95.420.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Hi95.420)

Lo80.420.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Lo80.420)
Lo80.420.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Lo80.420)

Hi80.420.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Hi80.420)
Hi80.420.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Hi80.420)

# 500
PF.500.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$PF.500)
PF.500.last <- findForecastLast(model.arima.best.NSA.predict.tbl$PF.500)

Lo95.500.first <- NULL  # 95% CI low curve doesn't hit 500 ppm
Lo95.500.last <- NULL  # 95% CI low curve doesn't hit 500 ppm

Hi95.500.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Hi95.500)
Hi95.500.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Hi95.500)

Lo80.500.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Lo80.500)
Lo80.500.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Lo80.500)

Hi80.500.first <- findForecastFirst(model.arima.best.NSA.predict.tbl$Hi80.500)
```

```r
Hi80.500.last <- findForecastLast(model.arima.best.NSA.predict.tbl$Hi80.500)

predict.420 = list(Lo95.420.first, Lo95.420.last, Lo80.420.first,
    Lo80.420.last, PF.420.first, PF.420.last, Hi80.420.first,
    Hi80.420.last, Hi95.420.first, Hi95.420.last)
predict.500 = list(Lo95.500.first, Lo95.500.last, Lo80.500.first,
    Lo80.500.last, PF.500.first, PF.500.last, Hi80.500.first,
    Hi80.500.last, Hi95.500.first, Hi95.500.last)
col.names <- c("First time lower estimate with 95% CI", "Last time lower estimate with 95% CI"
    "First time lower estimate with 80% CI", "Last time lower estimate with 80% CI",
    "First time Point Estimate", "Last time Point Estimate",
    "First time higher estimate with 80% CI", "Last time higher estimate with 80% CI",
    "First time higher estimate with 95% CI", "Last time higher estimate with 95% CI")
results.df <- data.frame(predict.420)
results.df[2, ] = predict.500
colnames(results.df) <- col.names
rownames(results.df) <- c("Predictions for 420 ppm", "Predictions for 500 ppm")
results.df.t <- t(results.df)

results.df.t[order(results.df.t[, 1]), ]
```

```
##                                        Predictions for 420 ppm
## First time higher estimate with 95% CI "2021 W08"
## First time higher estimate with 80% CI "2021 W12"
## First time Point Estimate              "2022 W12"
## Last time higher estimate with 80% CI  "2022 W27"
## Last time higher estimate with 95% CI  "2022 W32"
## First time lower estimate with 80% CI  "2024 W12"
## Last time Point Estimate               "2024 W30"
## First time lower estimate with 95% CI  "2026 W11"
## Last time lower estimate with 80% CI   "2027 W30"
## Last time lower estimate with 95% CI   "2030 W30"
##                                        Predictions for 500 ppm
## First time higher estimate with 95% CI "2043 W51"
## First time higher estimate with 80% CI "2047 W04"
## First time Point Estimate              "2057 W06"
## Last time higher estimate with 80% CI  "2048 W29"
## Last time higher estimate with 95% CI  "2044 W23"
## First time lower estimate with 80% CI  "2101 W50"
## Last time Point Estimate               "2059 W25"
## First time lower estimate with 95% CI  NA
## Last time lower estimate with 80% CI   "2116 W17"
## Last time lower estimate with 95% CI   NA
```

Our programatic approach tells us that week 12 in 2022, the first time point estimate of 420 ppm is reached. The last time point estimate for 420 ppm is in week 30 of 2024. In week 6 of 2057, first time point estimate of 500 ppm is reached and the last time point estimate is week 25 of 2059.

The higher estimates of the confidence intervals are as follows: When using a 95% confidence interval, the first time for 420 ppm is in week 8 of 2021 420 ppm and in week 32 of 2022 the last time estimate is reached. Additionally, the first time estimate for 500 ppm is reached in week 51 of 2043 and the last time estimate is reached in week 23 of 2044. Finally, using an 80% confidence interval, in week 12 of 2021 420 ppm is reached, and in week 4 of 2047 500 ppm is reached.

The lower estimates of the confidence intervals are as follows: When using a 95% confidence interval, the first time for 420 ppm is in week 11 of 2026 420 ppm and in week 30 of 2030 the last time estimate is reached. With a 95% confidence interval the value 500 ppm is not reached for the low estimate. Finally, using an 80% confidence interval, in week 12 of 2024 420 ppm is reached, and in week 50 of 2101 500 ppm is reached.

Also note that if we keep forecasting any further, the 95% CI forecast turns downwards and may touch 420 ppm again, so above table would change if we take even wider forecast window.

Now lets generate prediction for year 2100:

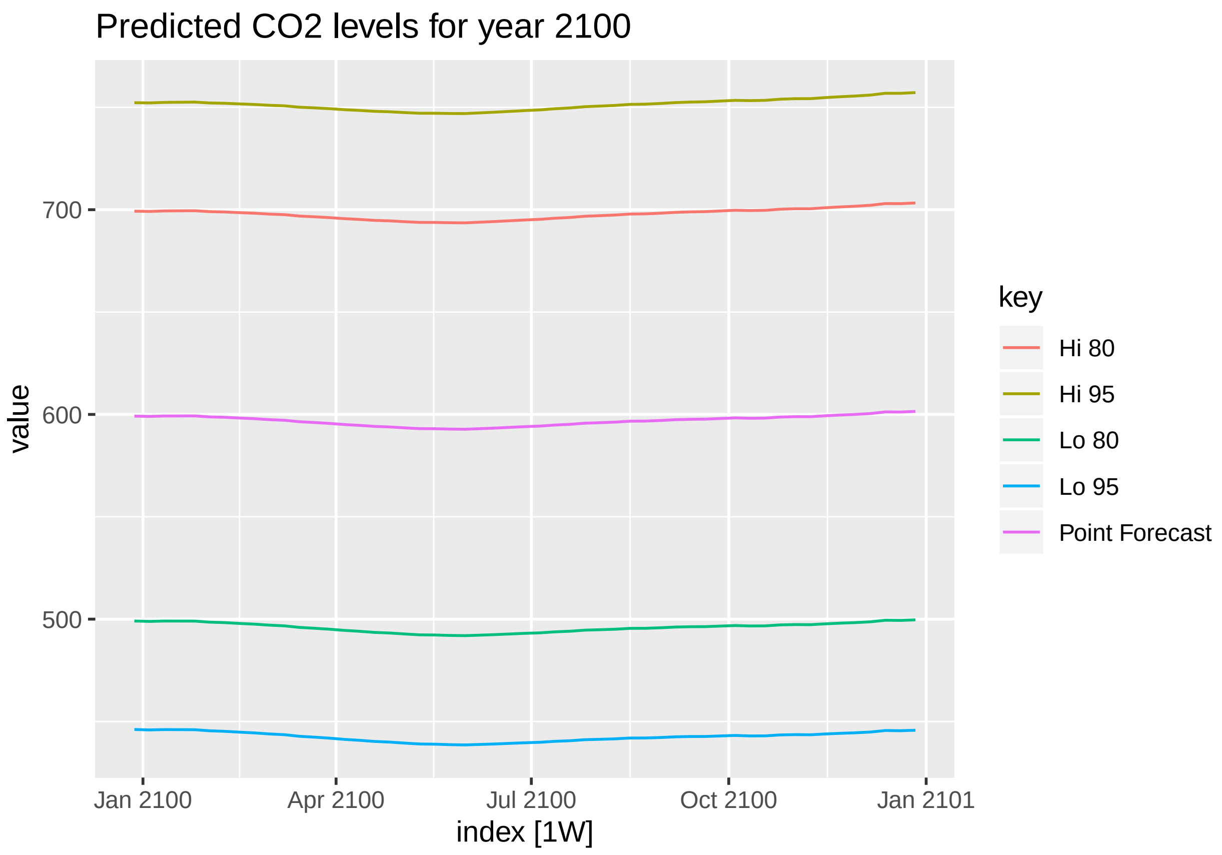
```
model.arima.best.NSA
```

```
## Series: as.ts(co2.noaa.weekly.ts.training)
## ARIMA(1,1,1)(0,1,1)[52]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.2490  -0.7955  -0.8049
## s.e.  0.0326   0.0232   0.0130
##
## sigma^2 estimated as 0.1895:  log likelihood=-1316.6
## AIC=2641.19   AICc=2641.21   BIC=2664.02
```

```
co2.noaa.weekly.ts.training
```

```
## # A tsibble: 2,277 x 2 [1W]
##       index value
##      <week> <dbl>
##  1 1974 W20  333.
##  2 1974 W21  333.
##  3 1974 W22  332.
##  4 1974 W23  332.
##  5 1974 W24  332.
##  6 1974 W25  332.
##  7 1974 W26  332.
##  8 1974 W27  331.
##  9 1974 W28  331.
## 10 1974 W29  331.
## # ... with 2,267 more rows
```

```
co2.noaa.weekly.ts.2100 = yearweek(seq(as.Date("2100-01-01"),
    as.Date("2100-12-31"), by = "1 week"))

model.arima.best.NSA.predict.ft %>% as_tibble() %>% ts(start = 2017.99905280457,
    frequency = 52.18) %>% as_tsibble() %>% filter_index("2100-01-01" ~
    .) %>% filter_index(. ~ "2100-12-31") %>% autoplot(.vars = value) +
    labs(title = "Predicted CO2 levels for year 2100")
```



Point estimate for CO2 levels in year 2100 is around 600 ppm. 80% CI CO2 levels predictions are from 500 ppm to 700 ppm whereas 95 % CI CO2 levels predictions are even wider from 450 ppm to 750 ppm.