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

Kevin Fu, Sang-Hyeb Lee, Heather Reed

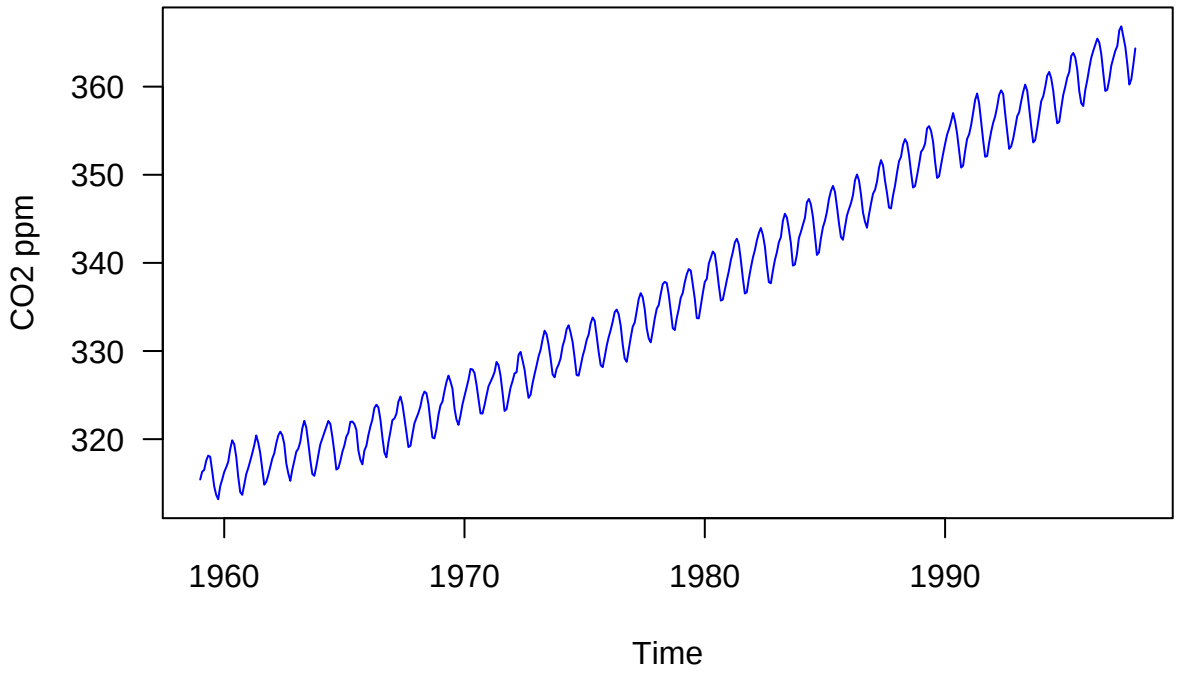
The Keeling Curve

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

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

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

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


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

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.

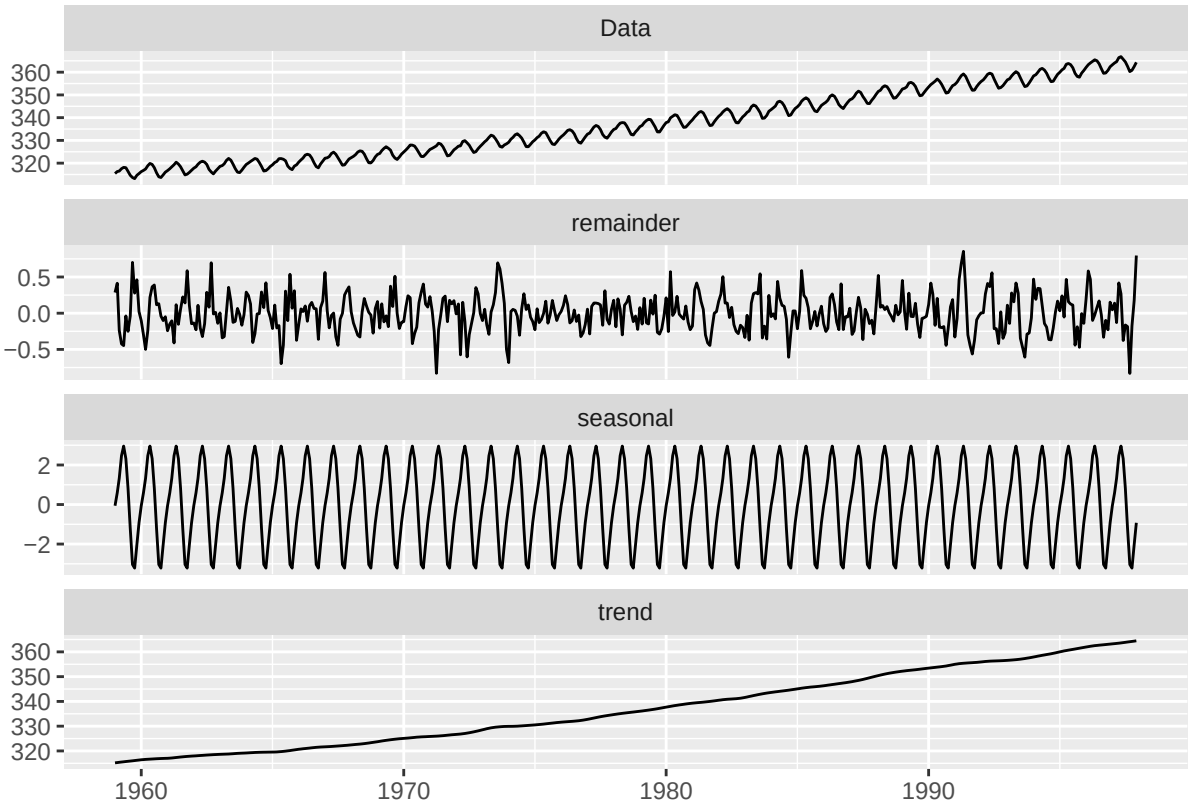
First we load all our packages.

Then we do some basic EDA: plotting the time series, its histogram, ACF, PACF.

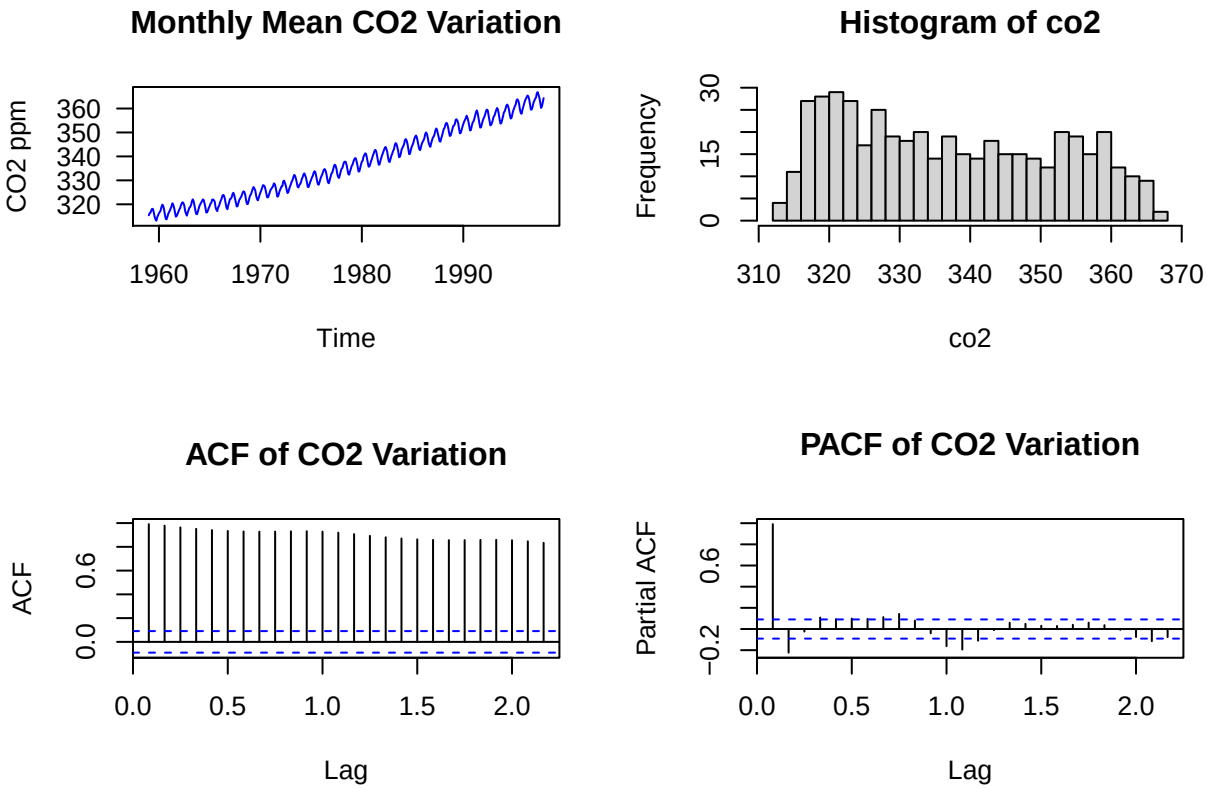
We see that there is a clear trend in the `co2` data (visual inspection), as well as an elevated ACF that does not approach zero quickly to support this hypothesis. There is also a clear seasonal trend in the `co2` data.

The histogram of `co2` is not normally distributed, it looks more uniformly distributed than normally distributed.

```
# Source: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html
autoplot(stl(co2, s.window = 'periodic'), ts.colour = 'black')
```



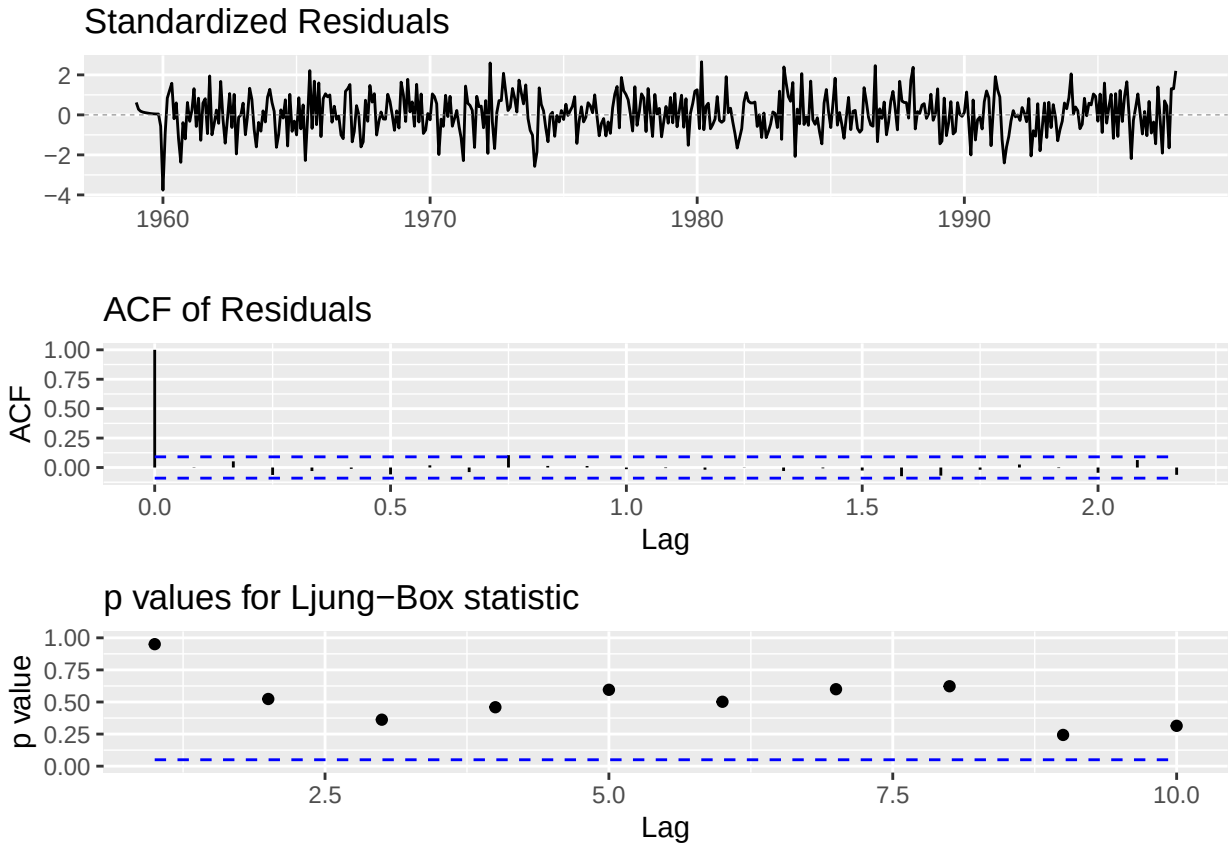

```
par(mfrow=c(2,2))
plot(co2, ylab = expression("CO2 ppm"), col = 'blue', las = 1)
title(main = "Monthly Mean CO2 Variation")
hist(co2, breaks=20)
acf(co2, main="ACF of CO2 Variation")
pacf(co2, main="PACF of CO2 Variation")
```



We examine the plot's residuals, the residuals' ACF and p values over different lags. The residuals resemble white noise (a good thing), do not have very signifcant spikes in the ACF residuals graph (a good thing), and also maintain a high p value in the Ljung-Box test (a good thing, means that the residuals are independent).

```
# Source: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html
ggtsdiag(auto.arima(co2))
```

```
## Warning: `mutate_()` was deprecated in dplyr 0.7.0.
## Please use `mutate()` instead.
## See vignette('programming') for more help
```



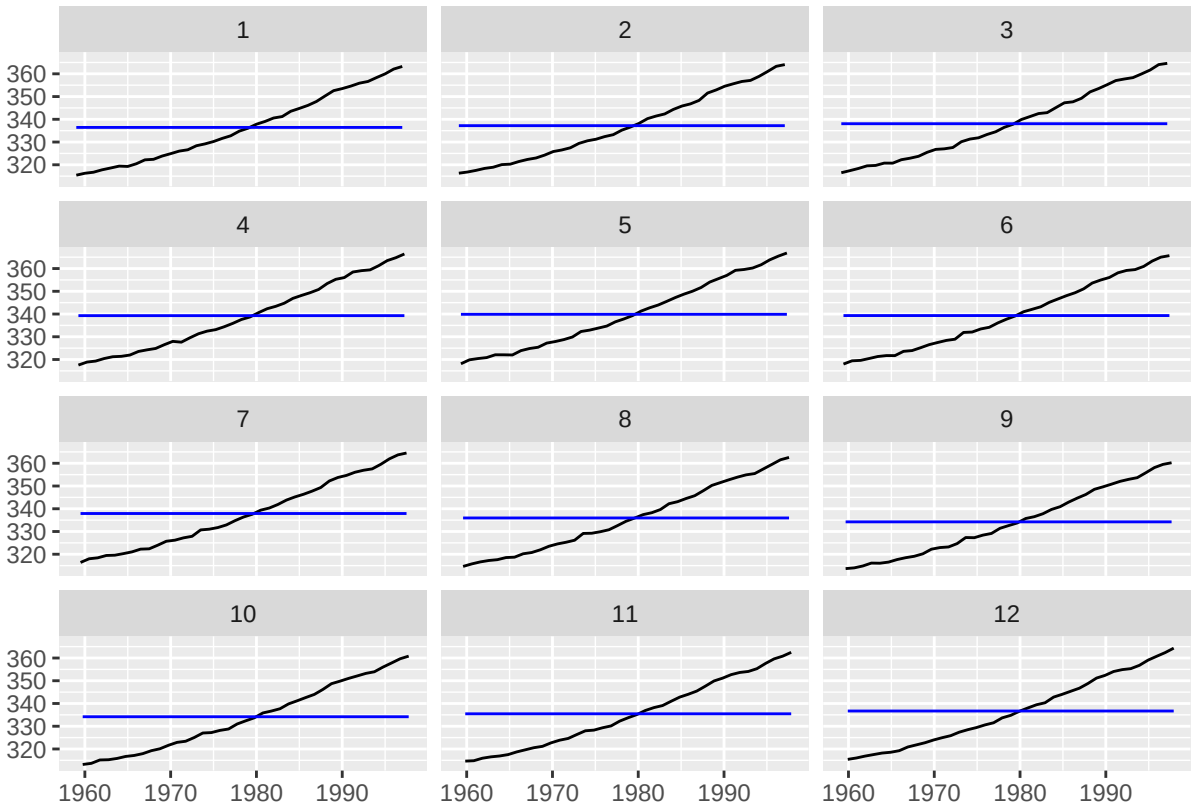
We then examine seasonality. We see that across all months, we have a steadily increasing, fairly linear trend of `co2` emissions over time. The mean across the 12 months looks to be relatively stable between the 330 and 340 range.

Even though no single month stands out as being the leader of co2 emissions across all 12 months, if we look at the boxplot of the co2 emissions by month, we can see that the months of April, May, and June do generate more co2 emissions than other months on average (though not by much).

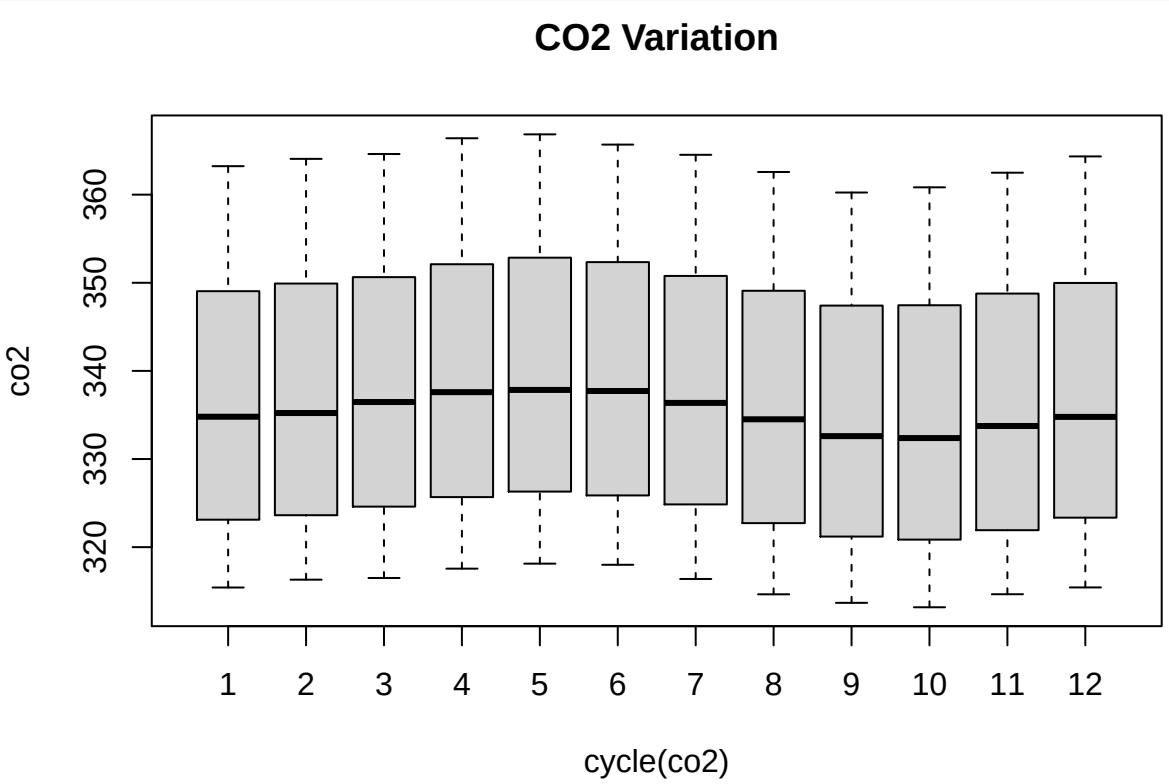
```
# Source: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_ts.html
ggfreqplot(co2)
```

```
## Warning: `summarise_()` was deprecated in dplyr 0.7.0.
## Please use `summarise()` instead.
```

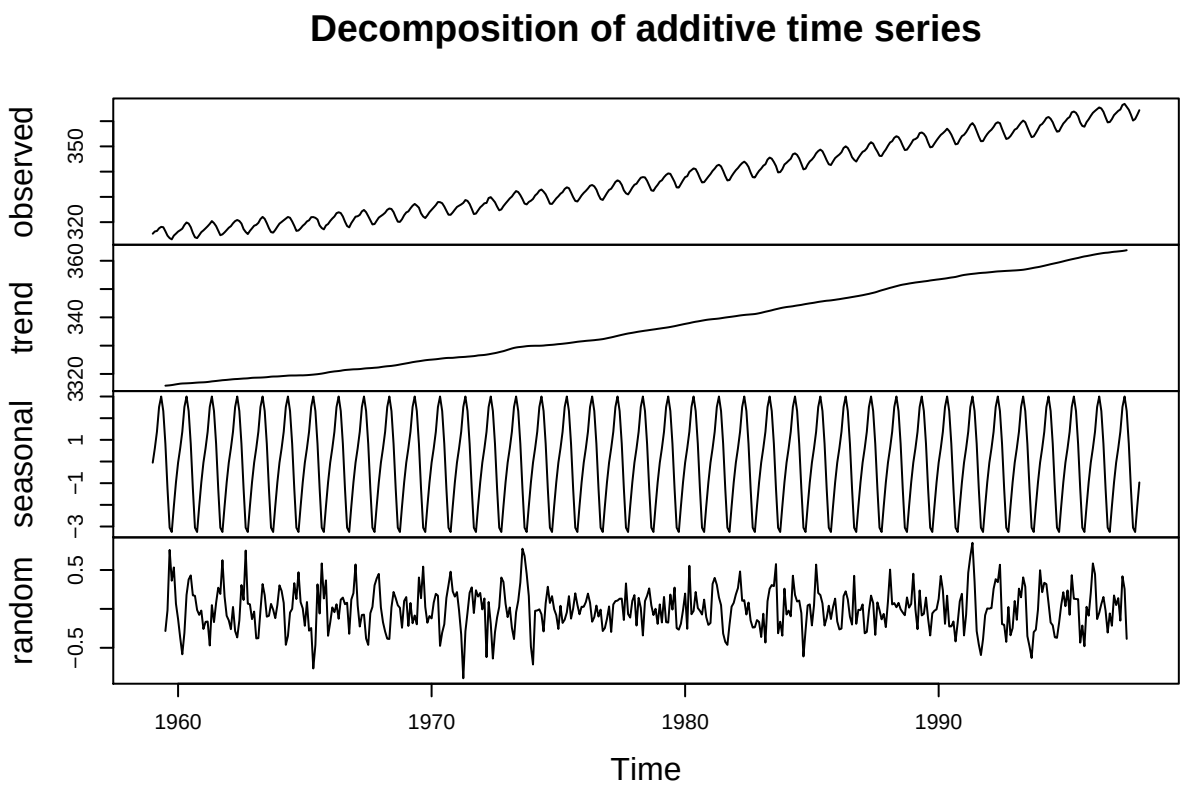
```
## Warning: `group_by_()` was deprecated in dplyr 0.7.0.
## Please use `group_by()` instead.
## See vignette('programming') for more help
```




```
boxplot(co2 ~ cycle(co2), main="CO2 Variation")
```



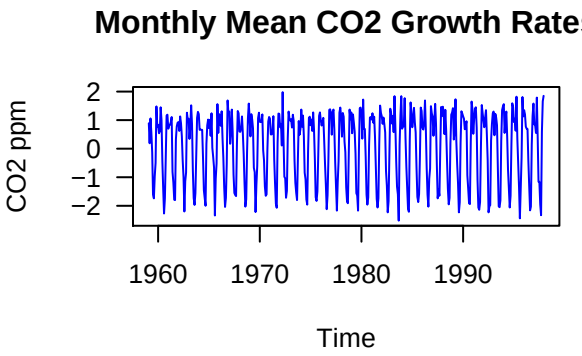
```
# Decomposition
# Source: https://rpubs.com/Mentors_Ubiqum/tslm
co2_decomp <- decompose(co2)
plot(co2_decomp)
```

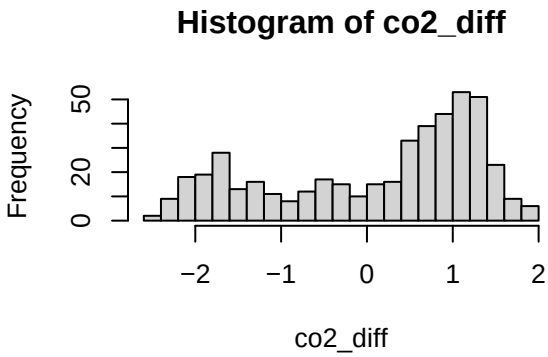


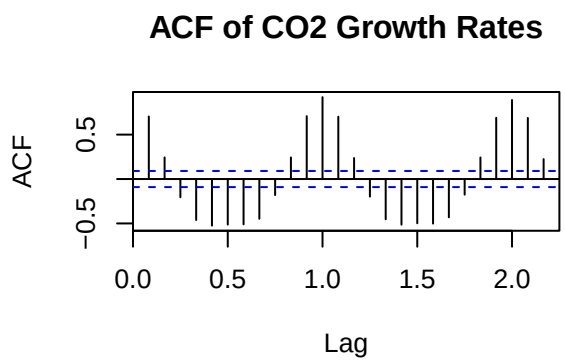
We first difference the data because the `co2` time series clearly has an upward trend, both from a visual perspective and from the elevated ACF plot. We also need to analyze the growth rates of the time series to see if it increase, decreases, or stays the same over time.

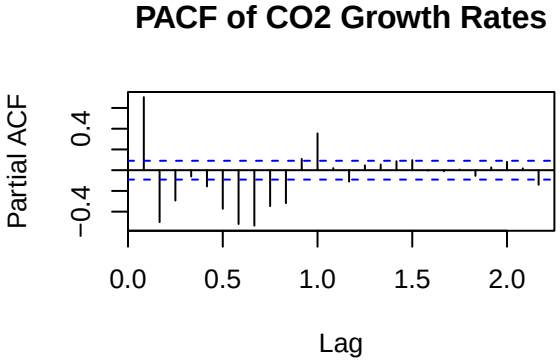
We see from the plots below that there is a pickup in volatility in the monthly mean growth rates after approx. the year 1983 and onwards. Before 1983, there was less volatility, oscillating between -2% and +1% change. After 1983, it looks like the C02 growth rates oscillate between -2% and +1.5% change. So far, we do not see evidence of the monthly mean growth rates leaving this range of -2% and +1.5% for the foreseeable future.

```
co2_diff <- diff(co2)
par(mfrow=c(2,2))
plot(co2_diff, ylab = expression("CO2 ppm"), col = 'blue', las = 1)
title(main = "Monthly Mean CO2 Growth Rates")
hist(co2_diff, breaks=20)
acf(co2_diff, main="ACF of CO2 Growth Rates")
pacf(co2_diff, main="PACF of CO2 Growth Rates")
```









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 up to the present.

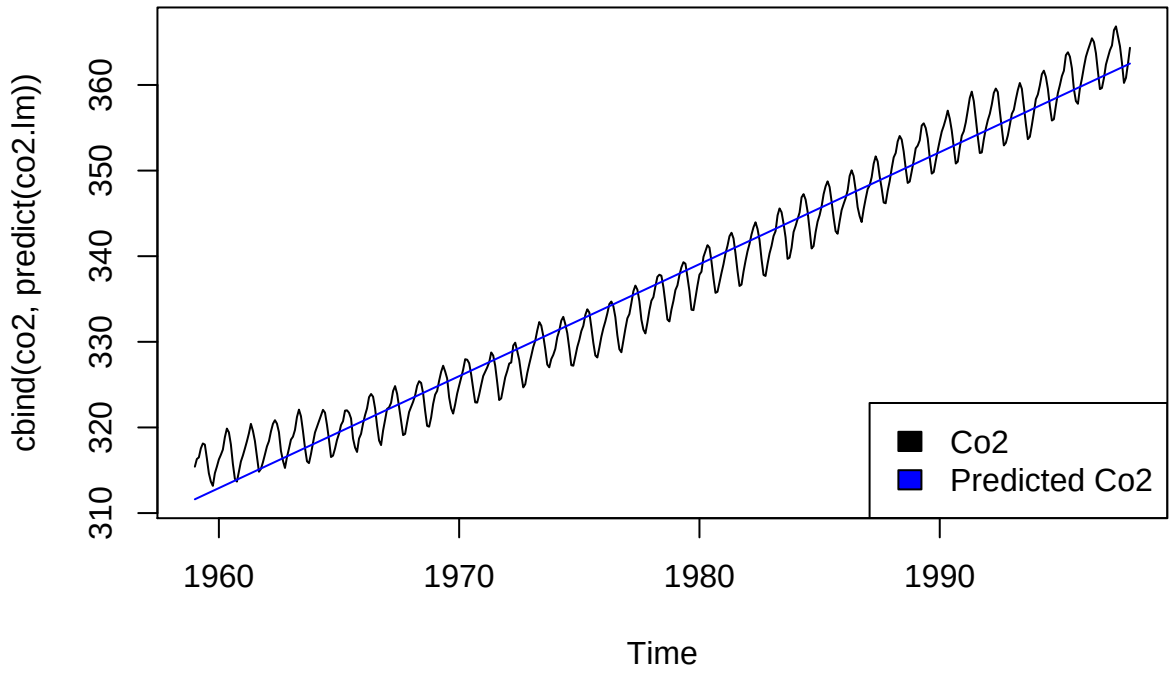
- Fit a linear time trend model to the `co2` series, and examine the characteristics of the residuals.

```
# Week 7
co2.lm = lm(co2 ~ time(co2))
summary(co2.lm)
```

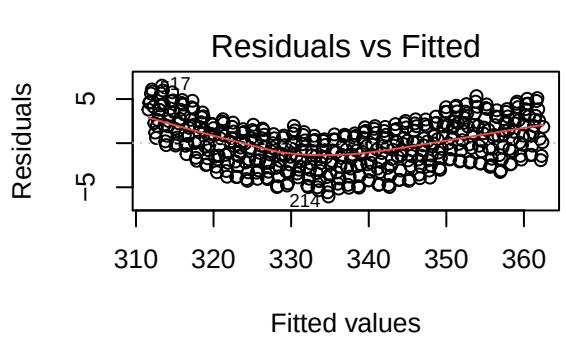
```
##
## Call:
## lm(formula = co2 ~ time(co2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0399 -1.9476 -0.0017  1.9113  6.5149
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.250e+03  2.127e+01  -105.8   <2e-16 ***
## time(co2)    1.308e+00  1.075e-02   121.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

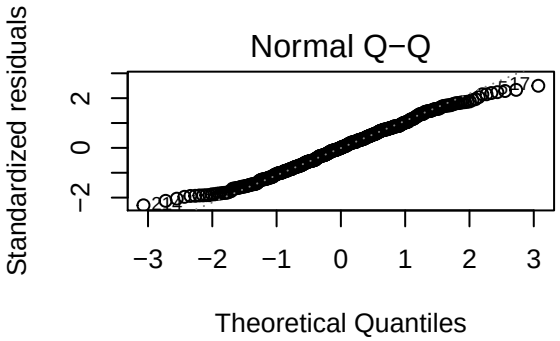
```
##
## Residual standard error: 2.618 on 466 degrees of freedom
## Multiple R-squared:  0.9695, Adjusted R-squared:  0.9694
## F-statistic: 1.479e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

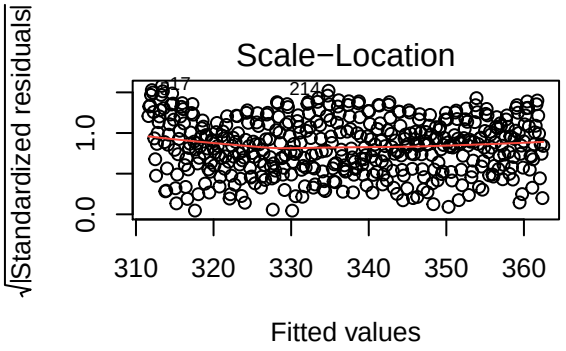
```
plot(cbind(co2, predict(co2.lm)), type="l",
  plot.type = "single", col = c("black", "blue"))
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```

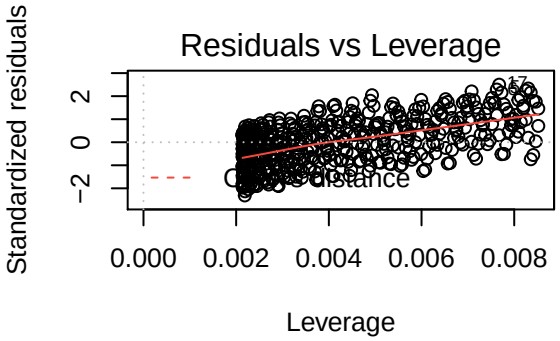



```
# Examine the characteristics of the residuals.
par(mfrow=c(2,2))
plot(co2.lm)
```

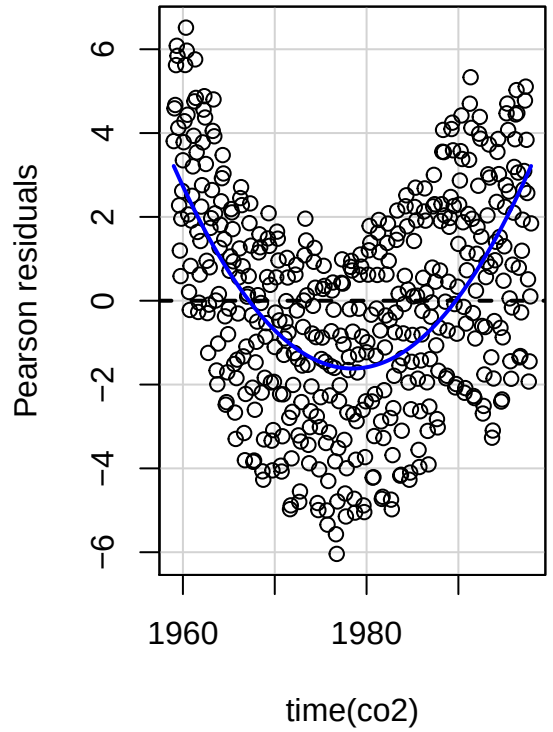


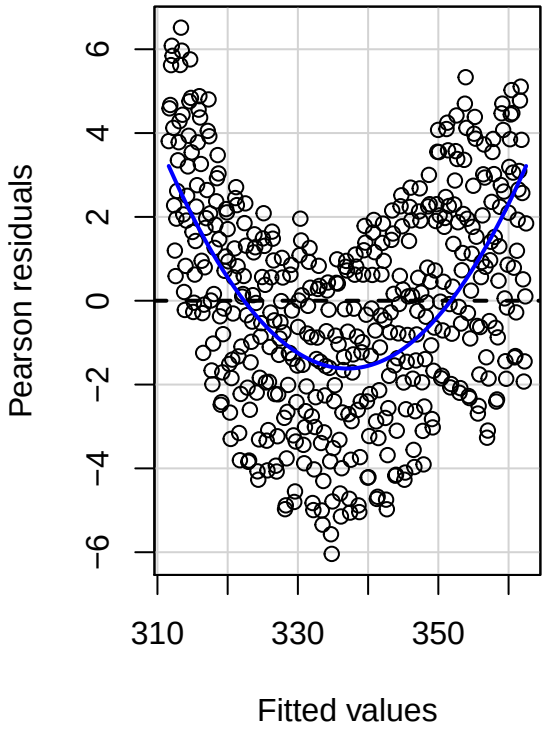







```
residualPlots(co2.lm)
```






```
##            Test stat Pr(>|Test stat|)
## time(co2)     14.342        < 2.2e-16 ***
## Tukey test    14.342        < 2.2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Keep the important #'s in a dataframe.
co2_df_lm = data.frame(
  month = time(co2),
  co2_observation = co2,
  fitted.values = co2.lm$fitted.values,
  residuals = co2.lm$residuals
)
```

The Scale Location plot looks good since the square root of standardized residuals are evenly spread out, forming a flat, horizontal red line.

However, we see some nonlinearity in the shape of a parabola in both the Residuals vs. Fitted graph as well as the Pearson residual graph. There is some nonlinearity towards the beginning and the end of the Normal Q-Q plot too. The Residuals vs. Leverage plot forms an ascending line as opposed to the flat horizontal line we would like to see. Since this is a basic linear model, we do not expect it to be the best and for the residuals to fulfill all the criteria, so we will move onto the next model, the quadratic model.

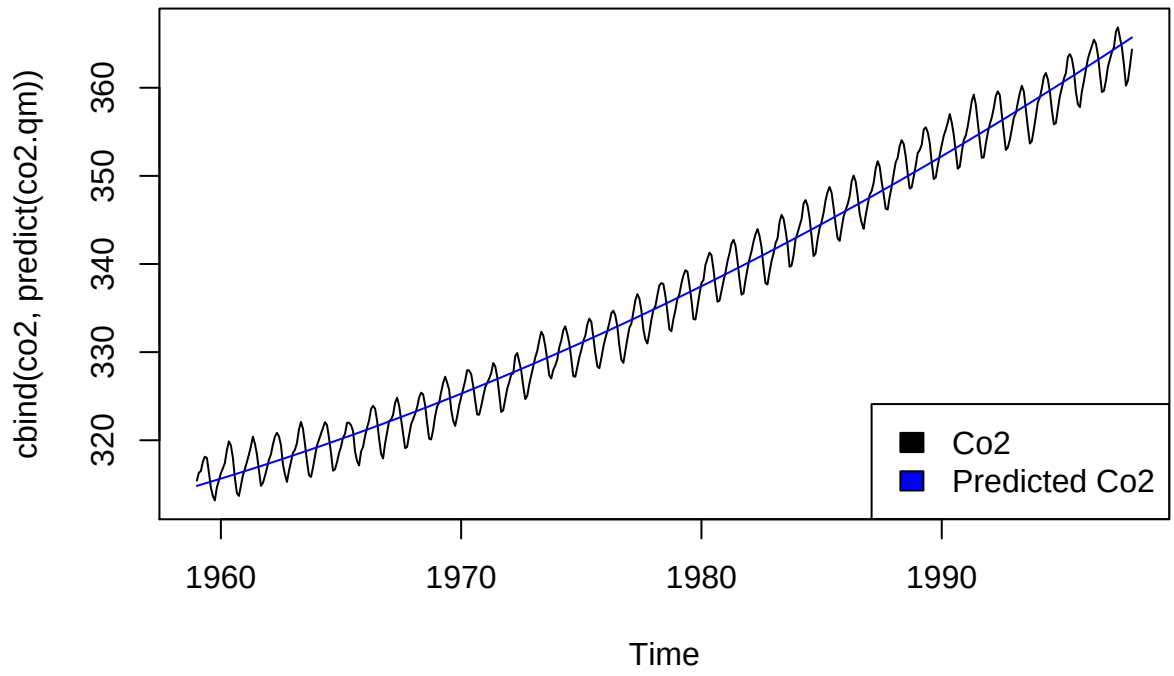
- Compare this to a higher-order polynomial time trend model.

```
# Week 7
# Source: https://rstudio-pubs-static.s3.amazonaws.com/161970_d6e74b8dc3ac4753a9416b4c21c04ec2
co2.qm = lm(co2 ~ time(co2) + I(time(co2)^2))
summary(co2.qm)
```

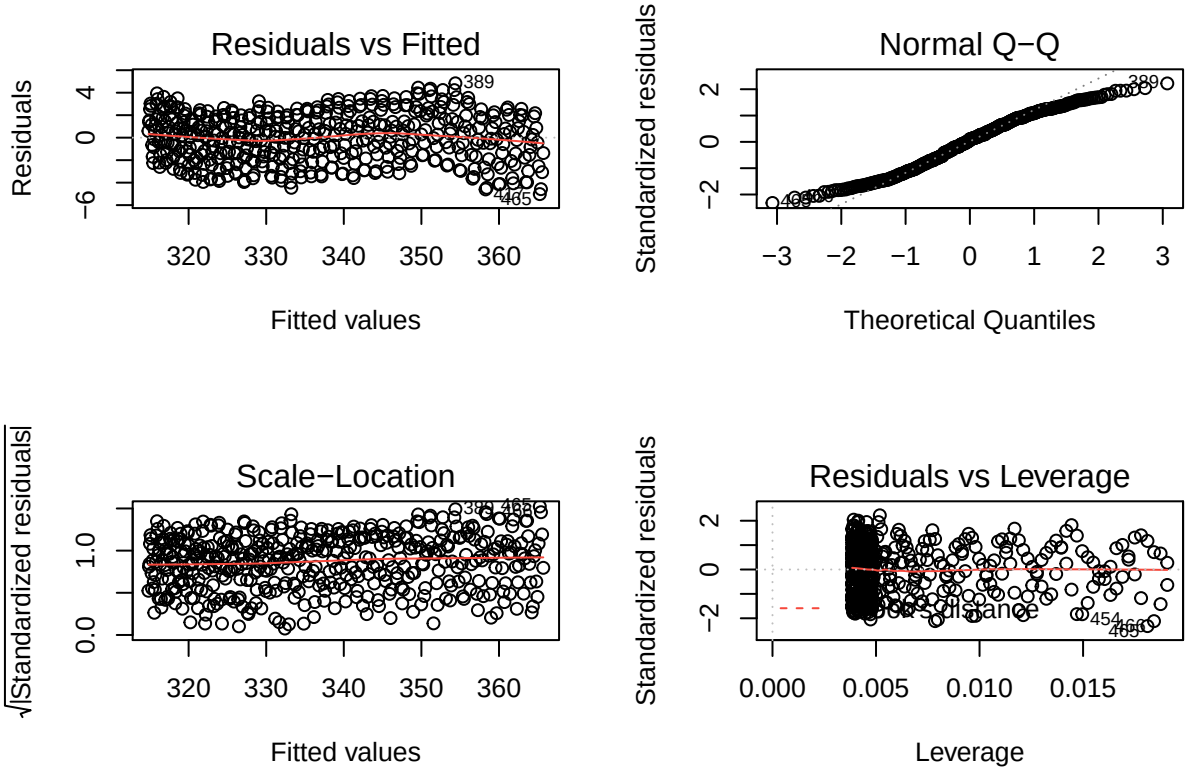
```
##
## Call:
## lm(formula = co2 ~ time(co2) + I(time(co2)^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.770e+04  3.483e+03   13.70   <2e-16 ***
## time(co2)      -4.919e+01  3.521e+00  -13.97   <2e-16 ***
## I(time(co2)^2)  1.276e-02  8.898e-04   14.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.182 on 465 degrees of freedom
## Multiple R-squared:  0.9788, Adjusted R-squared:  0.9787
## F-statistic: 1.075e+04 on 2 and 465 DF,  p-value: < 2.2e-16
```

```
plot(cbind(co2, predict(co2.qm)), type="l",
  plot.type = "single", col = c("black", "blue"))
legend("bottomright",
```

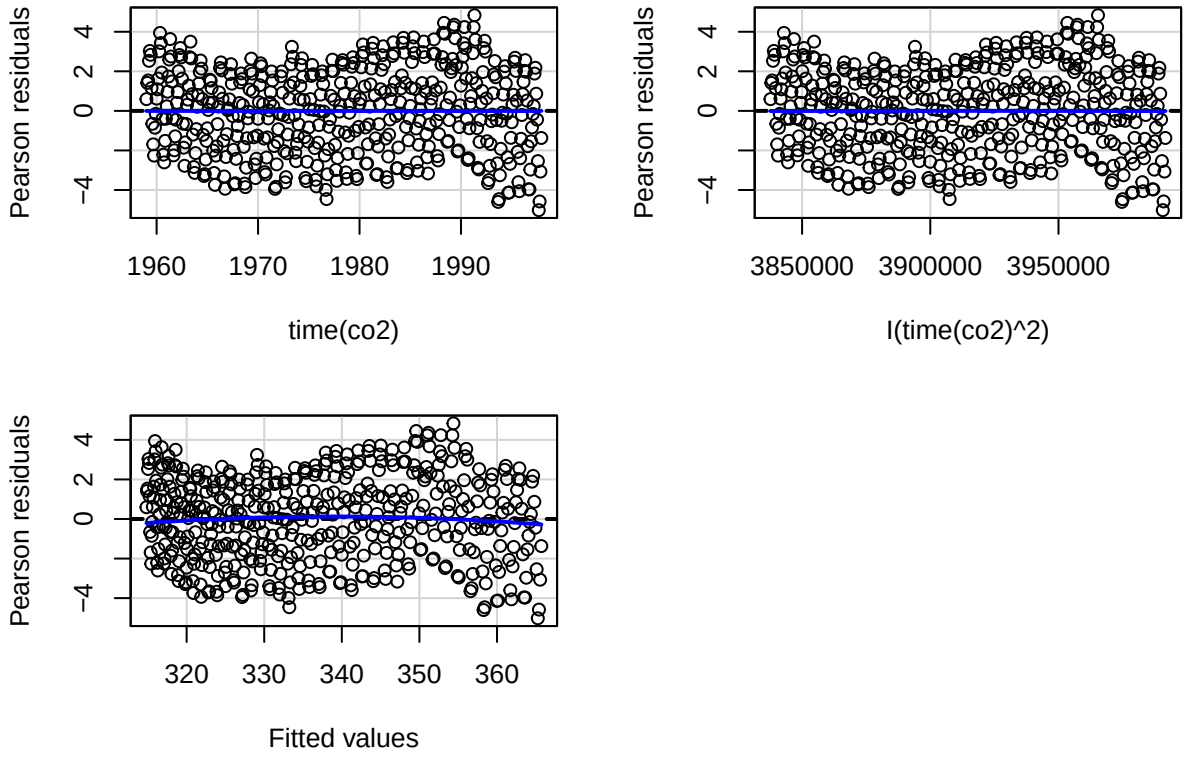
```
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```




```
# Examine the characteristics of the residuals.
par(mfrow=c(2,2))
plot(co2.qm)
```



```
residualPlots(co2.qm)
```




```
##                 Test stat Pr(>|Test stat|)
## time(co2)          0.6526           0.5144
## I(time(co2)^2)    -5.7640        1.500e-08 ***
## Tukey test        -5.8131        6.133e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Keep the important #'s in a dataframe.
co2_df_qm = data.frame(
  month = time(co2),
  co2_observation = co2,
  fitted.values = co2.qm$fitted.values,
  residuals = co2.qm$residuals
)
```

Comapred to the residual plots for our linear model, we see that most of the residual plots are what we would like to see. The Residuals vs. Fitted plot, the Scale Location plot, the Residuals vs. Leverage plot, and the Pearson residual plots all have evenly distributed standardized residuals, forming that flat red horizontal line we would like to see. The only exception here is that we see more nonlinearity in our Normal Q-Q plot. However, overall, this model's residuals is much better than the simple linear model.

```
AIC(co2.qm)
```

```
## [1] 2063.536
```

```
AIC(co2.lm)
```

```
## [1] 2232.961
```

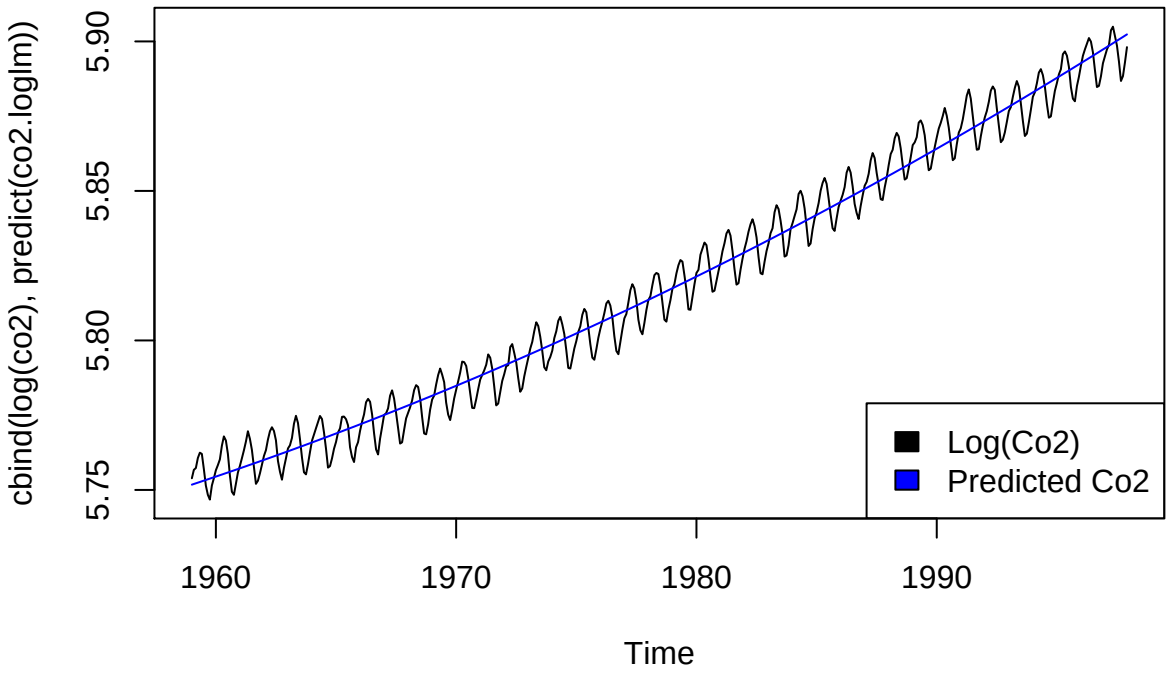
Since both the dependent variables are the same (co2), we can compare the AIC between the 2 models. As expected, the quadratic model generates a lower AIC error, which means we should prefer that one over the linear model.

- Discuss whether a logarithmic transformation of the data would be appropriate.

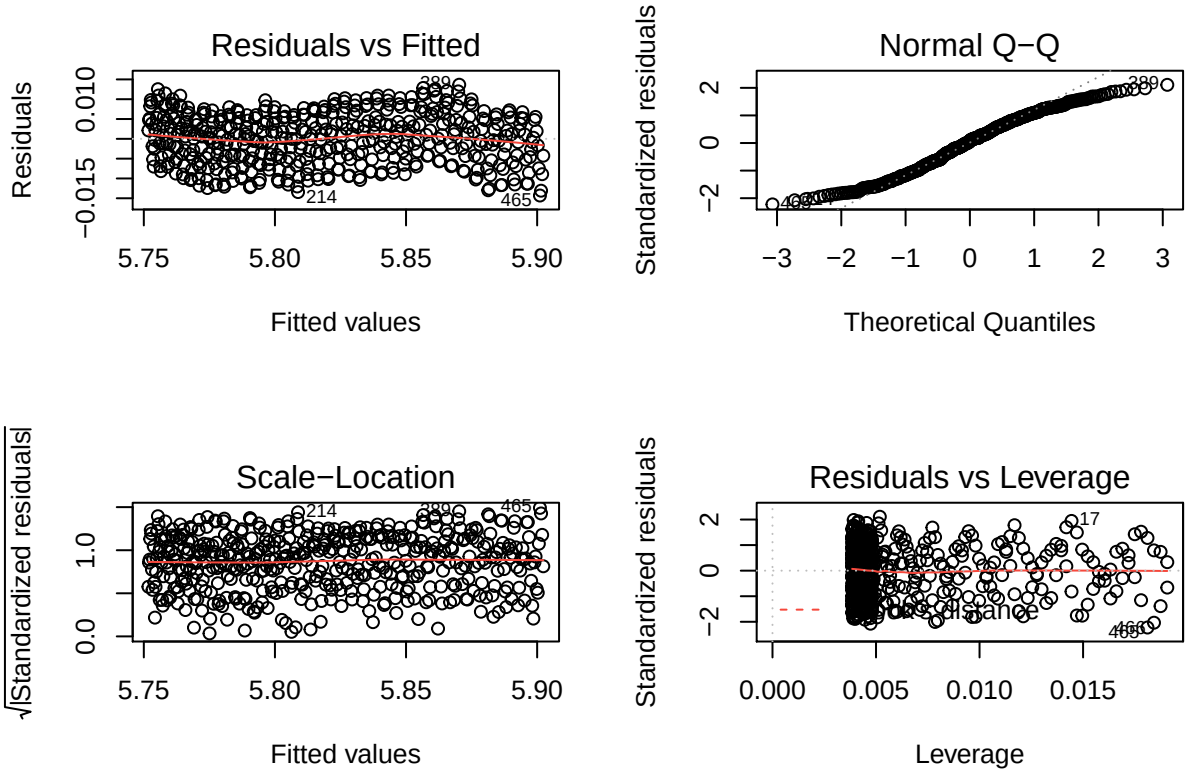
```
# Source: https://rstudio-pubs-static.s3.amazonaws.com/161970_d6e74b8dc3ac4753a9416b4c21c04ec2
# Source: 34:11 https://zoom.us/rec/play/wzgEibLbJUX1iHrUNP6iqLCz3OFSziDJljy8mHd6KdbzLWUQXJACk
co2.loglm = lm(log(co2) ~ time(co2) + I(time(co2)^2))
summary(co2.loglm)
```

```
##
## Call:
## lm(formula = log(co2) ~ time(co2) + I(time(co2)^2))
##
## Residuals:
##        Min          1Q      Median          3Q         Max
## -0.0143052 -0.0050832  0.0005277  0.0052757  0.0136508
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.193e+02  1.036e+01   11.52   <2e-16 ***
## time(co2)      -1.186e-01  1.047e-02  -11.32   <2e-16 ***
## I(time(co2)^2)  3.094e-05  2.646e-06   11.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00649 on 465 degrees of freedom
## Multiple R-squared:  0.9786, Adjusted R-squared:  0.9785
## F-statistic: 1.061e+04 on 2 and 465 DF,  p-value: < 2.2e-16
```

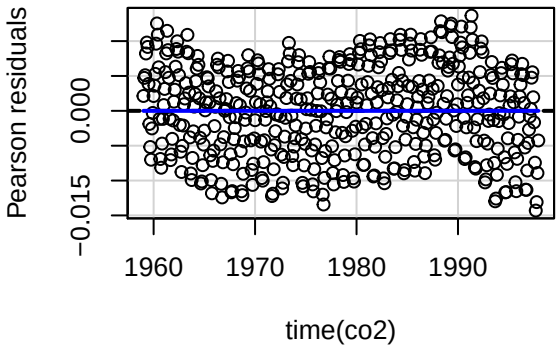
```
plot(cbind(log(co2), predict(co2.loglm)), type="l",
  plot.type = "single", col = c("black", "blue"))
legend("bottomright",
       legend = c("Log(Co2)", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```

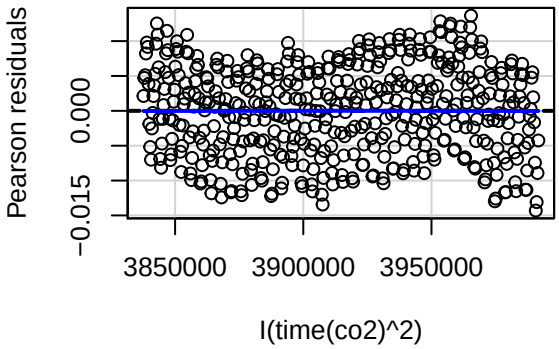


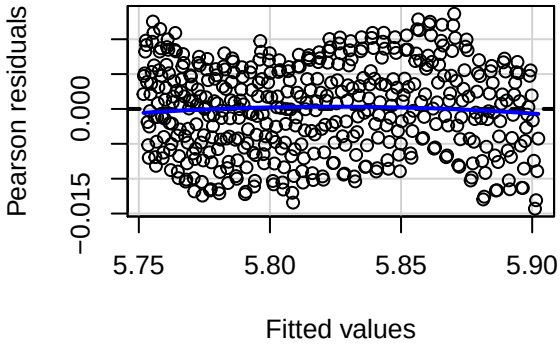

```
# Examine the characteristics of the residuals.
par(mfrow=c(2,2))
plot(co2.loglm)
```




```
residualPlots(co2.loglm)
```








```
##                 Test stat Pr(>|Test stat|)
## time(co2)          0.7214            0.471
## I(time(co2)^2)    -6.2340        1.023e-09 ***
## Tukey test        -6.2666        3.690e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Keep the important #'s in a dataframe.
co2_df_loglm = data.frame(
  month = time(co2),
  co2_observation = co2,
  fitted.values = exp(co2.loglm$fitted.values),
  residuals = co2 - exp(co2.loglm$fitted.values)
)
```

We see very similar residual plots in our exponential model vs. our quadratic model. Therefore, we cannot rely on how good the residual plots are to pick our model - we will use the sum of squared residuals below to determine if we should implement a log transformation or not.

```
sum(co2_df_lm$residuals^2)
```

```
## [1] 3194.08
```

```
sum(co2_df_qm$residuals^2)
```

```
## [1] 2214.454
```

```
sum(co2_df_loglm$residuals^2)
```

```
## [1] 2241.911
```

We cannot look at the R^2 to determine if this model is good or not since we can get very high R^2's from models that are not that effective.

We also cannot look at the AIC (an additional penalty for additional terms we add), since our dependent variable is different between the log() model vs. the linear & quadratic models. Therefore, we will use the sum of squared residuals to compare across our 3 models.

Since the sum of square residuals is the least in the quadratic model (2214.45), we should proceed with the quadratic model. A log transformation is not necessary since that model produced a higher sum of squared residuals (2241.91).

- Fit a polynomial time trend model that incorporates seasonal dummy variables, and use this model to generate forecasts up to the present.

```
# Source: https://rstudio-pubs-static.s3.amazonaws.com/161970_d6e74b8dc3ac4753a9416b4c21c04ec2
# least squares to fit a seasonal-means plus linear time trend to the logged co2
month = season(co2)
co2.lm2 = lm(co2 ~ month + time(co2) + I(time(co2)^2))
summary(co2.lm2)
```

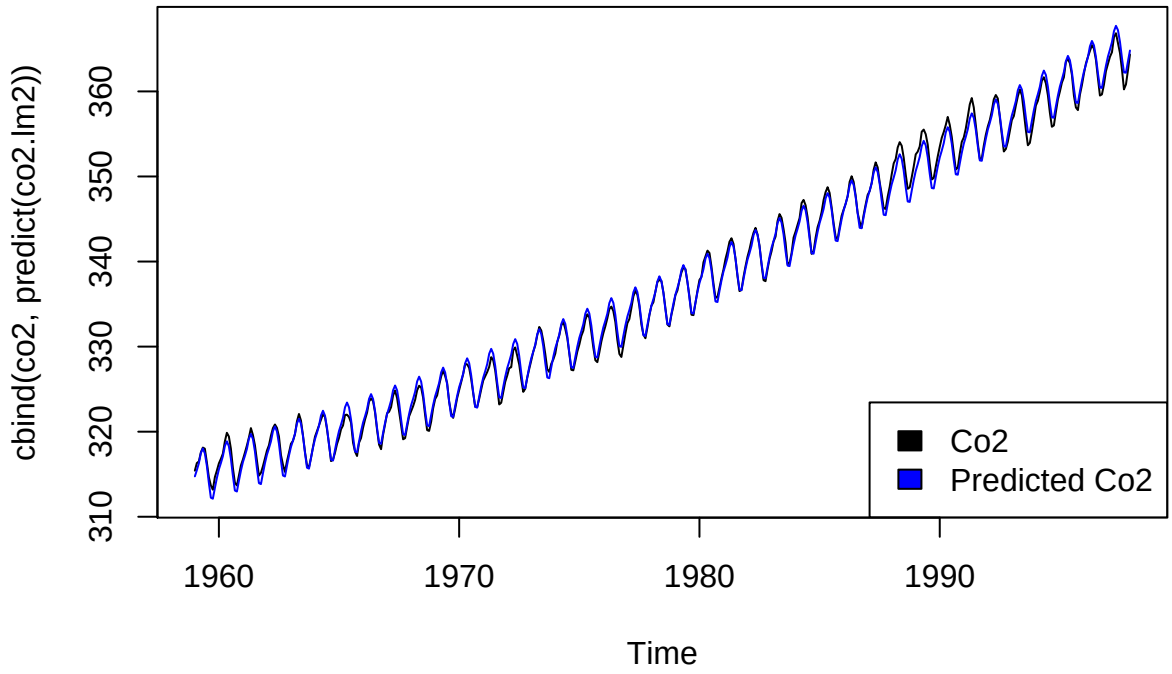
```
##
## Call:
## lm(formula = co2 ~ month + time(co2) + I(time(co2)^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99478 -0.54468 -0.06017  0.47265  1.95480
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.771e+04  1.156e+03  41.289  < 2e-16 ***
## monthFebruary   6.642e-01  1.640e-01   4.051 5.99e-05 ***
## monthMarch      1.407e+00  1.640e-01   8.582  < 2e-16 ***
## monthApril      2.538e+00  1.640e-01  15.480  < 2e-16 ***
## monthMay        3.017e+00  1.640e-01  18.400  < 2e-16 ***
## monthJune       2.354e+00  1.640e-01  14.357  < 2e-16 ***
## monthJuly       8.331e-01  1.640e-01   5.081 5.50e-07 ***
## monthAugust    -1.235e+00  1.640e-01  -7.531 2.75e-13 ***
## monthSeptember -3.059e+00  1.640e-01 -18.659  < 2e-16 ***
## monthOctober   -3.243e+00  1.640e-01 -19.777  < 2e-16 ***
## monthNovember  -2.054e+00  1.640e-01 -12.526  < 2e-16 ***
## monthDecember  -9.374e-01  1.640e-01  -5.717 1.97e-08 ***
## time(co2)      -4.920e+01  1.168e+00 -42.120  < 2e-16 ***
## I(time(co2)^2)  1.277e-02  2.952e-04  43.242  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.724 on 454 degrees of freedom
```

```
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9977
## F-statistic: 1.531e+04 on 13 and 454 DF,  p-value: < 2.2e-16
```

We take our quadratic model from above that performed the best (in terms of lowest sum of squared residuals) and then we add the seasonal dummy variable to it (`month`).

We then plot what our quadratic model would generate based on the existing data set (from 1959 to 1998) below.

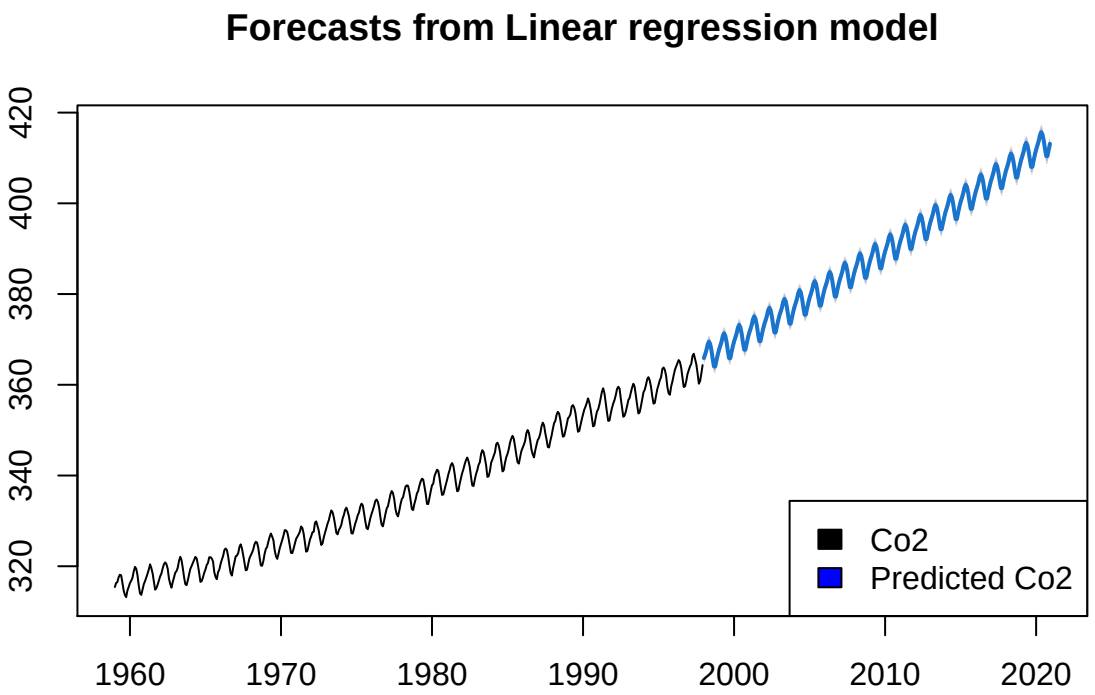
```
# Forecast from 1959 to 1998
plot(cbind(co2, predict(co2.lm2)), type="l",
  plot.type = "single", col = c("black", "blue"))
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```



The plot above fits well. We then proceed to forecast using that model from 1998 to 2021 below.

```
# Forecast from 1998 to 2021
# Source: https://pkg.robjhyndman.com/forecast/reference/tslm.html
# Source: https://rpubs.com/Mentors_Ubiqum/tslm
# Source: http://course1.winona.edu/bdeppa/FIN%20335/Handouts/Regression_Review__part2_.html
n.months <- (2021-1998)*12

my_df_ts <- data.frame(co2 = co2, as.numeric(time(co2)))
names(my_df_ts) <- c("co2", "time")
fit <- tslm(co2 ~ season+poly(trend,2), my_df_ts)
my_fc <- forecast(fit, h=n.months)
plot(my_fc, type="l",
  plot.type = "single")
```

```
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```




```
#autoplot(my_fc, CI=TRUE, legend = TRUE)
         #, legendLabs=c("Co2", "Predicted Co2"))
```

Part 3 (4 points)

Following all appropriate steps, choose an ARIMA model to fit to this `co2` series. Discuss the characteristics of your model and how you selected between alternative ARIMA specifications. Use your model to generate forecasts to the present.

- Following all appropriate steps, choose an ARIMA model to fit to this `co2` series. Discuss the characteristics of your model and how you selected between alternative ARIMA specifications.

Per the above section, we have a trending, non stationary series with seasonality. Therefore, we need to apply first differencing to stabilize the mean, and to remove linear trend and seasonal effects in a monthly trend.

```
# Source: https://zoom.us/rec/play/tljO2dqFHsUHBUVOWGZQr4Sty2Q6V4jbhGKPg8cBEJcfPgdtjotUiS1OYZA
str(co2)
```

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

```
# TS OBJECT -> DATA FRAME
# Source: https://stackoverflow.com/questions/25353002/converting-ts-object-to-data-frame

# TS OBJECT -> TSIBBLE
# Source: https://robjhyndman.com/hyndsight/tsibbles/
```

```
library(tidyverse)
library(tibble)
str(co2)
```

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

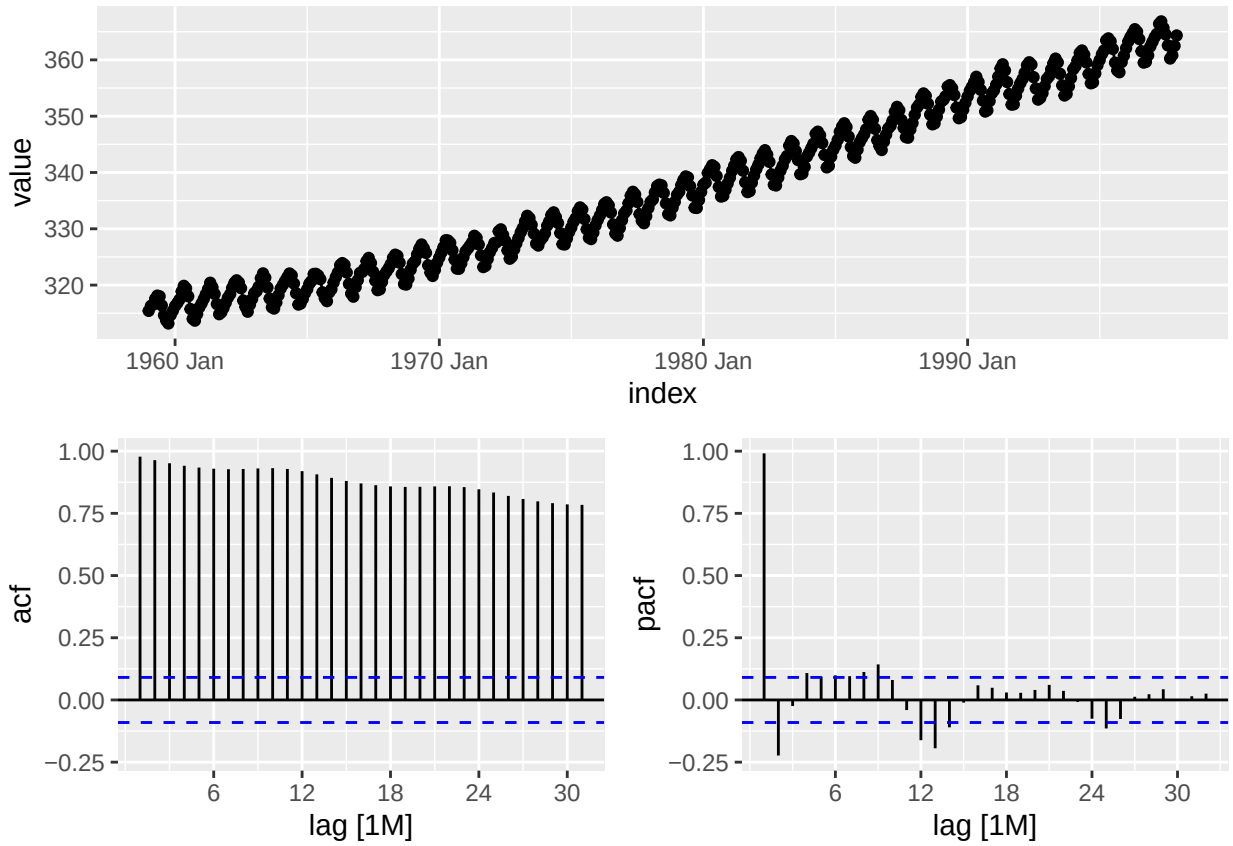
```
co2_tsibble <- co2 %>% as_tsibble()
str(co2_tsibble)
```

```
## tsibble [468 x 2] (S3: tbl_ts/tbl_df/tbl/data.frame)
##  $ index: mth [1:468] 1959 Jan, 1959 Feb, 1959 Mar, 1959 Apr, 1959 May, 1959 Jun...
##  $ value: num [1:468] 315 316 316 318 318 ...
##  - attr(*, "key")= tibble[,1] [1 x 1] (S3: tbl_df/tbl/data.frame)
##   ..$ .rows: list<int> [1:1]
##   .. ..$ : int [1:468] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##  - attr(*, "index")= chr "index"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "index"
##  - attr(*, "interval")= interval [1:1] 1M
##   ..@ .regular: logi TRUE
```

```
summary(co2_tsibble$value)
```

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

```
co2_tsibble %>% gg_tsdisplay(y=value, plot="partial", lag_max = 32)
```

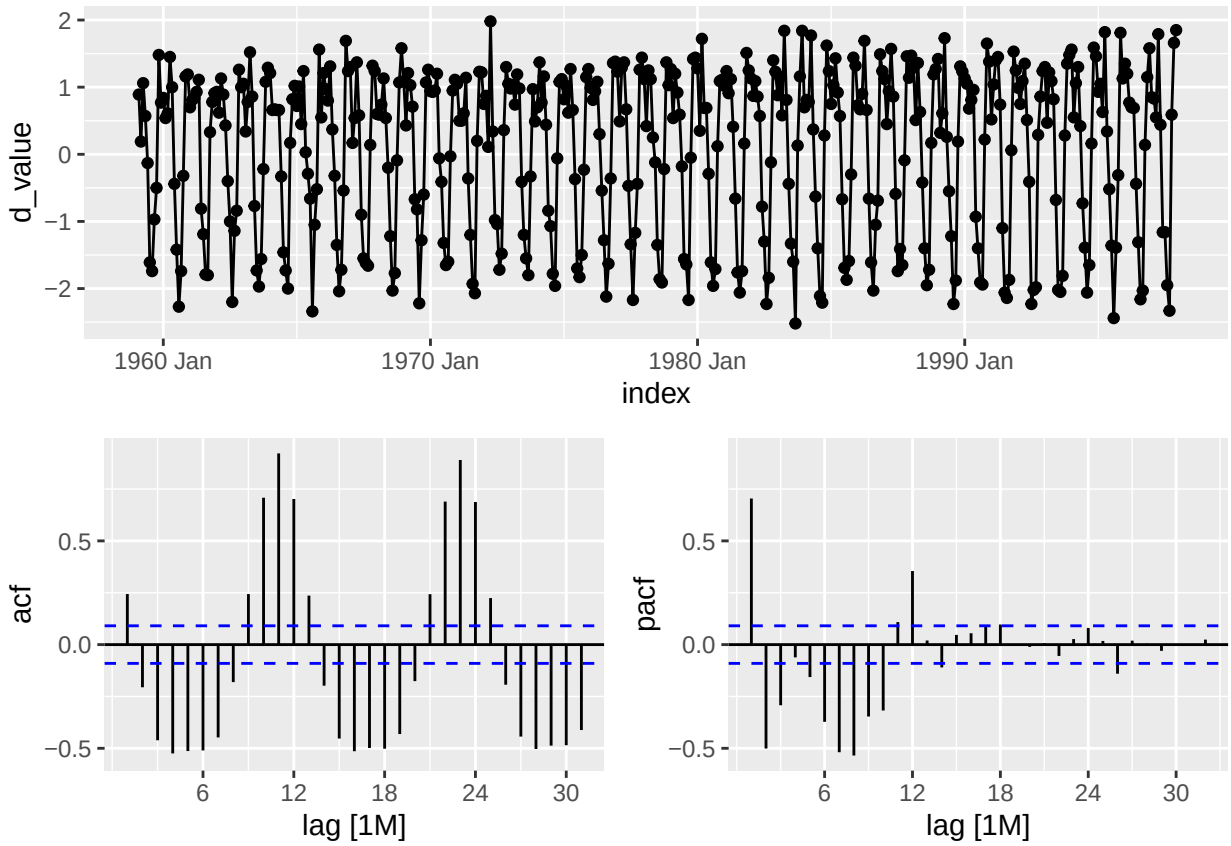


A sustained ACF graph, clear seasonality and trend in our time series means we need to do first differencing and seasonal differencing.

```
# First difference d=1
co2_tsibble_firstdiff <- co2_tsibble %>% mutate(d_value = difference(value, 1))
co2_tsibble_firstdiff %>% gg_tsdisplay(y = d_value, plot = "partial", lag_max = 32)
```

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

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

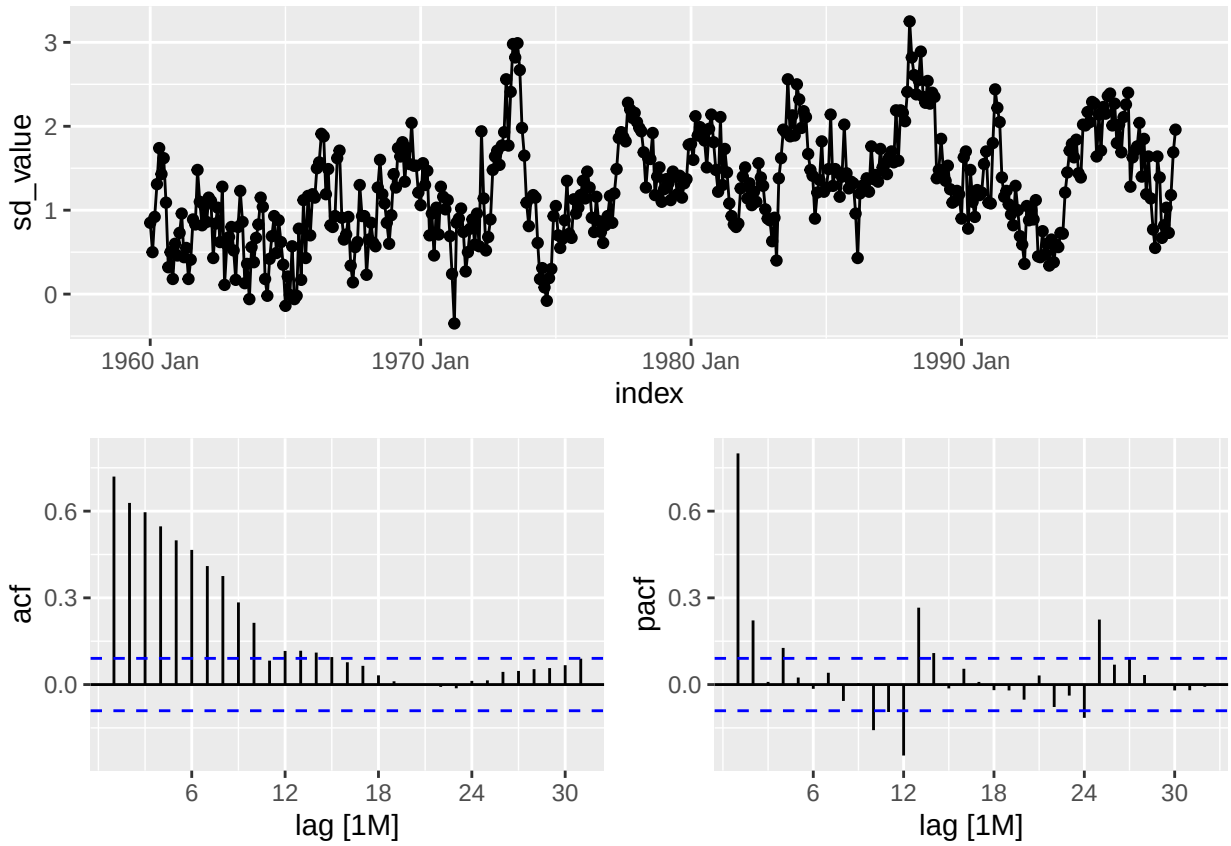


After first differencing, we have a non-trending, white noise-looking type of time series. However, there is clear seasonality. Our ACF plot also shows clear continued volatility as opposed to the slow decline towards zero that we would like to see. The PACF gradually shows less volatility and stays within the blue dotted lines over time (a good thing).

```
# Seasonal difference D=1
co2_tsibble_szndiff <- co2_tsibble %>% mutate(sd_value = difference(value, 12))
co2_tsibble_szndiff %>% gg_tsdisplay(y = sd_value, plot = "partial", lag_max = 32)
```

```
## Warning: Removed 12 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```



After seasonal differencing, we have what looks to be a stationary, white noise-like time series now. The ACF plot looks much better although we see some pickup volatility around lag 30. There are some values that breach the dotted blue lines on the PACF plot around lag 12 and 24.

```
co2_tsibble_firstdiff %>% features(d_value, unitroot_kpss)
```

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

```
co2_tsibble_szndiff %>% features(sd_value, unitroot_kpss)
```

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

We run the KPSS test to see if we should go further in terms of second differencing or not. We get a p value of 0.01 (which is <0.05) after the seasonal and first differencing, so that means we can stop here. At the 5% of level of significance, we can reject the null hypothesis of seasonality.

```
library(forecast)
co2.fit <- Arima(co2, order=c(1,1,1), seasonal=c(1,1,1))
summary(co2.fit)
```

```
## Series: co2
## ARIMA(1,1,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.2456  -0.5749  0.0300  -0.8583
## s.e.  0.1434   0.1237  0.0543   0.0276
##
## sigma^2 estimated as 0.08576:  log likelihood=-84.88
## AIC=179.76   AICc=179.89   BIC=200.36
##
## Training set error measures:
##                       ME      RMSE       MAE          MPE      MAPE      MASE
## Training set 0.01733093 0.2874745 0.2304755 0.005048606 0.0684787 0.1820237
##                       ACF1
## Training set -0.002339927
```

This fits ARIMA(1,1,1)(1,1,1)[12], which means: We differenced the dataset 1 time, using a AR(1) and MA(1), as well as a seasonal component with a difference of 1 time, with an AR(1) and MA(1) to model the seasonal effect.

We use auto.arima to confirm that our model is at least close to what auto.arima deems to be the best model based on the AICc figure.

```
# Source: http://rstudio-pubs-static.s3.amazonaws.com/314877_0c53022faafa4235acd544ae509060f9.
autoarima.fit <- auto.arima(co2)
summary(autoarima.fit)
```

```
## Series: co2
## ARIMA(1,1,1)(1,1,2)[12]
##
## Coefficients:
##          ar1      ma1     sar1     sma1     sma2
##       0.2569  -0.5847  -0.5489  -0.2620  -0.5123
## s.e.  0.1406   0.1203   0.5881   0.5703   0.4820
##
## sigma^2 estimated as 0.08576:  log likelihood=-84.39
## AIC=180.78   AICc=180.97   BIC=205.5
##
## Training set error measures:
##                       ME     RMSE       MAE          MPE       MAPE      MASE
## Training set 0.01742092 0.287159 0.2303994 0.005073769 0.06845665 0.1819636
##                       ACF1
## Training set -0.002858162
```

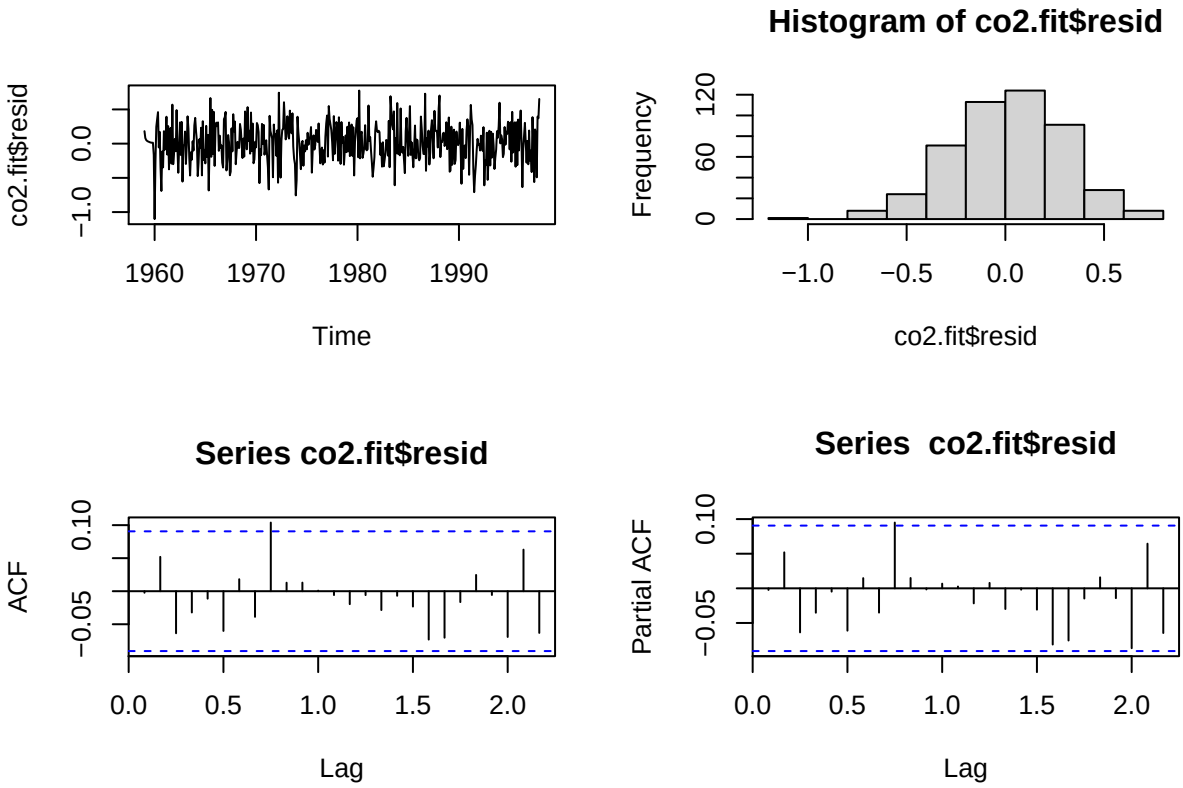
It looks like the auto.arima picked a ARIMA(1,1,1)(1,1,2)[12] model, with an AICc of 180.97. Our ARIMA model above of ARIMA(1,1,1)(1,1,1)[12] had a somewhat better AICc of 179.89. This means our model is in-line if not better than what we should use to model this `co2` variable.

We perform residual diagnostics for our ARIMA model (ARIMA(1,1,1)(1,1,1)[12]).

```
# Residual Diagnostics
#tsdisplay(co2.fit$residuals)
summary(co2.fit$resid)
```

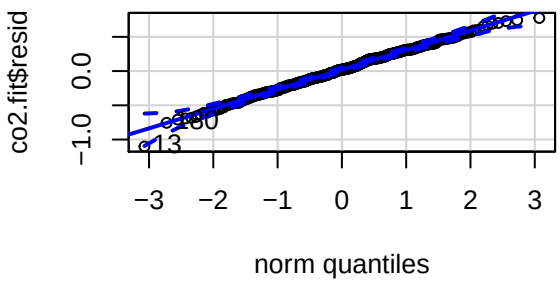
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.10176 -0.17177  0.01981  0.01733  0.21389  0.77527
```

```
par(mfrow= c(2,2))
plot(co2.fit$resid)
hist(co2.fit$resid)
acf(co2.fit$resid)
pacf(co2.fit$resid)
```




```
qqPlot(co2.fit$resid)
```

```
## [1]  13 180
```



We have residuals that resemble white noise, as well as a histogram that resembles a normal distribution (both good signs). In the ACF and PACF graphs, we see that they do not remain

significantly elevated, despite there being random spurts of volatility and picking up towards lag 2. However, since the bouts of volatility do not breach the blue dotted lines, it is not significant. The qqplot shows a textbook y=x line so there are also no issues there.

```
# Box test
Box.test(residuals(co2.fit), lag=24, type="Ljung")
```

```
##
##  Box-Ljung test
##
## data:  residuals(co2.fit)
## X-squared = 20.47, df = 24, p-value = 0.6697
```

```
# Unit root rest
# Source: http://rstudio-pubs-static.s3.amazonaws.com/314877_0c53022faafa4235acd544ae509060f9.
adf.test(residuals(co2.fit))
```

```
## Warning in adf.test(residuals(co2.fit)): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals(co2.fit)
## Dickey-Fuller = -8.2162, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

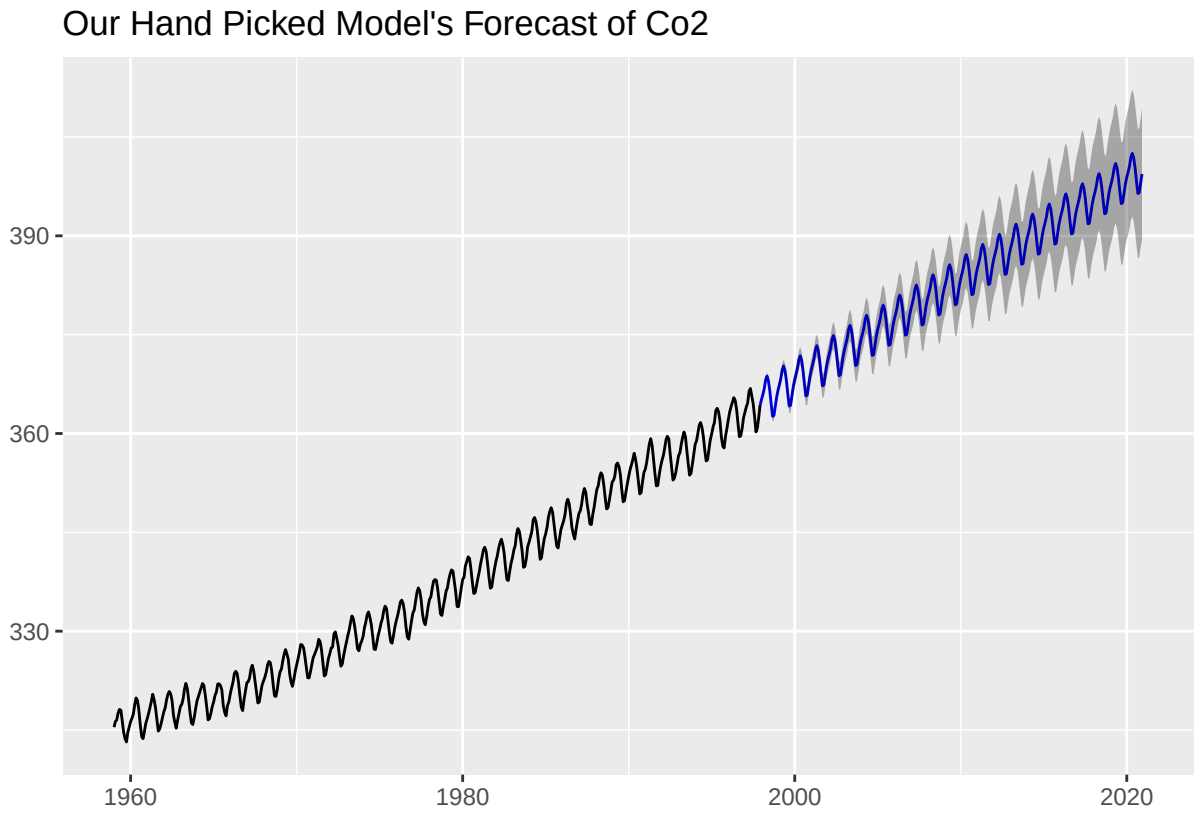
We perform the Box Ljung Test and the Augmented Dickey Fuller Test as well. We fail to reject the Box Ljung test, which is a good thing, meaning that our residuals are independent. We reject the null hypothesis in the Augmented Dickey Fuller Test, which is a good thing, meaning that we do not have a unit root and the residuals in our model are stationary.

- Use your model to generate forecasts to the present.

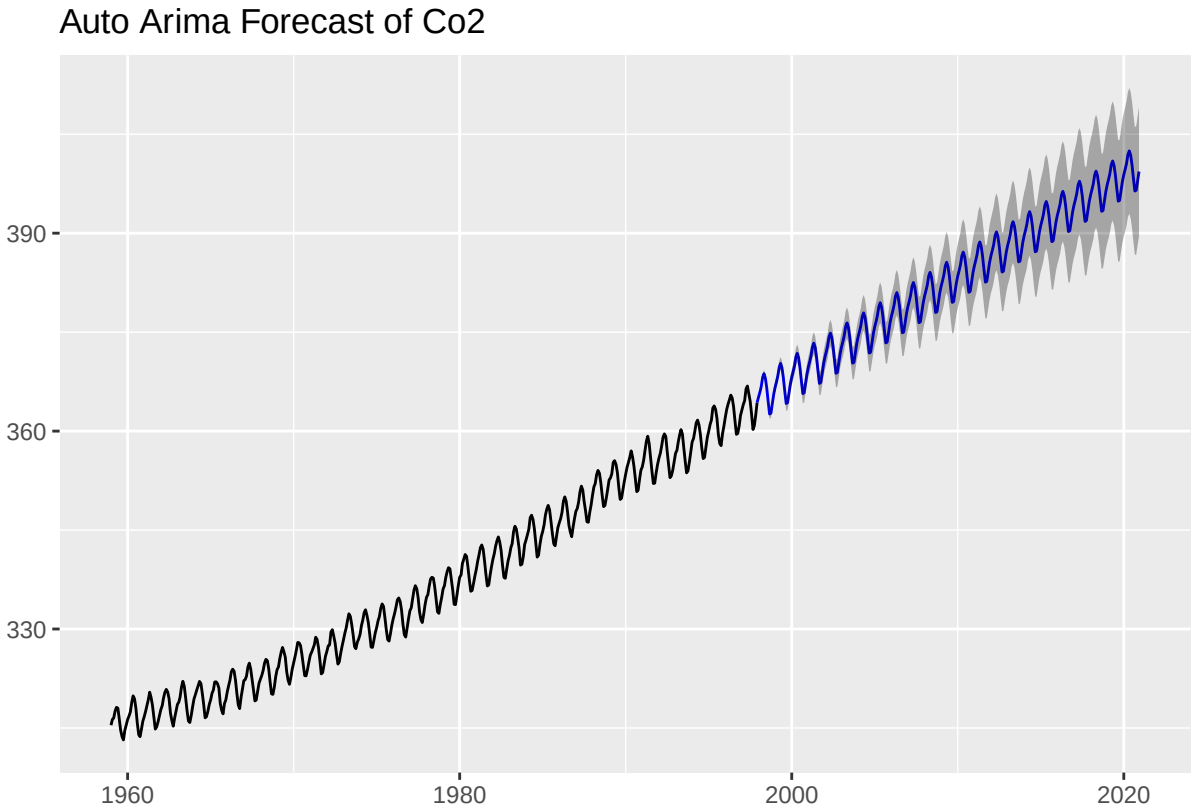
We will plot the model that we picked, and then the model that the auto arima function picked. Both seem like they produce a reasonable forecast, although as we previously discussed our model does produce a lower error score.

```
# Source: https://stats.stackexchange.com/questions/229948/plotting-predicted-values-in-arima-
n.months <- (2021-1998)*12

models_forecast <- co2.fit %>% forecast(h=n.months)
models_forecast %>% autoplot(co2) + ggtitle("Our Hand Picked Model's Forecast of Co2")
```

```
## Warning: `filter_()` was deprecated in dplyr 0.7.0.
## Please use `filter()` instead.
## See vignette('programming') for more help
```




```
# For auto arima model
models_forecast2 <- autoarima.fit %>% forecast(h=n.months)
models_forecast2 %>% autoplot(co2) + ggtitle("Auto Arima Forecast of Co2")
```



Part 4 (5 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, addressing the problem of missing observations and comparing the Keeling Curve's development to your predicitons from 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.

EDA on Weekly Data

Let's start by reading in the data and changing any extraneous observations to NA in order to determine how many missing variables exist within our dataset.

```
co2_weekly <- read.table("co2_weekly_mlo.txt",header = FALSE, sep = "", dec = ".")

co2_weekly <- co2_weekly[, c(1, 2, 3, 5)]
names(co2_weekly) <- c('year', 'month', 'day', 'co2ppm')
co2_weekly$date <- as.Date(paste(co2_weekly$year, co2_weekly$month, co2_weekly$day, sep = '-')
co2_weekly <- co2_weekly[, c('date', 'co2ppm')]
co2_weekly[co2_weekly$co2ppm == -999.99, ]$co2ppm = NA

sum(is.na(co2_weekly))
```

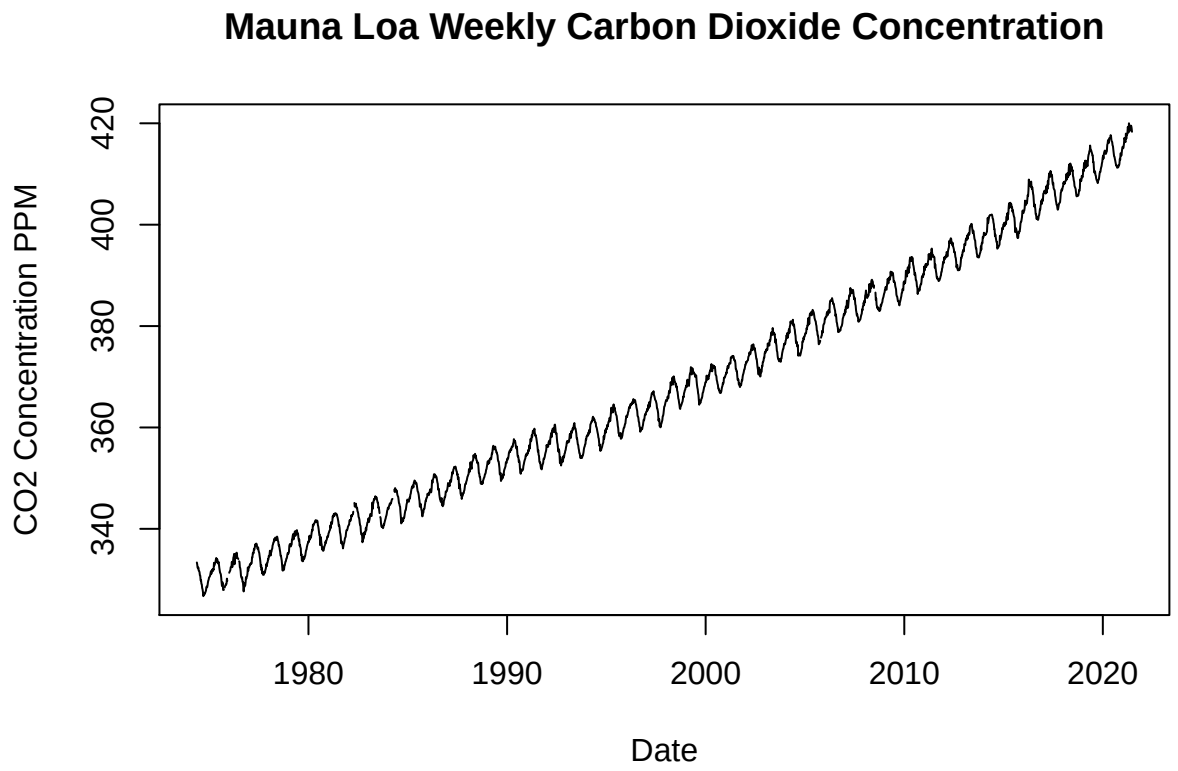
```
## [1] 18
```

```
#18 NA values

summary(co2_weekly)
```

```
##       date                co2ppm
##  Min.   :1974-05-19   Min.   :326.7
##  1st Qu.:1986-02-24   1st Qu.:347.5
##  Median :1997-12-03   Median :365.3
##  Mean   :1997-12-03   Mean   :368.4
##  3rd Qu.:2009-09-11   3rd Qu.:388.5
##  Max.   :2021-06-20   Max.   :420.0
##                       NA's   :18
```

There are 18 missing values within the dataset. Let's visualize our data.

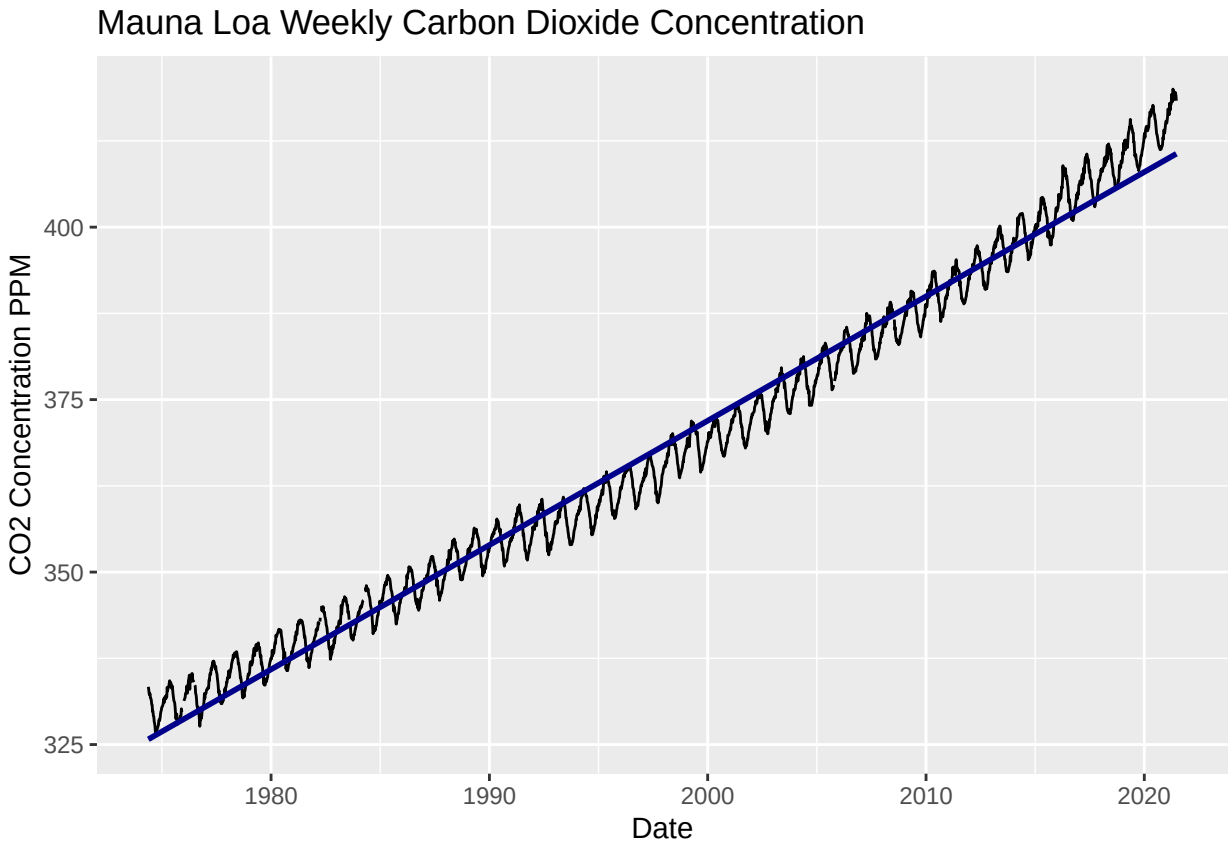
```
plot(
  co2_weekly$date,
  co2_weekly$co2ppm,
  type = 'l',
  xlab = 'Date',
  ylab = 'CO2 Concentration PPM',
  main = 'Mauna Loa Weekly Carbon Dioxide Concentration'
)
```



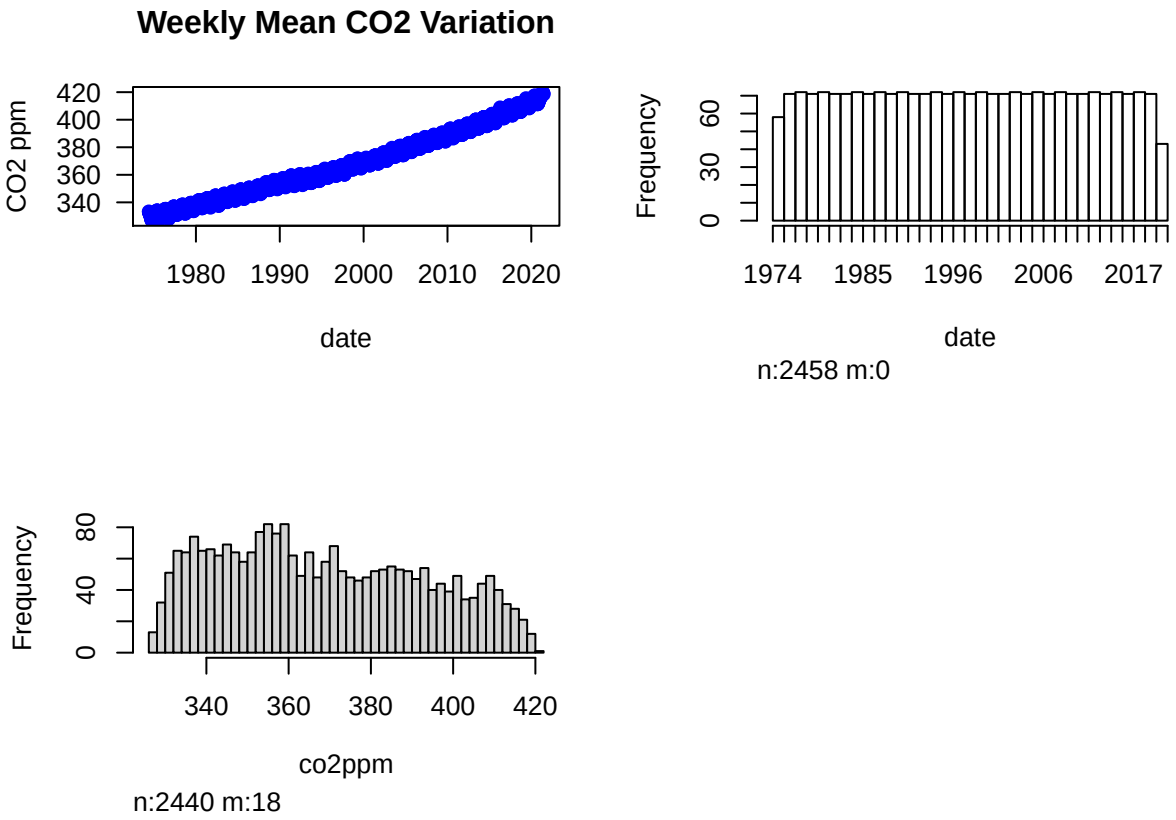

```
ggplot(data = co2_weekly, aes(date, co2ppm)) +
  geom_line() +
  xlab('Date') +
```

```
  ylab('CO2 Concentration PPM') +
  ggtitle('Mauna Loa Weekly Carbon Dioxide Concentration') +
  stat_smooth(method = lm, color = 'dark blue')
```

```
## Warning: Removed 18 rows containing non-finite values (stat_smooth).
```

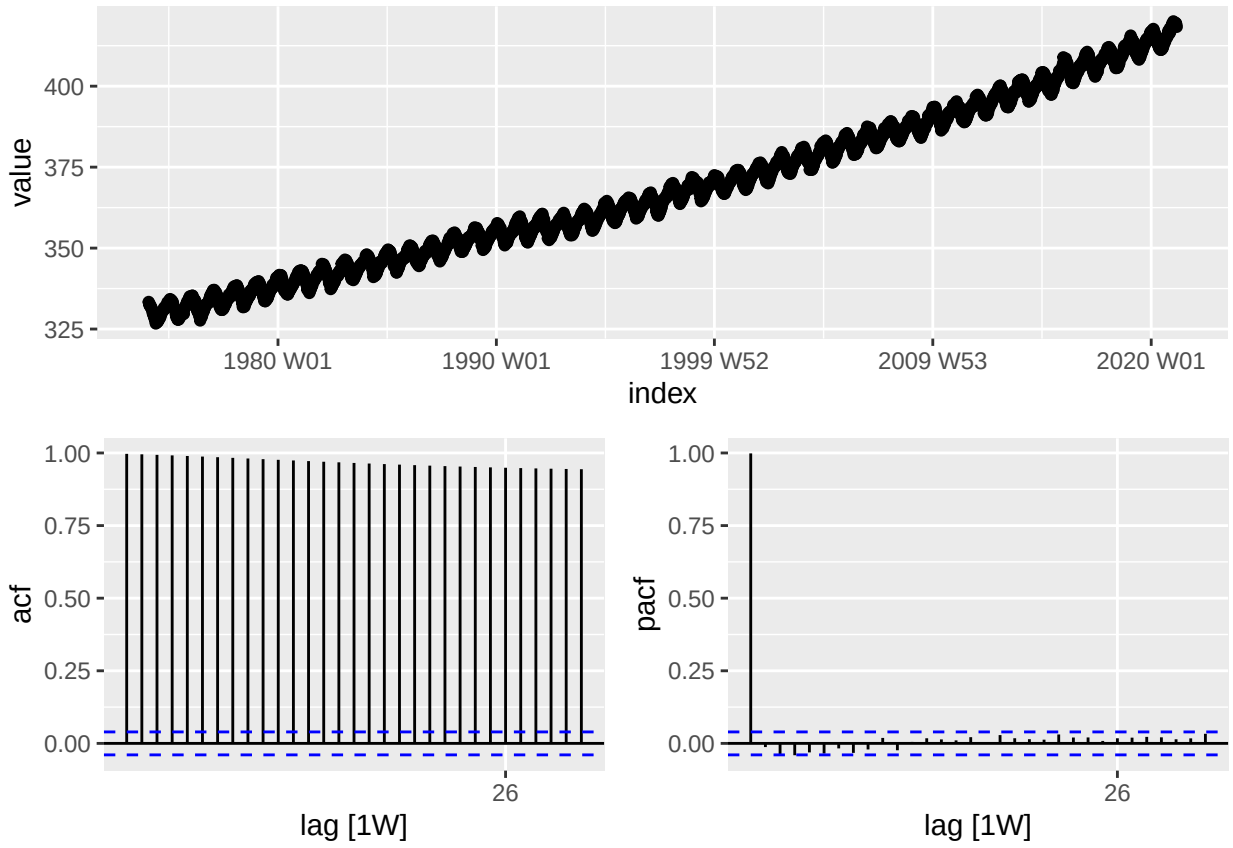



```
par(mfrow=c(2,2))
plot(co2_weekly, ylab = expression("CO2 ppm"), col = 'blue', las = 1)
title(main = "Weekly Mean CO2 Variation")
hist(co2_weekly, breaks=20)
```

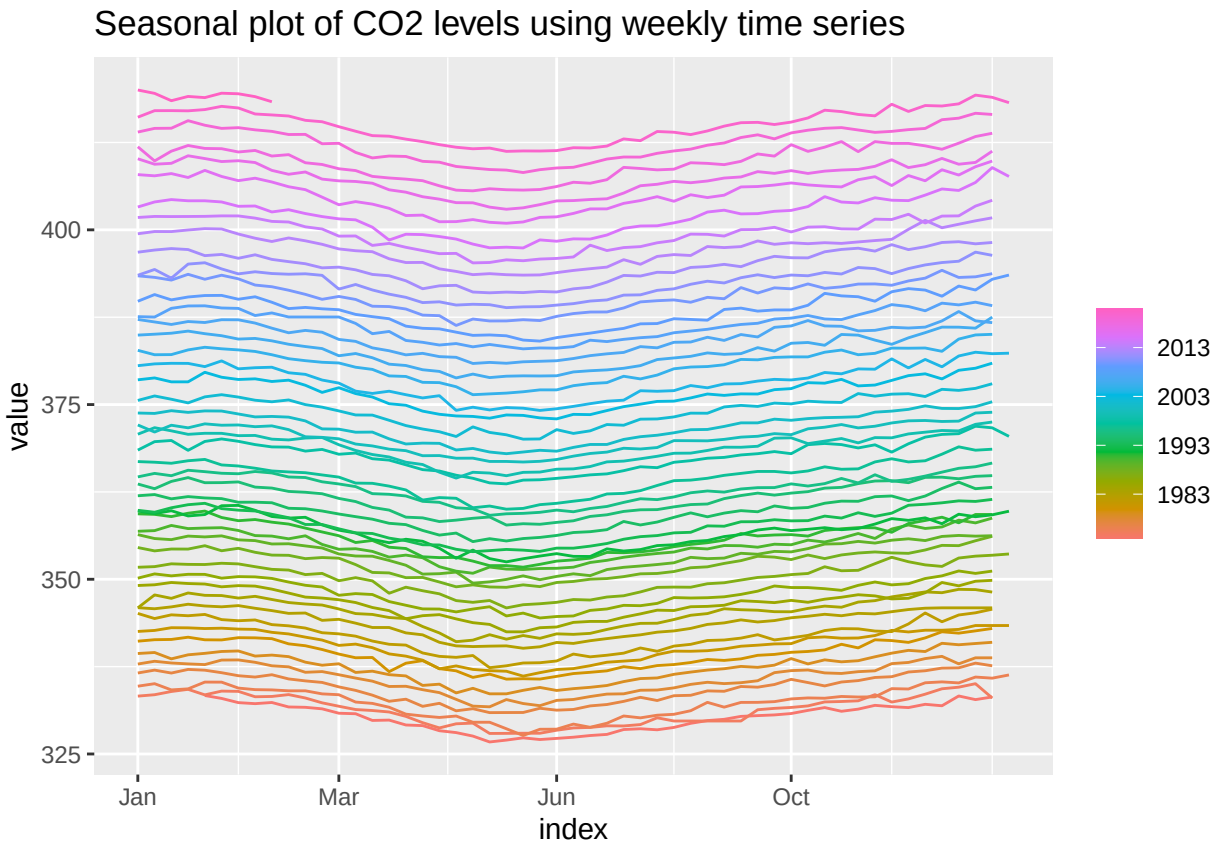


Now we will use the last observation carried forward method to eliminate the missing values from our data and then convert our weekly data to a time series and visualize the time series.

```
co2_weekly <- na_locf(co2_weekly)

co2_weekly.ts <- ts(co2_weekly$co2ppm, start=c(1974,5,19), frequency=52.18)

co2_weekly.ts <- as_tsibble(co2_weekly.ts)

co2_weekly.ts %>% gg_tsdisplay(y=value, plot="partial", lag_max=32)
```




```
co2_weekly.ts %>% gg_season(y=value) +
ggtitle("Seasonal plot of CO2 levels using weekly time series")
```



The visualizations above show that the weekly observations follow a similar pattern to the Keeling's curve. Even though the data is segmented weekly instead of monthly like the data used to create the Keeling's curve, the upward trend and seasonal patterns still persist. The weekly series graph shows both upwards trends and seasonal trends, with the ACF and PACF correlograms confirming this finding. There is an obvious seasonal pattern – CO2 level goes down from January to June but goes back up from June to December. Since the magnitude of the seasonal fluctuation does not vary with the level of time series, an additive seasonal pattern should be more appropriate than a multiplicative seasonal pattern.

EDA on Monthly Data

First, let's convert our weekly data into monthly averages of CO2 concentration.

```
new_monthly <- data.frame(tapply(co2_weekly[[2]], format(co2_weekly[[1]], "%Y-%m"), mean))
describe(new_monthly)
```

```
## new_monthly
##
##  1  Variables      566  Observations
## ---------------------------------------------------------------------------
## tapply.co2_weekly..2....format.co2_weekly..1......Y..m....mean.
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      566        0      565        1    368.2    28.48    332.7    336.4
##      .25      .50      .75      .90      .95
##    347.2    365.1    388.1    404.5    410.5
```

```
##
## lowest : 327.3280 327.3350 328.3200 328.3425 328.4750
## highest: 417.2820 417.6200 418.9533 419.1125 419.1140
## ---------------------------------------------------------------------------
```

```
summary(new_monthly)
```

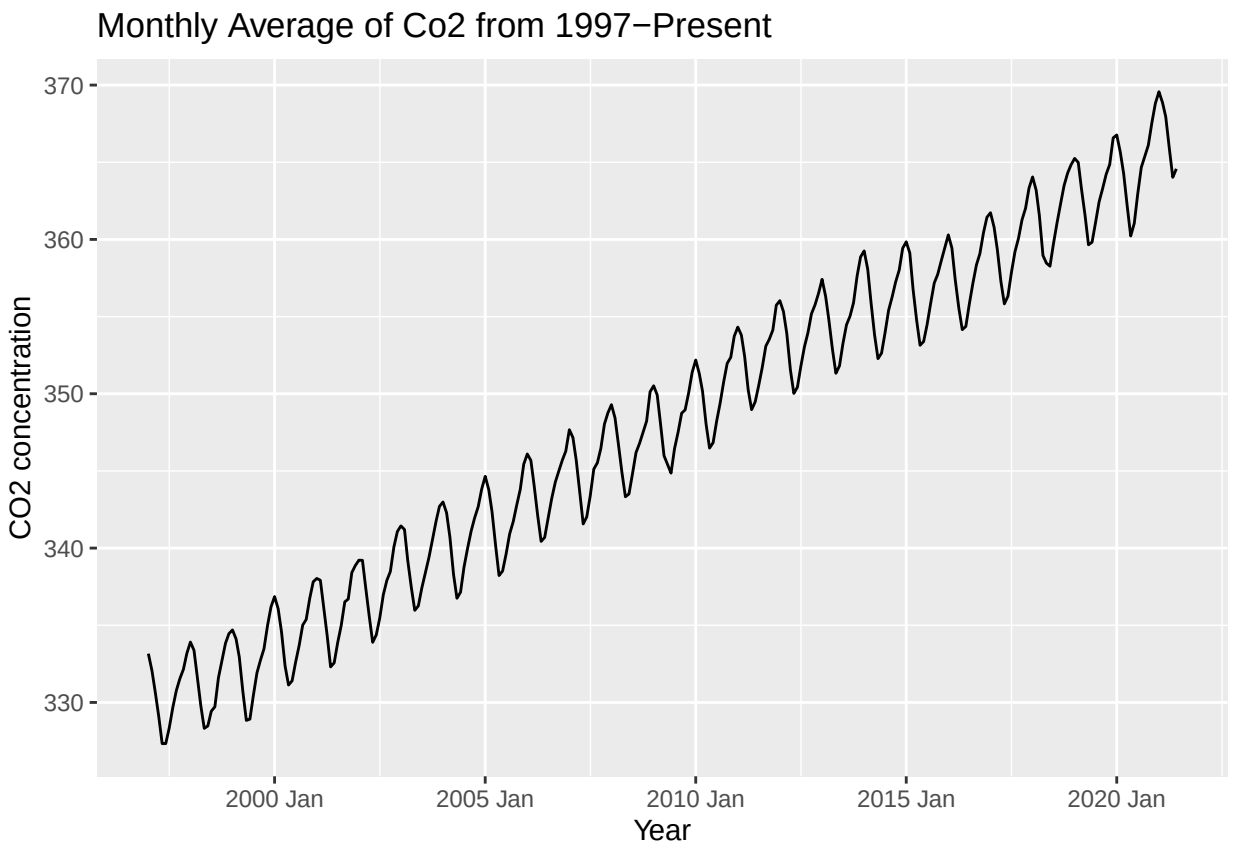
```
##  tapply.co2_weekly..2....format.co2_weekly..1......Y..m....mean.
##  Min.   :327.3
##  1st Qu.:347.2
##  Median :365.1
##  Mean   :368.2
##  3rd Qu.:388.1
##  Max.   :419.1
```

Let's convert our monthly averages data into a time series.

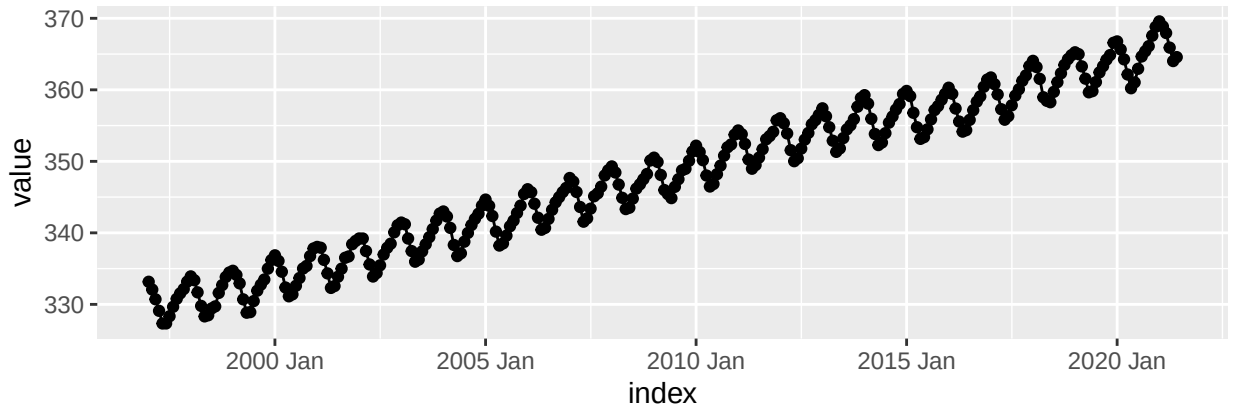
```
ts3 = as_tsibble(ts(new_monthly, frequency = 12, start=c(1997,1), end=c(2021,6)))
```

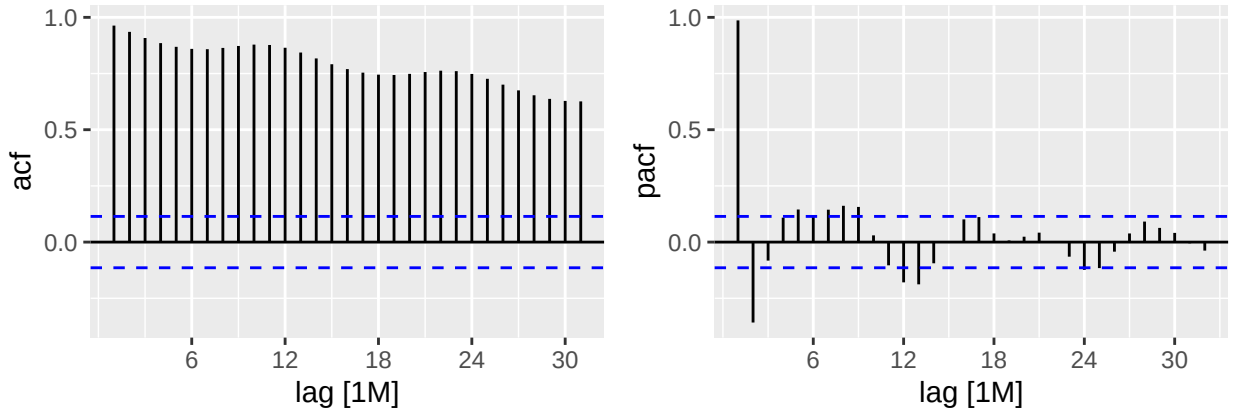
Let's visualize our monthly average time series data.

```
autoplot(ts3) + ggtitle("Monthly Average of Co2 from 1997-Present") +
xlab("Year") + ylab("CO2 concentration")
```

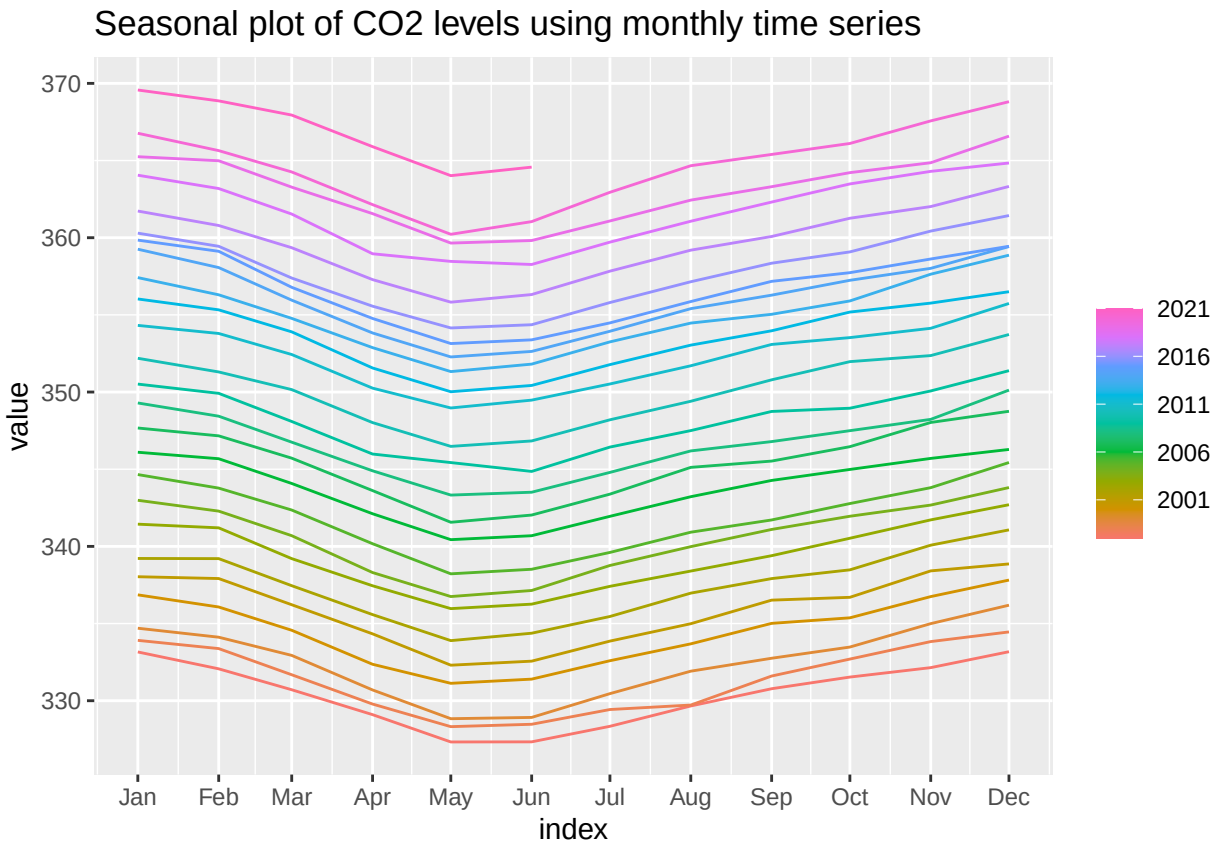



```
ts3 %>% gg_tsdisplay(y=value, plot="partial", lag_max=32)
```






```
ts3 %>% gg_season(y=value) + ggtitle("Seasonal plot of CO2 levels using monthly time series")
```



The monthly average time series graph is consistent with out findings from the weekly time series graphs above and confirms there is an upward and seasonal trend within the Mauna Loa data even after averaging the monthly CO2 concentration.

Generate Accuracy Metrics

Let's create predictions for the models generated in Parts 2 and 3 and generate accuracy metrics based on those predictions.

```
#Accuracy Metrics
num_months = 294

pred.lm <- predict(co2.lm, n.ahead=n.months, ci=0.95)
pred.lm2 <- predict(co2.lm2, n.ahead=num_months, ci=0.95)
pred.log <- predict(co2.loglm, n.ahead=n.months, ci=0.95)
pred.qm <- predict(co2.qm, n.ahead=num_months, ci=0.95)

ts_new <- ts(new_monthly, frequency = 12, start=c(1997,1), end=c(2021,6))
monthly <- window(ts_new, start=1997)

print("Linear Model 1 Accuracy Metrics")
```

```
## [1] "Linear Model 1 Accuracy Metrics"
```

```
accuracy(pred.lm, monthly)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 20.25658 20.42669 20.25658 5.812623 5.812623 0.8881054  16.28905
```

```
print("Logarithmic Model Accuracy Metrics")
```

```
## [1] "Logarithmic Model Accuracy Metrics"
```

```
accuracy(pred.log, monthly)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 342.0403 342.2136 342.0403 98.33383 98.33383 0.9863888   273.953
```

```
print("Polynomial Model Accuracy Metrics")
```

```
## [1] "Polynomial Model Accuracy Metrics"
```

```
accuracy(pred.qm, monthly)
```

```
##                ME     RMSE      MAE     MPE    MAPE     ACF1 Theil's U
## Test set 20.56499 20.87578 20.56499 5.88959 5.88959 0.939099  16.59875
```

```
print("Polynomial Time Trend Model Accuracy Metrics")
```

```
## [1] "Polynomial Time Trend Model Accuracy Metrics"
```

```
accuracy(pred.lm2, monthly)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 20.55338 21.07641 20.55338 5.884491 5.884491 0.9001741  16.76095
```

```
print("Time Series Model Accuracy Metrics")
```

```
## [1] "Time Series Model Accuracy Metrics"
```

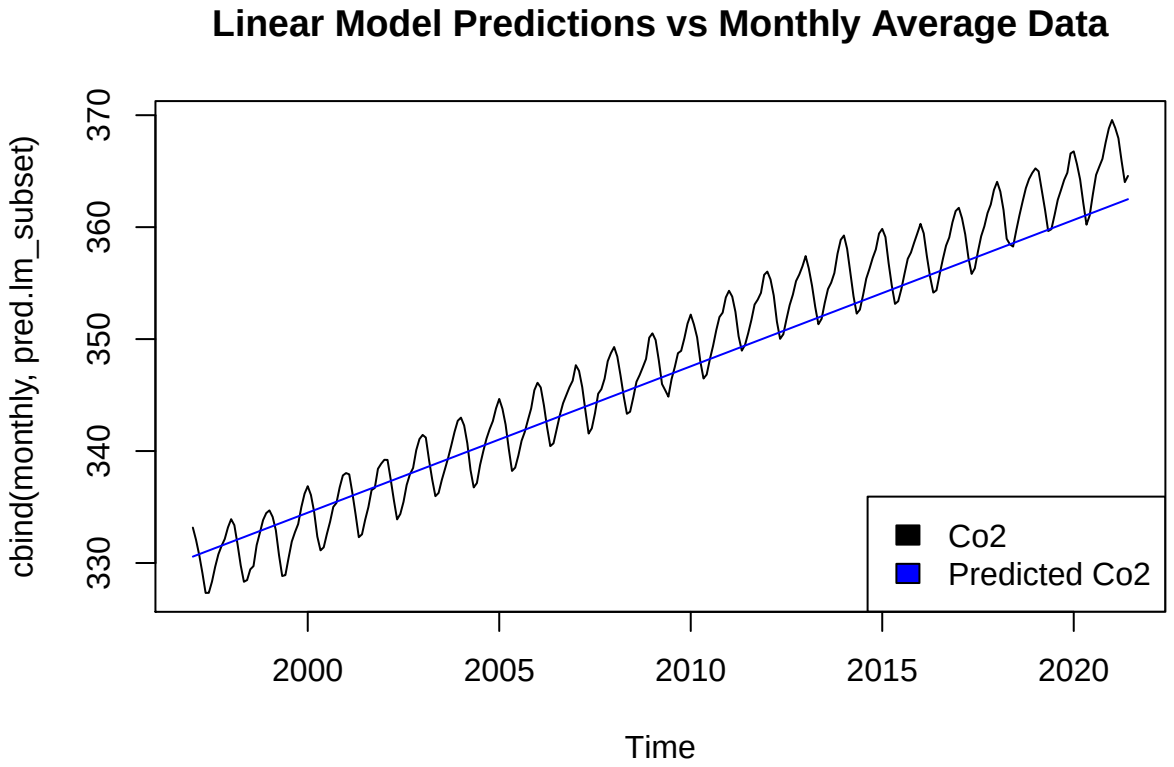
```
accuracy(forecast(co2.fit, h=n.months), monthly)
```

```
##                       ME       RMSE        MAE          MPE      MAPE
## Training set   0.01733093  0.2874745  0.2304755  0.005048606 0.0684787
## Test set     -34.47189122 34.6812761 34.4718912 -9.914606523 9.9146065
##                    MASE         ACF1 Theil's U
## Training set  0.1820237 -0.002339927        NA
## Test set     27.2250206  0.838227283  27.74029
```

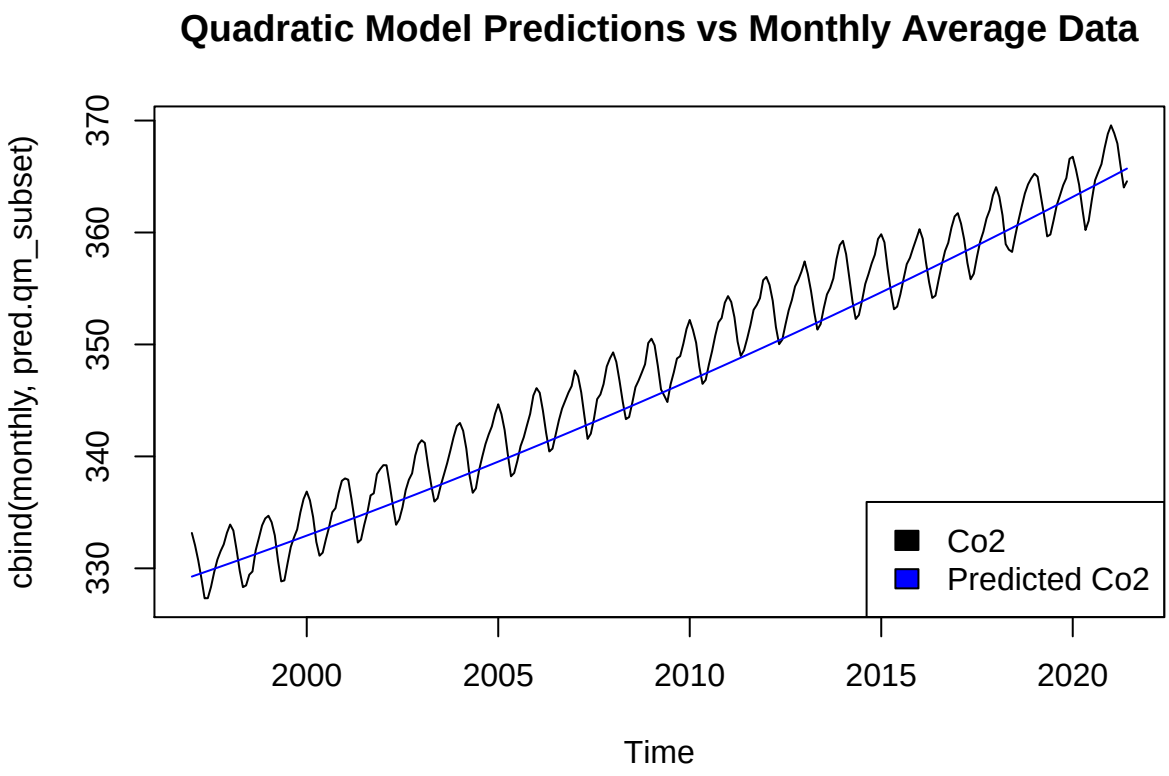
Next, we will visualize the models from Parts 2 and 3 and compare them to the monthly average data.

```
pred.lm_subset = tail(pred.lm,294)
plot(cbind(monthly, pred.lm_subset), type="l",
  plot.type = "single", col = c("black", "blue"))
title(main = "Linear Model Predictions vs Monthly Average Data")
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
```

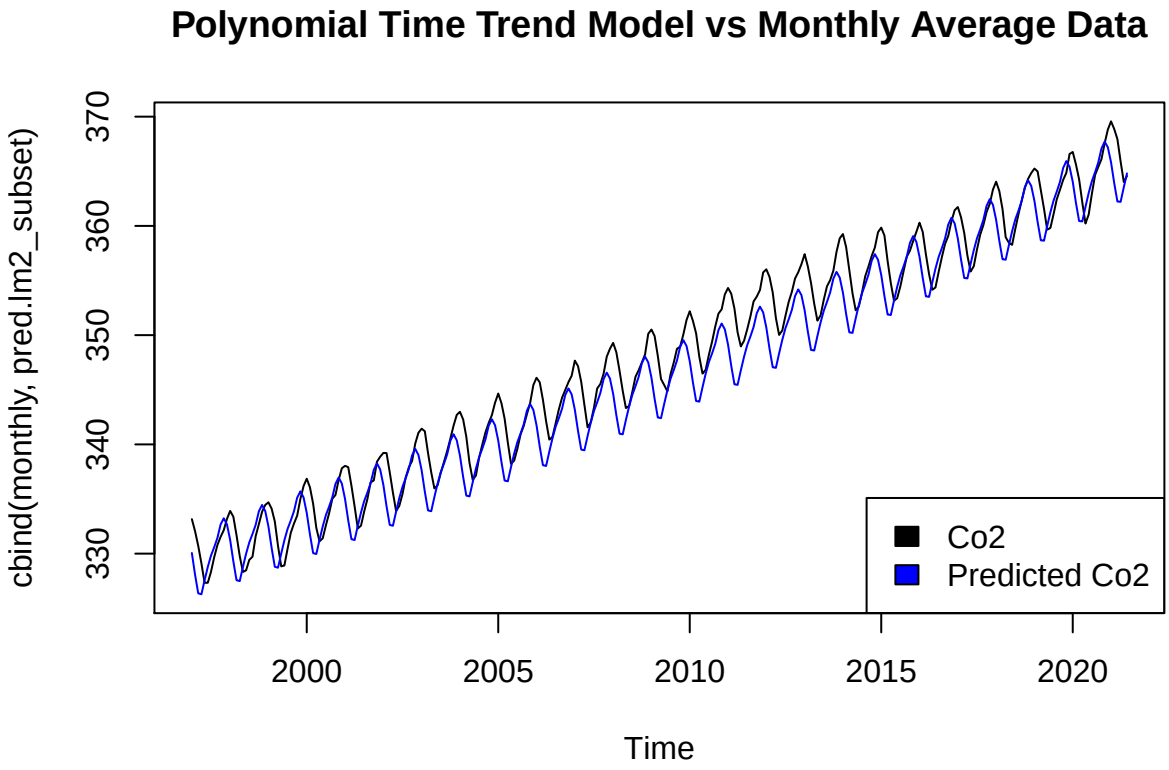
```
       fill = c("black", "blue"),
       border = "black")
```



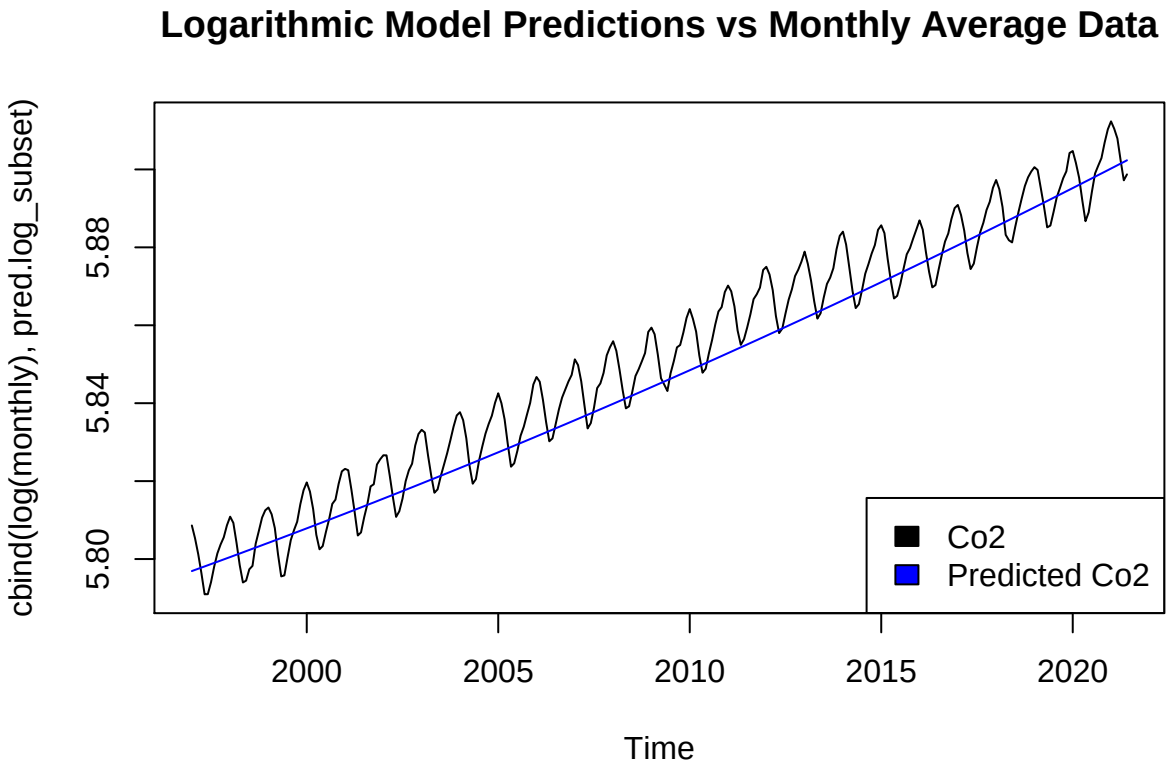

```
pred.qm_subset = tail(pred.qm,294)
plot(cbind(monthly, pred.qm_subset), type="l",
  plot.type = "single", col = c("black", "blue"))
title(main = "Quadratic Model Predictions vs Monthly Average Data")
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```



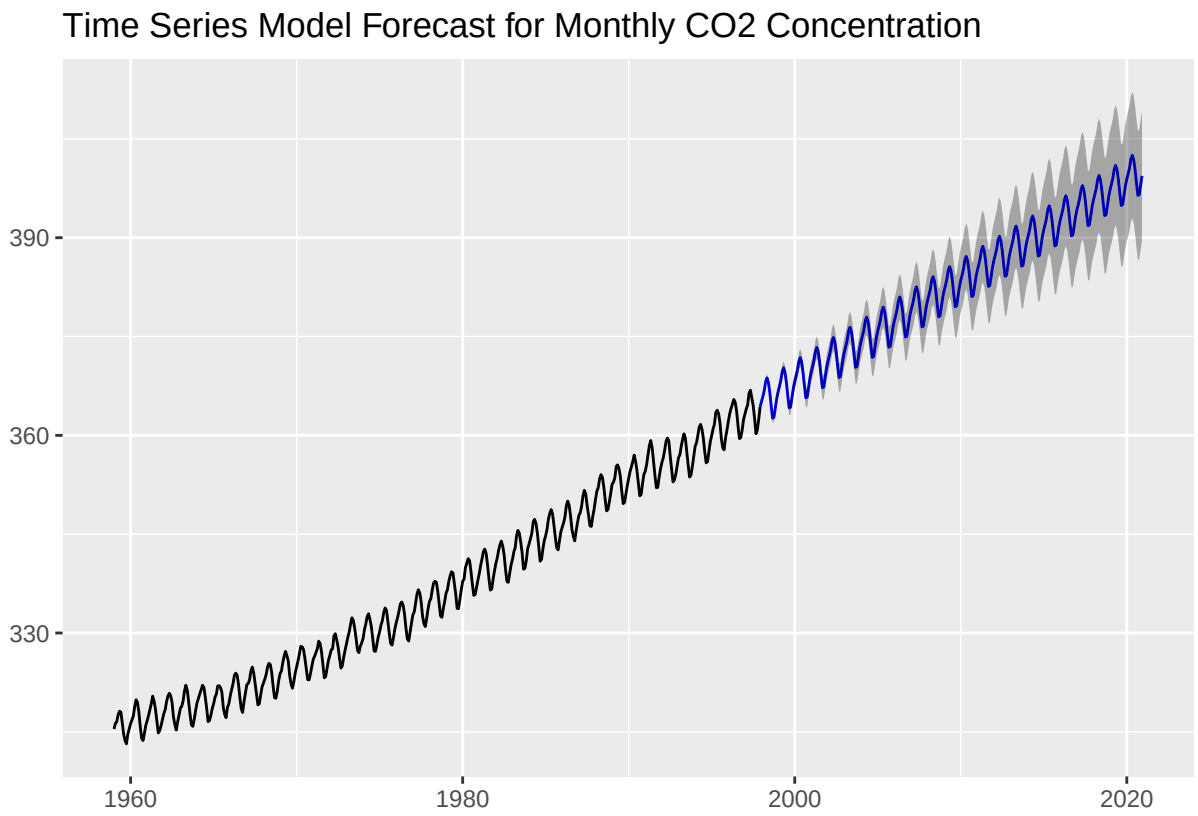

```
pred.lm2_subset = tail(pred.lm2,294)
plot(cbind(monthly, pred.lm2_subset), type="l",
  plot.type = "single", col = c("black", "blue"))
title(main = "Polynomial Time Trend Model vs Monthly Average Data")
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```




```
pred.log_subset = tail(pred.log,294)
plot(cbind(log(monthly), pred.log_subset), type="l",
  plot.type = "single", col = c("black", "blue"))
title(main = "Logarithmic Model Predictions vs Monthly Average Data")
legend("bottomright",
       legend = c("Co2", "Predicted Co2"),
       fill = c("black", "blue"),
       border = "black")
```




```
models_forecast %>% autoplot(monthly, start=c(1997,1)) +
ggtitle("Time Series Model Forecast for Monthly CO2 Concentration")
```



The accuracy metrics comparing our models from Parts 2 and 3 to the monthly averages time

series data show results that are consistent with our previous findings. The linear model, quadratic model, and polynomial time trend model all have very similar accuracy metrics when compared to the monthly averages data. The accuracy metrics show that the linear model has a slightly smaller Root Mean Square Error (RMSE) compared to the other models with an RMSE of 20.45. However, looking at the graph of the linear model predictions compared to the monthly averages data we can see that it is not the most accurate model. In Part 2, we determined that based on the sum of squared residuals that our quadratic model was the best model at forecasting the monthly CO2 concentration based on the data from 1959-1998 and crafted our polynomial time trend model based on that model. The accuracy metrics also confirmed the results from Part 2 that the logarithmic model was the least effective model prediction. The logarithmic model had the highest RMSE of all of the models with an RMSE of 342.45. The visualizations of our models compared to the monthly average data we generated confirm our results from Parts 2 and 3 and show that our polynomial time trend model is the most accurate fit for the monthly averages data.

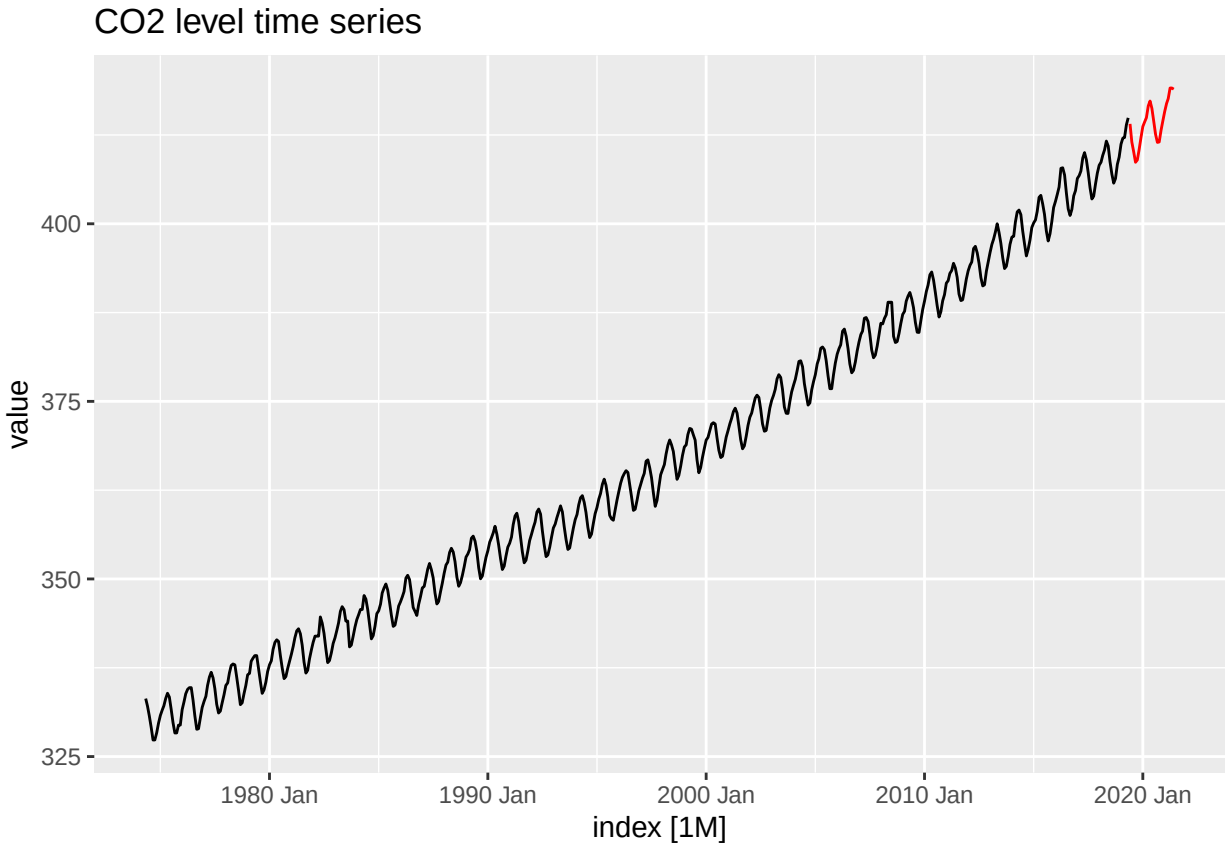
Part 5 (5 points)

Split the NOAA series into training and test sets, using the final two years of observations as the test set. Fit an ARIMA model to the series following all appropriate steps, including comparison of how candidate models perform both in-sample and (psuedo-) out-of-sample. Generate predictions for when atmospheric CO2 is expected to reach 450 parts per million, considering the prediction intervals as well as the point estimate. Generate a prediction for atmospheric CO2 levels in the year 2100. How confident are you that these will be accurate predictions?

Let's split the NOAA series into training and test sets, taking the final two years as test set period for measuring pseudo out-of-sample forecasting performance.

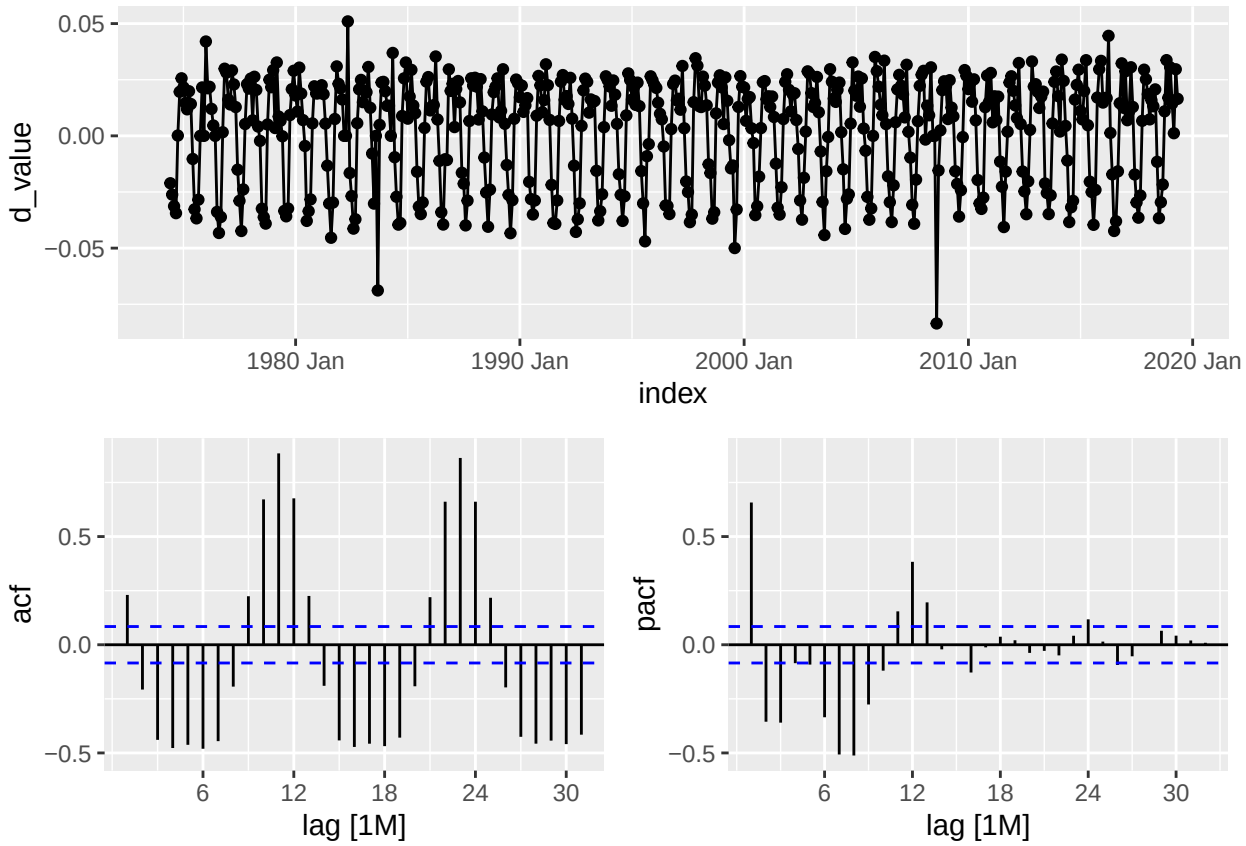
```
noaa.training <- nova.monthly.ts.boxcox %>% filter_index(~"2019 May")
noaa.test <- nova.monthly.ts.boxcox %>% filter_index("2019 June"~.)

noaa.training %>% autoplot(value) +
autolayer(noaa.test, colour="red") +
ggtitle("CO2 level time series")
```

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



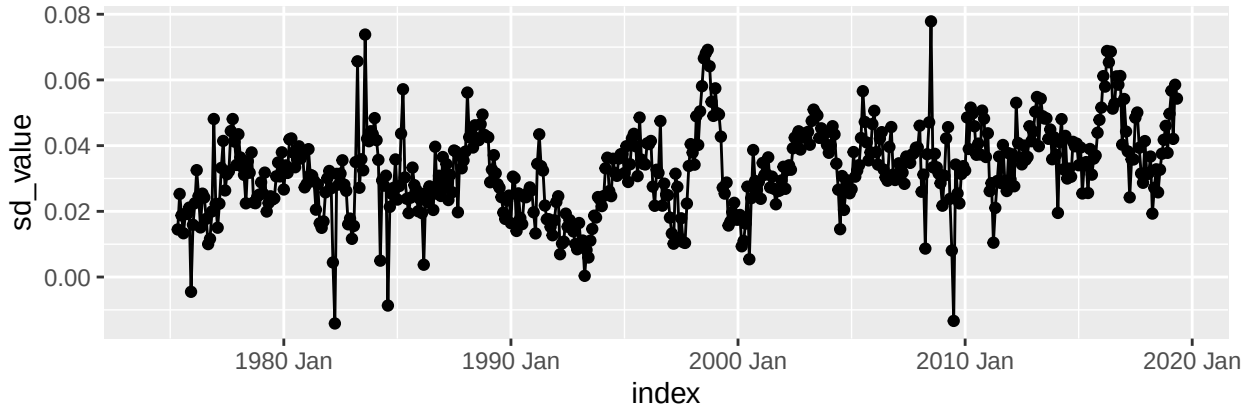
Black and red lines represent training and test sets, respectively. Let's apply Box-Cox transformation to stabilize the variance. Let's also apply first-differencing to the box-cox transformed series.

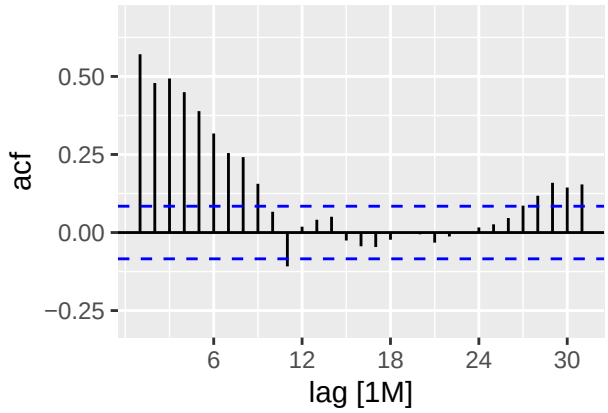
```
#choose lambda using guerrero's method
lambda <- nova.monthly.ts %>% features(value, features=guerrero) %>%
  pull(lambda_guerrero)
#print(lambda)
#apply Box-Cox transformation with a chosen lambda
nova.monthly.ts.boxcox <- nova.monthly.ts %>%
  mutate(value_bc = box_cox(value, lambda=lambda))

noaa.training <- noaa.training %>% mutate(d_value = difference(value_bc,1))
noaa.training %>% gg_tsdisplay(y=d_value, plot="partial", lag_max=32)
```

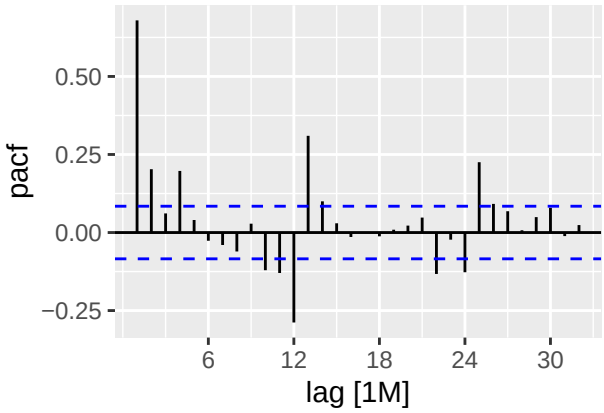


After the box-cox transformation and first-differencing, time series do not seem to have the upward trend but it still has a seasonal pattern. Let's apply the seasonal differencing to eliminate seasonality.

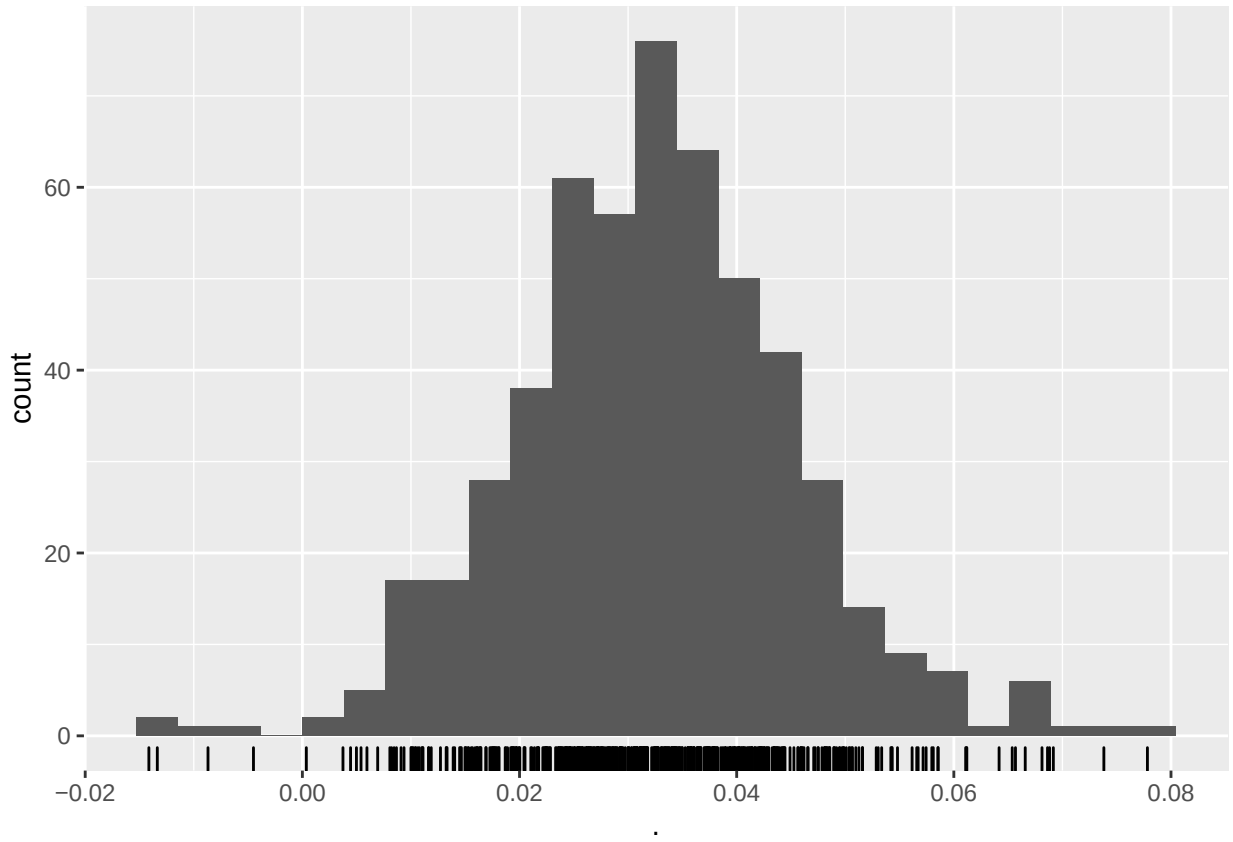
```
noaa.training <- noaa.training %>% mutate(sd_value = difference(value_bc,12))
noaa.training %>% gg_tsdisplay(y=sd_value, plot="partial", lag_max=32)
```








```
noaa.training$sd_value %>% gghistogram()
```



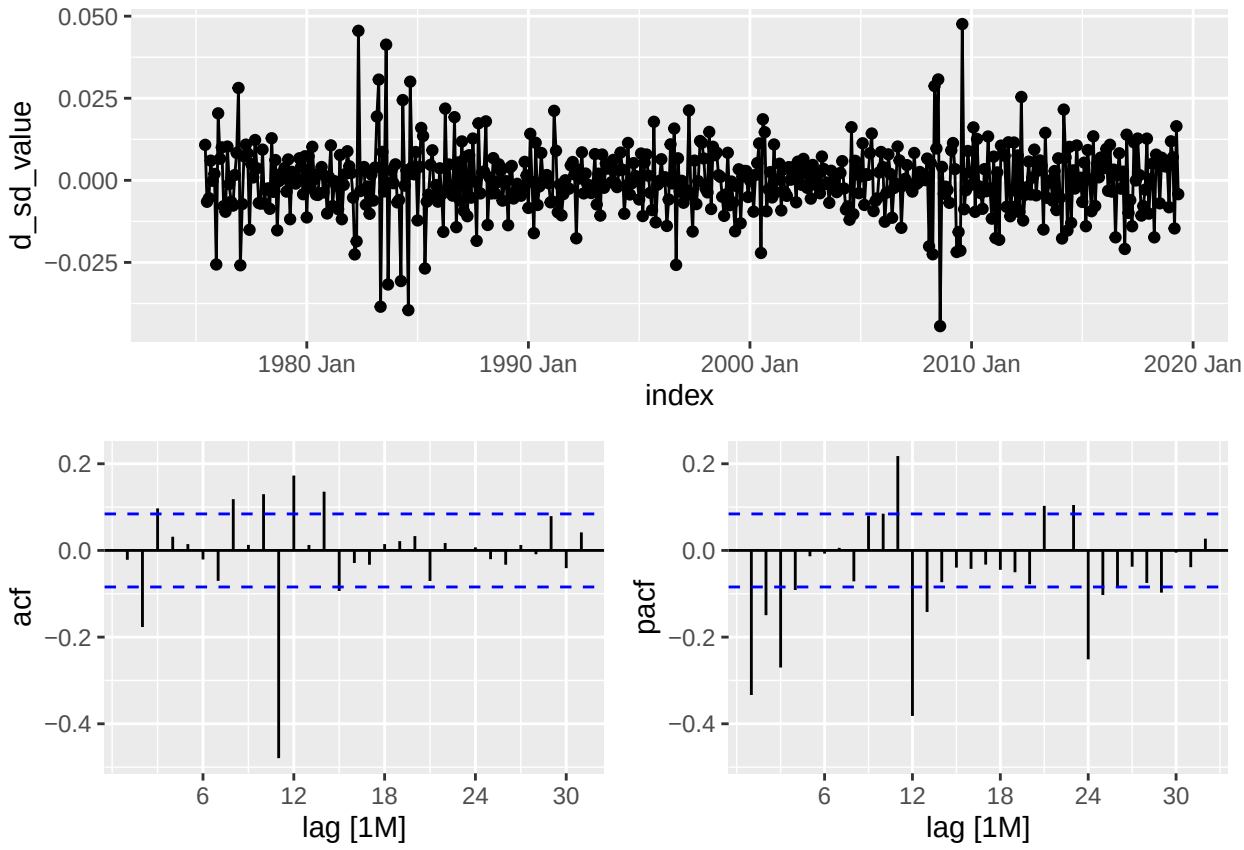
We applied differencing at a frequency of 12 since we are using a monthly data. Seasonality seems to be somewhat subdued. Let's apply KPSS unit root test to check the stationairty.

```
noaa.training %>% features(sd_value, unitroot_kpss)
```

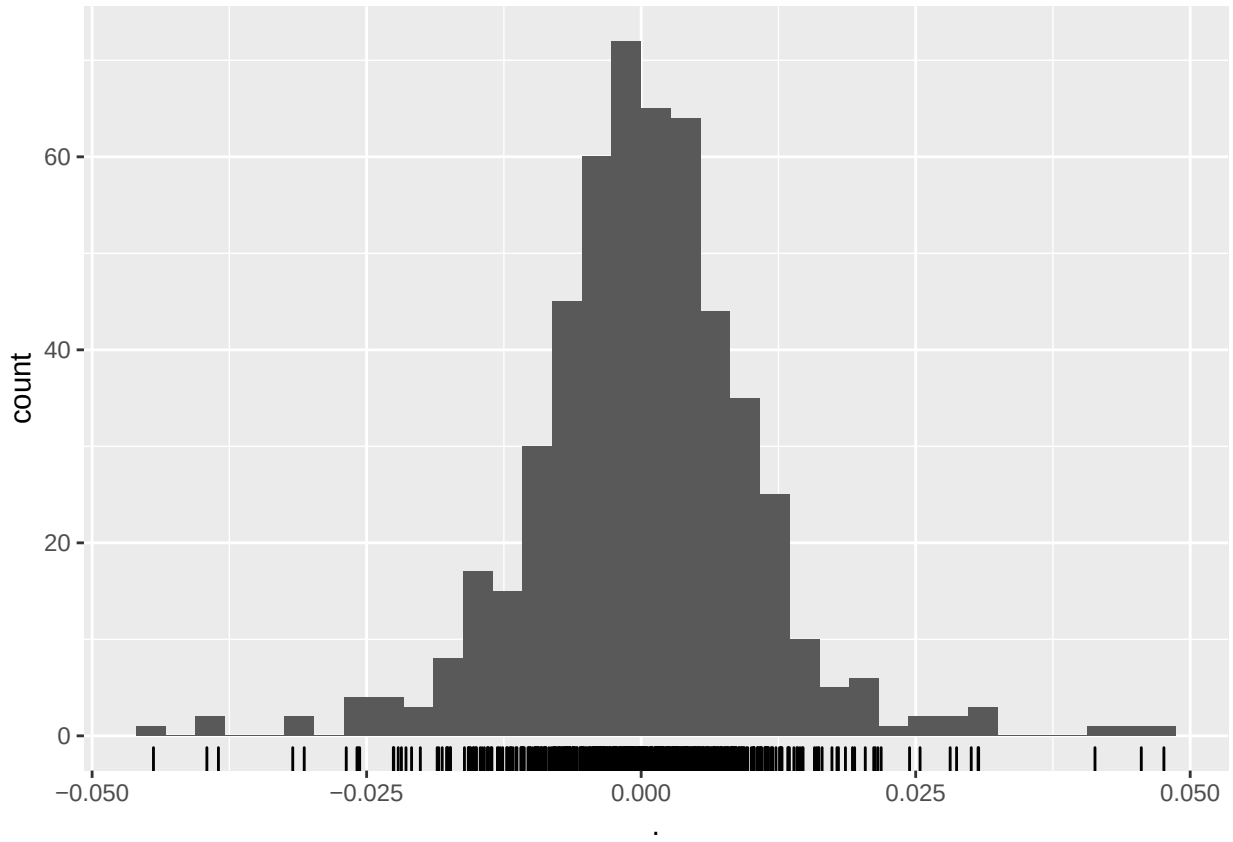
```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1      1.66        0.01
```

Under the KPSS unit root test, null hypotheses of stationarity would be rejected at 1% of significance. Let's apply both seasonal and nonseasonal differencing to improve our model's stability.

```
noaa.training <- noaa.training %>% mutate(d_sd_value = difference(sd_value))
noaa.training %>% gg_tsdisplay(y=d_sd_value, plot="partial", lag_max=32)
```




```
noaa.training$d_sd_value %>% gghistogram()
```




```
noaa.training %>% features(d_sd_value, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0117         0.1
```

Applying both seasonal and nonseasonal differencing seem to give more stationary series. P-value for KPSS test is 0.1. At alpha=0.05, we would not reject the null hypothesis of stationarity.

The correlogram of the first and seasonal-differened series shows significant spikes in ACF and PACF at lag 12, potentially suggesting a seasonal MA(1) and/or AR component. We can perform exhaustive search to find the best performing model. The following loop will perform a loop of $ARIMA(p,1,q)(P,1,Q)_{12}$ models up to a maximum value of 5 for p,q,P and Q. *AICc* and *RMSE* scores for training set and *RMSE* scores for test test are recorded for each model specification. A model is then selected based on the RMSE value on the held-out test set.

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

```
max_order = 5
for(p in 0:max_order) {
  for(q in 0:max_order) {
    for (P in 0:max_order) {
      for (Q in 0:max_order) {
        tryCatch({

          #training
          mod_training <- noaa.training %>% model(ARIMA(box_cox(value, lambda=lambda)~  0 +
pdq(p,1,q) + PDQ(P,1,Q)))

          #if the output contains the required metric
          current_trian_AICc <- NA
          current_test_RMSE <- NA
          current_training_RMSE <- NA

          if(has_name(glance(mod_training),"AICc")) {
            current_trian_AICc <- as.numeric(glance(mod_training)$AICc)
          }

          current_training_RMSE <- as.numeric(accuracy(mod_training)$RMSE)
          models_forecast <- mod_training %>% forecast(h=24)
          current_test_RMSE <- as.numeric(accuracy(models_forecast,noaa.test)$RMSE)


          #print the row
          #add to the matrix
          results <-results %>% add_row(p=p,
                                        q=q,
                                        P=P,
                                        Q=Q,
                                        train_AICc = current_trian_AICc,
                                        train_RMSE = current_training_RMSE,
                                        test_RMSE = current_test_RMSE)
          #print it out
          print(paste(p,q,P,Q, current_trian_AICc, current_training_RMSE,
                      current_test_RMSE))

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

```
}
```

It took 6 hours to run the above for loop. Environment variable, *results*, is written to the disk and *eval=FALSE* command is added to the R studio code chunk to avoid running this cell in the future run.

```
save.image(file="checkpoint.RData")
```

```
load(file="checkpoint.RData")
```

Let's find out a model which has the lowest RMSE on the held-out test set.

```
#which configuration result in the lowest AICc
#compare not just AICc but also RMSE on held-out test set.
results[which.min(results$test_RMSE),]
```

```
##      p q P Q train_AICc train_RMSE test_RMSE
## 1104 5 0 3 5  -3704.587   0.395265 0.2509743
```

As shown above, $ARIMA(5,1,0)(3,1,5)_{12}$ has the lowest RMSE on the held out test set. Let's find out if this model also has the lowest RMSE on the training set.

```
#which configuration result in the lowest AICc
#compare not just AICc but also RMSE on held-out test set.
results[which.min(results$train_RMSE),]
```

```
##     p q P Q train_AICc train_RMSE test_RMSE
## 856 3 5 4 3   -3703.74  0.3897357 0.3091411
```

$ARIMA(3,1,5)(4,1,3)_{12}$ has the lowest RMSE on the train set. However, this model has slightly higher test set RMSE than $ARIMA(5,1,0)(3,1,5)_{12}$.

Finally, let's find out the model which has the lowest AICc.

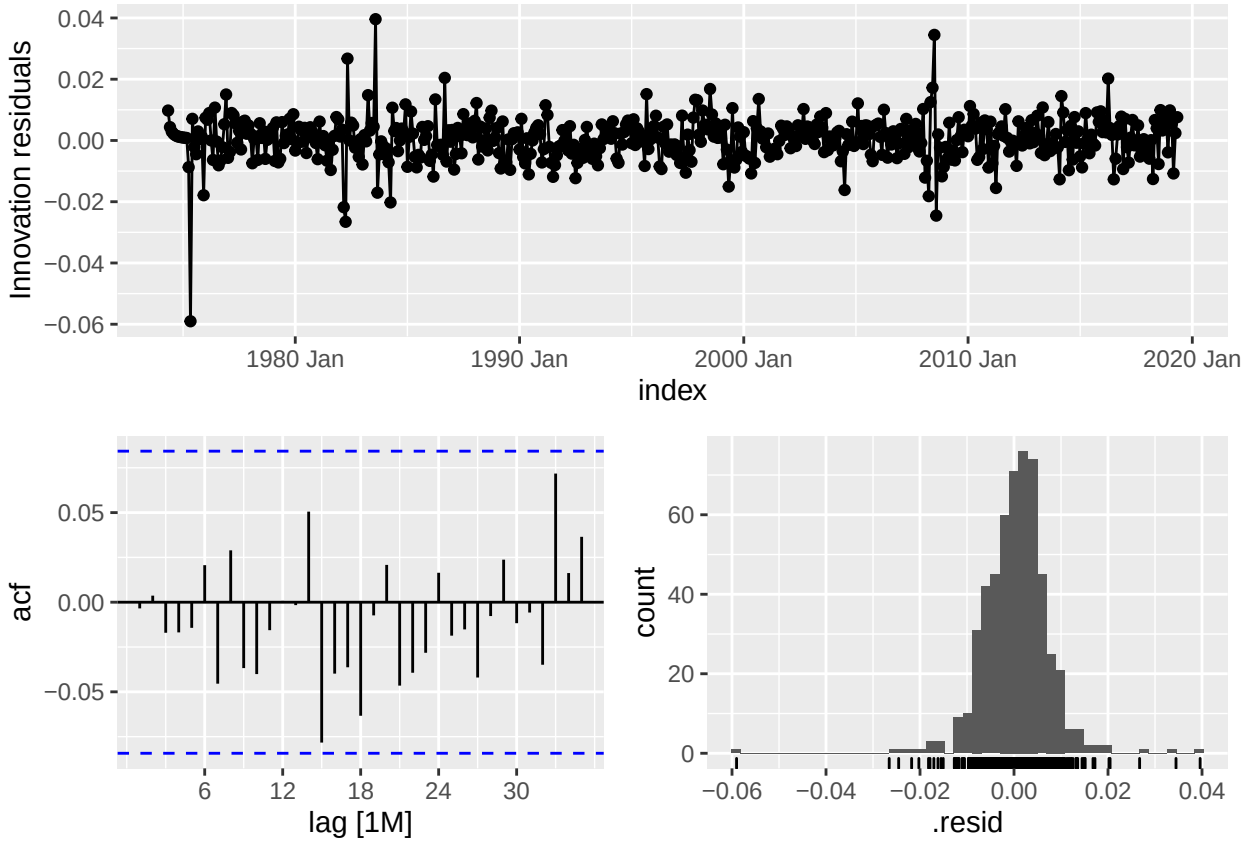
```
#which configuration result in the lowest AICc
#compare not just AICc but also RMSE on held-out test set.
results[which.min(results$train_AICc),]
```

```
##     p q P Q train_AICc train_RMSE test_RMSE
## 254 1 1 0 1  -3712.051  0.4073687 0.3221125
```

$ARIMA(1,1,1)(0,1,1)_{12}$ has the lowest AICc score on the train set. However, this model has much higher test set RMSE value than the other two models.

Before we perform any forecasting, let's check if our model $ARIMA(5,1,3)(3,1,5)_{12}$ is a good fit by examining its residuals and other model properties for any sign of auto-correlation.

```
model %>% gg_tsresiduals(lag_max=36)
```




```
augment(model) %>% features(.resid, ljung_box, lag=24)
```

```
## # A tibble: 1 x 3
##   .model                                                    lb_stat lb_pvalue
##   <chr>                                                       <dbl>     <dbl>
## 1 "ARIMA(box_cox(value, lambda = lambda) ~ 0 + pdq(p, 1, q) +~    14.3     0.940
```

Residuals are symmetrically distributed and it resembles the white noise. Ljung-Box Test with lag-24 also shows that this model is a good fit. Lag of 24 is chosen since it is recommended to go back two seasonal periods. We can see that p-value is significantly larger than 0.05. This result indicates that there is strong evidence for discrete white noise being a good fit to the residuals. Therefore, $ARIMA(5,1,3)(3,1,5)_{12}$ is a good fit as expeted.

Let's generate predictions for when atmospheric CO2 is expected to reach 450 parts per million using a point estimate.

```
current_h = 150
models_forecast <- model %>% forecast(h=current_h)
while(tail(models_forecast,1)$.mean<450) {
  models_forecast <- model %>% forecast(h=current_h)
  current_h <- current_h+1
}
```

```
any(models_forecast$.mean>450)
```

```
## [1] TRUE
```

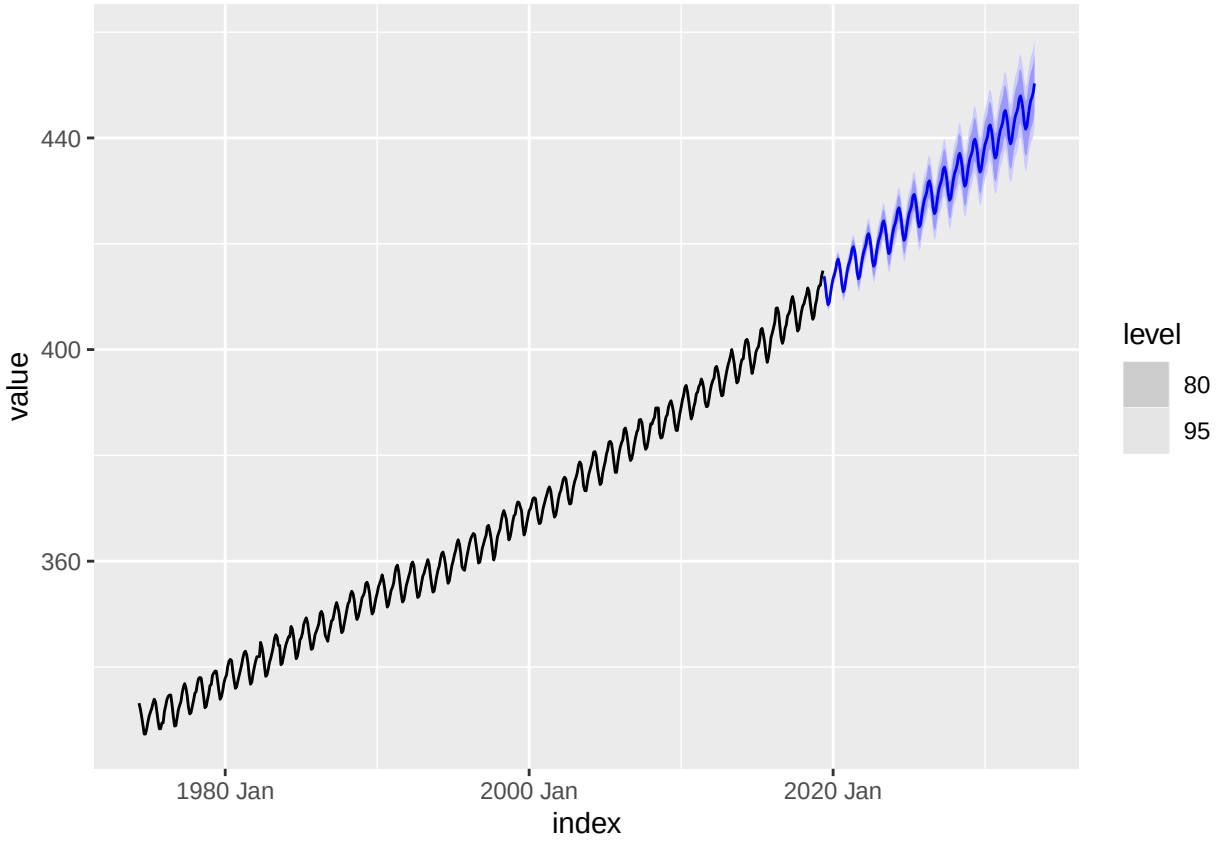
```
max(models_forecast$.mean)
```

```
## [1] 450.2947
```

```
tail(models_forecast,1)
```

```
## # A fable: 1 x 4 [1M]
## # Key:     .model [1]
##   .model                                                index            value .mean
##   <chr>                                                 <mth>           <dist> <dbl>
## 1 "ARIMA(box_cox(value, lambda = lambda) ~ 0 + ~ 2033 Apr t(N(19, 0.0044))  450.
```

```
models_forecast %>% autoplot(noaa.training)
```



Using a point estimate, atmospheric CO2 is expected to reach 450 parts per million in April 2033.

Let's generate predictions for when atmospheric CO2 is expected to reach 450 parts per million using 95% prediction interval

```
prediction_interval <- 0.95
#function for finding the prediction interval
findEstimate <- function(interval) {
    current_h = 10
  models_forecast <- model %>% forecast(h=current_h)
```

```
  while((quantile(tail(models_forecast,1)$value[[1]]$dist,interval)
        + tail(models_forecast,1)$.mean)<450 ) {
    models_forecast <- model %>% forecast(h=current_h)
    current_h <- current_h+1
    #print(paste(current_h,(quantile(tail(models_forecast,1)$value[[1]]$dist,interval)
    #    + tail(models_forecast,1)$.mean),interval))
  }

  return(tail(models_forecast,1))

}
result <- findEstimate(prediction_interval)
result$index
```

```
## <yearmonth[1]>
## [1] "2026 Apr"
```

```
c((quantile(result$value[[1]]$dist,prediction_interval) + result$.mean),
  (result$.mean - quantile(result$value[[1]]$dist,prediction_interval)
        ))
```

```
## [1] 450.1829 412.7390
```

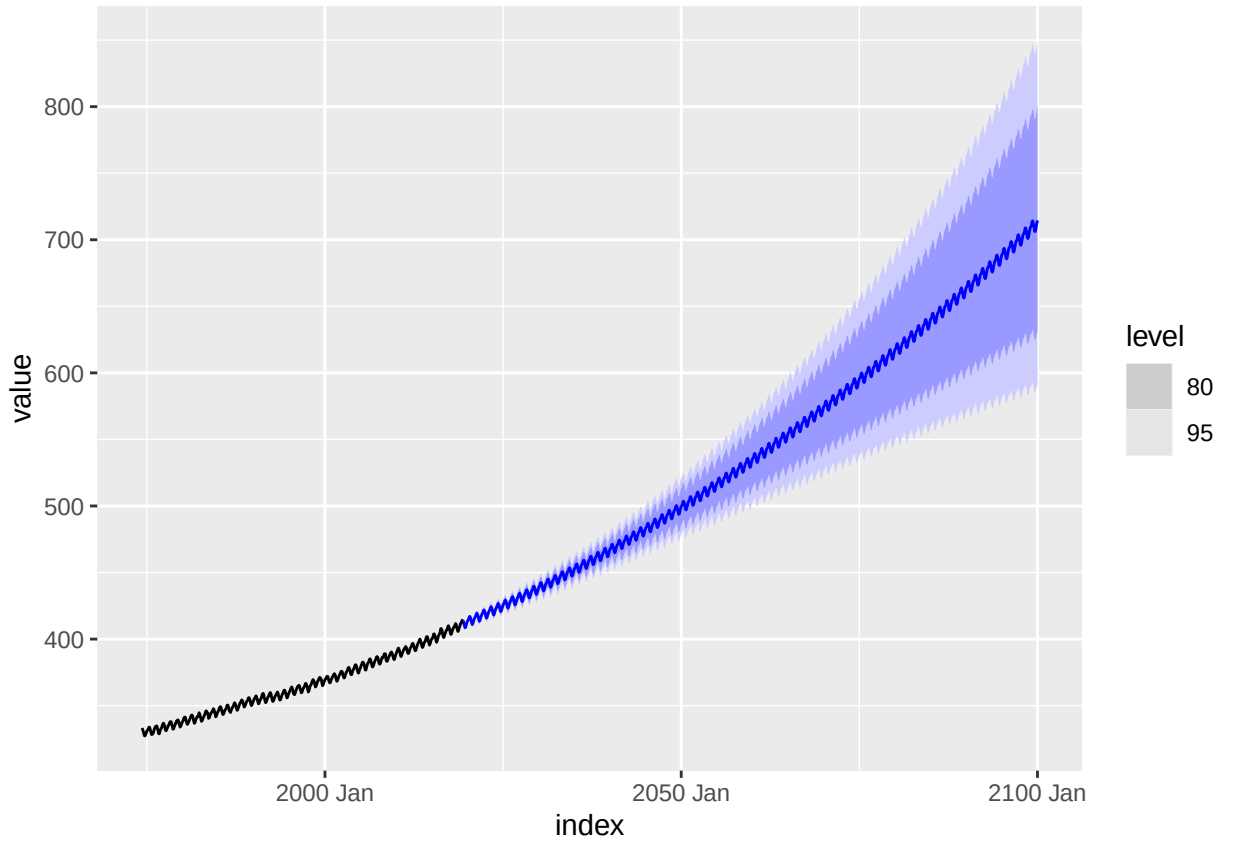
Using a 95% prediction interval, atmospheric CO2 is expected to reach 450 parts per million in April 2026. Its 95% prediction interval is expected to be $412 < CO2 < 450$ in April 2026.

Let's generate a prediction for atmospheric CO2 levels in the year 2100.

```
models_forecast2100 <- model %>% forecast(h=968)
#tail(models_forecast2100,1)$index
result <-tail(models_forecast2100,1)
prediction_interval <- 0.95
rev(c(result$.mean,(quantile(result$value[[1]]$dist,prediction_interval) + result$.mean),
  (result$.mean - quantile(result$value[[1]]$dist,prediction_interval)
        )))
```

```
## [1] 690.8367 738.2186 714.5277
```

```
models_forecast2100 %>% autoplot(noaa.training)
```




```
prediction_interval <- 0.8
rev(c(result$.mean,(quantile(result$value[[1]]$dist,prediction_interval) + result$.mean),
  (result$.mean - quantile(result$value[[1]]$dist,prediction_interval)
        )))
```

```
## [1] 691.4430 737.6124 714.5277
```

The mean point estimate of atmospheric CO2 level in January 2100 is 714.5. Using a 95% prediction interval, atmospheric CO2 level is expected to be $690.8 < CO2 < 738.2$ in January 2100. Using 80% prediction interval, atmospheric CO2 level is expected to be $691.4 < CO2 < 737.6$ in January 2100. These prediction will be accurate if we assume that the historical data generating process continues to behave in a similar fashion in the year 2100. This is very unlikely since our model is trained with data from 1974 to 2019 and CO2 emission pattern would be very different in the year 2100.