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

Due Monday July 12 2021 11:59pm

Song Park, Mikayla Pugel, Alan Zhang

Instructions (Please Read Carefully):

- Submit by the due date. **Late submissions will not be accepted**
- No page limit, but be reasonable
- Do not modify fontsize, margin or line-spacing settings
- One student from each group should submit the lab to their student github repo by the deadline
- Submit two files:

  1. A pdf file that details your answers. Include all R code used to produce the answers
  2. The R markdown (Rmd) file used to produce the pdf file

  The assignment will not be graded unless **both** files are submitted
- Name your files to include all group members names. For example, if the students' names are Stan Cartman and Kenny Kyle, name your files as follows:

  - `StanCartman_KennyKyle_Lab2.Rmd`
  - `StanCartman_KennyKyle_Lab2.pdf`
- Although it sounds obvious, please write your name on page 1 of your pdf and Rmd files
- All answers should include a detailed narrative; make sure that your audience can easily follow the logic of your analysis. All steps used in modelling must be clearly shown and explained; do not simply 'output dump' the results of code without explanation
- If you use libraries and functions for statistical modeling that we have not covered in this course, you must provide an explanation of why such libraries and functions are used and reference the library documentation
- For mathematical formulae, type them in your R markdown file. Do not e.g. write them on a piece of paper, snap a photo, and use the image file
- Incorrectly following submission instructions results in deduction of grades
- Students are expected to act with regard to UC Berkeley Academic Integrity.

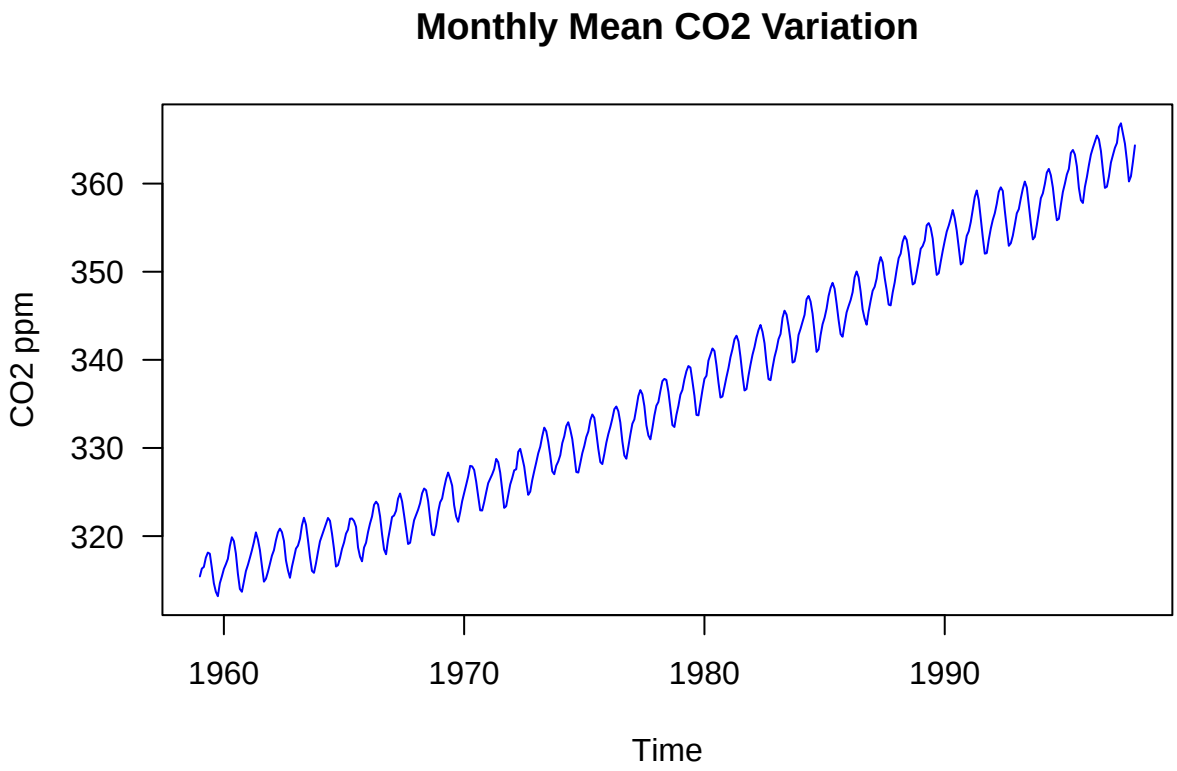
The Keeling Curve

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")
```



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.

Response: We started off the EDA with looking at the general statistics of the data. The data contains a value every month starting in 1959, it contains some missing values, and the frequency of the data is 12, which makes sense since there are 12 months in a year. The mean of the data set is 337.1 and the range is from 313.2 to 366.8. The histogram of the data shows that the data is generally equal throughout the range, the data is not skewed in any direction. From the boxplot graph, we can see that mean of the data seems to decrease in the second half of the year, showing there might be a bit of a seasonal trend in the data.

The decomposition plot shows a strong trend in the data as well as a strong seasonality trend, which shows to us that we might think about using a seasonal model later on and that the data might need to be difference in order to be made stationary. Lastly, from the ACF, the data does not quickly go to zero and therefore this may mean that the data is not stationary and we may need to difference the data, and the PACF graph shows two spikes in the beginning and another two spikes 12 months in, which makes us think that a model will need not only a moving average term of 1 or 2 but also a seasonal moving average term of 1 or 2.

Overall, the data looks very clean with very strong trend and seasonality impacts which will need to be correctly modeled to obtain good time series predictions.

```
head(co2)
```

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

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

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

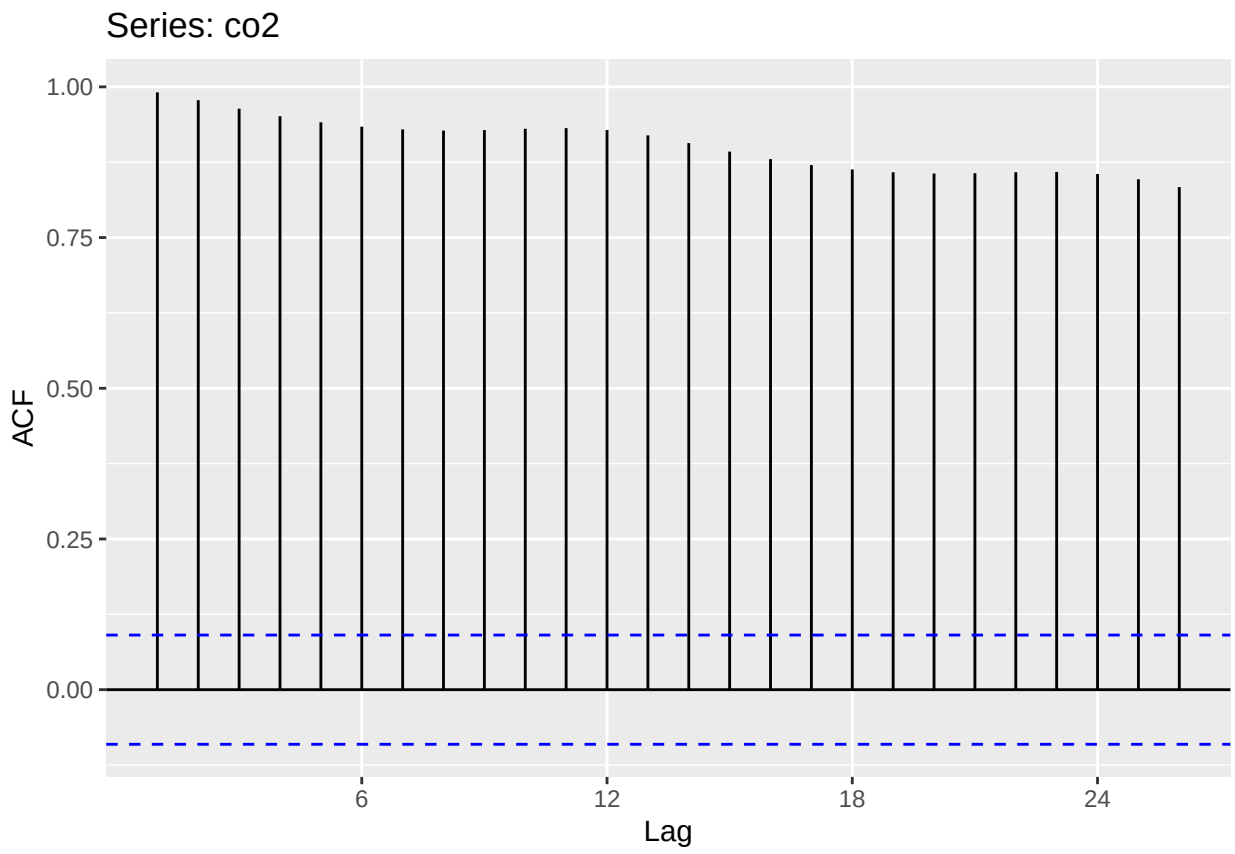
```
frequency(co2)
```

```
## [1] 12
```

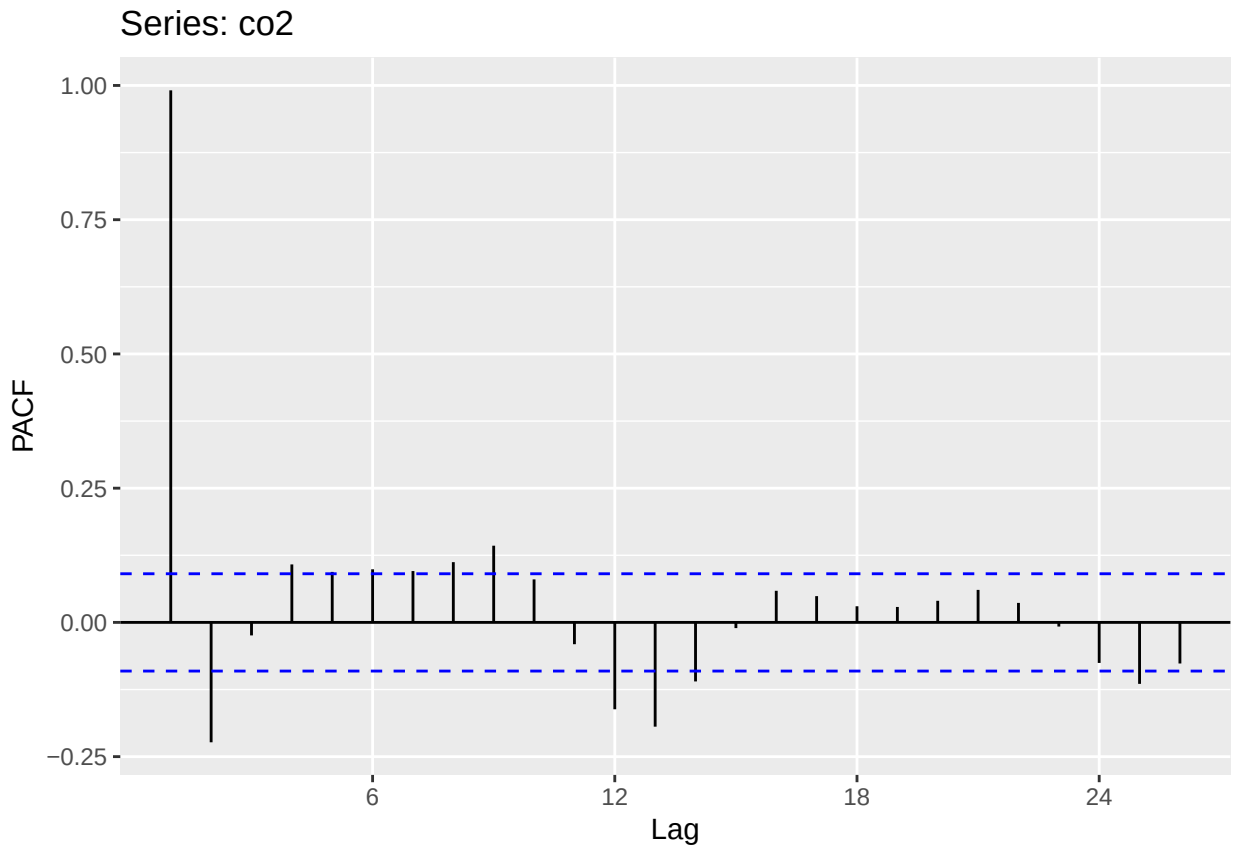
```
summary(co2)
```

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

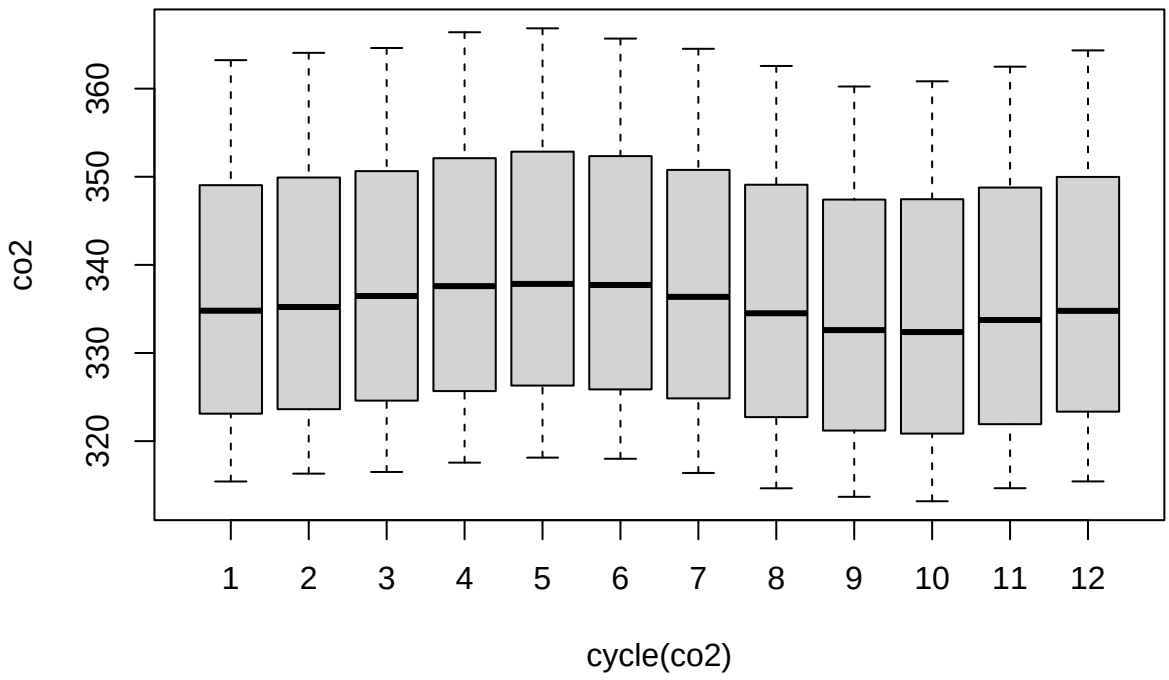
```
ggAcf(co2)
```



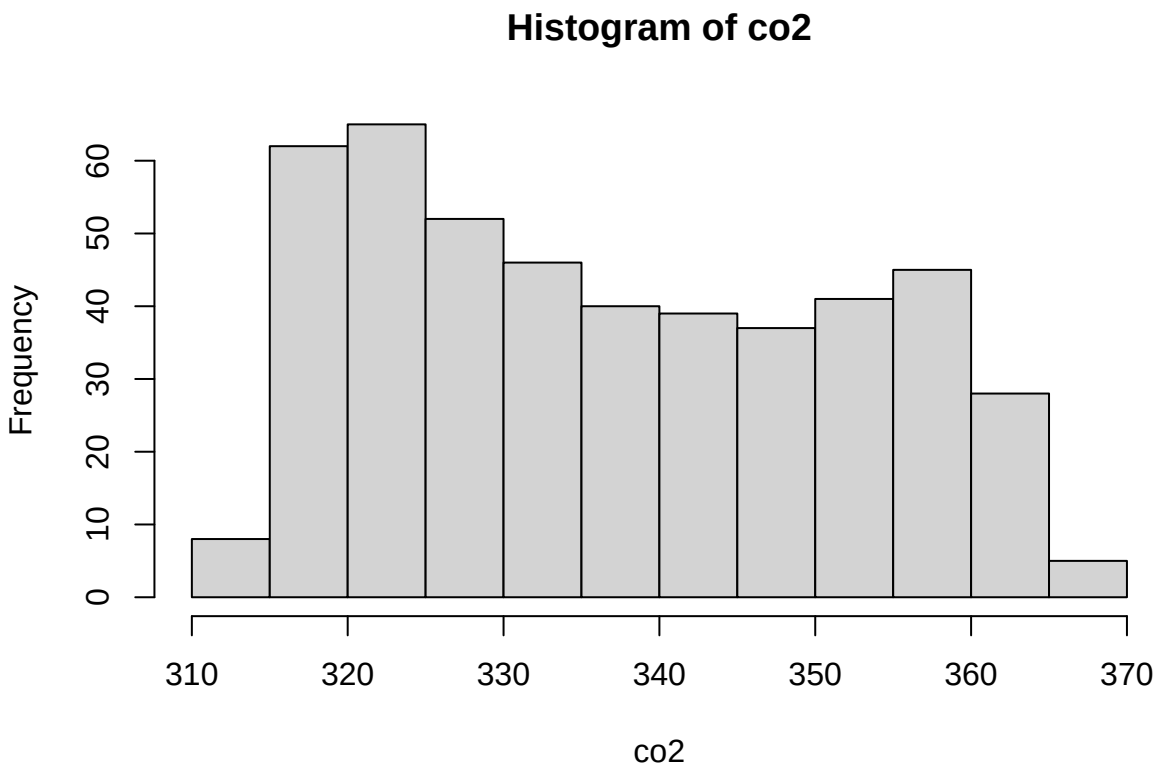

```
ggPacf(co2)
```



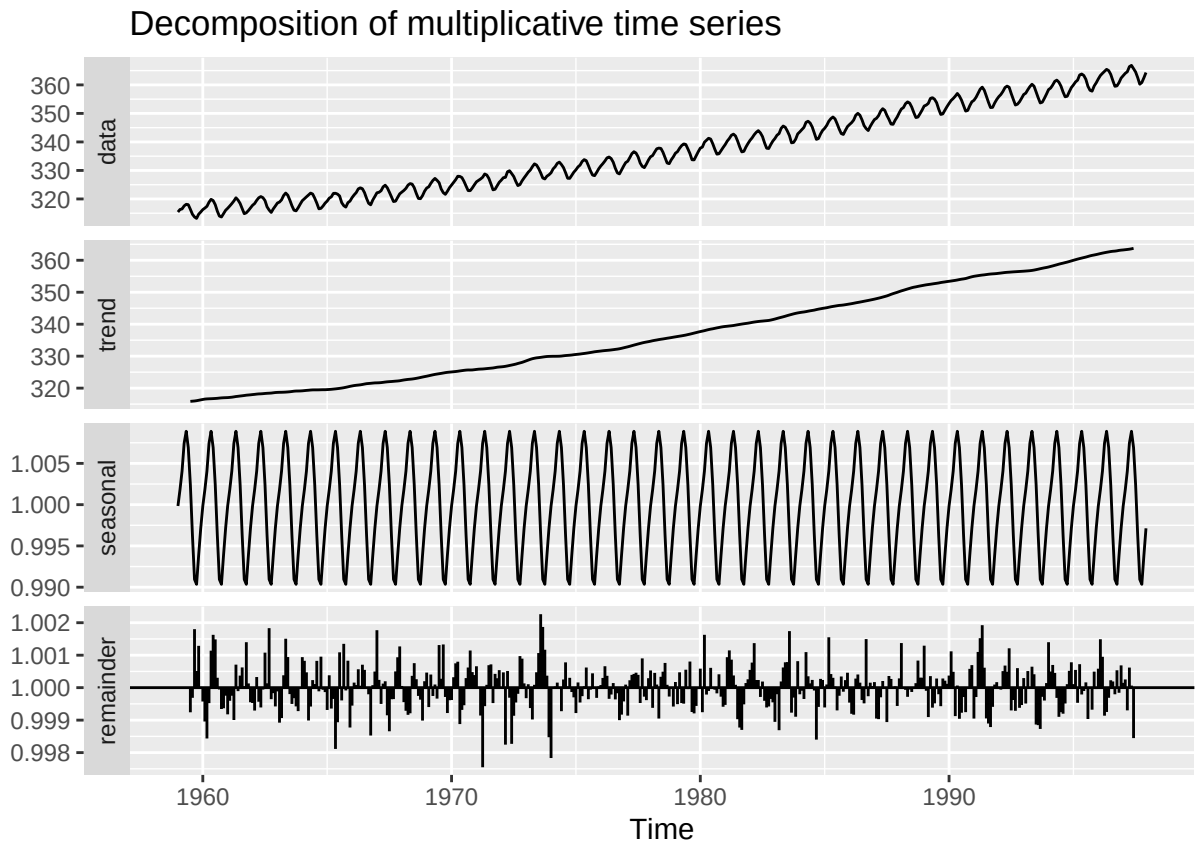

```
boxplot(co2 ~ cycle(co2))
```




```
hist(co2)
```




```
decompose_data1 <- decompose(co2, "multiplicative")
autoplot(decompose_data1)
```



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.

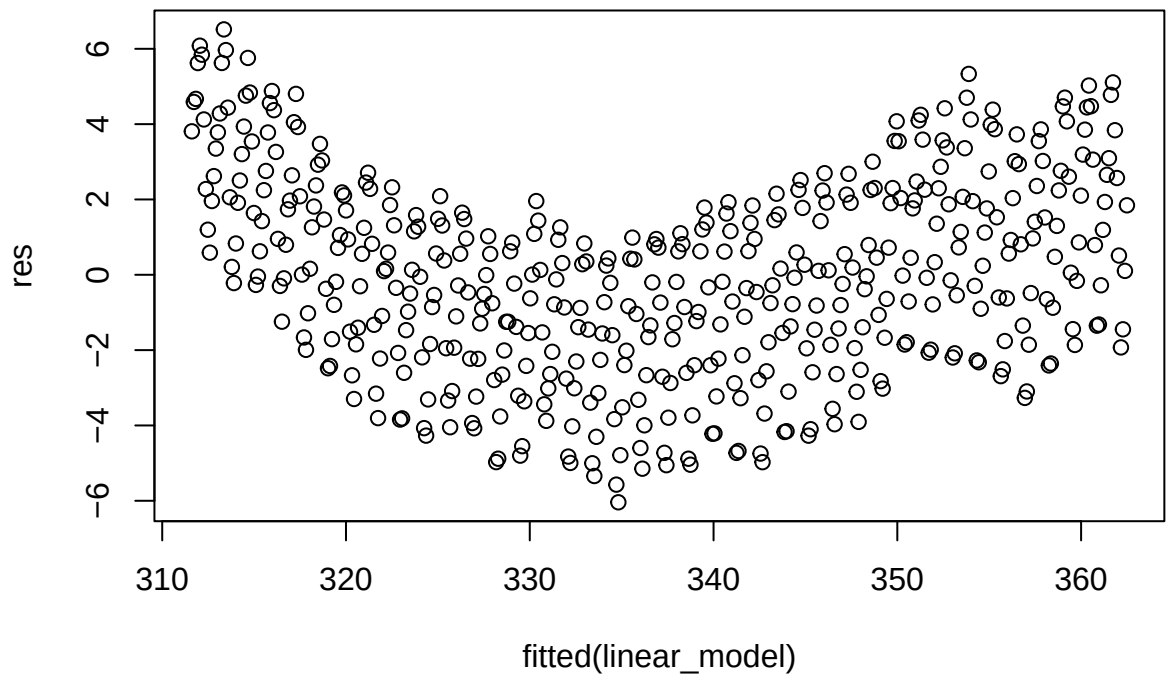
Response: The linear time trend model and the higher-order polynomial time trend model are shown below. For the two models, the time trend is significant for each scenario, and the residuals of each model actually show a reverse pattern of each other. Based on the residuals plot, and the Q-Q plot, the linear-time trend model would not pass the CLM assumptions. The data would also not be appropriate for a logarithmic transformation because we want to be able to use the predictions from the model to know the carbon dioxide levels, with a logarithmic transformation in a model, it makes it much harder to place the model estimates into context. Also, we normally use log transformation to make the data more normal, however that is not necessary in this case.

```
linear_model <- tslm(co2 ~ trend)
summary(linear_model)
```

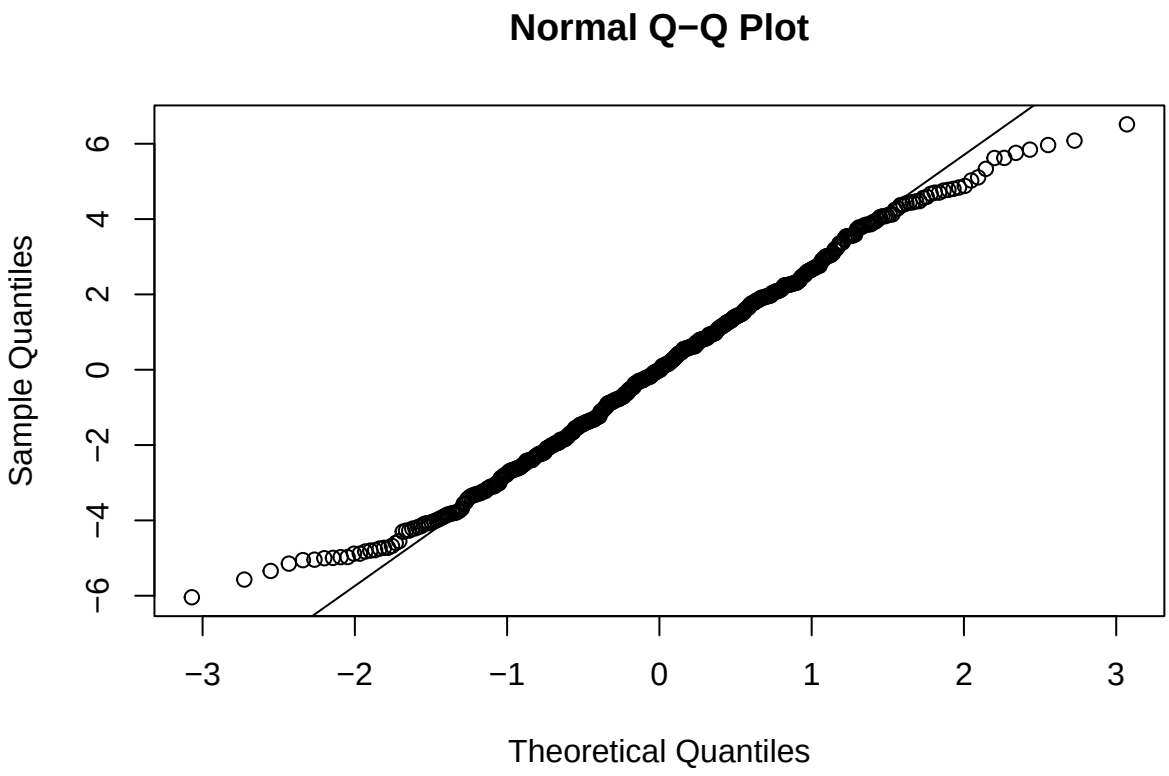
```
##
## Call:
## tslm(formula = co2 ~ trend)
##
## Residuals:
```

```
##     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 ***
## trend       1.090e-01  8.958e-04   121.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.618 on 466 degrees of freedom
## Multiple R-squared:  0.9695, Adjusted R-squared:  0.9694
## F-statistic: 1.479e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

```r
res <- resid(linear_model)

plot(fitted(linear_model), res)
```



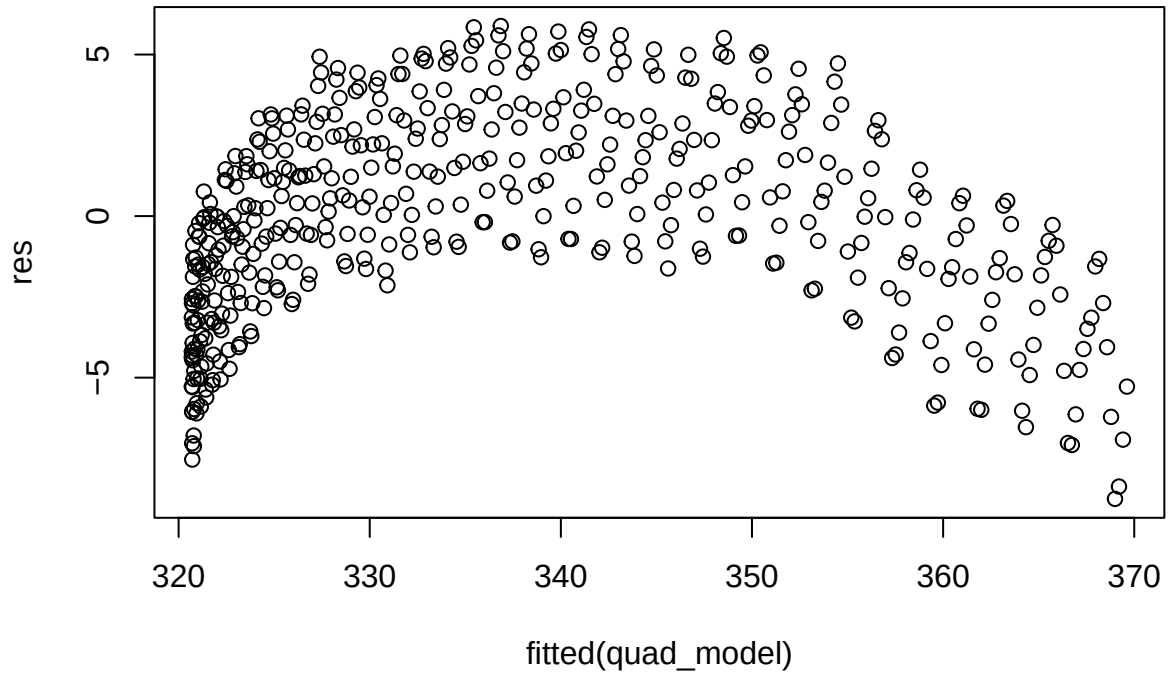

```r
qqnorm(res)
qqline(res)
```



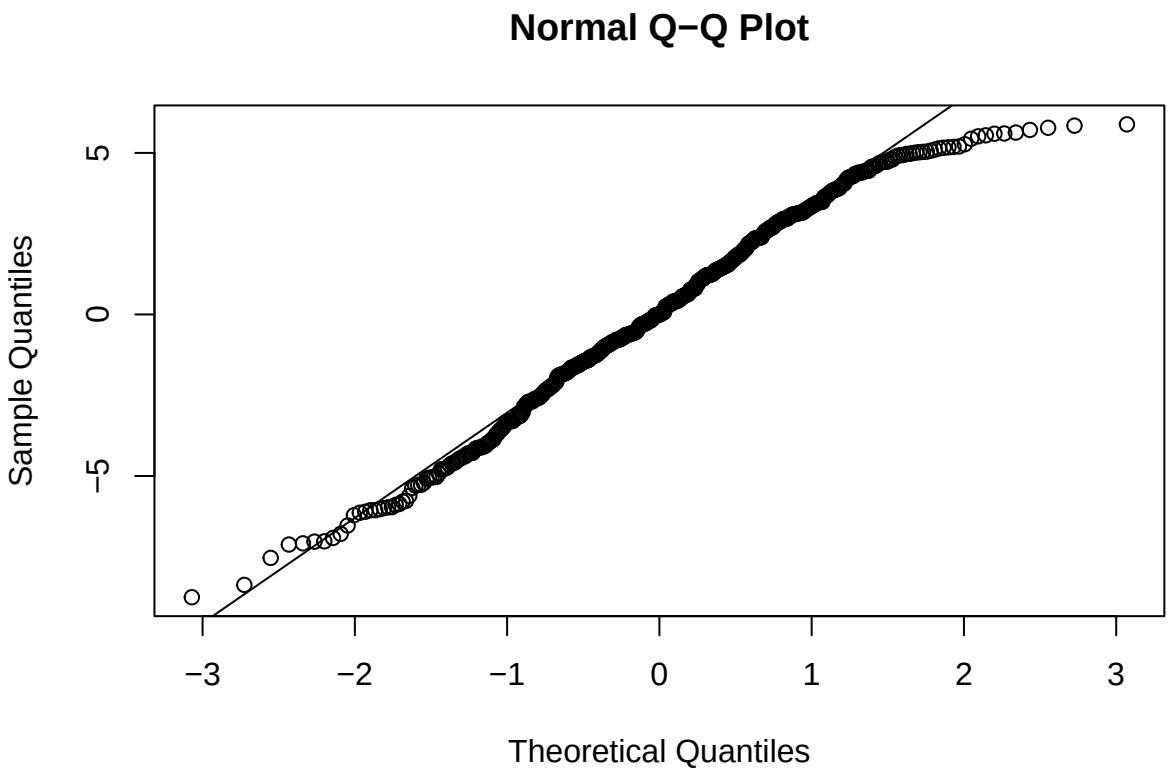

```
quad_model <- tslm(co2 ~ I(trend^2))
summary(quad_model)
```

```
##
## Call:
## tslm(formula = co2 ~ I(trend^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7523 -1.9818 -0.0023  2.4092  5.8833
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.207e+02  2.187e-01  1466.4   <2e-16 ***
## I(trend^2)  2.234e-04  2.227e-06   100.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.152 on 466 degrees of freedom
## Multiple R-squared:  0.9557, Adjusted R-squared:  0.9556
## F-statistic: 1.006e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

```r
res <- resid(quad_model)

plot(fitted(quad_model), res)
```




```r
qqnorm(res)
qqline(res)
```



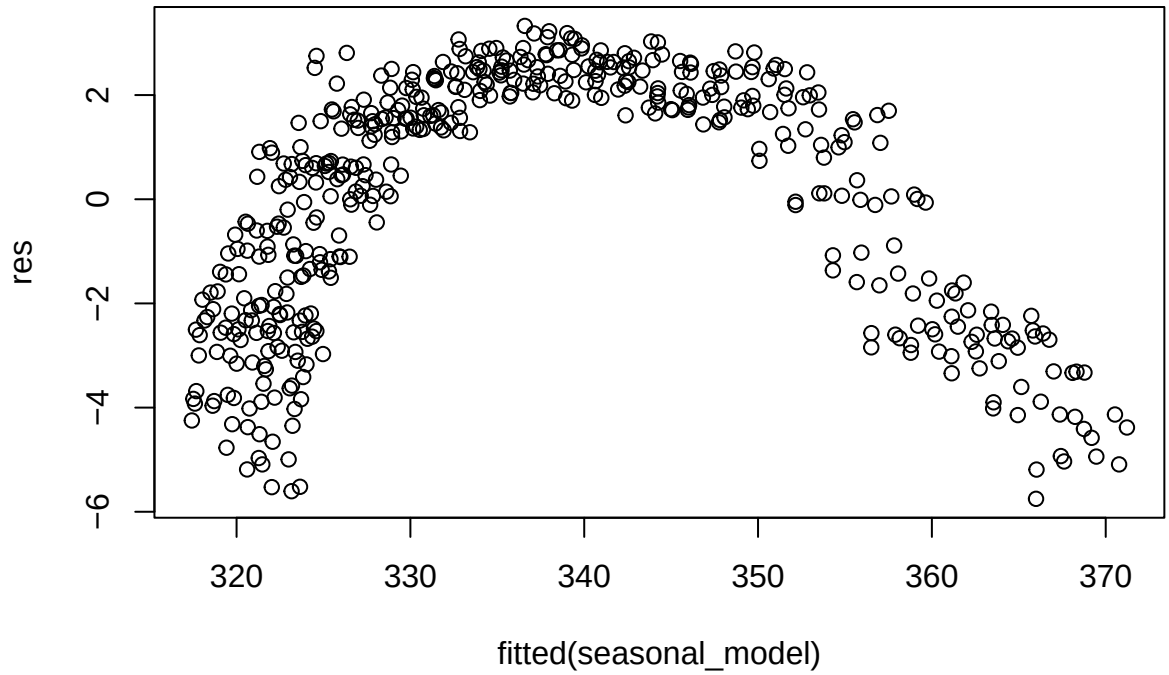
Below is the seasonal model that contains a quadratic trend value and seasonal dummy variables. We can see that certain months are significant however certain months are not significant to the model. The residuals plot also has a large curve in it and the Q-Q plot is does not follow the linear line. The forecasted plot is also shown below. From the forecast we can see that the trend seems to increase with intensity in the forecasted values and overall looks like it might over predict some of the values.

```
seasonal_model <- tslm(co2 ~ I(trend^2) + season)

summary(seasonal_model)
```

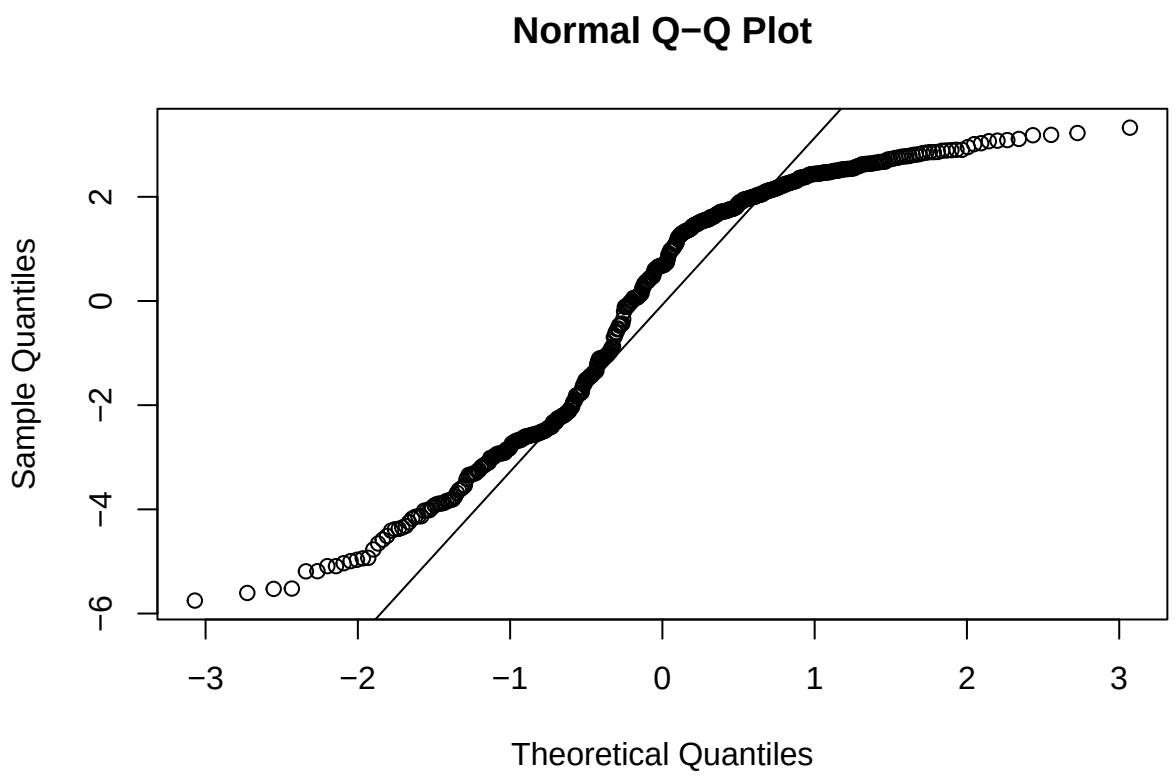
```
##
## Call:
## tslm(formula = co2 ~ I(trend^2) + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7514 -2.2333  0.6857  2.0903  3.3276
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.206e+02  4.063e-01 789.029  < 2e-16 ***
## I(trend^2)   2.239e-04  1.712e-06 130.774  < 2e-16 ***
```

```
## season2      6.698e-01  5.486e-01   1.221 0.222717
## season3      1.418e+00  5.486e-01   2.585 0.010057 *
## season4      2.554e+00  5.486e-01   4.656 4.24e-06 ***
## season5      3.038e+00  5.486e-01   5.537 5.21e-08 ***
## season6      2.379e+00  5.486e-01   4.337 1.78e-05 ***
## season7      8.624e-01  5.486e-01   1.572 0.116646
## season8     -1.202e+00  5.486e-01  -2.190 0.029019 *
## season9     -3.023e+00  5.486e-01  -5.509 6.03e-08 ***
## season10    -3.202e+00  5.486e-01  -5.837 1.01e-08 ***
## season11    -2.011e+00  5.486e-01  -3.665 0.000277 ***
## season12    -8.911e-01  5.486e-01  -1.624 0.105024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.422 on 455 degrees of freedom
## Multiple R-squared:  0.9745, Adjusted R-squared:  0.9738
## F-statistic:  1448 on 12 and 455 DF,  p-value: < 2.2e-16
```

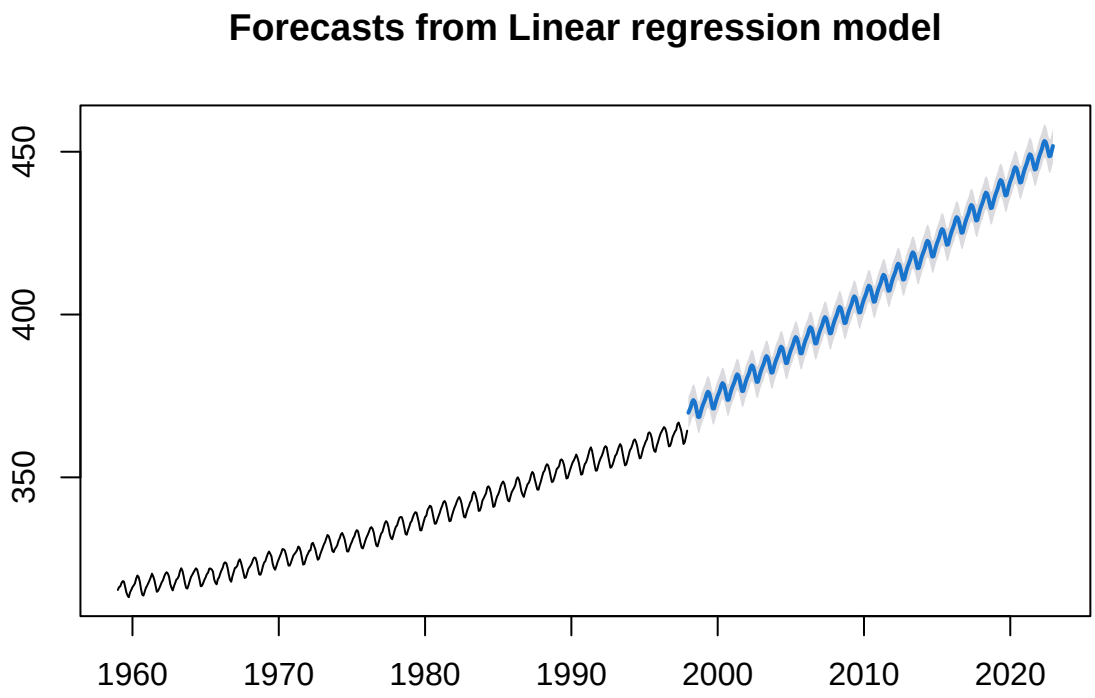
```r
res <- resid(seasonal_model)

plot(fitted(seasonal_model), res)
```



```
qqnorm(res)
qqline(res)
```




```
forecast <- forecast(seasonal_model, h = 300, level = 95)

plot(forecast)
```



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.

Response:

Below I have fitted the best ARIMA model from multiple iterations, the starting point coming from looking at some of the EDA graphs completed above. Since there is a strong seasonality cycle in the decomposition plot, I will be using a seasonal ARIMA model. In a seasonal ARIMA model the format is shown below:
ARIMA(p,d,q)(P,D,Q)m

From this model, the lowercase letters are for the non-seasonal terms and the uppercase letters are the seasonal terms with m being the seasonal frequency. The p is the auto-regressive value, the d is the differencing term, and the q is the moving average term.

From the EDA completed above, there were a key points to take into consideration when deciding what kind of ARIMA model to use. First, from the decomposition plot, there is an obvious trend, which will need to be taken out by differencing once. However, I was not sure if it should be a seasonal or non-seasonal differencing value, therefore I tested both to see which had better results. From the ACF plot in the EDA section, the values do not decline towards zero quickly, therefore the auto-regressive value will definitely not be zero. Lastly, for the PACF graph shows that the model may have a moving average value of 1 or 2 and a seasonal moving average value of 1 or 2.

With all of these values as starting points, we iteratively tried the different combination to look for

the model with the lowest AIC value (172.71). Some of the models that came close to having the minimized AIC value are shown below, however we did complete more models at trials than are shown below. We also looked through the residual values and the ACF and PACF graphs of the residuals to see how the model performed. The model we chose as the best has non-seasonal values of p = 1, d = 1, q = 1, and seasonal values of p = 2, d = 1, q = 1, with the seasonal frequency being 12 for the months. The residuals of this model are all fairly low, and the ACF and PACF graphs only have one spike that goes past the blue dotted line.

Final ARIMA Model: ARIMA(1,1,1)(2,1,2)[12]

Final ARIMA Equation: $(1-\Phi 1B12-\Phi 2B24)(1-B12)(1-B)yt=(1+ 1B)(1+\Theta 1B12+\Theta 2B24)(1+\Theta 1B+\Theta 2B)et$

Lastly, there is a forecast plot of the model below that forecasts the next 25 years of data, with a confidence interval on the graph. The confidence level is pretty tight in the beginning but grows wider as we try to predict further out. Overall, the Arima model seems to do a good job.

```
arima_model1 <- arima(co2, order = c(1, 1, 2), seas = list(order = c(2,
    1, 2), 12))

arima_model1
```

```
##
## Call:
## arima(x = co2, order = c(1, 1, 2), seasonal = list(order = c(2, 1, 2), 12))
##
## Coefficients:
##          ar1      ma1     ma2    sar1     sar2     sma1    sma2
##       0.5895  -0.9249  0.1382  0.9574  -0.1368  -1.8120  0.8525
## s.e.  0.2406   0.2454  0.1117  0.0794   0.0601   0.0715  0.0629
##
## sigma^2 estimated as 0.07887:  log likelihood = -78.66,  aic = 173.32
```

```
arima_model2 <- arima(co2, order = c(1, 1, 2), seas = list(order = c(1,
    1, 2), 12))

arima_model2
```

```
##
## Call:
## arima(x = co2, order = c(1, 1, 2), seasonal = list(order = c(1, 1, 2), 12))
##
## Coefficients:
##          ar1      ma1     ma2     sar1     sma1     sma2
##       0.5959  -0.9284  0.1413  -0.5055  -0.3059  -0.4784
## s.e.  0.2402   0.2445  0.1101   0.4996   0.4857   0.4118
##
## sigma^2 estimated as 0.08168:  log likelihood = -83.66,  aic = 181.31
```

```r
arima_model3 <- arima(co2, order = c(1, 1, 1), seas = list(order = c(1,
    1, 1), 12))

arima_model3
```

```
##
## Call:
## arima(x = co2, order = c(1, 1, 1), seasonal = list(order = c(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 0.08217:  log likelihood = -84.88,  aic = 179.76
```

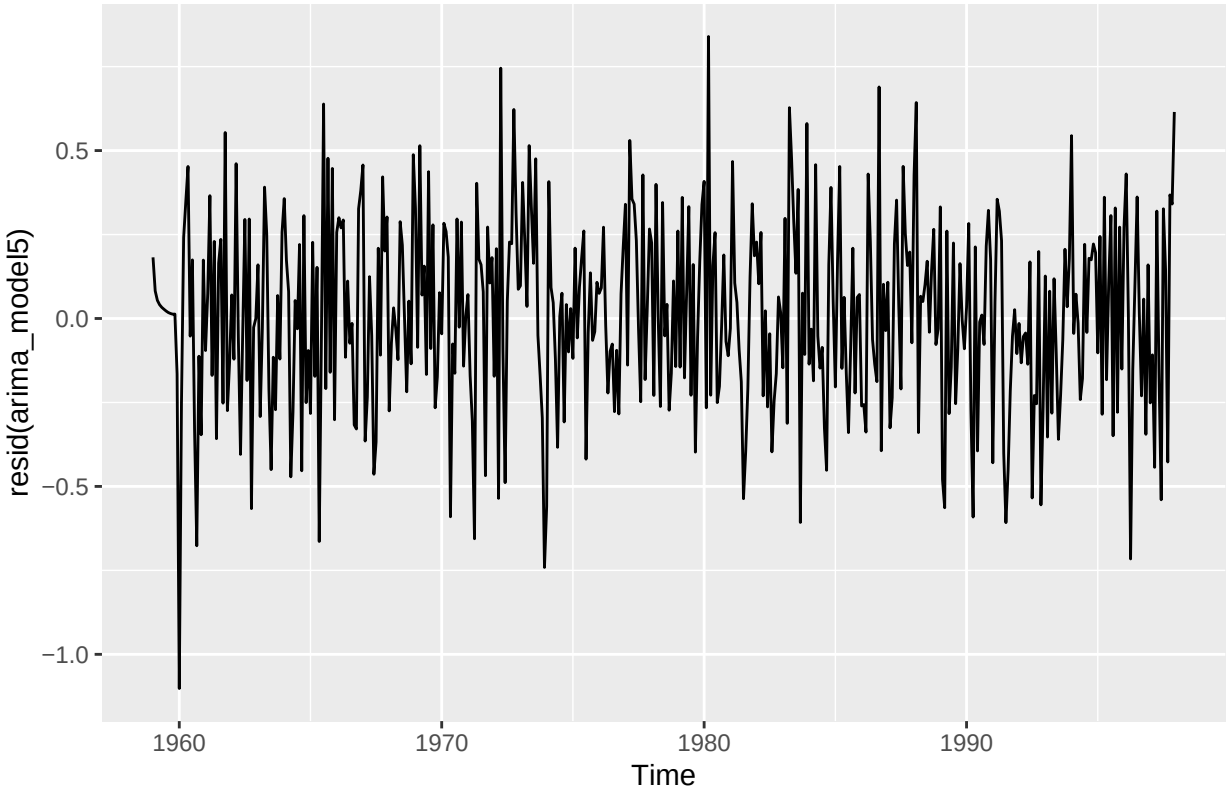
```r
arima_model4 <- arima(co2, order = c(1, 1, 1), seas = list(order = c(2,
    1, 1), 12))

arima_model4
```

```
##
## Call:
## arima(x = co2, order = c(1, 1, 1), seasonal = list(order = c(2, 1, 1), 12))
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1
##       0.2595  -0.5902  0.0113  -0.0869  -0.8369
## s.e.  0.1390   0.1186  0.0558   0.0539   0.0332
##
## sigma^2 estimated as 0.08163:  log likelihood = -83.6,  aic = 179.2
```

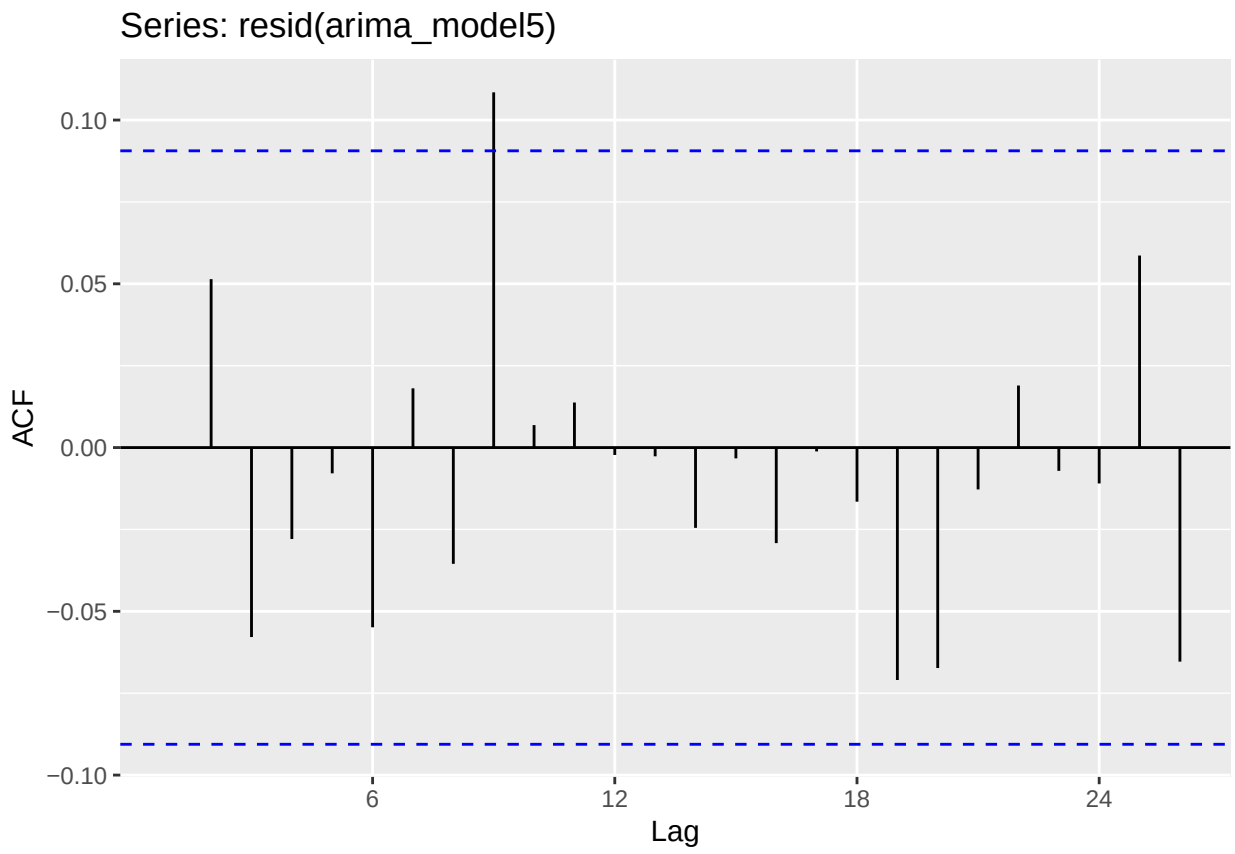
```r
arima_model5 <- arima(co2, order = c(1, 1, 1), seas = list(order = c(2,
    1, 2), 12))

arima_model5
```

```
##
## Call:
## arima(x = co2, order = c(1, 1, 1), seasonal = list(order = c(2, 1, 2), 12))
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1    sma2
##       0.2652  -0.5950  0.9613  -0.1335  -1.8169  0.8564
## s.e.  0.1355   0.1152  0.0789   0.0603   0.0712  0.0624
##
## sigma^2 estimated as 0.0791:  log likelihood = -79.35,  aic = 172.71
```

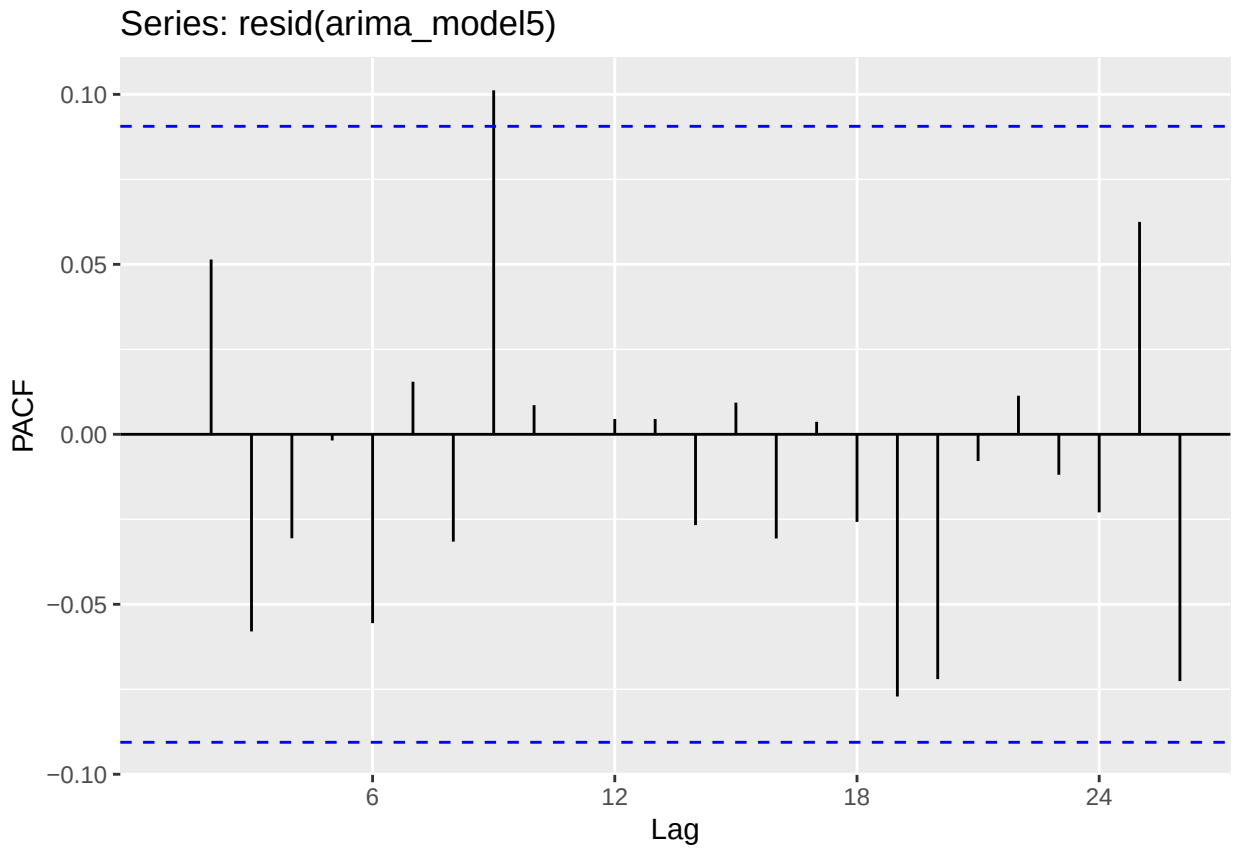
```
autoplot(resid(arima_model5))
```



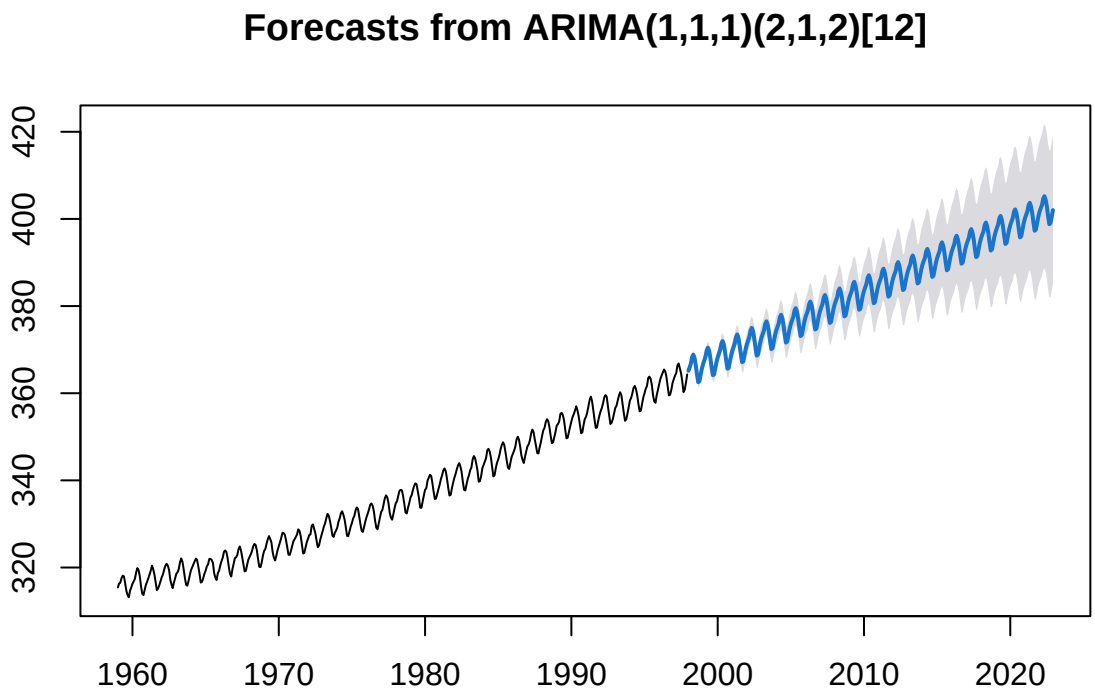

```
ggAcf(resid(arima_model5))
```




```
ggPacf(resid(arima_model5))
```




```
myforecast <- forecast(arima_model5, level = c(95), h = 300)

plot(myforecast)
```

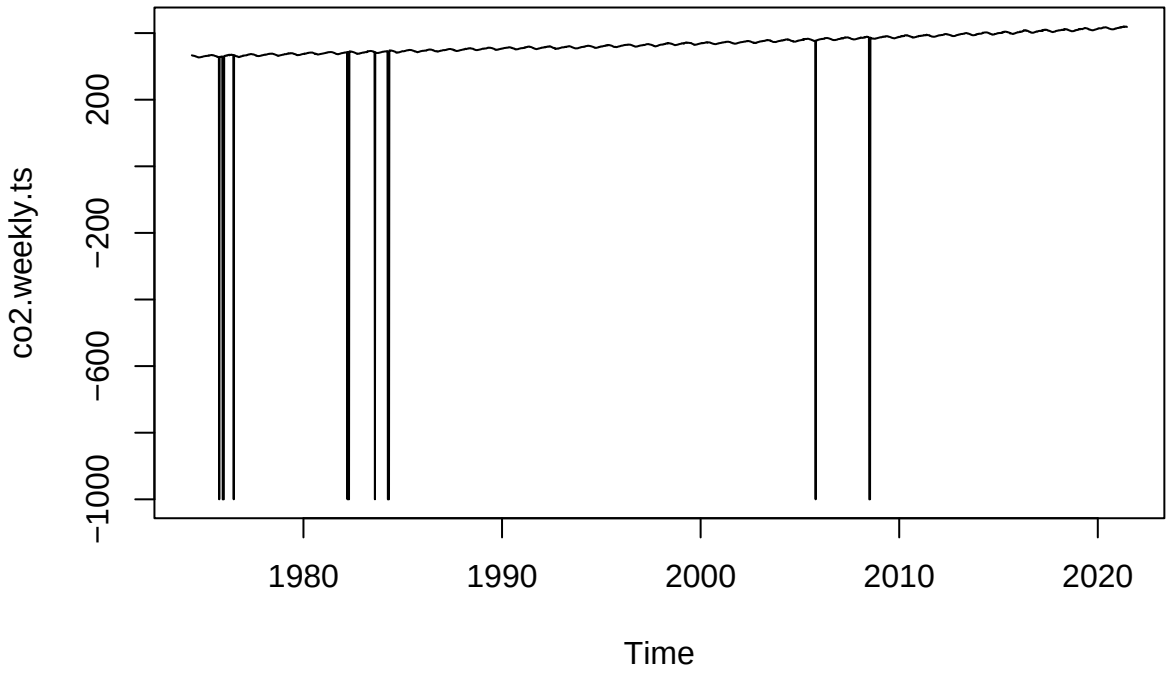


Part 4 (5 points)

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 predictions 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.

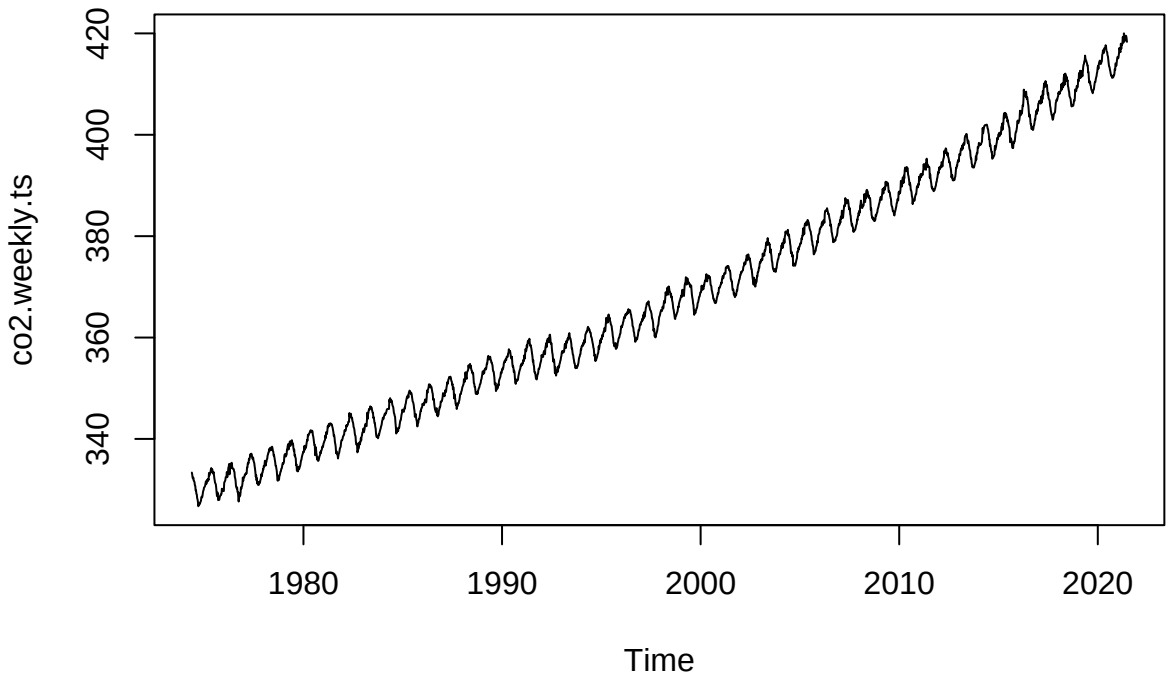
```
co2_weekly <- read.table("co2_weekly_mlo.txt", sep = "\t", fill = FALSE,
    strip.white = TRUE)
co2_weekly <- gsub("\\s+", " ", co2_weekly)
co2_weekly <- str_remove_all(co2_weekly, ",")
co2_weekly <- str_remove_all(co2_weekly, "\"")
co2_weekly <- str_sub(co2_weekly, 3, nchar(co2_weekly) - 1)

# create a dataframe from the cleaned txt file convert from
# char to numeric
co2_weekly_df <- data.frame(strsplit(co2_weekly, " "))
co2_weekly_df <- data.frame(matrix(unlist(t(co2_weekly_df)),
    byrow = T, 2458, 9))
co2_weekly_df <- data.frame(apply(co2_weekly_df, 2, as.numeric))
colnames(co2_weekly_df) <- c("yr", "mon", "day", "decimal", "ppm",
```

```
    "days", "1 yr ago", "10 yr ago", "since 1800")

co2.weekly.ts <- ts(co2_weekly_df$ppm, start = co2_weekly_df$decimal[1],
    frequency = 365.25/7)
plot(co2.weekly.ts)
```



There are several obvious missing values from this dataset. We will convert the -999 data points to NA first and then fill the NA values with previously available data points using na.locf function. We created a new column named ppm_fill that will represent the ppm values without NAs so we still have the original datasets to try different fill methods if needed.

```
co2_weekly_df[co2_weekly_df == -999.99] <- NA
co2_weekly_df <- co2_weekly_df %>%
    mutate(ppm_fill = na.locf(ppm, na.rm = FALSE))

co2.weekly.ts <- ts(co2_weekly_df$ppm_fill, start = co2_weekly_df$decimal[1],
    frequency = 365.25/7)
plot(co2.weekly.ts)
```



After filling in the missing values, the plot of the ppm over time is very similar to what we saw in question 1.

```
head(co2.weekly.ts)
```

```
## Time Series:
## Start = 1974.3795
## End = 1974.47532477755
## Frequency = 52.1785714285714
## [1] 333.37 332.95 332.35 332.20 332.37 331.73
```

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

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

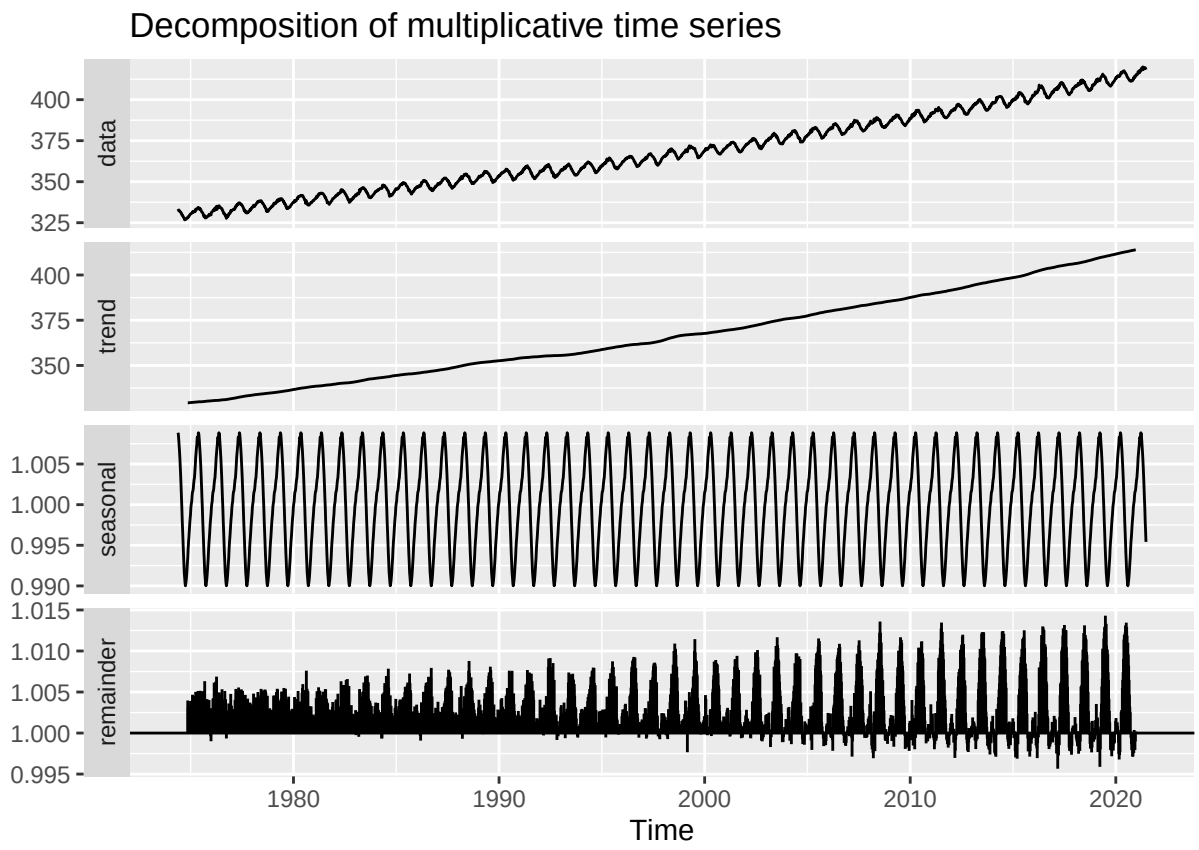
```
frequency(co2.weekly.ts)
```

```
## [1] 52.17857
```

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

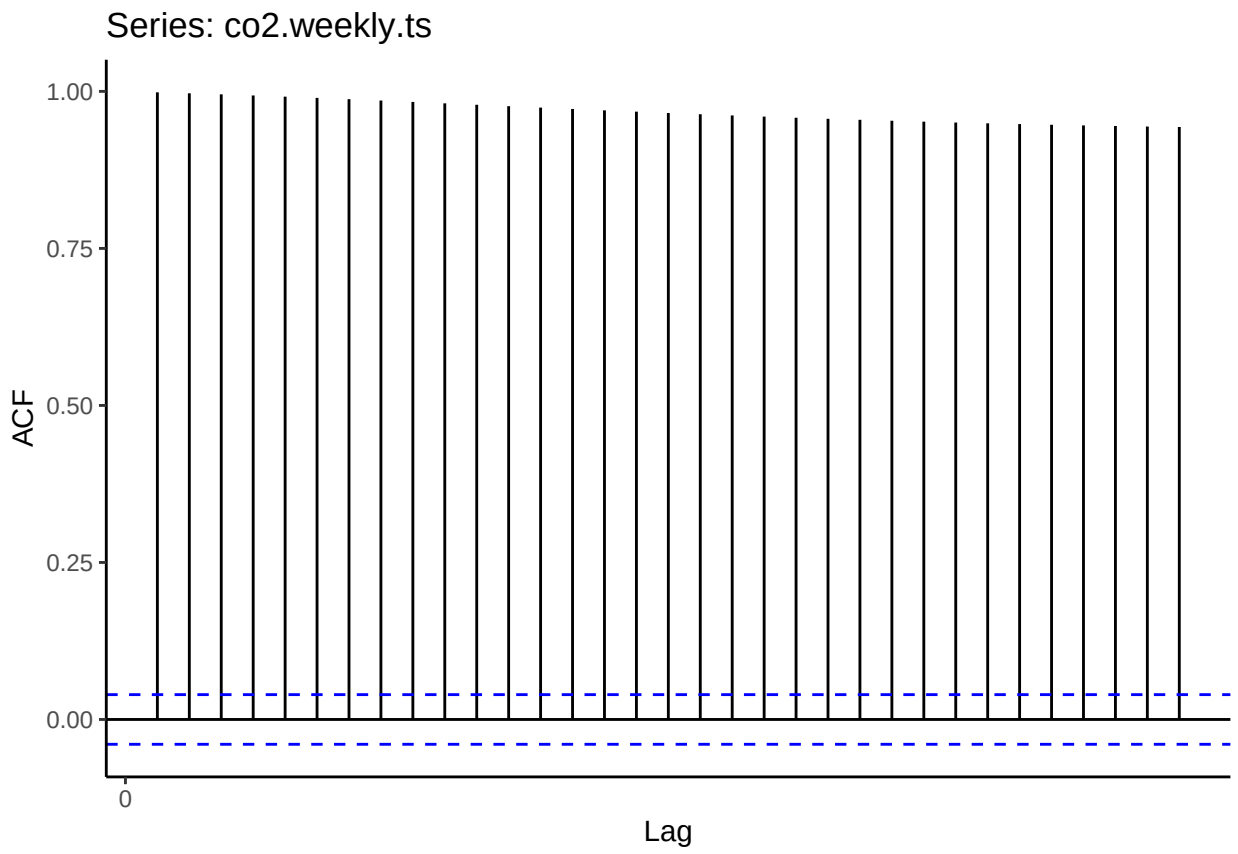
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   326.7   347.3   365.2   368.2   388.4   420.0
```

```
autoplot(decompose(co2.weekly.ts, "multiplicative"))
```

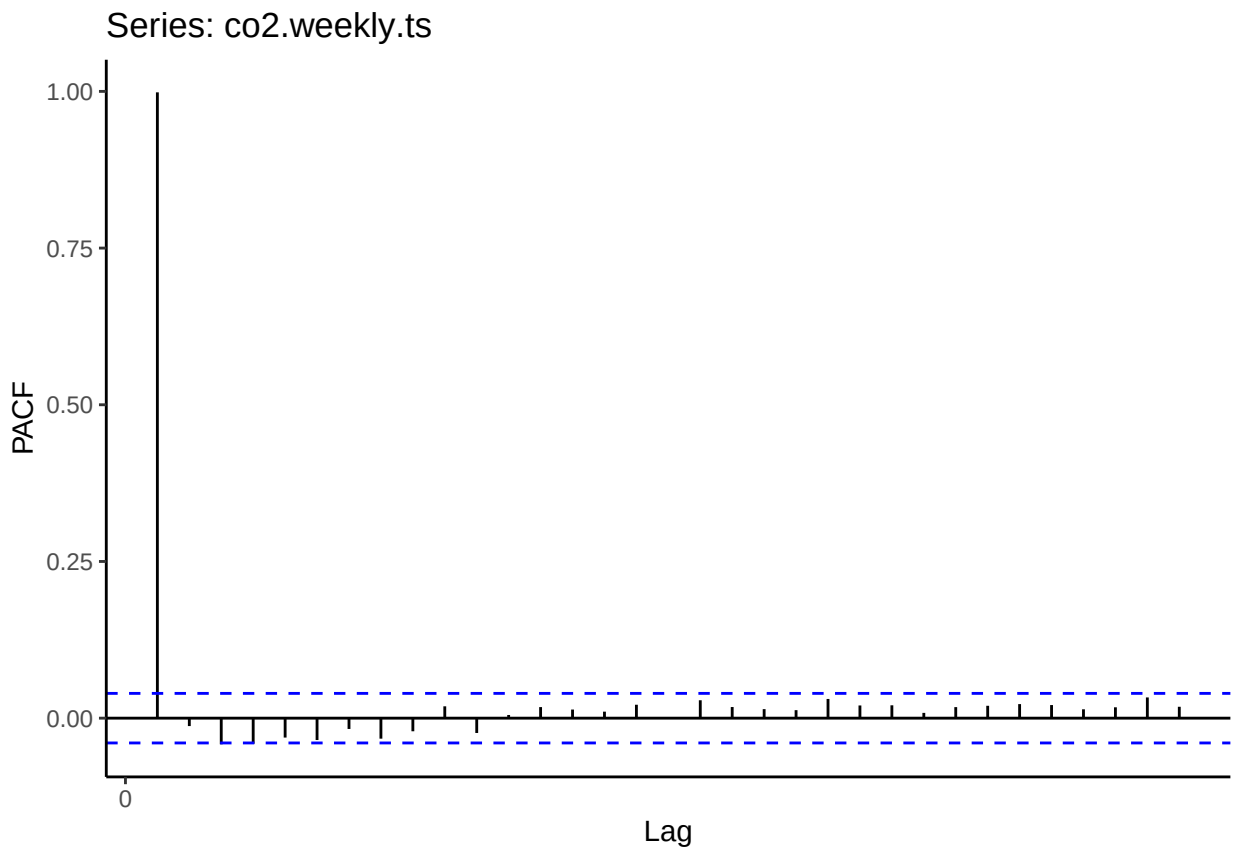


We then conduct EDA for the NOAA data. The range for this dataset is close to the one from question 1. The different is that the frequency of the data is weekly instead of monthly. In the later parts, we will use the xts object function to convert the weekly data into monthly data. From the decomposition plots, we could see clear upward trend from 1970 to 2020 and relatively stable seasonality. There is also an increasing trend in the residuals.

```
autoplot(acf(co2.weekly.ts, plot = FALSE)) + theme_classic()
```




```
autoplot(pacf(co2.weekly.ts, plot = FALSE)) + theme_classic()
```



For the ACF plot, as lag increases, ACF gradually decreases. For the PACF plot, after lag 1, the values for the other lags are close to zero.

```
co2_weekly_df <- co2_weekly_df %>%
    mutate(date = paste0(yr, "-", mon, "-", day))

co2.weekly.xts <- xts(co2_weekly_df$ppm_fill, order.by = as.Date(co2_weekly_df$date))
co2.monthly.avg <- apply.monthly(co2.weekly.xts, mean)
```

In order to take the average of the weekly data and convert the data series to a monthly data series. We will first generate a list of dates that comes from the combination of the year, month, and day columns. We would also convert the data object to xts, so we could leverage the apply.monthly function.

```
# slice forecast and actual monthly results
forecast_df <- data.frame(myforecast)
rownames(forecast_df) <- as.yearmon(rownames(forecast_df))
forecast_df <- forecast_df[1:282, ]

# forecast starts in Jan 1998
co2.monthly.avg_df <- data.frame(co2.monthly.avg)
rownames(co2.monthly.avg_df) <- as.yearmon(rownames(co2.monthly.avg_df))
```

```
co2.monthly.avg.forecast <- data.frame(co2.monthly.avg_df[285:nrow(co2.monthly.avg_df),
    ])
forecast_df <- cbind(forecast_df, co2.monthly.avg.forecast)
colnames(forecast_df) <- c("predicted", "low", "high", "actual")

rmse(forecast_df$actual, forecast_df$predicted)
```

```
## [1] 7.954469
```

```
rmse(forecast_df$actual, forecast_df$predicted)/mean(forecast_df$actual)
```

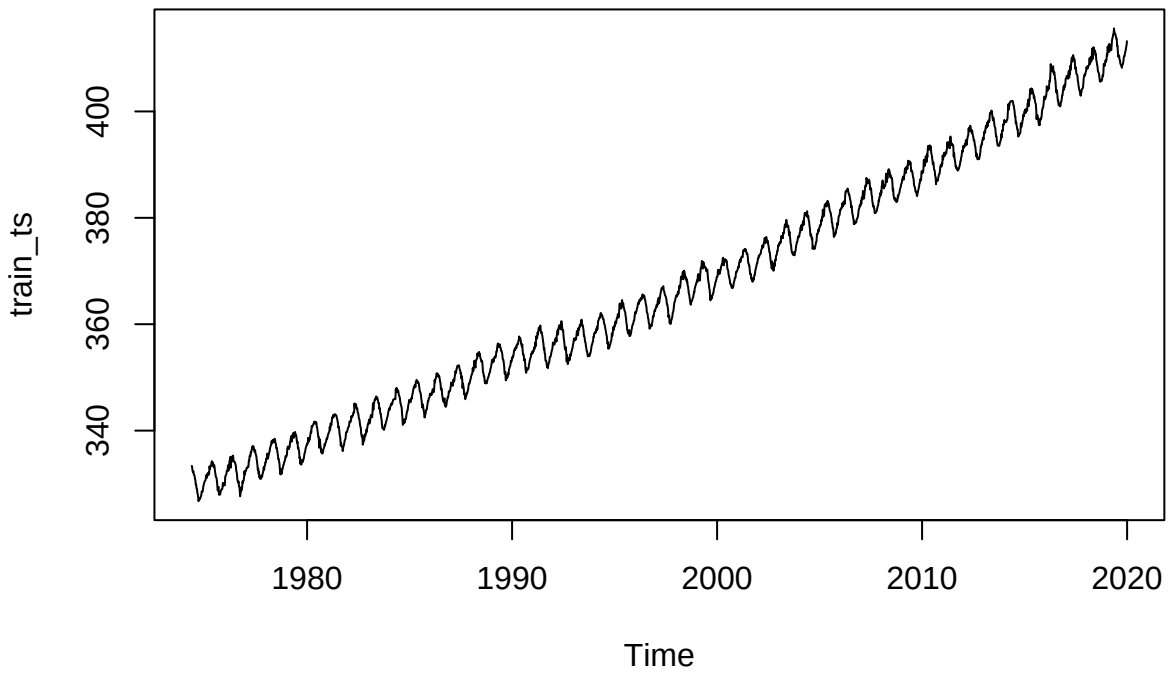
```
## [1] 0.02042741
```

The measurement we used is RMSE. We first sliced the predicted values from the forecast in part 3, and then compared it against the actual values from the NOAA dataset. The RMSE is about 7.95. However, this number is meaningless unless compared to the scale of the data. We then divided the RMSE by the mean of the monthly average ppm. The ratio is about 2.04%, which is a relatively small figure. Our forecast model from part 3 is a good prediction model based on the RMSE metric.

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?

```
# Subset data to training and test data sets
train <- co2_weekly_df[co2_weekly_df$yr < 2020, ]
test <- co2_weekly_df[co2_weekly_df$yr >= 2020, ]

# Transform training and test sets to ts
train_ts <- ts(train$ppm_fill, start = train$decimal[1], frequency = 365.25/7)

test_ts <- ts(test$ppm_fill, start = test$decimal[1], frequency = 365.25/7)

plot(train_ts)
```



Based on the EDA in Part 4, it's evident that there is strong seasonality, so I chose to use a seasonal ARIMA model. The decomposition plot also showed an obvious trend, which will be removed by difference once. The ACF shows that the auto-regressive value will not be zero, and the PACF chart shows that the model will have a moving average value of 1 or 2 and a seasonal moving average value of 1 or 2. Using the available information, I iteratively tried various models, with a few listed below.

```
# Try different ARIMA models and look at AIC

model1 <- arima(train_ts, order = c(1, 1, 1), seas = list(order = c(2,
    1, 2), period = 12))
model1$aic
```

```
## [1] 3980.329
```

```
model2 <- arima(train_ts, order = c(1, 1, 2), seas = list(order = c(1,
    1, 2), period = 12))
model2$aic
```

```
## [1] 3652.058
```

```
model3 <- arima(train_ts, order = c(1, 1, 2), seas = list(order = c(0,
    1, 1), period = 52))
model3$aic
```

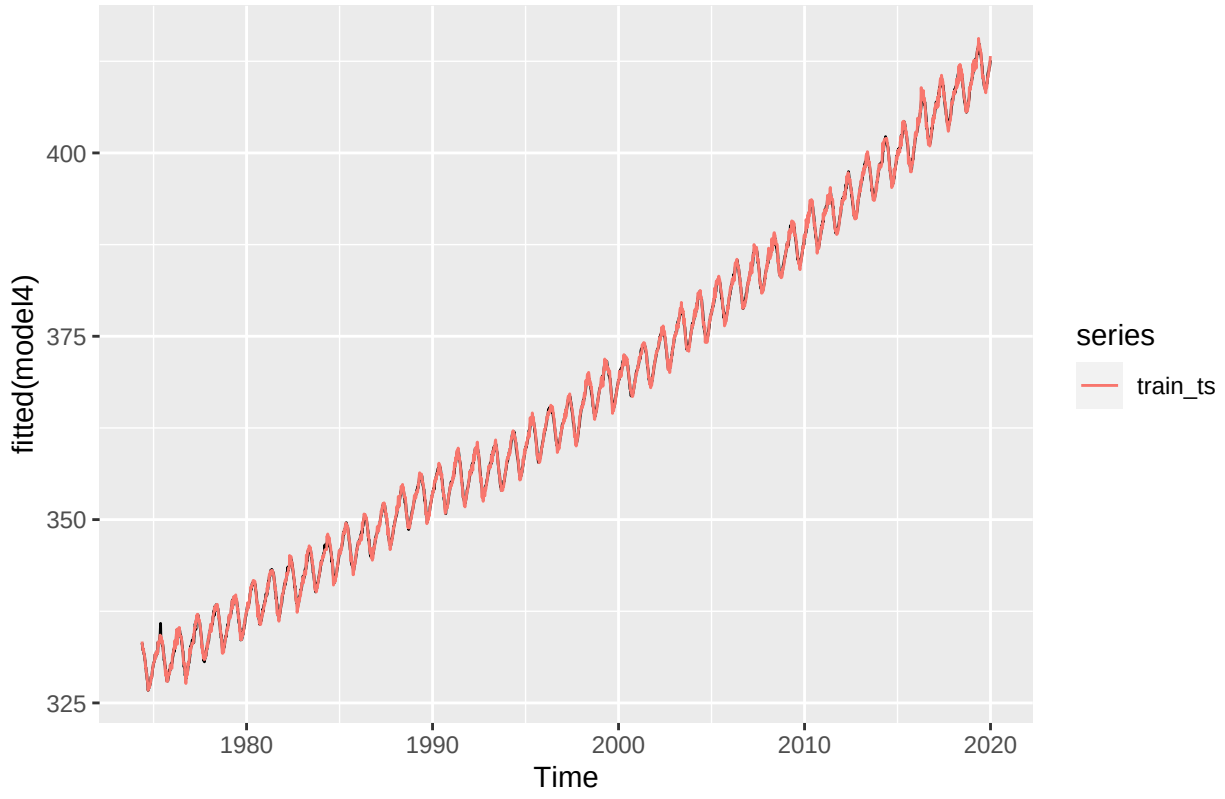
```
## [1] 2856.013
```

```
model4 <- arima(train_ts, order = c(1, 1, 1), seas = list(order = c(0,
    1, 1), period = 52))
model4$aic
```

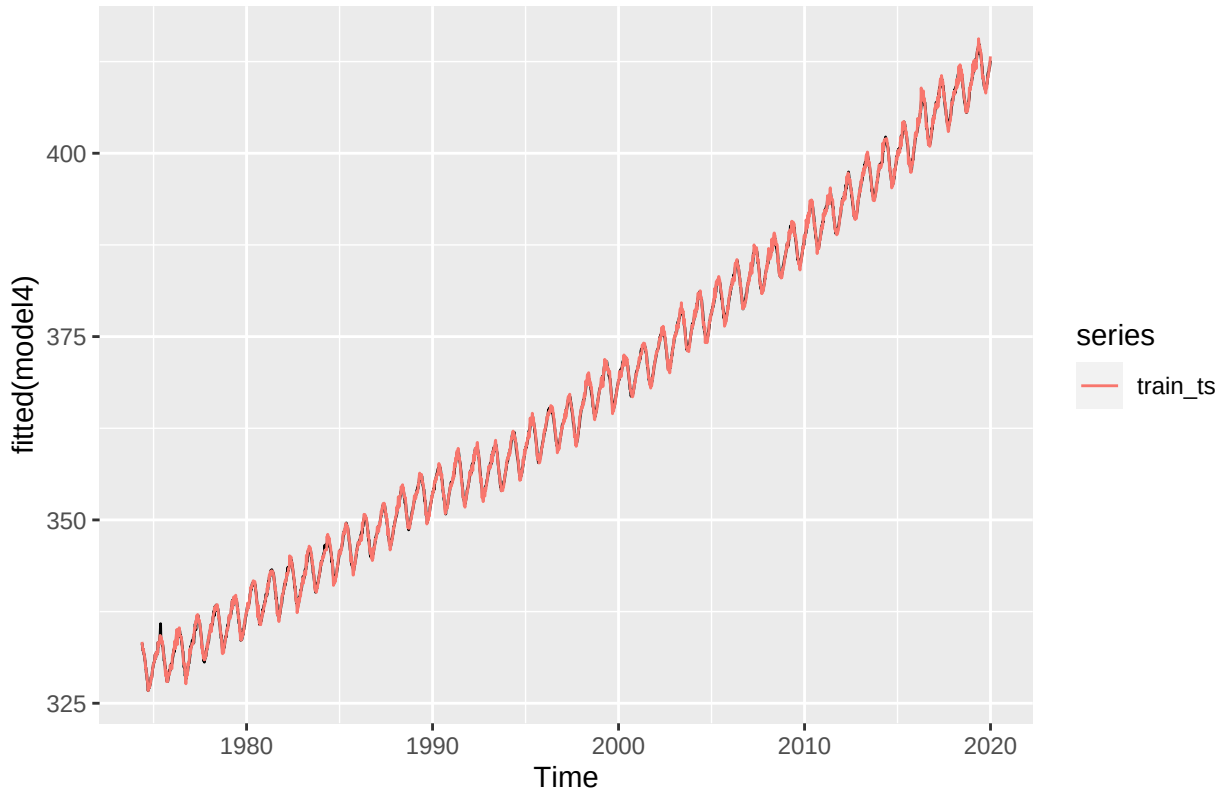
```
## [1] 2854.026
```

The fourth model ARIMA(1,1,1)(0,1,1)[52] shows the lowest AIC, but we also examine in-sample performance (residuals & fitted values vs train set) and out-sample performance (forecast vs test set) to confirm the differences between models.

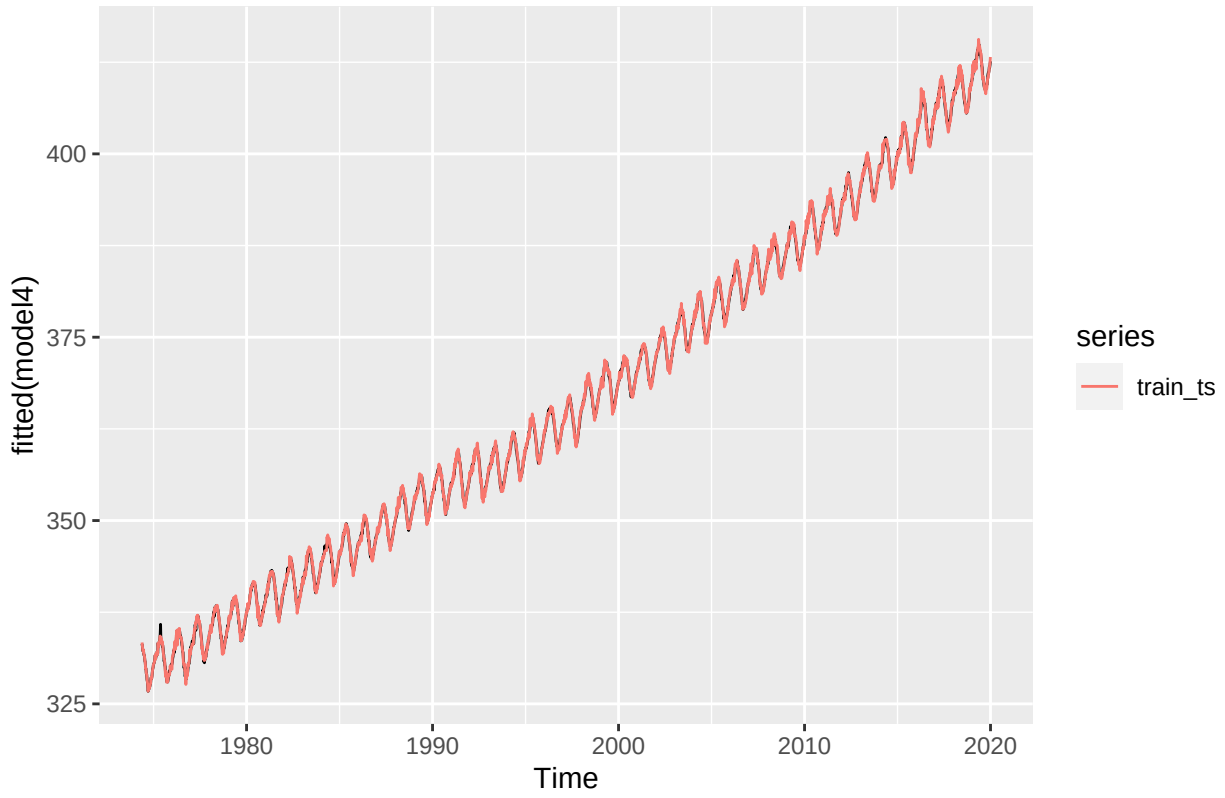
```
# Plot fitted values vs actual for the training set
autoplot(fitted(model4)) + autolayer(train_ts)
```



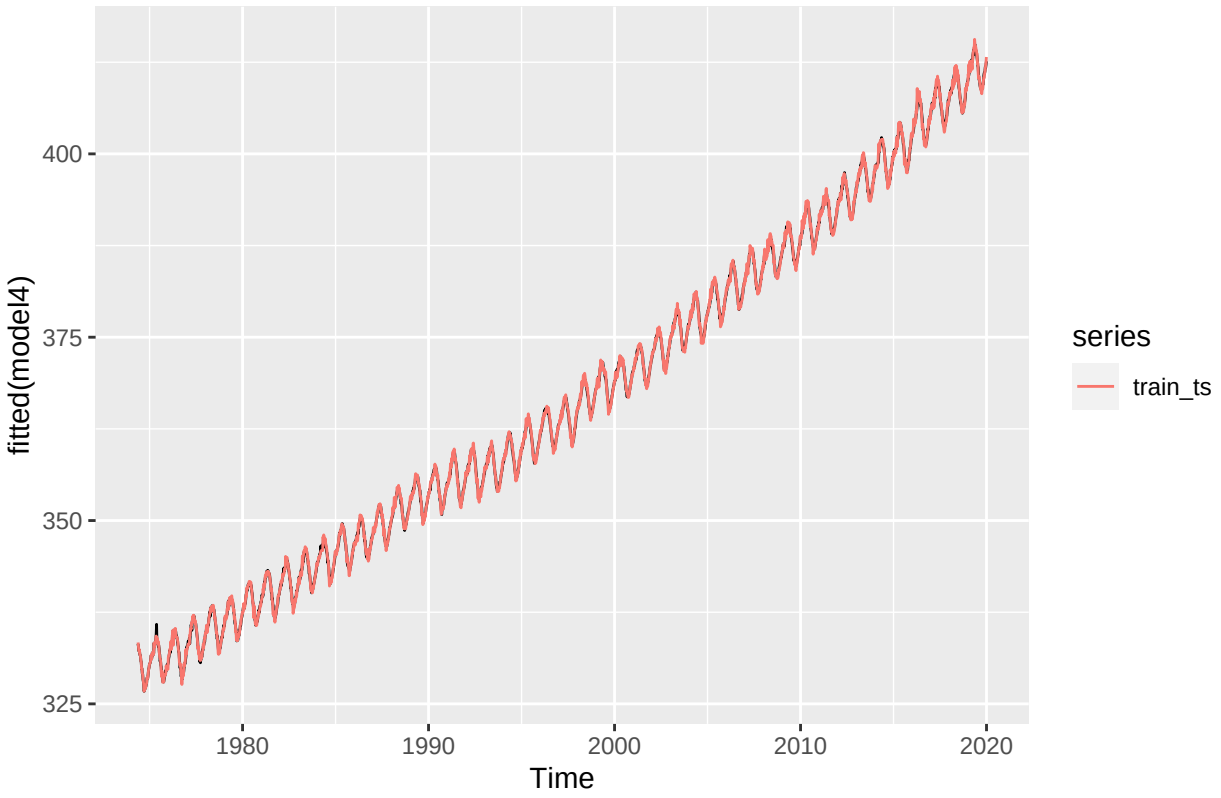
```
autoplot(fitted(model4)) + autolayer(train_ts)
```



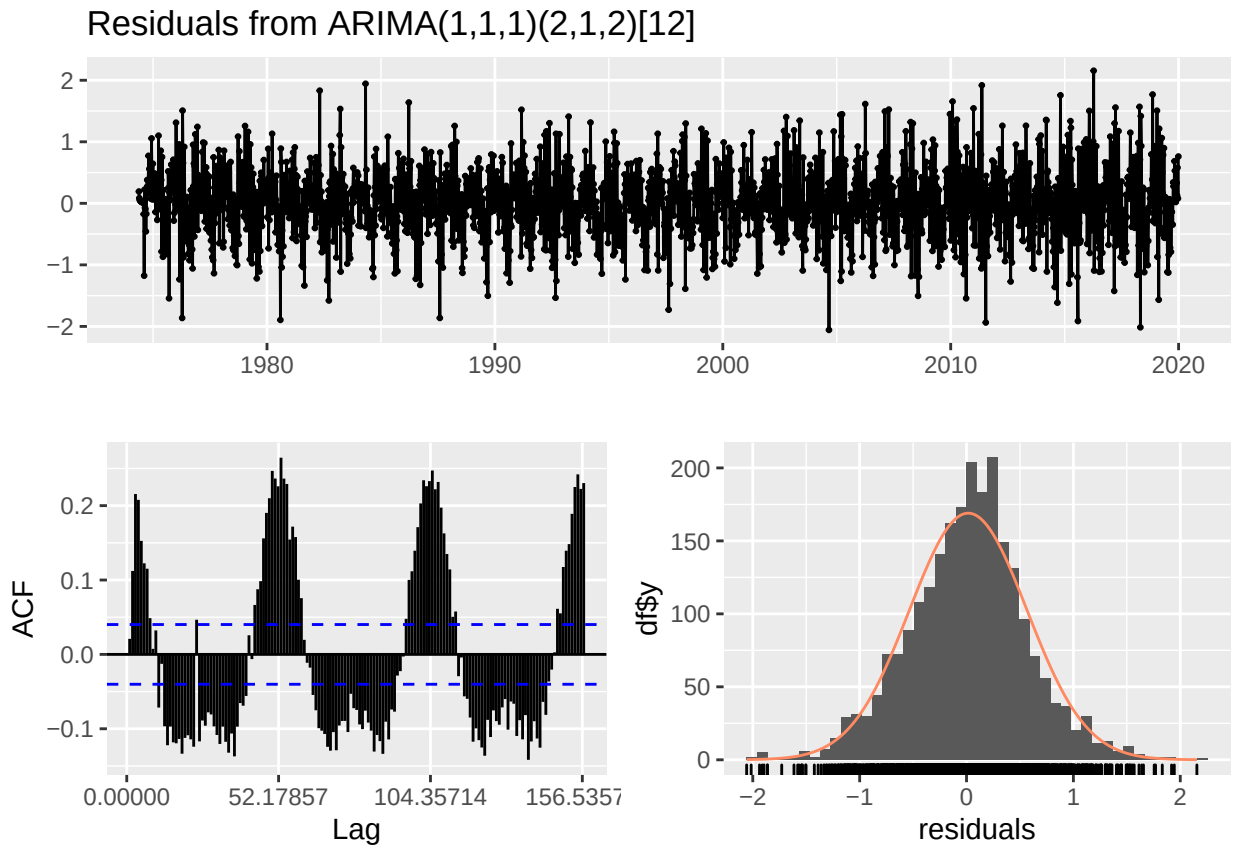

```
autoplot(fitted(model4)) + autolayer(train_ts)
```




```
autoplot(fitted(model4)) + autolayer(train_ts)
```

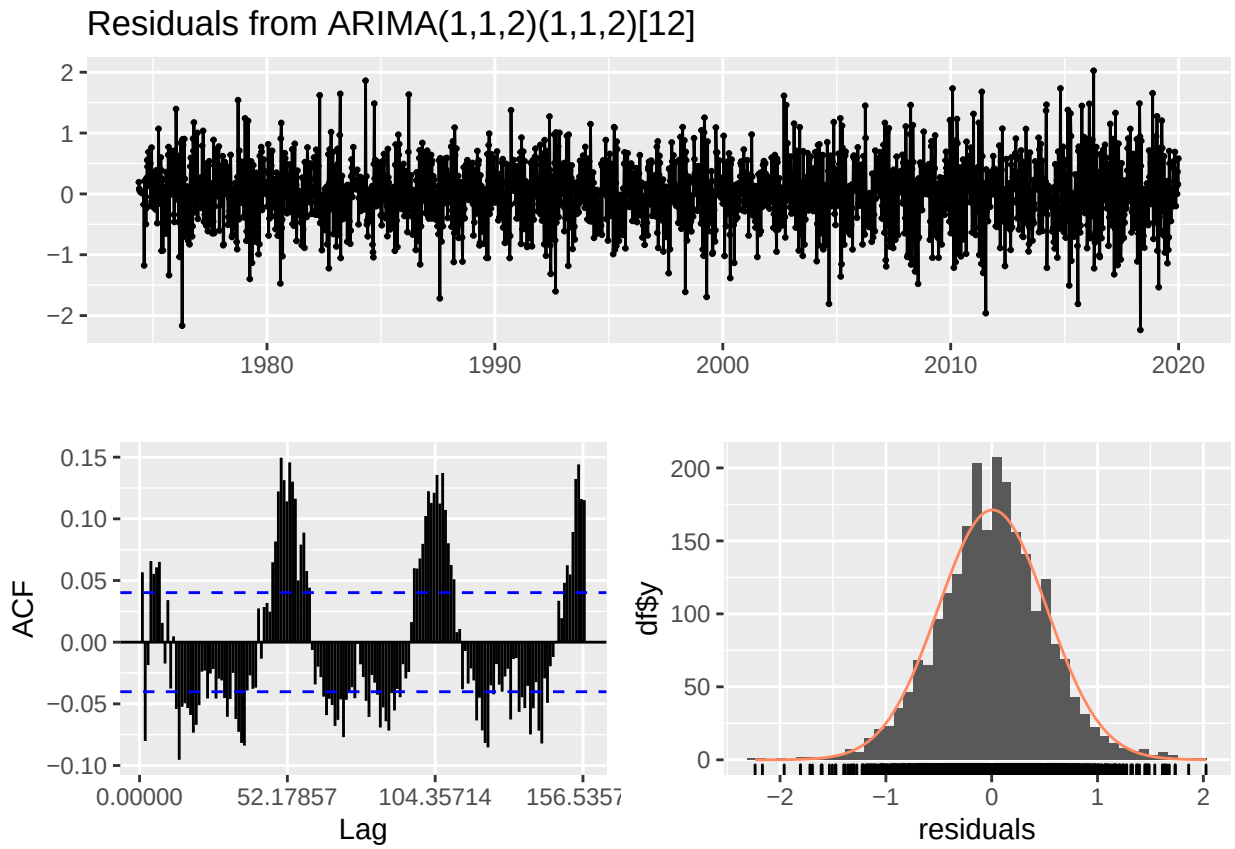



```
# Check the residuals for the different models
checkresiduals(model1)
```



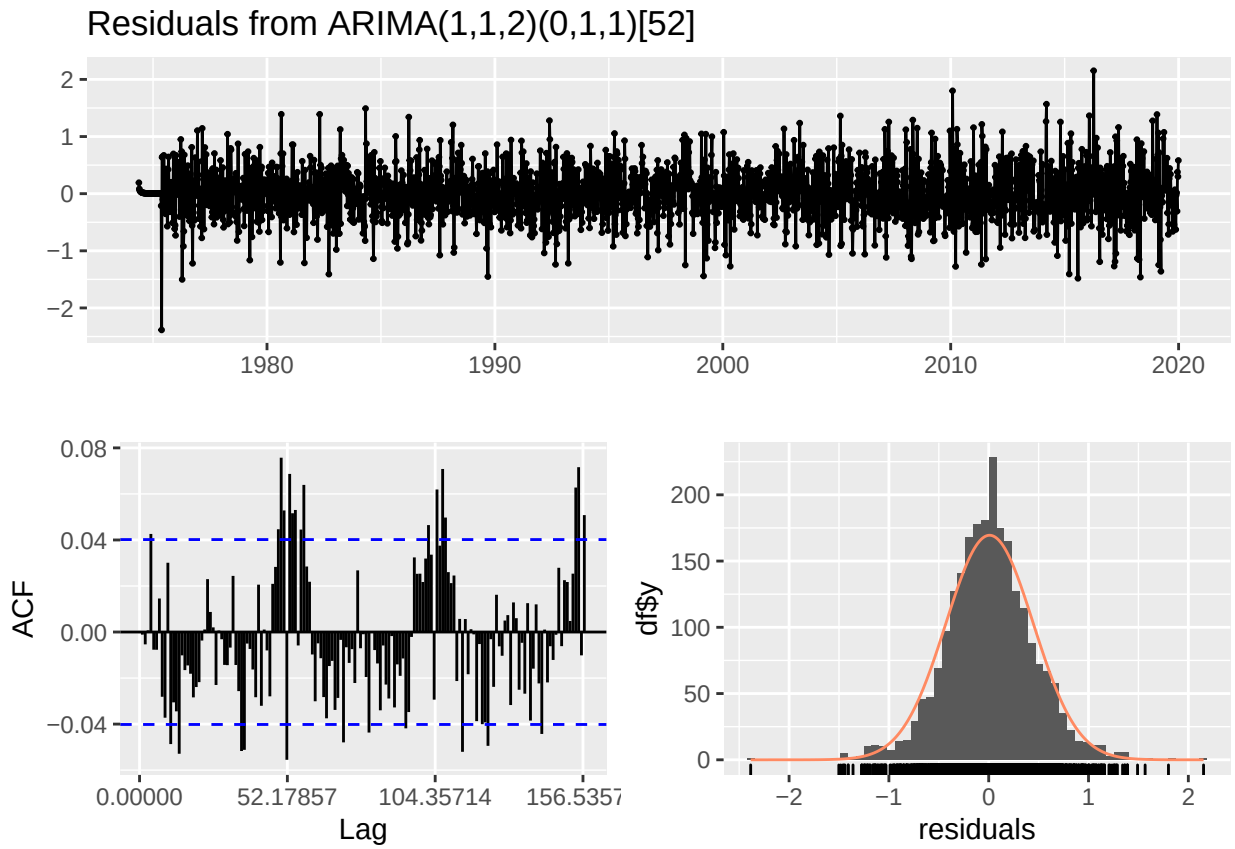

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(2,1,2)[12]
## Q* = 3936.4, df = 98.357, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 104.357142857143
```

```
checkresiduals(model2)
```



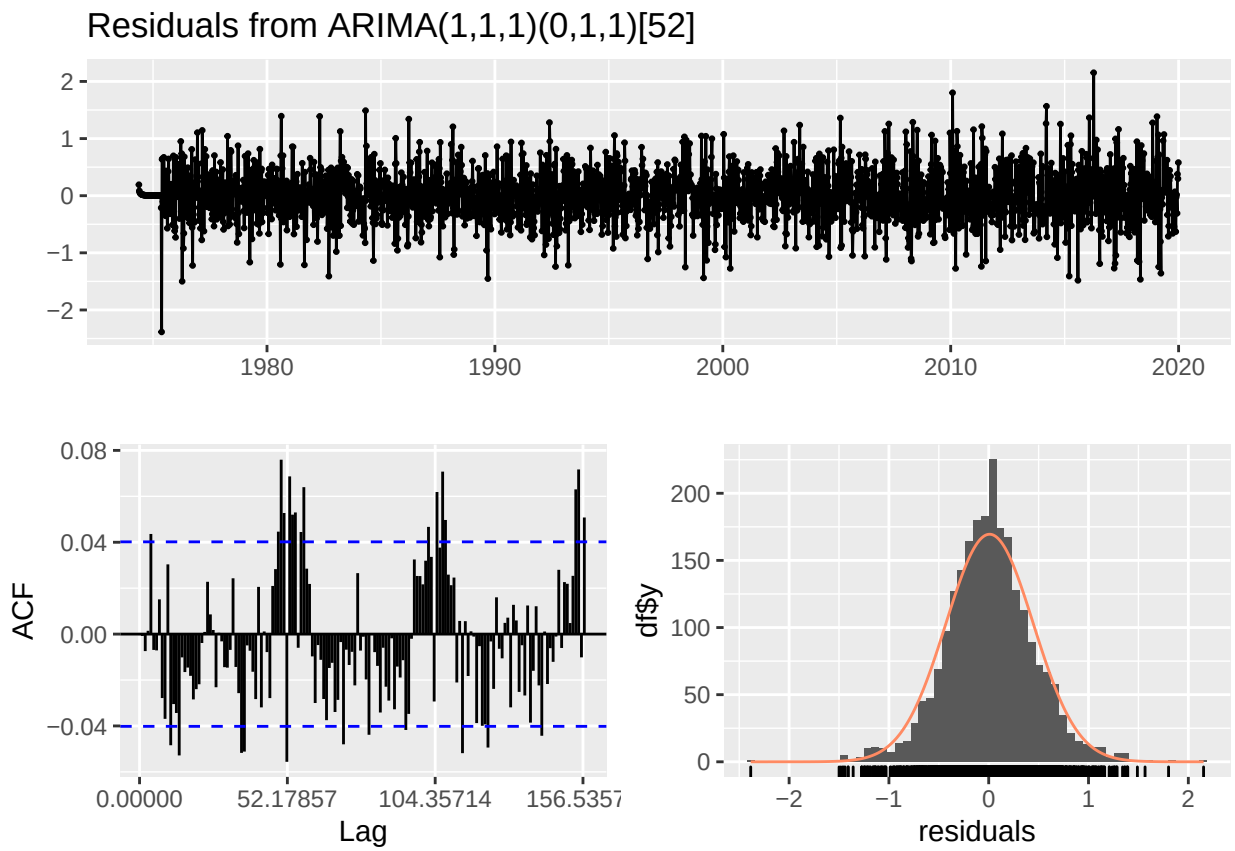

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,2)(1,1,2)[12]
## Q* = 1000.3, df = 98.357, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 104.357142857143
```

```
checkresiduals(model3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,2)(0,1,1)[52]
## Q* = 211.85, df = 100.36, p-value = 5.778e-10
##
## Model df: 4.   Total lags used: 104.357142857143
```

```
checkresiduals(model4)
```

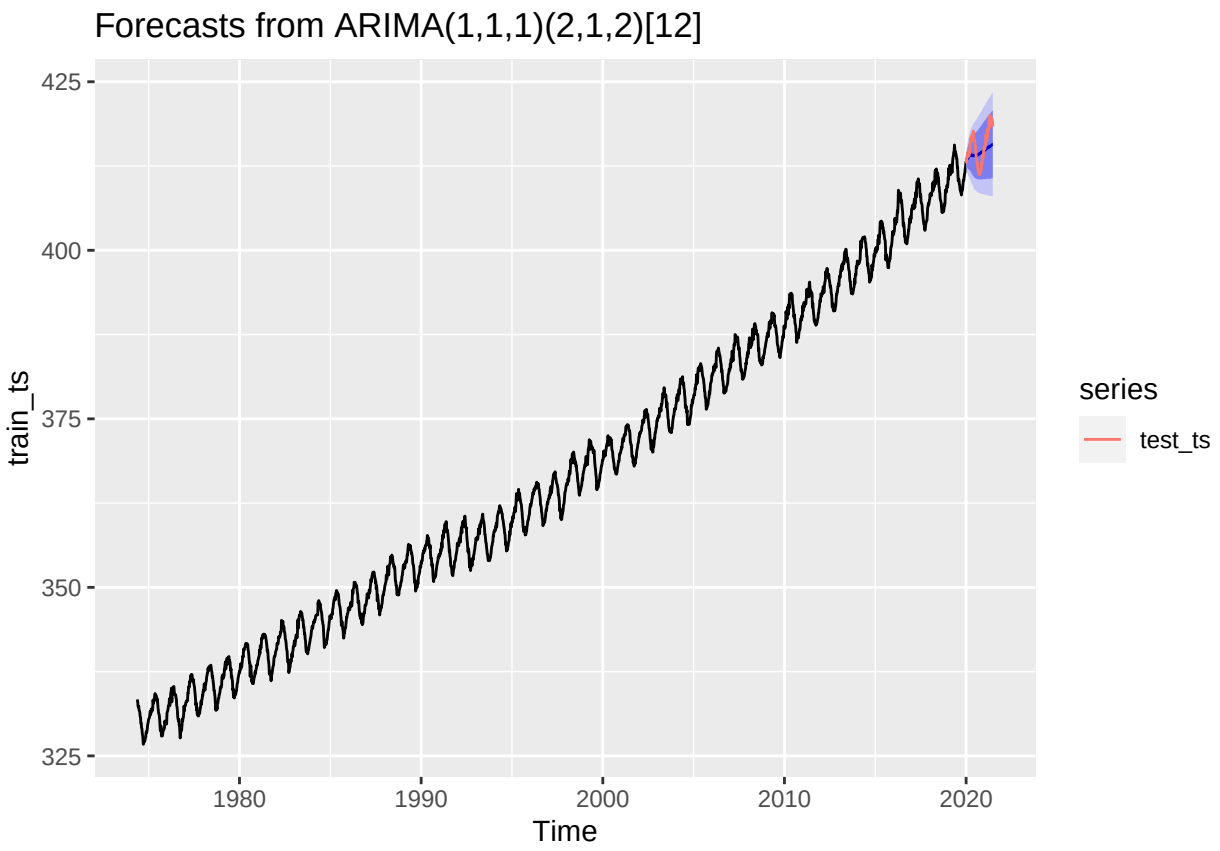



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,1,1)[52]
## Q* = 212.3, df = 101.36, p-value = 7.521e-10
##
## Model df: 3.   Total lags used: 104.357142857143
```

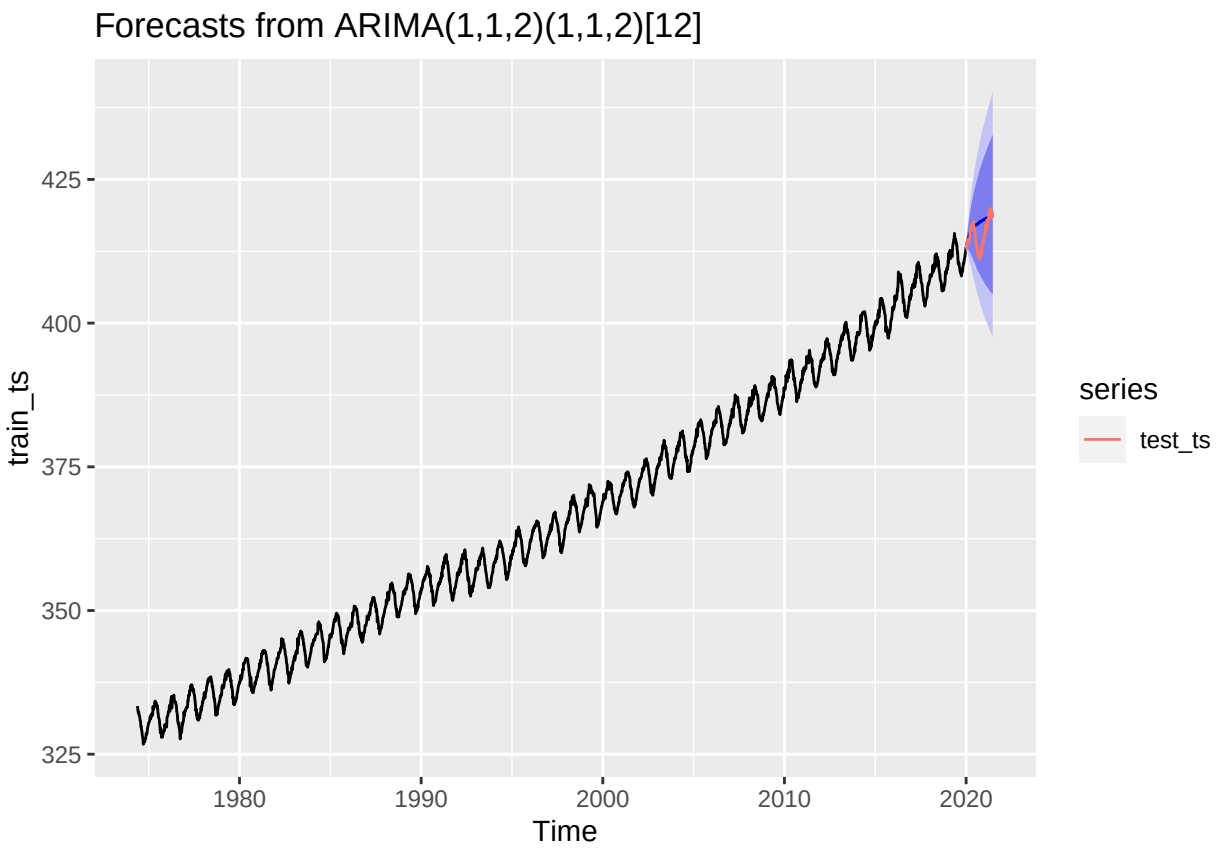
Examining the fitted values and residuals, there doesn't appear to be any drastic differences. The charts of the fitted vs actual values, when examined visually, also do not show major differences. However, models 3 and 4, which had the lowest AIC, also show slightly better performance in terms of having smaller residuals. The charts below also show the forecasted values for the test set.

Model 4 had the smallest AIC and smaller prediction intervals for the predicted values, so we continue our analysis using Model 4.

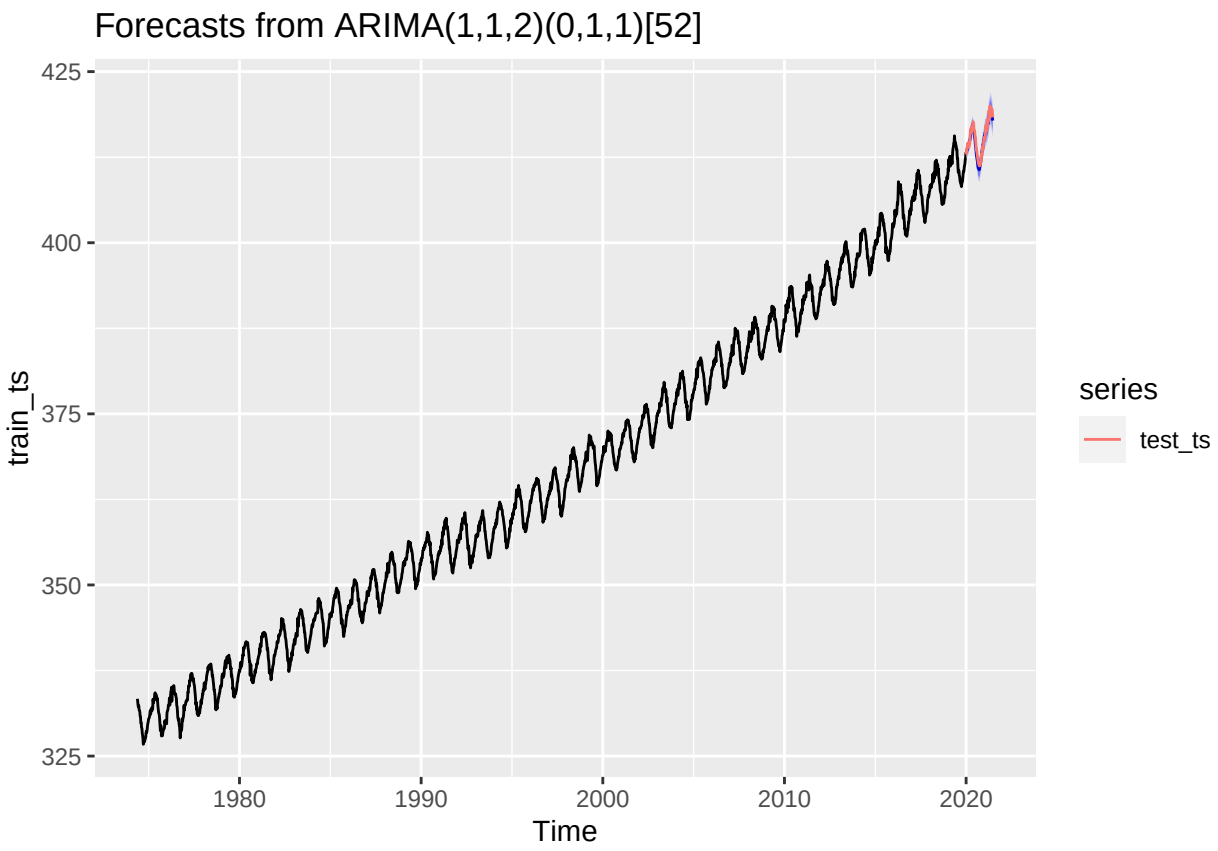
```
# Plot forecast for Model 1 vs test set
forecast1 <- model1 %>%
    forecast(h = 77)
autoplot(forecast1) + autolayer(test_ts)
```



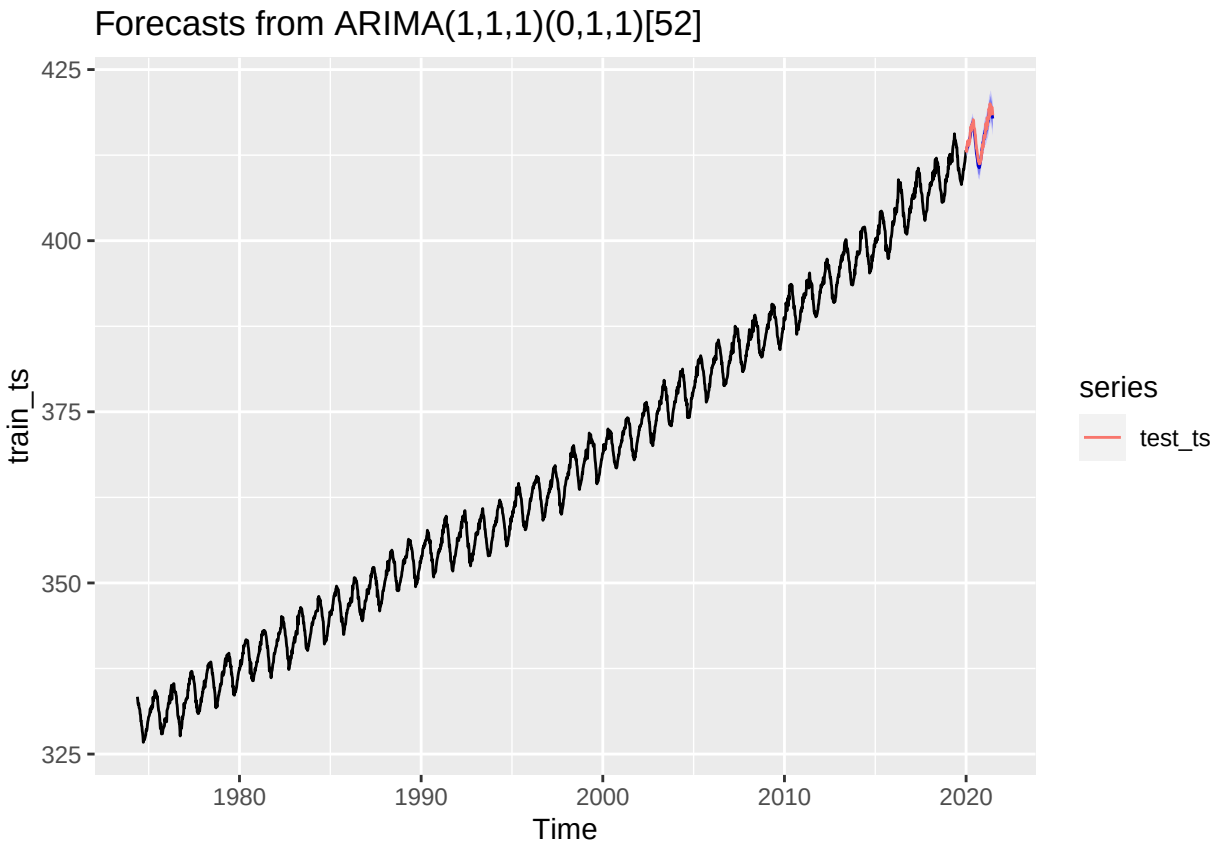

```
# Plot forecast for Model 2 vs test set
forecast2 <- model2 %>%
    forecast(h = 77)
autoplot(forecast2) + autolayer(test_ts)
```




```
# Plot forecast for Model 3 vs test set
forecast3 <- model3 %>%
    forecast(h = 77)
autoplot(forecast3) + autolayer(test_ts)
```




```
# Plot forecast for Model 4 vs test set
forecast4 <- model4 %>%
    forecast(h = 77)
autoplot(forecast4) + autolayer(test_ts)
```




```
# Forecast out to a longer period
forecast <- model4 %>%
    forecast(h = 4300)
model_predictions <- as.data.frame(forecast)
model_predictions <- setDT(model_predictions, keep.rownames = TRUE)[]
names(model_predictions) <- c("date", "point_forecast", "lo_80",
    "hi_80", "lo_95", "hi_95")
```

```
# Prediction for when atmospheric CO2 is expected to reach
# 450 parts per million Considering prediction intervals as
# well as the point estimate
format(date_decimal(as.numeric(model_predictions[model_predictions$hi_95 >=
    450][1]$date)), "%d-%m-%Y")
```

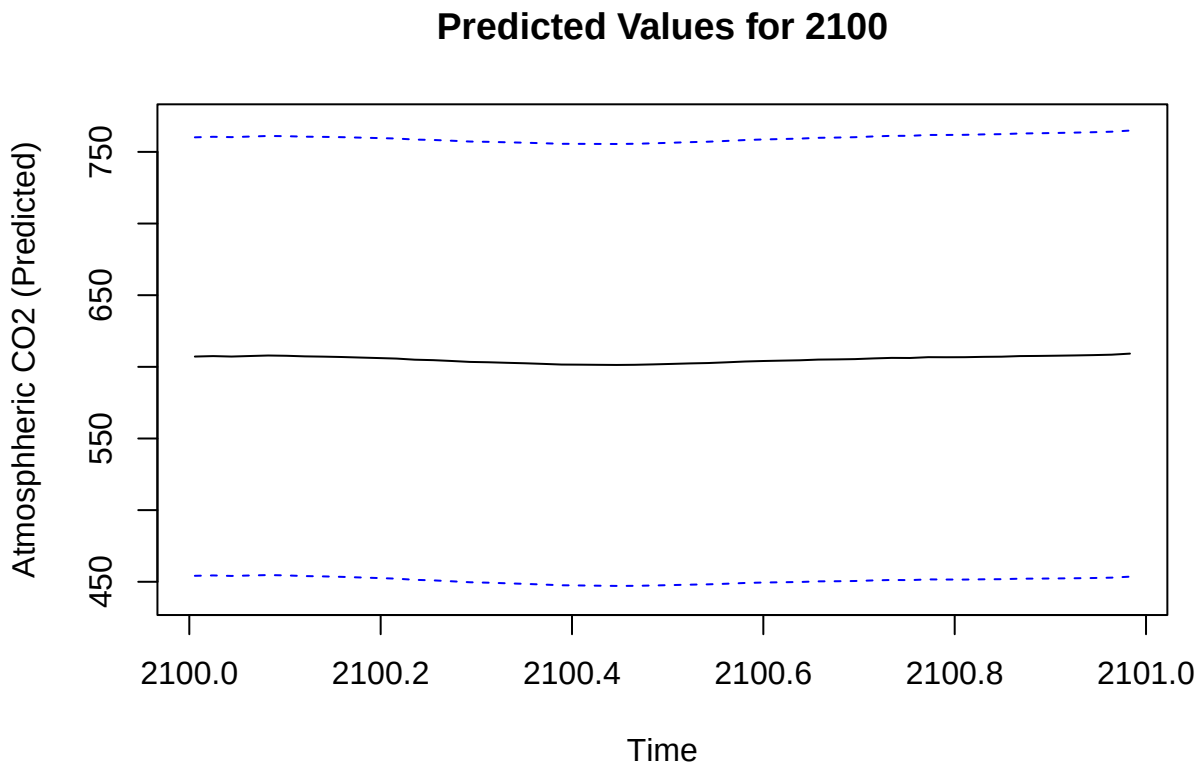
```
## [1] "10-03-2030"
```

```
format(date_decimal(as.numeric(model_predictions[model_predictions$point_forecast >=
    450][1]$date)), "%d-%m-%Y")
```

```
## [1] "02-04-2034"
```

October 3, 2030 is when the atmospheric CO2 level of 450 enters the 95% prediction interval, but the point at which the point estimate reaches 450 is February 4, 2034.

```
# Subset to 2100 data
forecast_2100 <- as.data.frame(window(forecast$mean, start = 2100,
    end = 2101))
forecast_2100_lower <- as.data.frame(window(forecast$lower, start = 2100,
    end = 2101)[, 2])
forecast_2100_upper <- as.data.frame(window(forecast$upper, start = 2100,
    end = 2101)[, 2])


# Plot predicted values for 2100 with 95% prediction
# intervals
plot(forecast_2100$x, main = "Predicted Values for 2100", ylim = c(440,
    770), ylab = "Atmosphheric CO2 (Predicted)", pch = 19)

lines(forecast_2100_lower$x, col = "blue", lty = 2)
lines(forecast_2100_upper$x, col = "blue", lty = 2)
```




```
summary(forecast_2100$x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   601.3   603.2   605.8   605.3   607.2   609.2
```

As seen from the plot of predicted values for the year 2100, the predicted point estimates for atmospheric CO2 in 2100 range from 601.3 to 609.2. However, the 95% predicted intervals range from 447.1 to 765.9, indicating very low confidence in the accuracy of the predictions.