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

Group 3: Maurice Williams, Mikra Walekova, and David Linnard Wheeler

## The Keeling Curve

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

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

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

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

### Import libraries

```
library(astsa)
library(car)
library(data.table) # to enable creation and coercion of data tables
library(fable) # for forecasting time series
library(feasts)
library(forecast)
library(fpp2)
library(fpp3)
library(GGally) # for scatterplot matrices
library(ggExtra)
library(ggfortify) # for visualization
library(ggplot2) # for plotting
```

```
library(gridExtra) # for arranging multiple plots together
library(lindia) # for diagnostic plots via gg_diagnose()
# library(lubridate)
library(plotly)
library(tidyr)
library(tidyverse)
library(tsibble) # for time series graphic and analyses
# library(readr)
# library(xts)
library(timetk)
```

### Part 1 (3 points)

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

### Load and inspect data

```
# Load data
head(co2)
```

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

```
# Inspect structure
str(co2)
```

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

```
# Coerce data into tsibble object
co2 = as_tsibble(co2)

# Sanity check
str(co2)
```

```
## 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 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
```

```
# Are there any missing values/NAs?
co2[!complete.cases(co2),]
```

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

```
# Annual average data
co2AA <- co2 %>% index_by(Year = ~ year(.)) %>%
   summarise(AnnualAverage = mean(value))
```

The data consists of two columns and index comprised of month and year in the 'YYYY Mmm' format and a column representing atmospheric carbon dioxide concentrations measured in ppm (parts per million), equivalent to 0.0001%. There are 468 records, representing 38 year of monthly data (January 1959 to December 1997). There are no missing values in the `co2` data set.

### Exploratory Data Analysis - co2 series

#### Data Review $ STL decomposition

```
# STL Decomposition
co2.stl <- co2 %>%
  model(STL(value))
# Components: seasonal, trend, remainder
# components(co2.stl)

# Plot yearly data
p=co2 %>%
  autoplot(value, color='gray') +
  autolayer(components(co2.stl),
            season_adjust, color='steelblue') +
  ylab(expression("CO"[2]~ " (ppm)")) +
  labs(x="Time (1959-1997)") +
    scale_y_continuous(name = expression(paste(" CO"[2]~"(ppm)")))+
  theme_classic() + geom_point(col="transparent")
ggMarginal(p, type="histogram",
           margins = "y",
           col="grey",
           fill = "grey83")
```

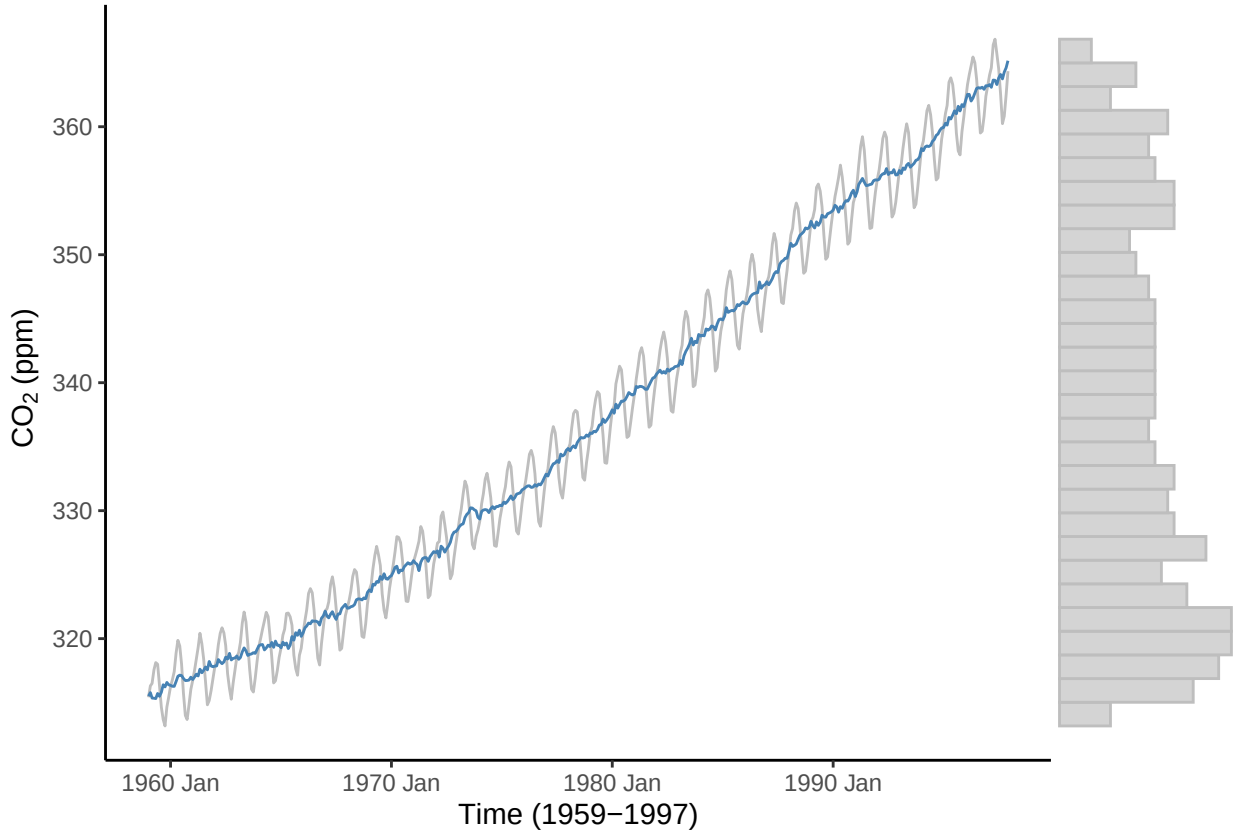



Figure 1: CO2 as function of time

There are two patterns of interest: an overall upward trend, and a cyclical trend. The subsequent sections of this document first explore the trend and seasonality components. The series appears nearly uniformly distributed with a slight negative skew.

To further understand the time series components, we decomposed the series into the seasonal & trend & remainder components:

```
# Seasonal and Trend decomposition using Loess (STL)
co2 %>%
  model(STL(value ~ trend(window=13) + season(),
    robust = TRUE)) %>%
  components() %>%
  autoplot() +
  labs(x="Time (1959-1997)") +
  xlab('Time') +
  theme_minimal()
```

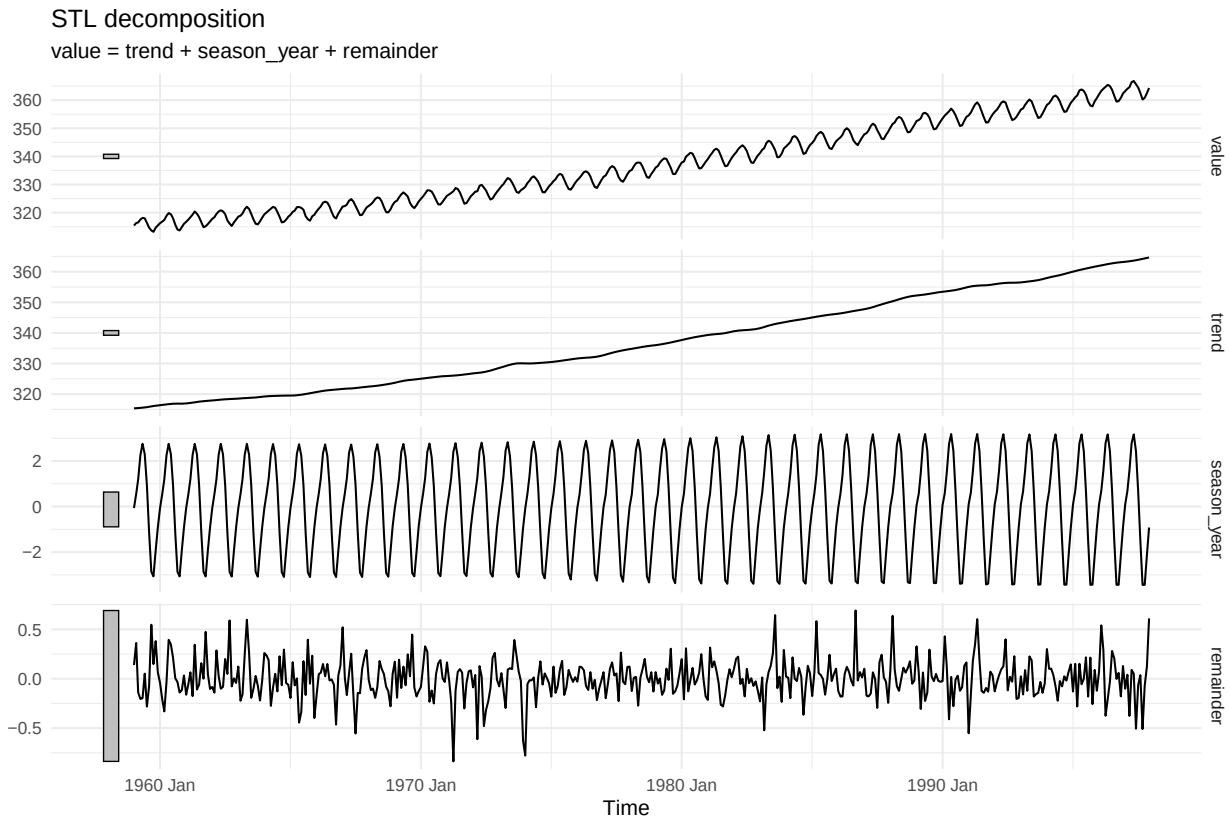



Figure 2: CO2 Seasonal and Trend Decomposition

**Trend**

```r
# Overall trend by fitting a straight line to the data
ggplot(co2, aes(x = index, y = value)) +
  geom_line() +
  theme_minimal() +
  stat_smooth(method = "lm", se = FALSE, aes(col = "red")) +
  stat_smooth(method = "lm",formula = y ~ x + I(x^2), se = FALSE, aes(col = "blue")) +
  stat_smooth(method = "lm",formula = y ~ x + I(x^2) + I(x^3), se = FALSE,
              aes(col = "green")) + scale_color_identity(name = "Model fit",
                                breaks = c("red", "blue", "green"),
                                labels = c("Linear", "Quadratic", "Cubic"),
                                guide = "legend")
```

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

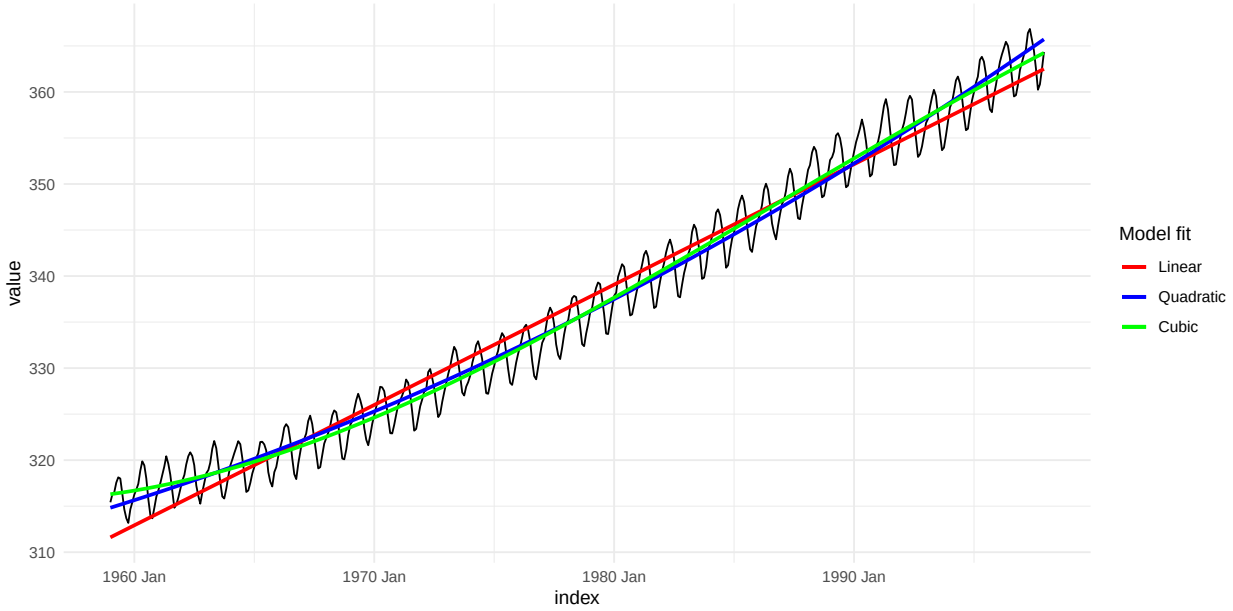



Figure 3: CO2 Trend Analysis

Linear, quadratic and cubic regression models are fitted using the standard regression analysis procedure to evaluate the trend visually.

$$Linear\ Trend: x_t = \alpha_0 + \alpha_1 \times t + z_t$$

$$Quadratic\ Trend: x_t = \alpha_0 + \alpha_1 \times t + +\alpha_2 \times t + \alpha_3 \times t^2 + z_t$$

$$Cubic\ Trend: x_t = \alpha_0 + \alpha_1 \times t + \alpha_2 \times t^3 + z_t$$

The overall trend is appears to be sligthly convex with a minor "hump" around the 1980 time period. The quadratic and cubic trend seem to fit the data better than the linear trend. To understand which trend is more appropriate the residuals of linear models will be examined in a later section of this document.

**Trend Growth Rates** (consider expressing longer-run growth rates as annualized averages)

```
# Annual average data
co2AA <- co2 %>% index_by(Year = ~ year(.)) %>%
   dplyr::summarise(AnnualAverage = mean(value))

# Plot annualized average trend
p1<-co2AA %>% autoplot(AnnualAverage) +
  theme_classic() +
  scale_y_continuous(name = expression(paste("Annualized average CO"[2]~"(ppm)")))+
  labs(x="Time (1959-1997)")

# Subset df by January
p2<-co2[grep("Jan", co2$index),] %>%
    # define yt_1 as yt - yt-1
    dplyr::mutate(Yt_1 = value - lag(value),
      # define aa as rolling average
      AnnualizedAverage = zoo::rollmean(Yt_1, k = 10,
                                        fill = NA)) %>%
  # Plot
  autoplot(AnnualizedAverage) +
  theme_classic() +
  theme(axis.text.x = element_text(angle=90))+
  scale_y_continuous(name = expression(paste("CO"[2]~"annualized 10 year growth rate")))+
  labs(title=expression("Annual trend growth rate"),
     x="Time (1959-1997)")

grid.arrange(p1, p2, ncol=2)
```

```
## Warning: Removed 10 row(s) containing missing values (geom_path).
```

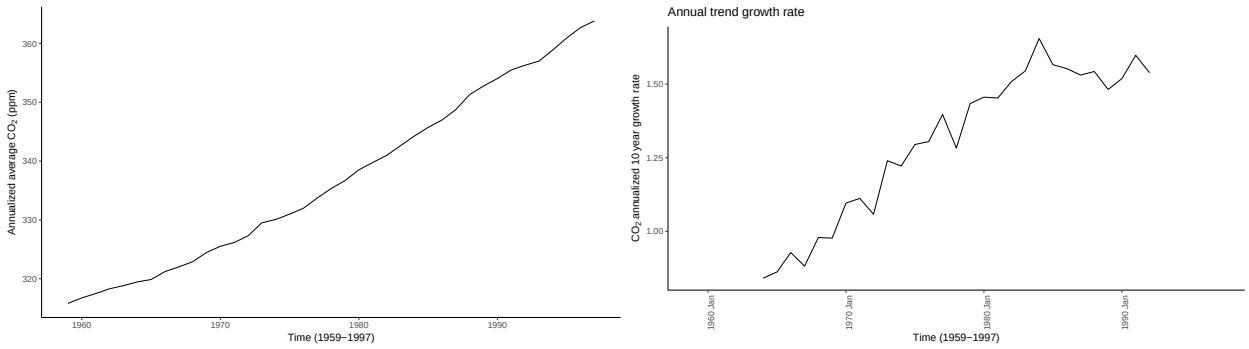



Figure 4: Trend Growth Rates

While the annual $CO_2$ ppm average seems to be growing at a constant rate from figure above it can be observed that the annual trend growth rate stalled between the years 1980 and 1990. This observation explains why the simple linear trend in previous figure does not look like a best fit.

### Seasonal Variation

```
ggplot(data = co2 %>% subset(year(index) == 1985), aes(index, value)) +
 geom_line() #+
```

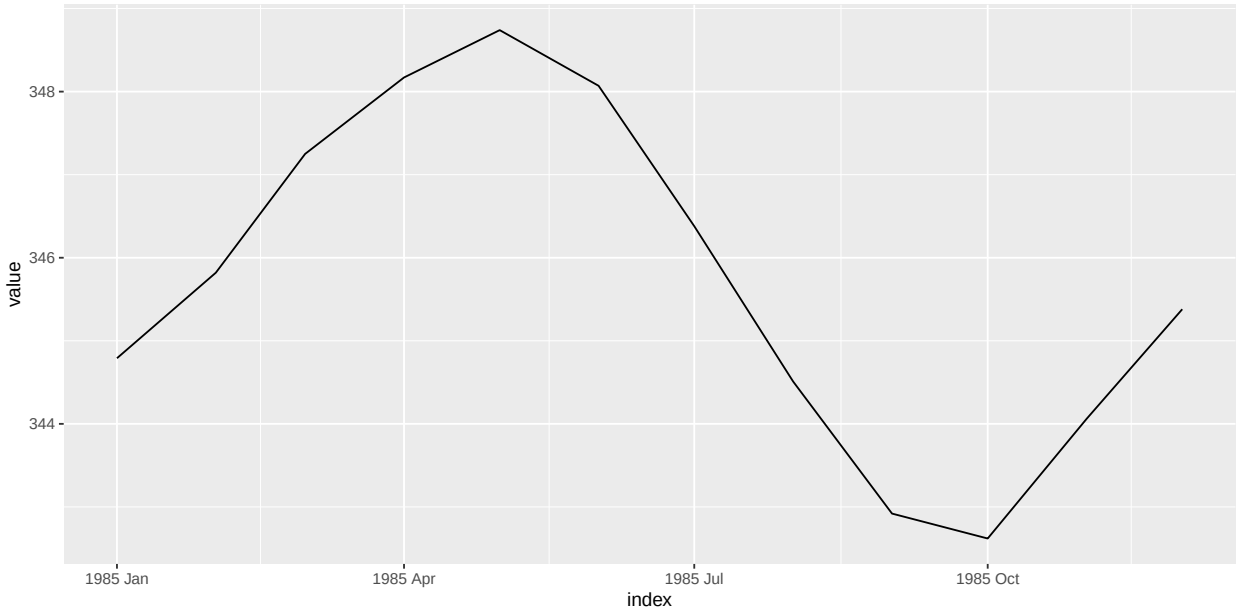



Figure 5: CO2 Seasonality Component

```
  theme_minimal() +
  xlab('Time') +
  ylab('CO2 Concentration PPM') +
  labs(title=expression("CO"[2]~"Seasonality component"),
     x="Time (1959-1997)") +
 ggtitle('Carbon Dioxide Concentration in 1985')
```

```
# Plot time series
p1 <- co2 %>% gg_subseries(value) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=90))+
  scale_y_continuous(name = expression(paste("CO"[2]~"(ppm)")))+
  labs(title=expression("CO"[2]~"as a function of time by month"),
       x="Monthly seasonal variation (1959 - 1997)")

p2 <- co2 %>% gg_season(value, period = "year") +
  theme_minimal() +
  scale_y_continuous(name = expression(paste("CO"[2]~"(ppm)")))+
  labs(title=expression("CO"[2]~"as a function of time by season"),
       x="Time (1959 - 1997)")
```

```
grid.arrange(p1, p2, ncol=2)
```

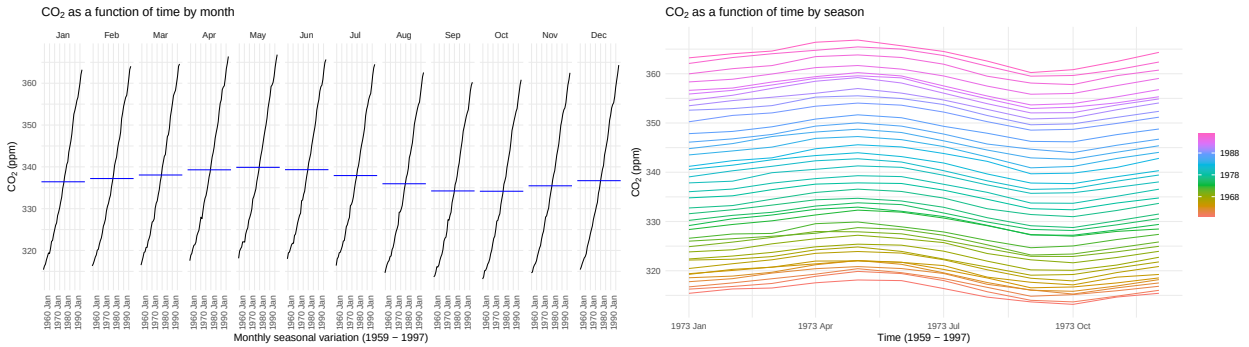
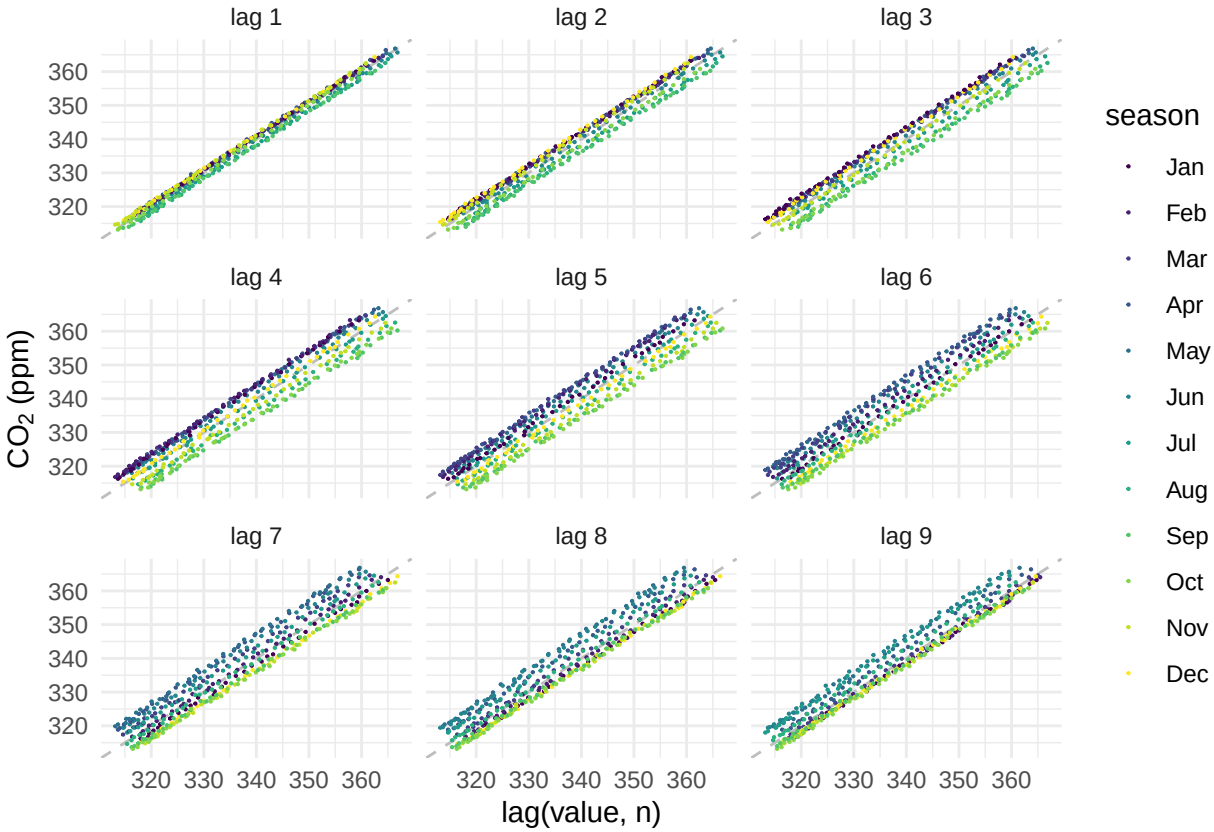



Figure 6: CO2 Seasonality Analysis

The data contains a seasonality component, where the $CO_2$ ppm climbs until the spring commences and then decreases continuously until the end of summer when it begins to climb again. This cycle is explained by the land mass in northern versus southern hemisphere and the photosynthesis. The seasonality is further explored in subsequent sections after regression model fitting.

### Irregular Elements

```
# Plot lag plot
co2 %>% gg_lag(value, geom = "point", size = .1) +
 theme_minimal() +
 scale_y_continuous(name = expression(paste("CO"[2]~"(ppm)")))
```




The data shows a linear pattern, a positive linear trend (i.e. going upwards from left to right) is suggestive of positive autocorrelation.

### Other

Is there any linear correlation between lagged values of $CO_2$? What about when we control for correlations between lags $< k$?

```
# Plot ACF
acf<-co2 %>% ACF(value, lag_max = nrow(co2)) %>% autoplot()+
  theme_classic() +
  theme(axis.text.x = element_text(angle=90))+
 scale_y_continuous(name = expression(paste("acf")))+
 labs(title=expression("autocorrelation of CO"[2]~"(ppm)"),
      x="lag")

# Plot PACF
pacf<-co2 %>% PACF(value, lag_max = nrow(co2)) %>% autoplot()+
  theme_classic() +
  theme(axis.text.x = element_text(angle=90))+
 scale_y_continuous(name = expression(paste("acf")))+
 labs(title=expression("partial autocorrelation of CO"[2]~"(ppm)"),
      x="lag")
```

```
# All together
grid.arrange(acf, pacf, ncol= 2)
```

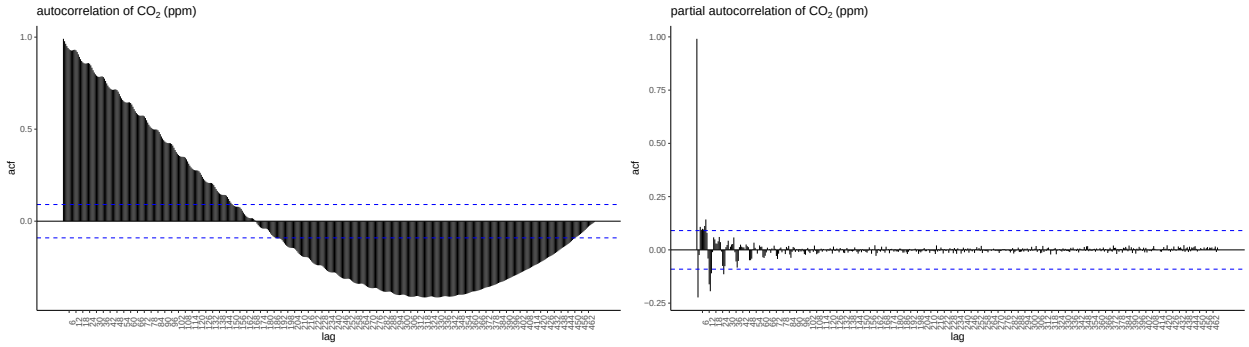



Figure 7: ACF and PACF

The ACF plot describes how well the present value of the series is related with its past values, the ACF indicates about 12 years of correlation between the $CO_2$ values. However if the residuals are examined and the only unexplained correlation is taken into account - the PACF - indicates 1-2 year period.

Further analysis is carried our later in this document on the residual data once seasonality and trend are removed after fitting a suitable regression model.

**Part 2 (3 points)**

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

## Linear time trend model residuals

$$Linear\ Trend\ Model: x_t = \alpha_0 + \alpha_1 \times t + z_t$$

```
# Aggregate residuals by year
co2$Month <- month(co2$index, label=TRUE)
co2$Date <- co2$index
co2.row <- co2 %>% dplyr::mutate(index = dplyr::row_number())
co2 = as_tsibble(co2, index = index)

model_diagnostic = function(model) {
  plot(model)
  residualPlots(model)
}
```

```r
# linear time trend
lin.trend <- co2 %>%
  model(TSLM(value ~ trend())) %>%
  report()
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -6.039885 -1.947575 -0.001671  1.911271  6.514852
##
## 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: < 2.22e-16
```

```r
# plotting residuals
lin.trend %>% gg_tsresiduals()
```

```r
# Generate a model object from the regression fit
linear.model <- lm(value ~ index, co2)
co2$residuals.linear <- linear.model$residuals

# Plot the residuals from the model object
p1 <- ggplot(co2, aes(x = Date, y = residuals.linear)) +
  theme_minimal() +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE, col='red')

p2 <- ggplot(co2, aes(x = Month, y = residuals.linear)) +
  theme_minimal() +
  geom_boxplot()

grid.arrange(p1, p2, ncol=2)
```

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

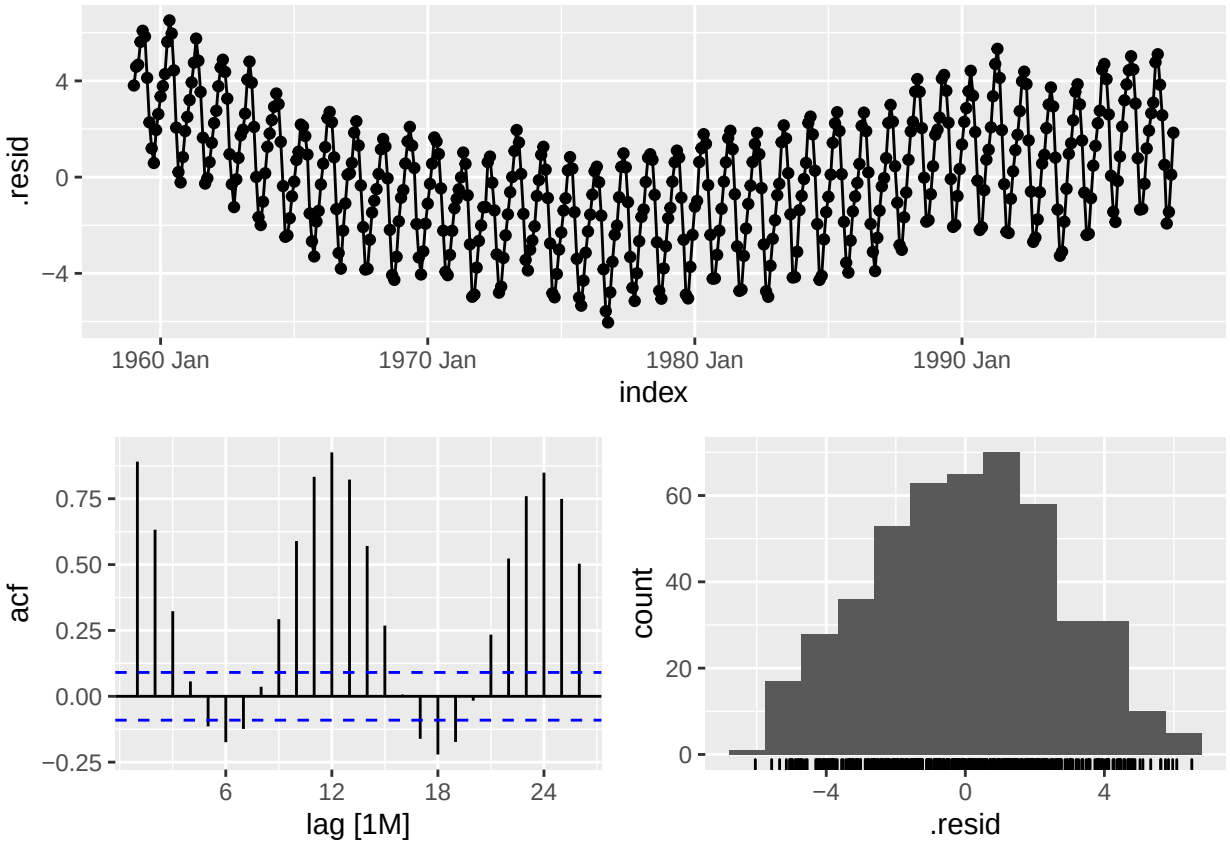
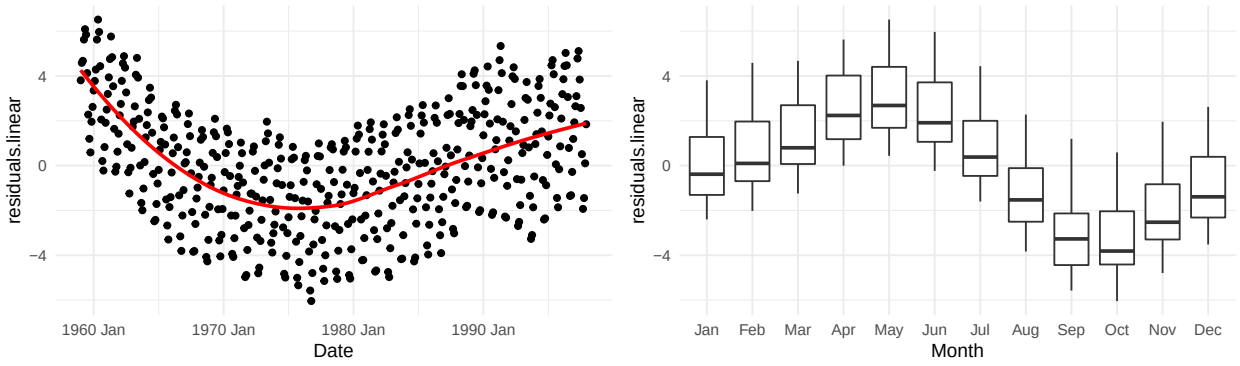



Figure 8: Linear Trend Residual Analysis - part1




By leveraging simple regression, a linear trend is subtracted from the data and residuals are visualised in the resulting plots clearly display a lack of fit. The residuals clearly visualise seasonality that is consistent with the observation described in the previous section.

### Quadratic time trend model residuals

$$Quadratic\ Trend\ Model: x_t = \alpha_0 + \alpha_1 \times t + \alpha_2 \times t^2 + z_t$$

```
# quadratic time trend
quad.trend <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2))) %>%
  report()
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.148e+02  3.039e-01 1035.65   <2e-16 ***
## trend()      6.739e-02  2.993e-03   22.52   <2e-16 ***
## I(trend()^2) 8.862e-05  6.179e-06   14.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.182 on 465 degrees of freedom
## Multiple R-squared: 0.9788,  Adjusted R-squared: 0.9787
## F-statistic: 1.075e+04 on 2 and 465 DF, p-value: < 2.22e-16
```

```
# plotting residuals
quad.trend %>% gg_tsresiduals()
```

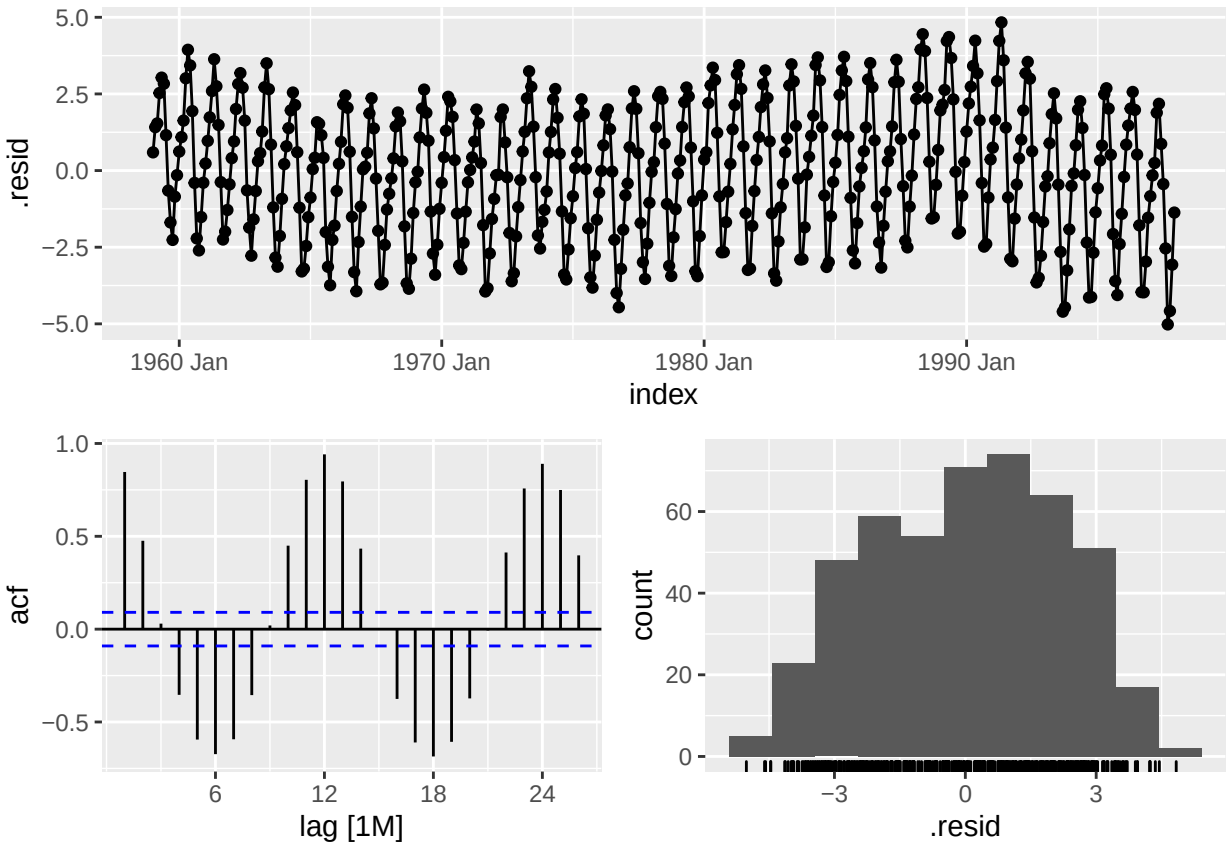



Figure 9: Quadratic Trend Residual Analysis - part1

```r
# Generate a model object from the regression fit y ~ x + I(x^2)
quadratic.model <- lm(value ~ index + I(index^2), co2.row)
co2$residuals.quadratic <- quadratic.model$residuals

# Plot the residuals from the model object
p1 <- ggplot(co2, aes(x = index, y = residuals.quadratic)) +
  theme_minimal() +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE, col='blue')

p2 <- ggplot(co2, aes(x = Month, y = residuals.quadratic)) +
  theme_minimal() +
  geom_boxplot()

grid.arrange(p1, p2, ncol=2)
```

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

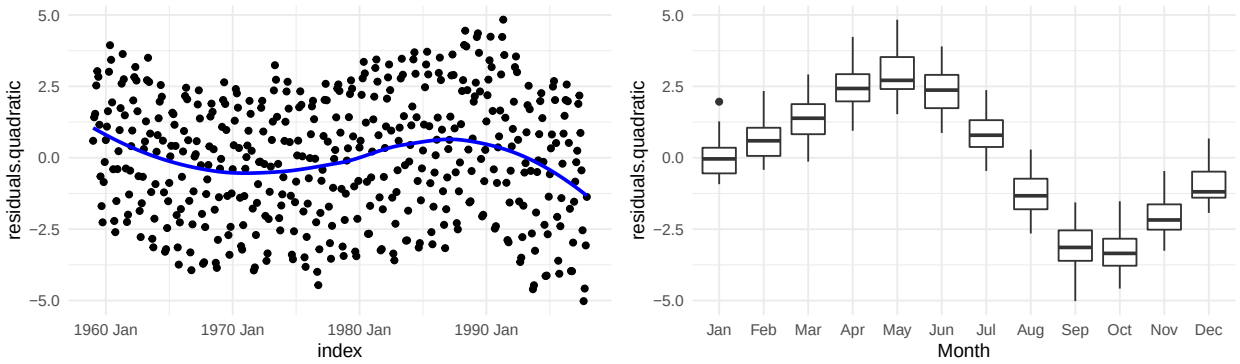



Figure 10: Quadratic Trend Residual Analysis - part2

The spread of residuals and the fit have improved in comparison to the linear model however the lack of consideration for seasonality in this model is clearly visualised by the obvious seasonality patter in the residuals.

### Cubic time trend model residuals

$$Cubic\ Trend\ Model: x_t = \alpha_0 + \alpha_1 \times t + \alpha_2 \times t^2 + \alpha_2 \times t^3 + z_t$$

```r
# quadratic time trend
cube.trend <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2)+ I(trend()^3))) %>%
  report()
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5786 -1.7299  0.2279  1.8073  4.4318
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.163e+02  3.934e-01 804.008  < 2e-16 ***
## trend()       2.905e-02  7.256e-03   4.004 7.25e-05 ***
## I(trend()^2)  2.928e-04  3.593e-05   8.149 3.44e-15 ***
## I(trend()^3) -2.902e-07  5.036e-08  -5.763 1.51e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.11 on 464 degrees of freedom
## Multiple R-squared: 0.9802,  Adjusted R-squared: 0.9801
## F-statistic:  7674 on 3 and 464 DF, p-value: < 2.22e-16
```

```
# plotting residuals
cube.trend %>% gg_tsresiduals()
```

```
# Generate a model object from the regression fit y ~ x + I(x^2)
cube.model <- lm(value ~ index + I(index^2) + I(index^3), co2.row)
co2$residuals.cube <- cube.model$residuals

# Plot the residuals from the model object
p1 <- ggplot(co2, aes(x = index, y = residuals.cube)) +
  theme_minimal() +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE, col='blue')

p2 <- ggplot(co2, aes(x = Month, y = residuals.cube)) +
  theme_minimal() +
  geom_boxplot()

grid.arrange(p1, p2, ncol=2)
```

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

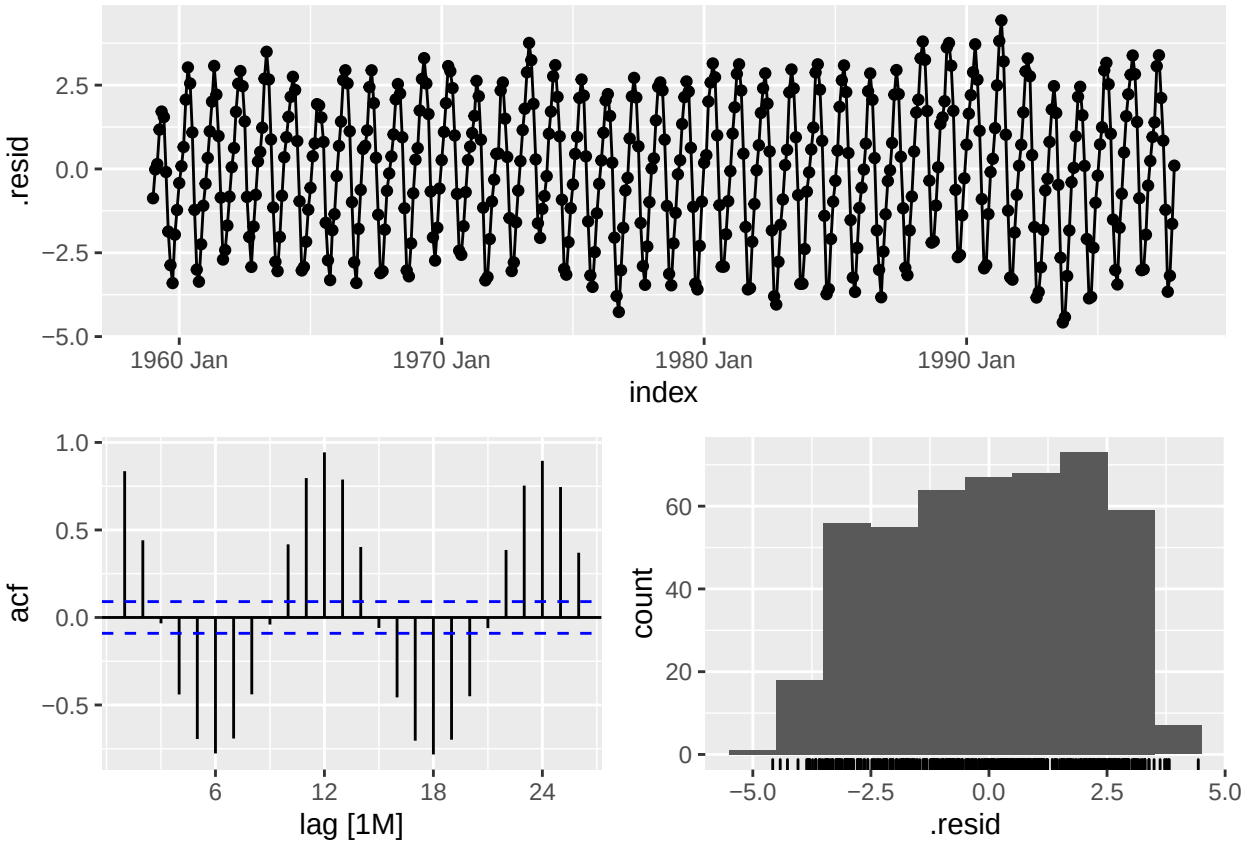
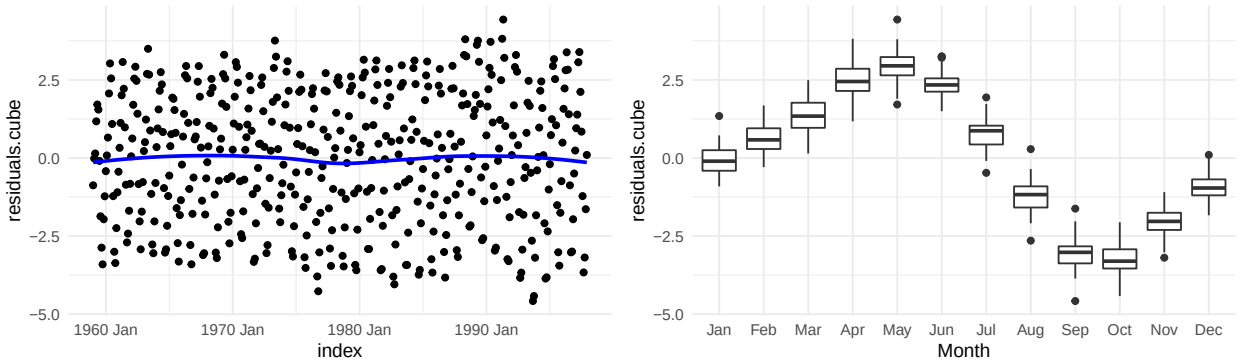



Figure 11: Cubic Trend Residual Analysis - part1




Despite the visible seasonality it can be observed that the cubic model fit is substantially better than the quadratic model.

### Logarithmic transformation of the data

Log transformations are generally more suitable for time series with exponentially growing trend. The exploratory data analysis in previous section has confirmed that this is not the case for the $CO_2$ data.

Nevertheless a log transform model is fitted and reviewed below:

$$Logarithmic\ Trend\ Model: log(x_t) = \alpha_0 + \alpha_1 \times t + z_t$$

```
# quadratic time trend
expo.trend <- co2 %>%
  model(TSLM(log(value) ~ trend())) %>%
  report()
```

```
## Series: value
## Model: TSLM
## Transformation: log(.x)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0172650 -0.0056145  0.0002764  0.0053760  0.0187770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.744e+00  6.829e-04  8410.5   <2e-16 ***
## trend()     3.224e-04  2.523e-06   127.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.007375 on 466 degrees of freedom
## Multiple R-squared: 0.9722,  Adjusted R-squared: 0.9722
## F-statistic: 1.633e+04 on 1 and 466 DF, p-value: < 2.22e-16
```

```r
# plotting residuals
expo.trend %>% gg_tsresiduals()
```

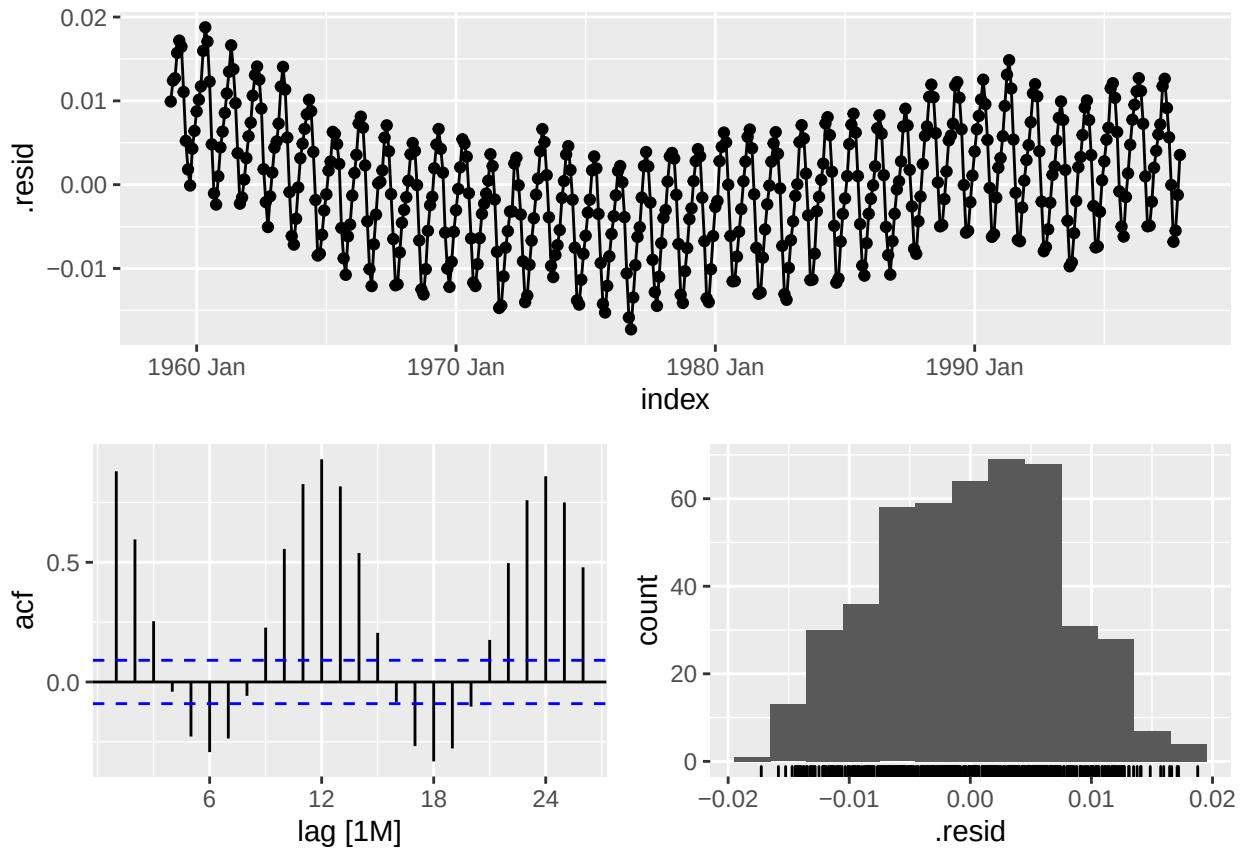
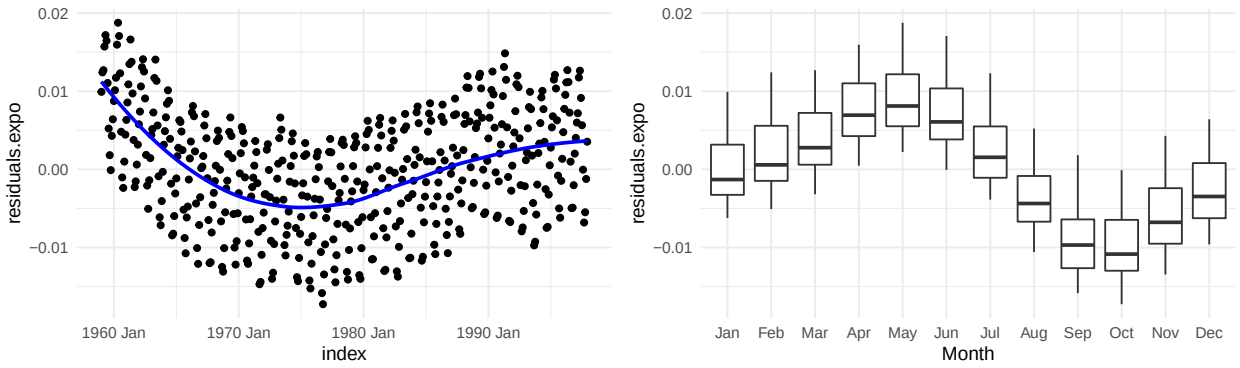



Figure 12: Exponential Trend Residual Analysis - part1

```r
# Generate a model object from the regression fit y ~ x + I(x^2)

expo.model <- lm(formula = log(value) ~ index,data = co2.row)
co2$residuals.expo <- expo.model$residuals

# Plot the residuals from the model object
p1 <- ggplot(co2, aes(x = index, y = residuals.expo)) +
  theme_minimal() +
  geom_point() +
  stat_smooth(method = "loess", se = FALSE, col='blue')

p2 <- ggplot(co2, aes(x = Month, y = residuals.expo)) +
  theme_minimal() +
  geom_boxplot()

grid.arrange(p1, p2, ncol=2)
```

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




As per expectation the log transformation, given that the trend is not exponentially growing did not result in better fit.

```
data.frame(
"linear" = deviance(linear.model),
"quadratic" = deviance(quadratic.model),
"cubic" = deviance(cube.model),
"log-transformed quadratic"= sum(co2$value - exp(expo.model$fitted.values))^2)
```

```
##    linear quadratic    cubic log.transformed.quadratic
## 1 3196.17  2214.454 2066.558                    22.184
```

The deviance metric indicates that the cubic trend is a better fit than the quadratic trend, therefore in subsequent section we leverage cubic terms.

### Polynomial time trend model with incorporates seasonal dummy variables

$$Polynomial\ Trend\ Model\ incl.\ Seasonality: \ x_t = \alpha_0 + \alpha_1 \times t + \alpha_2 \times t^2 + \alpha_2 \times t^3 + s_t + z_t$$

```
# polynomial with seasonals
poly.model <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2) + I(trend()^3)+ season() )) %>%
  report()
```

```
## Series: value
## Model: TSLM
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.5573094 -0.3312054  0.0008042  0.2880086  1.5039635
##
## Coefficients:
##                  Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept)     3.160e+02  1.210e-01 2611.629  < 2e-16 ***
## trend()         3.275e-02  1.740e-03   18.827  < 2e-16 ***
## I(trend()^2)    2.744e-04  8.614e-06   31.850  < 2e-16 ***
## I(trend()^3)   -2.640e-07  1.207e-08  -21.863  < 2e-16 ***
## season()year2   6.700e-01  1.145e-01    5.852 9.32e-09 ***
## season()year3   1.419e+00  1.145e-01   12.390  < 2e-16 ***
## season()year4   2.555e+00  1.145e-01   22.319  < 2e-16 ***
## season()year5   3.040e+00  1.145e-01   26.550  < 2e-16 ***
## season()year6   2.383e+00  1.145e-01   20.811  < 2e-16 ***
## season()year7   8.678e-01  1.145e-01    7.578 2.00e-13 ***
## season()year8  -1.194e+00  1.145e-01  -10.429  < 2e-16 ***
## season()year9  -3.013e+00  1.145e-01  -26.311  < 2e-16 ***
## season()year10 -3.191e+00  1.145e-01  -27.860  < 2e-16 ***
## season()year11 -1.996e+00  1.145e-01  -17.428  < 2e-16 ***
## season()year12 -8.738e-01  1.145e-01   -7.628 1.41e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5056 on 453 degrees of freedom
## Multiple R-squared: 0.9989,  Adjusted R-squared: 0.9989
## F-statistic: 2.92e+04 on 14 and 453 DF, p-value: < 2.22e-16
```

```
# plotting residuals
poly.model %>% gg_tsresiduals()
```

### Seasonality when trend is removed

```
co2.month <- co2 %>%
  group_by(Month) %>%
  mutate(residuals.cube.med = residuals.cube - median(residuals.cube))

ggplot(co2.month, aes(x = Month, y = residuals.cube.med)) +
  theme_minimal() +
  labs(title = 'Analysis of Seasonal component (incl. Median adjustment)') +
  geom_boxplot()
```

```
ggplot(co2.month, aes(x = index, y = residuals.cube.med)) +
  geom_point(col = "grey", cex = .6)  +
  theme_minimal() +
  stat_smooth(se = FALSE, method = "lm") +
  facet_wrap(~ Month, nrow = 1) +
  theme(axis.text.x=element_text(angle=45, hjust = 1.5, vjust = 1.6,  size = 7))
```

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

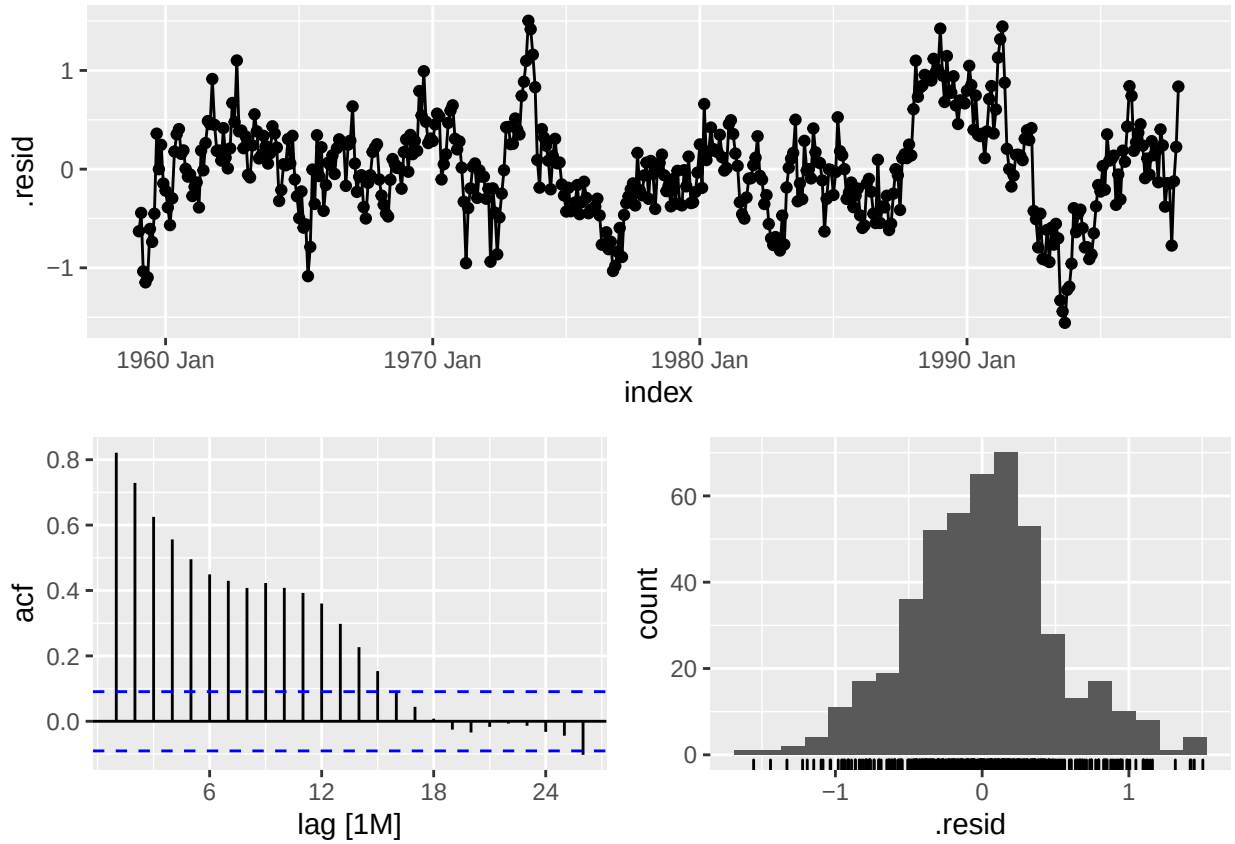



Figure 13: Polynomial Seasonal Model Residual Analysis - part1

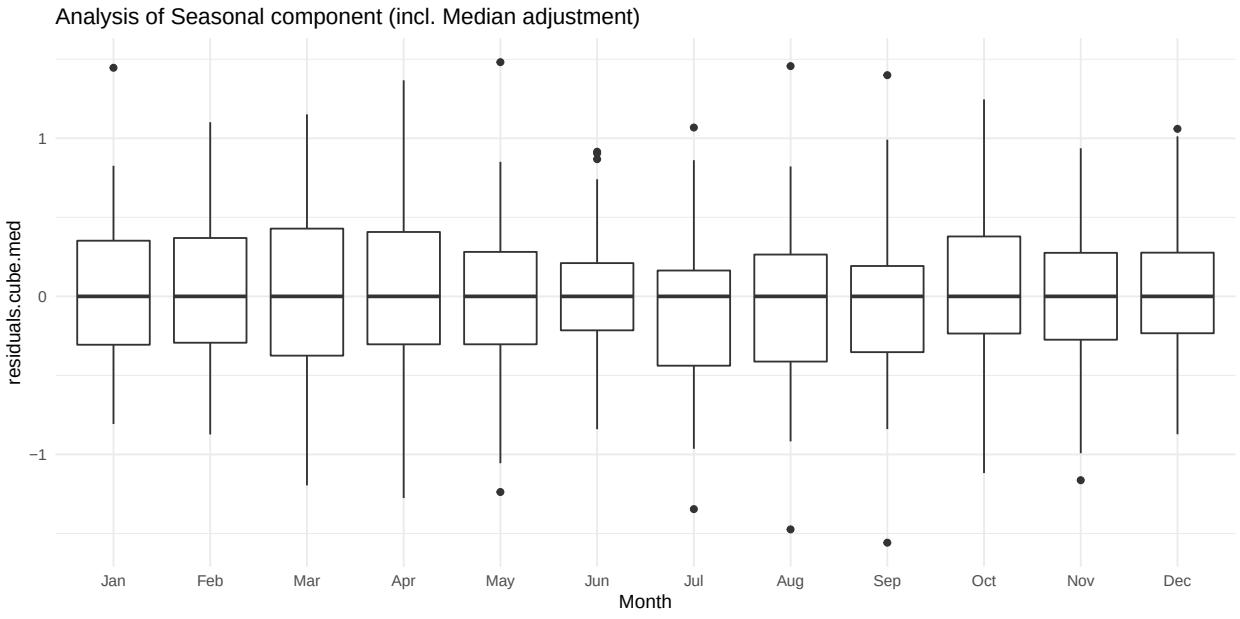



Figure 14: Analysis of Seasonal component (incl. Median adjustment) - part 1

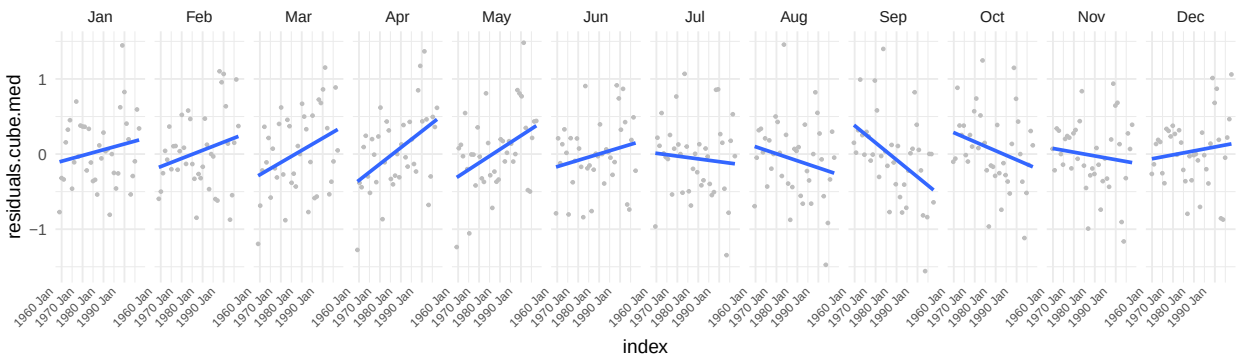



Figure 15: Analysis of Seasonal component (incl. Median adjustment) - part 2

Residual trends in data can be observed where there is an increase in CO2 in first half of each year and a decrease in CO2 for the remainder of the year.

**Examination of remaining data after having accounted for both overall trend and seasonality.**

```
p1<- ggplot(co2.month, aes(x = index, y = residuals.cube.med)) +
  geom_line(col = "grey") +
  theme_minimal() +
  stat_smooth(method = "lm", se = FALSE, span = 0.1 ,
                method.args = list(family = "symmetric", degree = 2))

p2<- ggplot(co2.month, aes(x = index, y = residuals.cube.med)) +
  geom_line(col = "grey") +
  theme_minimal() +
  stat_smooth(method = "loess", se = FALSE, span = 0.1 ,
                method.args = list(family = "symmetric", degree = 2))

grid.arrange(p1, p2, nrow=2)
```

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

## Warning: In lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
##     ...) :
##  extra arguments 'family', 'degree' will be disregarded

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

Upon further examination of the residual data, while we can see that mean expected value is approximately 0, the residual data cannot be described as white noise. Next figure looks at evaluating the model by comparing the data to fitted values.

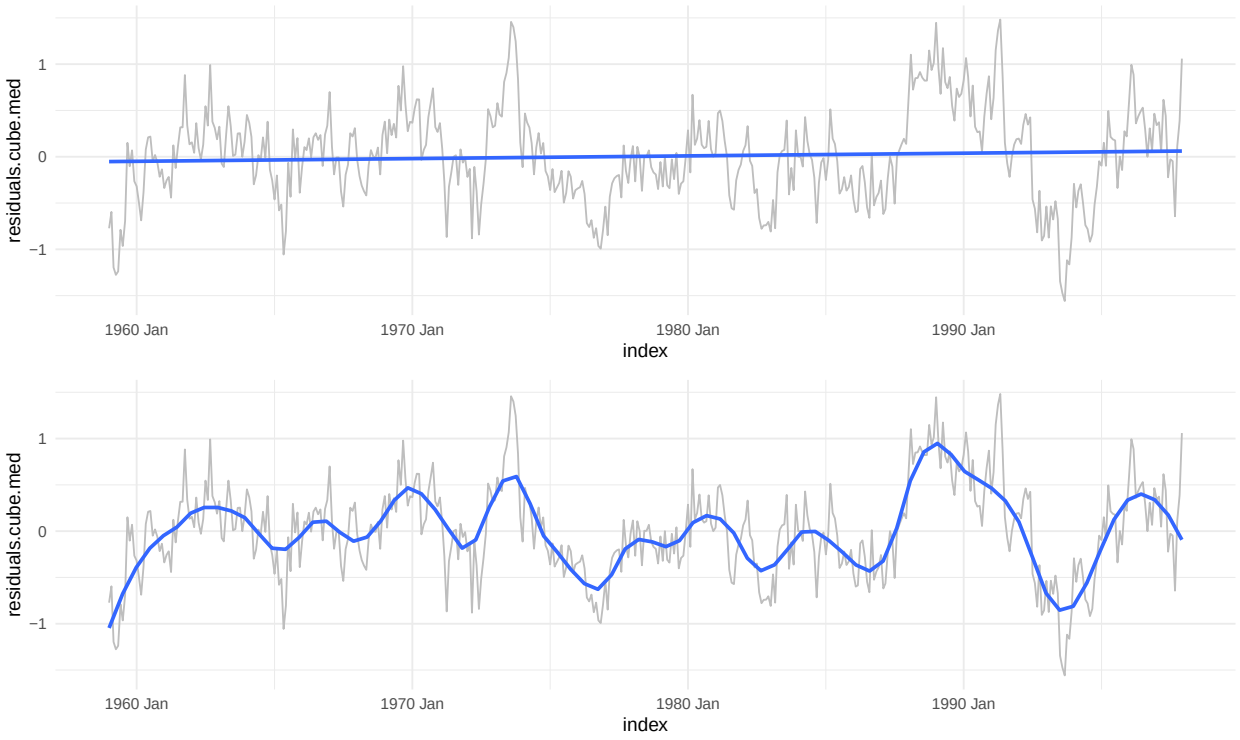



Figure 16: Analysis of Remaning Data

```
augment(poly.model) %>%
  ggplot(aes(x = index)) +
  theme_minimal() +
  geom_line(aes(y = value, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(x = "Index", y = "CO_2 ppm",
       title = "Cubic Polynomial time trend model")
```

The Fitted values do not seem to depart from the actual data significantly, hence we proceed by leveraging this model in subsequent forecast.

### 2020 forecasts

```
# Forecasts for all of models specified previously
linear.model.forecast <- co2 %>%
  model(TSLM(value ~ trend())) %>% forecast(h = 300)
quadratic.model.forecast <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2))) %>% forecast(h = 300)
cube.model.forecast <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2) + I(trend()^3))) %>% forecast(h = 300)
expo.model.forecast <- co2%>%
```

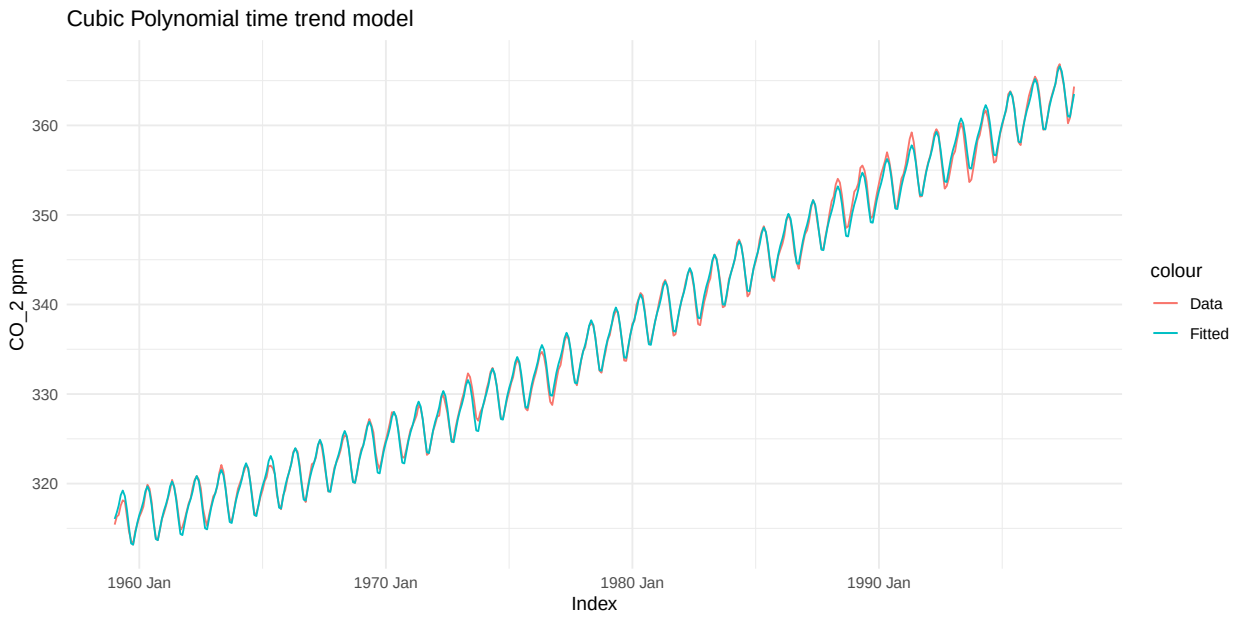



Figure 17: Fitted vs Actual Data for Polynomial Time Trend Model

```
  model(TSLM(log(value) ~ trend())) %>% forecast(h = 300)
poly.model.forecast1 <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2) + I(trend()^3) + season() )) %>%
  forecast(h = 300)
poly.model.forecast2 <- co2 %>%
  model(TSLM(value ~ trend() + I(trend()^2) + season() )) %>% forecast(h = 300)

# Comparison of forecasts
p1<-co2 %>% autoplot(value) +
  autolayer(rbind(linear.model.forecast, quadratic.model.forecast,
                  cube.model.forecast, expo.model.forecast, poly.model.forecast1)) +
  theme_minimal()
```

```
## Warning: 'rbind' is deprecated.
## Use 'bind_rows()' instead.
## See help("Deprecated")
```

```
p2<-co2 %>% autoplot(value) +
  autolayer(rbind(poly.model.forecast1, poly.model.forecast2)) +
  theme_minimal()
```

```
## Warning: 'rbind' is deprecated.
## Use 'bind_rows()' instead.
## See help("Deprecated")
```

```
grid.arrange(p1, p2, nrow=2)
```

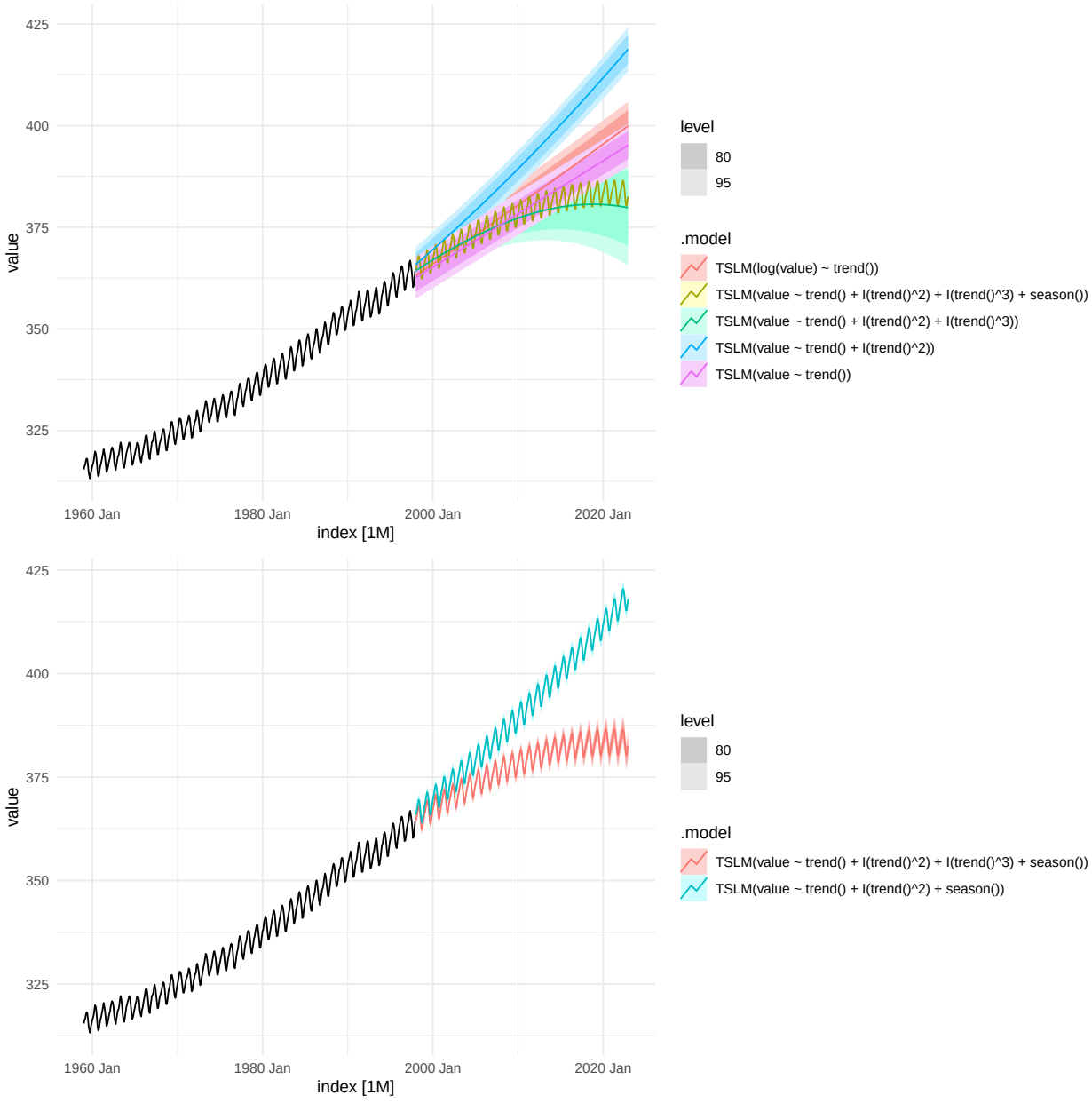



Figure 18: Model Forecast Review

Comparing forecast up to 2020, the quadratic version of the polynomial model trend looks more consistent and in line with expectation in comparison to the cubic version of the polynomial model trend. Therefore the section below reexamines the quadratic polynomial model residuals more closely.

```
# Polynomial with seasonals
poly.model.quad <- co2 %>%
```

```
  model(TSLM(value ~ trend() + I(trend()^2) + season() ))

# Plotting residuals
poly.model.quad %>% gg_tsresiduals()
```

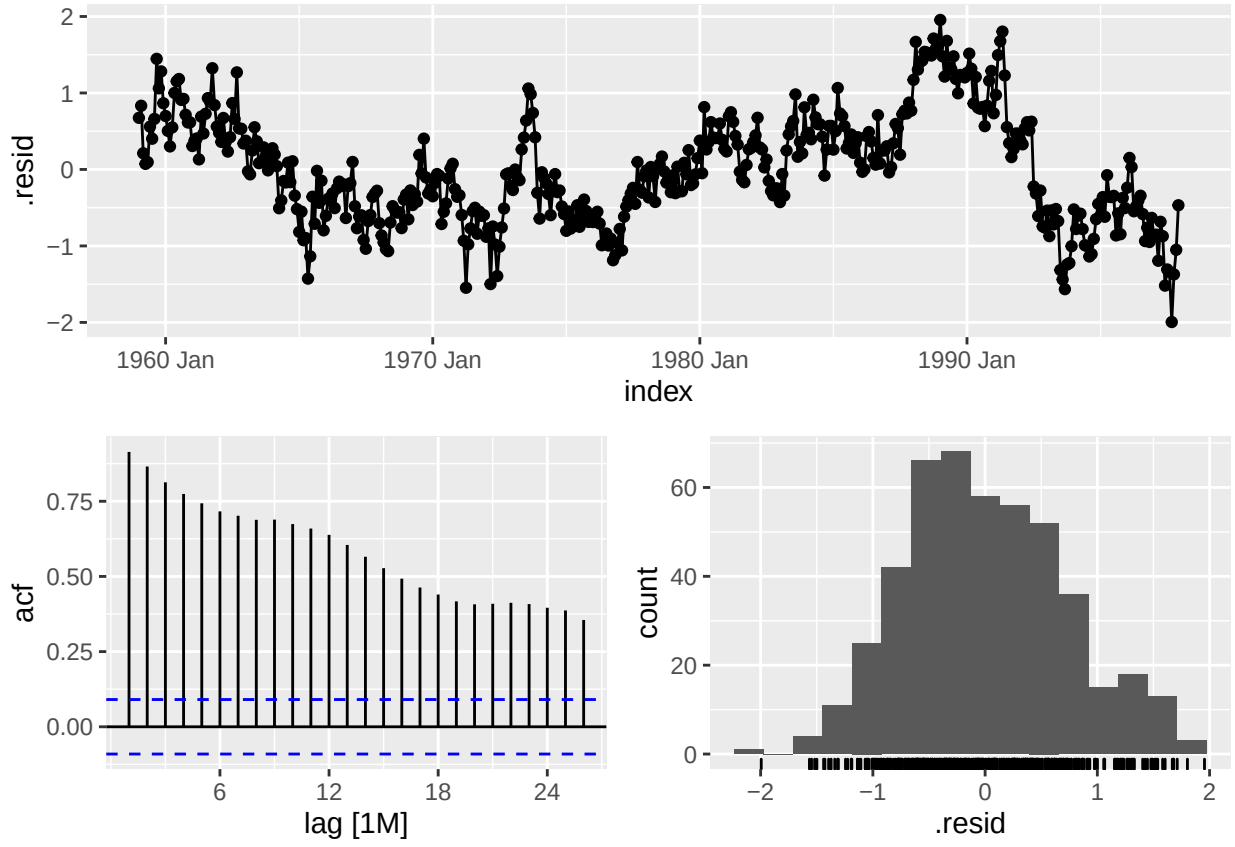



Figure 19: Residuals of Quadratic Polynomial Model with Seasonality

```
co2 <- co2 %>%
  group_by(Month) %>%
  mutate(residuals.quadratic.med = residuals.quadratic - median(residuals.quadratic))

p1<-ggplot(co2, aes(x = Month, y = residuals.quadratic.med)) +
  theme_minimal() +
  geom_boxplot()

p2<-ggplot(co2, aes(x = index, y = residuals.quadratic.med)) +
  geom_point(col = "grey", cex = .6)  +
  theme_minimal() +
  stat_smooth(se = FALSE, method = "lm") +
  facet_wrap(~ Month, nrow = 1) +
  theme(axis.text.x=element_text(angle=45, hjust = 1.5, vjust = 1.6,  size = 7))
```

```
p3<- ggplot(co2, aes(x = index, y = residuals.quadratic.med)) +
  geom_line(col = "grey") +
  theme_minimal() +
  stat_smooth(method = "lm", se = FALSE, span = 0.1 ,
                method.args = list(family = "symmetric", degree = 2))

p4<- ggplot(co2, aes(x = index, y = residuals.quadratic.med)) +
  geom_line(col = "grey") +
  theme_minimal() +
  stat_smooth(method = "loess", se = FALSE, span = 0.1 ,
                method.args = list(family = "symmetric", degree = 2))

grid.arrange(p1, p2, p3, p4, nrow=2, ncol=2)
```

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

```
## Warning: In lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
##     ...) :
##  extra arguments 'family', 'degree' will be disregarded
```

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

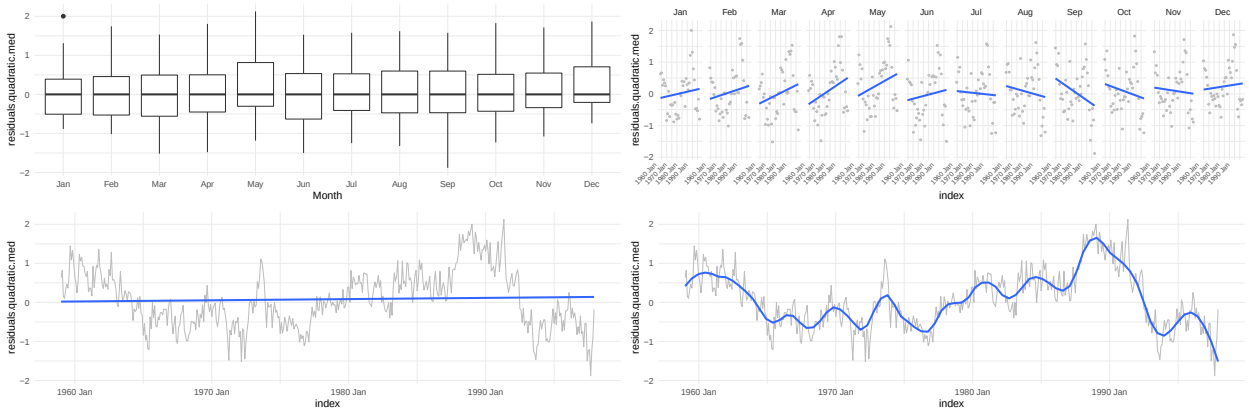



Figure 20: Quadratic Polynomial Model Review

```
augment(poly.model.quad) %>%
  ggplot(aes(x = index)) +
  theme_minimal() +
  geom_line(aes(y = value, colour = "Data")) +
  geom_line(aes(y = .fitted, colour = "Fitted")) +
  labs(x = "Index", y = "CO_2 ppm",
       title = "Quadratic Polynomial time trend model")
```

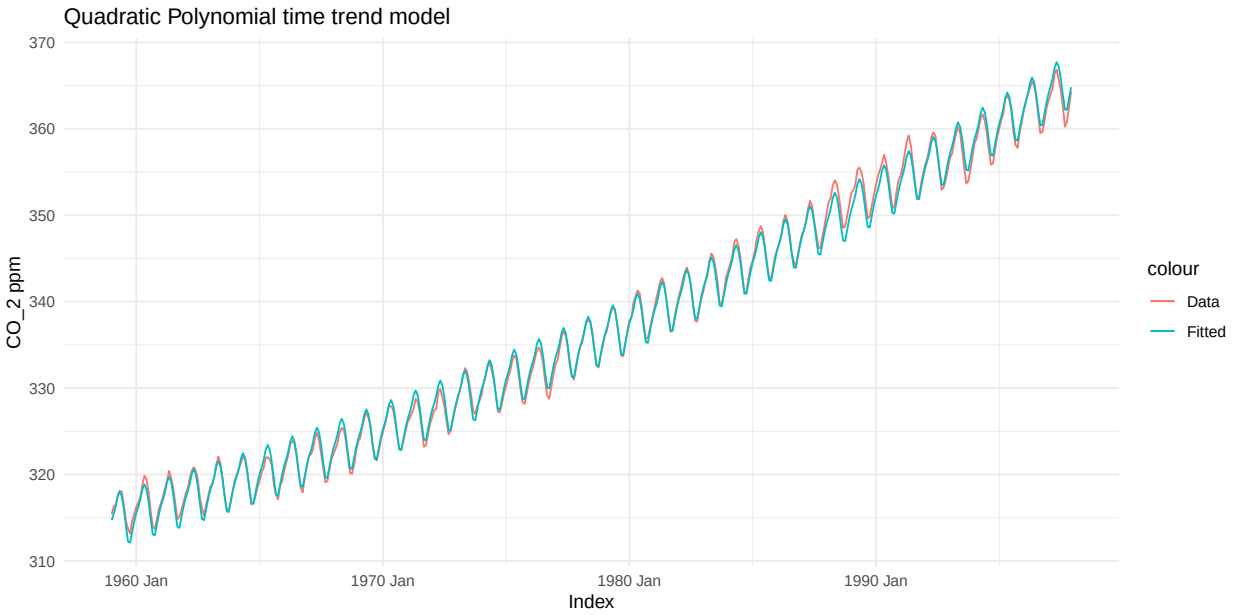



Figure 21: Quadratic Polynomial Model Forecast Review

Visual examination of the quadratic version of the polynomial model looks largely consistent with the conclusion drawn from the examination of the cubic version, however as observed the forecast performance of the quadratic version of the polynomial model is more consistent with the expectation.

**Part 3 (3 points)**

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

## Inspect differenced data

As depicted in the figures in the previous sections of this document, there is a clear trend and seasonality and the properties of CO2 concentration depend on time which is evidence that the series is non-stationary. Differencing procedures need to be applied to the data.

```
# differenced data
dd <- co2 %>%
  transmute(
    `index` = index,
    `co2` = value, # raw co2
    `LogCo2` = log(value),# log co2
    `d1` = difference(value), # first diff
    `Logd1` = difference(log(value)), # first log diff
    `d2` = difference(difference(value)), # second diff
    `Logd2` = difference(difference(log(value))), # second log diff
    `D1` = difference(value, 12), # seasonal first diff co2
```

```
    `LogD1` = difference(log(value), 12), # seasonal first diff in log co2
    `D1d1` = difference(difference(value, 12), 1), # D=1,d=1
    `LogD1d1` = difference(difference(log(value), 12), 1)) %>% #  D=1,d=1
  as_tsibble(index = index)

# plot differences
dd %>%
gather("series","value",-index,-Month) %>%
    mutate(series = factor(series,
                           levels = c('co2', 'LogCo2',
                                      'd1','Logd1',
                                      'd2', 'Logd2',
                                      'D1', 'LogD1',
                                      'D1d1','LogD1d1'))) %>%
    ggplot(aes(x = index, y = value)) +
    geom_line(colour = "grey", size = 1) +
    facet_wrap(~ series, ncol = 2, scales = "free") +
    theme_classic() + theme(legend.position = "none")  +
    ylab(expression(paste("CO"[2]))) +
    labs(title=expression(~" "),
       x=" ")+
    theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.title.y = element_text(size = 5),
    axis.text.y = element_text(size = 4))
```

Since the variance is relatively stable, the Box-Cox transformation is not necessary. Before fitting searching for ARIMA models, a $H_0^1$ test that the data is independently distributed - not serially correlated and $H_0^2$ test that the data are stationary is carried out. Ljung-Box test is used to test $H_0^1$ and identify whether the residuals resemble white noise or not and a KPSS test is used to test $H_0^2$ to test if the observable time series is stationary around a the trend.

```
# stationarity testing using kpss
stationary.tests <- dd %>%

  select(-c(index,Month)) %>%

  # applying stationarity and differencing tests
  features_all(unitroot_kpss)

data.frame(t(stationary.tests))
```

```
##                     t.stationary.tests.
## co2_kpss_stat                7.81727838
## co2_kpss_pvalue              0.01000000
## LogCo2_kpss_stat             7.83892328
```

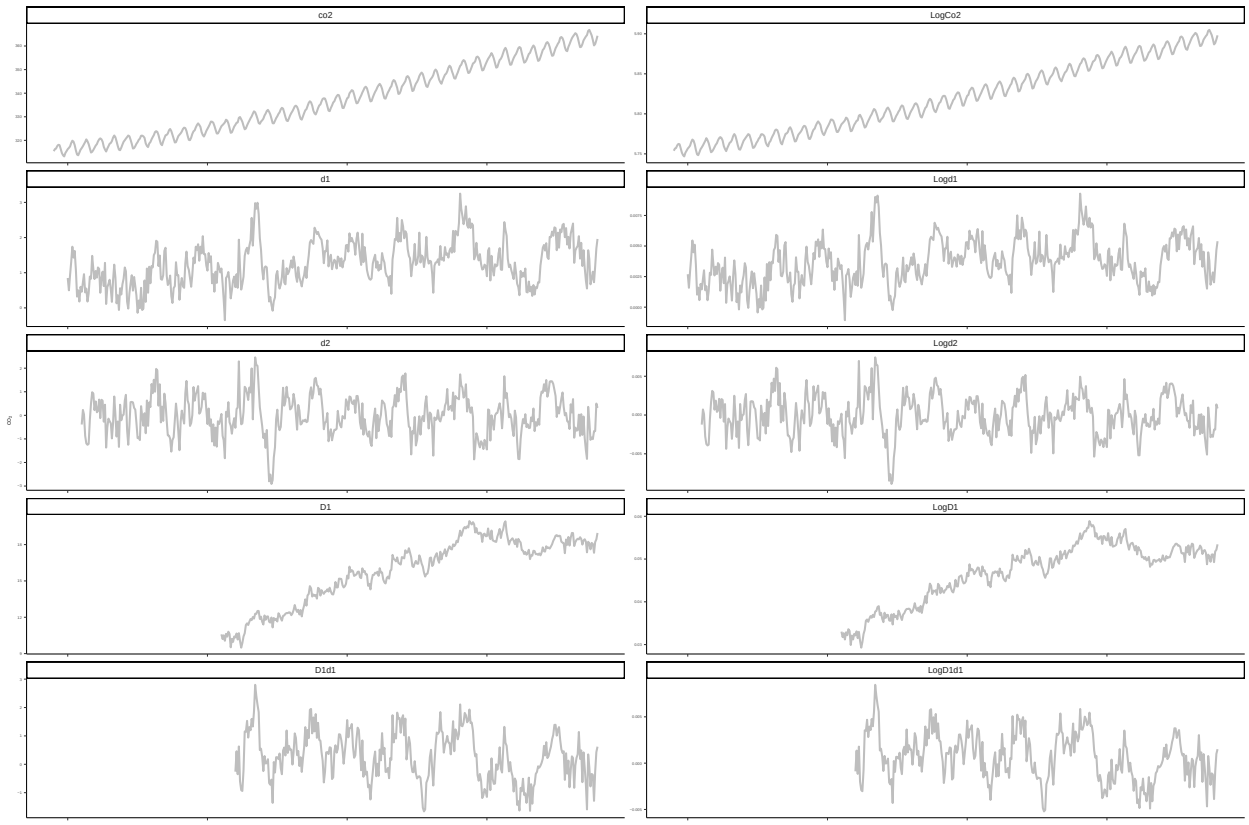



Figure 22: Differencing Procedure Analysis

```
## LogCo2_kpss_pvalue          0.01000000
## d1_kpss_stat                1.94387468
## d1_kpss_pvalue              0.01000000
## Logd1_kpss_stat             1.45610033
## Logd1_kpss_pvalue           0.01000000
## d2_kpss_stat                0.03398211
## d2_kpss_pvalue              0.10000000
## Logd2_kpss_stat             0.03366093
## Logd2_kpss_pvalue           0.10000000
## D1_kpss_stat                4.84147252
## D1_kpss_pvalue              0.01000000
## LogD1_kpss_stat             4.49337274
## LogD1_kpss_pvalue           0.01000000
## D1d1_kpss_stat              0.54543848
## D1d1_kpss_pvalue            0.03143278
## LogD1d1_kpss_stat           0.63080057
## LogD1d1_kpss_pvalue         0.01983631
```

As shown above the $H_0^1$ is rejected for each transformed version of $CO_2$. Conversely, $H_0^2$ cannot be rejected for series with first and second differences and first and seasonal differences. Moreover, the plots in **Figure 22** illustrate that, while d=1 and d=2 are stationary, there are still some seasonal patterns in the data. Thus, first differenced (d=1) and seasonal differenced (D=1) data series is used in the subsequent analyses.

### Fit ARIMA models with different p,d,q & P,D,Q values to the series:

**Approach 1**: Iteratively fit models with all possible values of $p, q$ and $P, Q$ between 0 and 3.

```
co2_train <- co2 %>% filter_index(~ '1987')
co2_test <- co2 %>% filter_index('1987' ~ .)
```

```
# # dataframe, ARIMAdf, to populate with ARIMA parameters (p,d,q)(P,D,Q)s
# ARIMA.df <- data.frame(index=rep(0, 1024),
#                        p=rep(0, 1024),
#                        d=rep(0, 1024),
#                        q=rep(0, 1024),
#                        P=rep(0, 1024),
#                        D=rep(0, 1024),
#                        Q=rep(0, 1024),
#                        AIC=rep(0, 1024),
#                        BIC=rep(0, 1024))
#
# # Index
# idx <- 1
# # Seasonal ARIMA models evaluation
# for (p in 0:3){
```

```
#   for (d in 0:1){
#     for (q in 0:3){
#       for (P in 0:3){
#         for (D in 0:1){
#           for (Q in 0:3){
#
#           # Estimate ARIMA (pdq)(PDQ)m
#           mod <- co2 %>%model(ARIMA(value ~ 0 + pdq(p,d,q) + PDQ(P,D,Q),
#                         stepwise=FALSE))
#
#           # populate dataframe with parameters and RMSE
#           ARIMA.df[idx,] <- c(idx, p, d, q, P, D, Q,
#                               try(select(glance(mod), AIC),silent=TRUE),
#                               try(select(glance(mod), BIC), silent=TRUE))
#           # Plus one
#           idx=idx+1
#           }
#         }
#       }
#     }
#   }
# }
# # Coerce characters to numerics
# ARIMA.df$AIC = as.numeric(ARIMA.df$AIC)
# ARIMA.df$BIC = as.numeric(ARIMA.df$BIC)
#
# # Order by IC and RMSE
# head(ARIMA.df[order(ARIMA.df$AIC, ARIMA.df$BIC),])
# # Ordered df
# odf = ARIMA.df[order(ARIMA.df$AIC, ARIMA.df$BIC),]
#
# # Save top 100 models in df
# write.csv(odf[c(1:100),],
#           "ARIMAdf3.csv")

# Open & view df
# head(read.csv("ARIMAdf3.csv"))
```

Three best models measured by AIC and BIC:

```
head(read.csv("ARIMAdf3.csv"),3)
```

```
##     X index p d q P D Q      AIC      BIC
## 1 759   759 2 1 3 2 1 2 171.0975 212.3005
## 2 752   752 2 1 3 1 1 3 171.3010 212.5039
## 3 247   247 0 1 3 2 1 2 171.6748 204.6372
```

The top models trained on data up to 1987:

- evaluated by AIC is ARIMA(2, 1, 3)(2, 1, 2)[12]

$$\Theta_0(B^{12})\theta_3(B)(1-B^{12})(1-B)x_t = \Phi_1(B^{12})\phi_3(B)w_t$$

- evaluated by BIC ARIMA(0, 1, 1)(0, 1, 1)[12]

$$\Theta_0(B^{12})\theta_0(B)(1-B^{12})(1-B)x_t = \Phi_1(B^{12})\phi_1(B)w_t$$

AIC top model:

```
fit.aic <- co2_train %>%model(arima = ARIMA(value ~ 0 + pdq(2,1,3) + PDQ(2,1,2))) %>%
report()
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
## Series: value
## Model: ARIMA(2,1,3)(2,1,2)[12]
##
## Coefficients:
##          ar1      ar2      ma1     ma2      ma3     sar1     sar2     sma1
##       0.2703  -0.0156  -0.6177  0.1108  -0.1339  -0.4223  -0.0493  -0.4275
## s.e.  0.0044   0.0063      NaN  0.0050      NaN   0.0091      NaN   0.0097
##          sma2
##       -0.3504
## s.e.      NaN
##
## sigma^2 estimated as 0.08595:  log likelihood=-57.56
## AIC=135.12   AICc=135.82   BIC=172.93
```

```
fit.aic2 <- co2_train %>%model(arima = ARIMA(value ~ 0 + pdq(0,1,3) + PDQ(0,1,1)))
fit.aic3 <- co2_train %>%model(arima = ARIMA(value ~ 0 + pdq(3,1,3) + PDQ(2,1,1)))

fit.aic %>% gg_tsresiduals()
```

The ACF plots of the residuals from the top ARIMA models show that all autocorrelations are largely within the threshold limits, indicating that the residuals are behaving like white noise.

```
p1<-gg_arma(fit.aic) + labs(title='AIC model #1')
#p2<-gg_arma(fit.bic) + labs(title='BIC model #1')
#p3<-gg_arma(fit.rmse) + labs(title='RMSE model #1')

p4<-gg_arma(fit.aic2) + labs(title='AIC model #2')
```

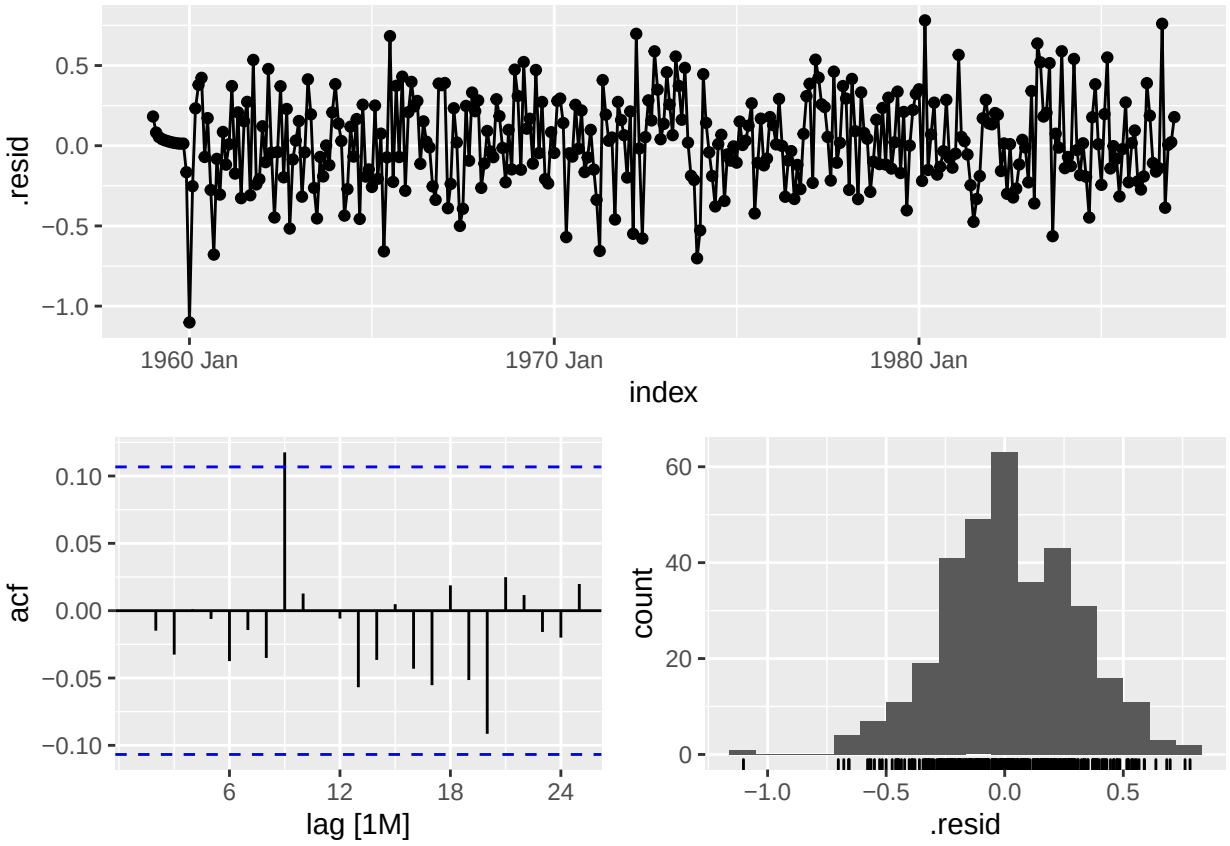



Figure 23: Residuals of a Top ARIMA Model evaluated by AIC

```
#p5<-gg_arma(fit.bic2) + labs(title='BIC model #2')
#p6<-gg_arma(fit.rmse2) + labs(title='RMSE model #2')

p7<-gg_arma(fit.aic3) + labs(title='AIC model #3')
#p8<-gg_arma(fit.bic3) + labs(title='BIC model #3')
#p9<-gg_arma(fit.rmse3) + labs(title='RMSE model #3')

# Arrange
grid.arrange(p1,p4,p7,nrow=3)
```

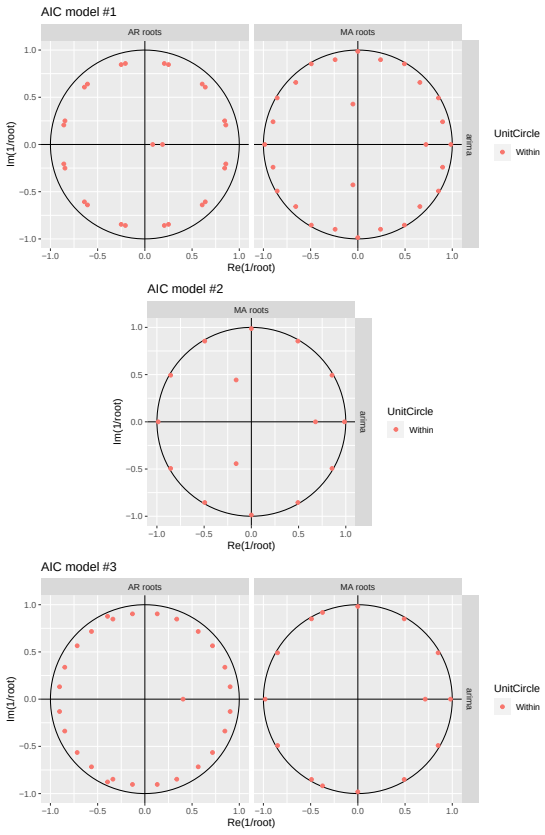



Figure 24: Best ARIMA Model evaluated by AIC, BIC and RMSE

The three red dots in the left hand plot correspond to $\phi(B)$ the roots of the polynomials, while the red dot in the right hand plot corresponds to the root of $\theta(B)$. They are all inside the unit circle. Any roots close to the unit circle may be numerically unstable, and the corresponding model will not be good for forecasting. Based on the unit circle charts it looks like the top two AIC models may more stable at forecasting.

**Approach 2**: Naive search of a subset of models with values of $p, q$ and $P, Q$ that do not approach the edge of the unit-cirle.

```
# Partial search & report
sarimaPS = co2_train %>%
```

```
  model(ARIMA(value)) %>%
  report(fit)
```

```
## Series: value
## Model: ARIMA(0,1,1)(2,1,2)[12]
##
## Coefficients:
##           ma1     sar1     sar2     sma1     sma2
##       -0.3726  -0.2333  -0.0805  -0.6278  -0.1629
## s.e.   0.0604   0.5726   0.0684   0.5726   0.4885
##
## sigma^2 estimated as 0.08668:  log likelihood=-61.15
## AIC=134.31   AICc=134.57   BIC=156.99
```

```
# Examine residuals
sarimaPS %>% gg_tsresiduals()
```

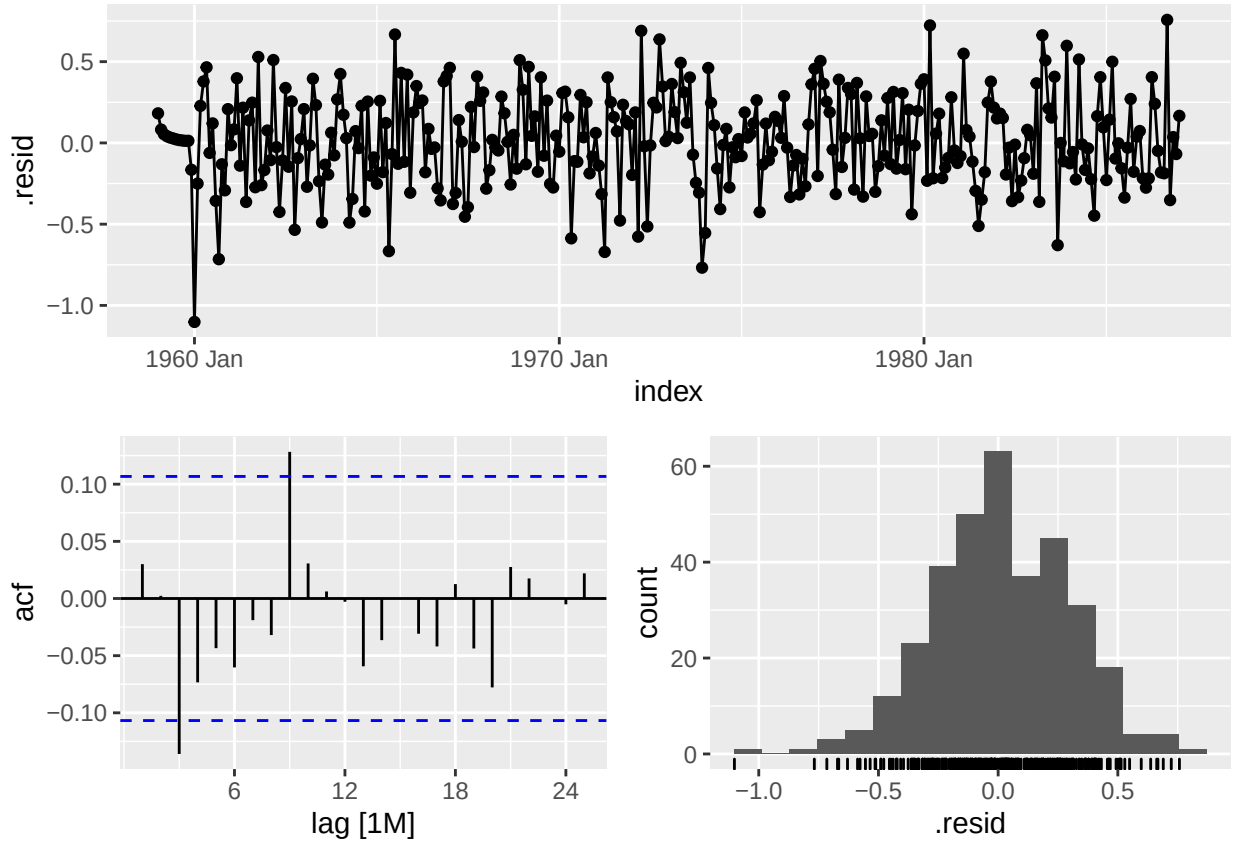



Figure 25: Naive Search ARIMA Model

```
# Test for autocorrelation of residuals
augment(sarimaPS) %>% features(.resid, ljung_box)
```

```
## # A tibble: 1 x 3
##   .model       lb_stat lb_pvalue
##   <chr>          <dbl>     <dbl>
## 1 ARIMA(value)   0.307     0.580
```

```
# Examine roots
glance(sarimaPS)[['ar_roots']]
```

```
## [[1]]
##  [1]  0.7068162+0.8567471i -0.8567471+0.7068162i -0.3885567-1.0404943i
##  [4]  1.0404943-0.3885567i  0.3885567+1.0404943i -1.0404943+0.3885567i
##  [7] -0.1837472-1.0953728i  1.0953728-0.1837472i  0.1837472+1.0953728i
## [10] -1.0953728-0.1837472i  0.1837472-1.0953728i  1.0953728+0.1837472i
## [13] -0.1837472+1.0953728i -1.0404943-0.3885567i  0.3885567-1.0404943i
## [16]  1.0404943+0.3885567i -0.3885567+1.0404943i -0.8567471-0.7068162i
## [19]  0.7068162-0.8567471i  0.8567471+0.7068162i -0.7068162+0.8567471i
## [22] -0.7068162-0.8567471i  0.8567471-0.7068162i -1.0953728+0.1837472i
```

```
# Inverse roots within unit circle
gg_arma(sarimaPS)
```

```
# modulus of roots exceed unity
Mod(polyroot(c(1, -coef(sarimaPS)[['estimate']])))
```

```
## [1] 1.145046 1.108494 1.108494 1.145046 3.810939
```

$$\Theta_1(B^{12})\theta_1(B)(1-B^{12})(1-B)x_t = \Phi_2(B^{12})\phi_1(B)w_t$$

For the model identified by the naive search ARIMA(1,1,1)(1,1,2)[12], the residuals are independently distributed- not serially correlated. We failed to reject the $H_0^1$. Thus the residuals are independently distributed! This is also apparent in the ACF plot only 1 lag exceeds the threshold.

The roots of the top models identified by both the exhaustive as well as naive search are close to the unit circle. This would suggest that there may be lack of stability in the forecasted data.

### Generate forecasts into 2020 with the both ARIMA models

To generate forecast, based on the unit circle analysis three models are selected:

- exhaustive search - top AIC model $\Theta_0(B^{12})\theta_3(B)(1-B^{12})(1-B)x_t = \Phi_1(B^{12})\phi_3(B)w_t$

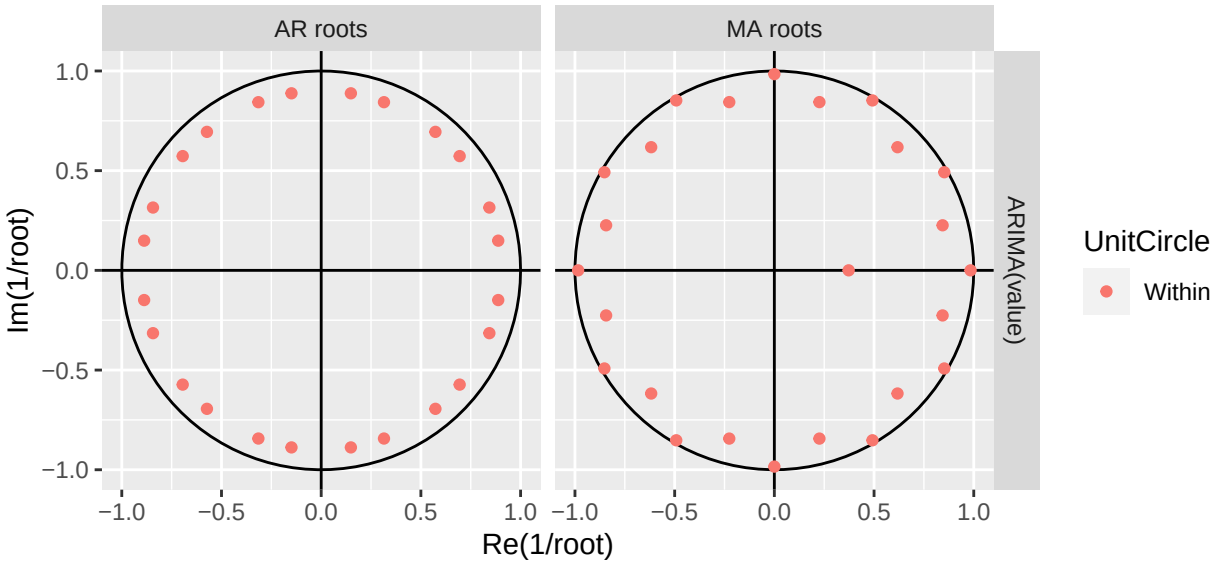



Figure 26: Naive Search ARIMA Model

- exhaustive search - second best BIC model $ _0(B^{12}) _1(B)(1-B^{12})(1-B)x_t= _1(B^{12}) _1(B)w_t $
- naive search model $ _1(B^{12}) _1(B)(1-B^{12})(1-B)x_t= _2(B^{12}) _1(B)w_t $

```
# First, fit ARIMA Model selected with approach 1:ARIMA(3,1,3)(0,1,1)12
# Extract report
sarimaES1 <- co2_train %>%
  model(ARIMA(value ~ 0 + pdq(3,1,3) + PDQ(0,1,1),
                        stepwise=FALSE)) %>%
  report()
```

```
## Series: value
## Model: ARIMA(3,1,3)(0,1,1)[12]
##
## Coefficients:
##           ar1     ar2     ar3      ma1      ma2      ma3     sma1
##       -0.0853  0.1428  0.0230  -0.2708  -0.1667  -0.1204  -0.8538
## s.e.   2.5951  0.3595  0.8692   2.5915   1.1557   0.7887   0.0352
##
## sigma^2 estimated as 0.08542:  log likelihood=-57.72
## AIC=131.44   AICc=131.9   BIC=161.69
```

```
# Generate forecasts from exhaustive search
esp1=sarimaES1 %>%
  forecast(h = 400) %>% autoplot() +
  autolayer(co2_train, value) +
  theme_classic() +
  labs(x = "Time (1959-2020)", y = expression(paste("CO"[2]~"(ppm)")),
       title = expression(~"CO"[2]~
                            "forecasts with ARIMA(3,1,3)(0,1,1)"[12]))

# First, fit ARIMA Model selected with approach 1:ARIMA(1,1,1)(0,1,1)12
# Extract report
sarimaES2 <- co2_train %>%
  model(ARIMA(value ~ 0 + pdq(1,1,1) + PDQ(0,1,1),
                        stepwise=FALSE)) %>%
  report()
```

```
## Series: value
## Model: ARIMA(1,1,1)(0,1,1)[12]
##
## Coefficients:
##          ar1      ma1     sma1
##       0.2969  -0.6387  -0.8527
## s.e.  0.1541   0.1283   0.0347
##
```

```
## sigma^2 estimated as 0.08593:  log likelihood=-60.78
## AIC=129.55   AICc=129.68   BIC=144.68
```

```
# Generate forecasts from exhaustive search
esp2=sarimaES2 %>%
  forecast(h = 400) %>% autoplot() +
  autolayer(co2_train, value) +
  theme_classic() +
  labs(x = "Time (1959-2020)", y = expression(paste("CO"[2]~"(ppm)")),
       title = expression(~"CO"[2]~
                            "forecasts with ARIMA(1,1,1)(0,1,1)"[12]))

# Generate forecasts from partial search
psp=sarimaPS %>%
  forecast(h = 400) %>% autoplot() +
  autolayer(co2_train, value) +
  theme_classic() +
  labs(x = "Time (1959-2020)", y = expression(paste("CO"[2]~"(ppm)")),
       title = expression(~"CO"[2]~
                            "forecasts with ARIMA(1,1,1)(1,1,2)"[12]))

# arrange
grid.arrange(esp1, esp2, psp)
```

```
#TOP AIC
p1<-sarimaES1 %>%
  forecast(h=120) %>%
  autoplot() +
    ylab("CO2 ppm") + xlab("Time") +
    labs(title='ARIMA(3,1,3)(0,1,1)12') +
  autolayer(co2_test, col= 'red')

#SECOND BEST BIC
p2<-sarimaES2 %>%
  forecast(h=120) %>%
  autoplot() +
    ylab("CO2 ppm") + xlab("Time") +
    labs(title='ARIMA(1,1,1)(0,1,1)12') +
  autolayer(co2_test, col= 'red')

#TOP NAIVE
p3<-sarimaPS %>%
  forecast(h=120) %>%
  autoplot() +
    ylab("CO2 ppm") + xlab("Time") +
    labs(title='ARIMA(1,1,1)(1,1,2)12') +
  autolayer(co2_test, col= 'red')
```

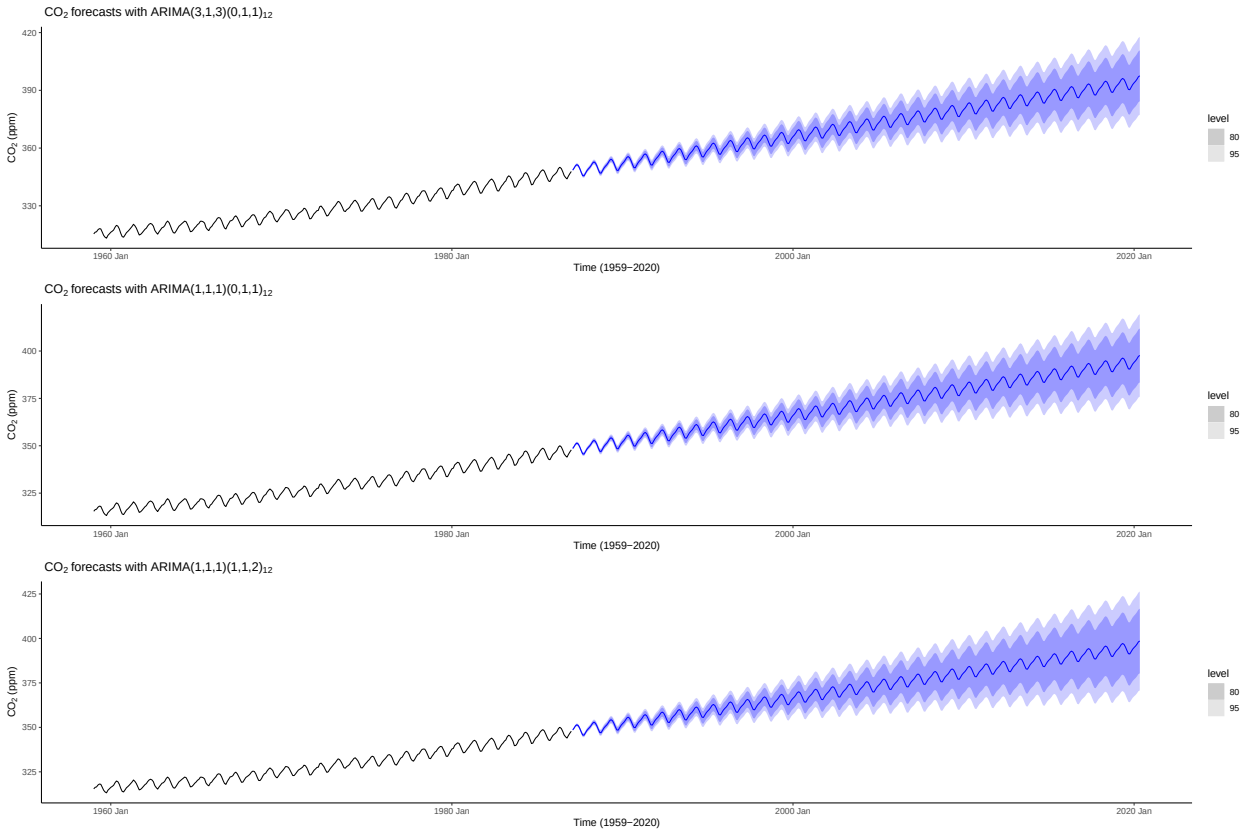



Figure 27: Best ARIMA Model Forecasts

```
#Arrange

grid.arrange(p1,p2,p3, nrow=3)
```

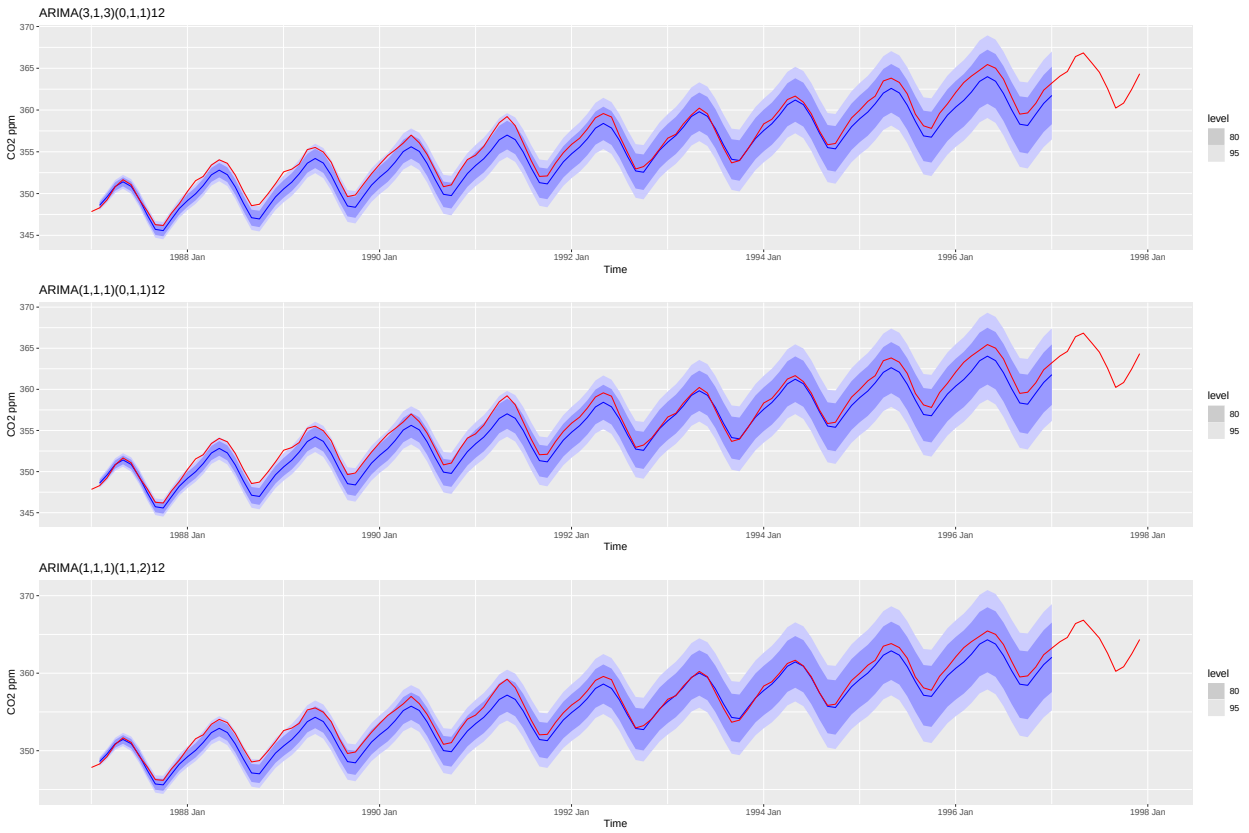



Figure 28: Best ARIMA Model Forecasts vs. Actual Data

The performance of our top three models is not visibly different, all three appear to make predictions of similar precision. If evaluated by AIC the ARIMA(2,1,3)(2,1,2)12 is the top model trained on 1959 to 1987 data. However, there are two observations that may suggest that forecasts delivered by this model may not be stable: - the roots of this model are close to the unit-circle - the exploratory data analysis revealed a notable change in the trend between 1980 and 1990. As the train and test data is split using 1987 year, this may have an impact on the stability of the forecasted data.

**Part 4 (4 points)**

The file `co2_weekly_mlo.txt` contains weekly observations of atmospheric carbon dioxide concentrations measured at the Mauna Loa Observatory from 1974 to 2020, published by the National Oceanic and Atmospheric Administration (NOAA). Convert these data into a suitable time series object, conduct a thorough EDA on the data, and address the problem of missing observations. Describe how the Keeling Curve evolved from 1997 to the present and compare current atmospheric CO2 levels to those predicted by your forecasts in Parts 2 and 3. Use the weekly data to generate a month-average series from 1997 to the present, and compare the overall forecasting performance of your models from Parts 2 and 3 over the entire period.

### Convert these data into a suitable time series object

```
# Load data - co2_weekly_mlo.txt
co2.weekly <- read.table("co2_weekly_mlo.txt")

# rename columns
names(co2.weekly) <- c('year', 'month', 'day', 'decimal','co2ppm', 'days', '1yr', '10yr', 'sin

# create date column
co2.weekly <- co2.weekly %>%
  mutate(date = tsibble::yearweek( make_date(year, month, day) ))

#replace missing with NA value
co2.weekly[co2.weekly$co2ppm == -999.99, ]$co2ppm = NA

# subset data
co2.weekly.subset <- co2.weekly[, c('date', 'co2ppm')]

# Coerce data into tsibble object
co2.weekly <- as_tsibble(co2.weekly.subset, index = date)
```

The data consists of nine variables out of which the following are required; - the `year`, `month`, `day` that index the data series by week - co2ppm - weekly carbon dioxide ppm concentrations measurement.

There are 2406 records, representing 46 years of weekly data (May 1974 to June 2020). There are missing values in the dataset.

### Plot Series

```
# plot
co2.weekly %>%
  autoplot(co2ppm, ylab = expression("CO2 ppm"),col = 'grey', las = 1) +
  theme_minimal()
```

```
# NA index positions
na.indx <- which(is.na(co2.weekly$co2ppm))

# imputed NA values
co2.weekly.imputed.ts <- co2.weekly %>%
               mutate(co2ppm.imp = zoo::na.approx(co2ppm))

# plot
co2.weekly.imputed.ts %>%
```

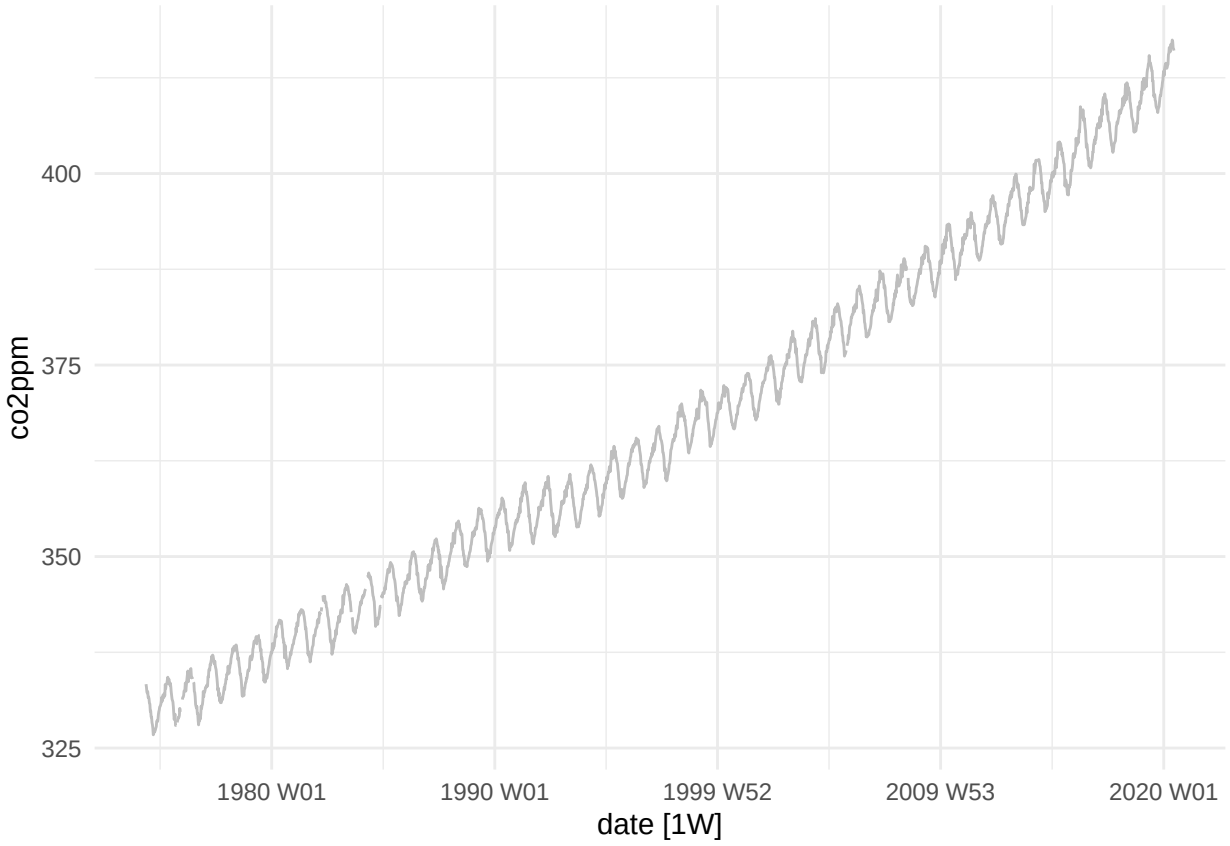



Figure 29: co2_weekly_mlo.txt weekly data

```
  ggplot(aes(x = date)) +
  geom_point(data = co2.weekly.imputed.ts %>%
               filter(is.na(co2ppm)),
             aes(y = co2ppm.imp,  color = 'salmon'), size = 1, alpha = .5) +
  geom_line(aes(y = co2ppm), color = 'grey', size = .4) +
  theme_classic() +
  ylab(expression("CO"[2]~"ppm")) +
  xlab("Year Week") +
  scale_color_identity(name = " ",
                       breaks = c("salmon"),
                       labels = c("imputed values"),
                       guide = "legend")
```

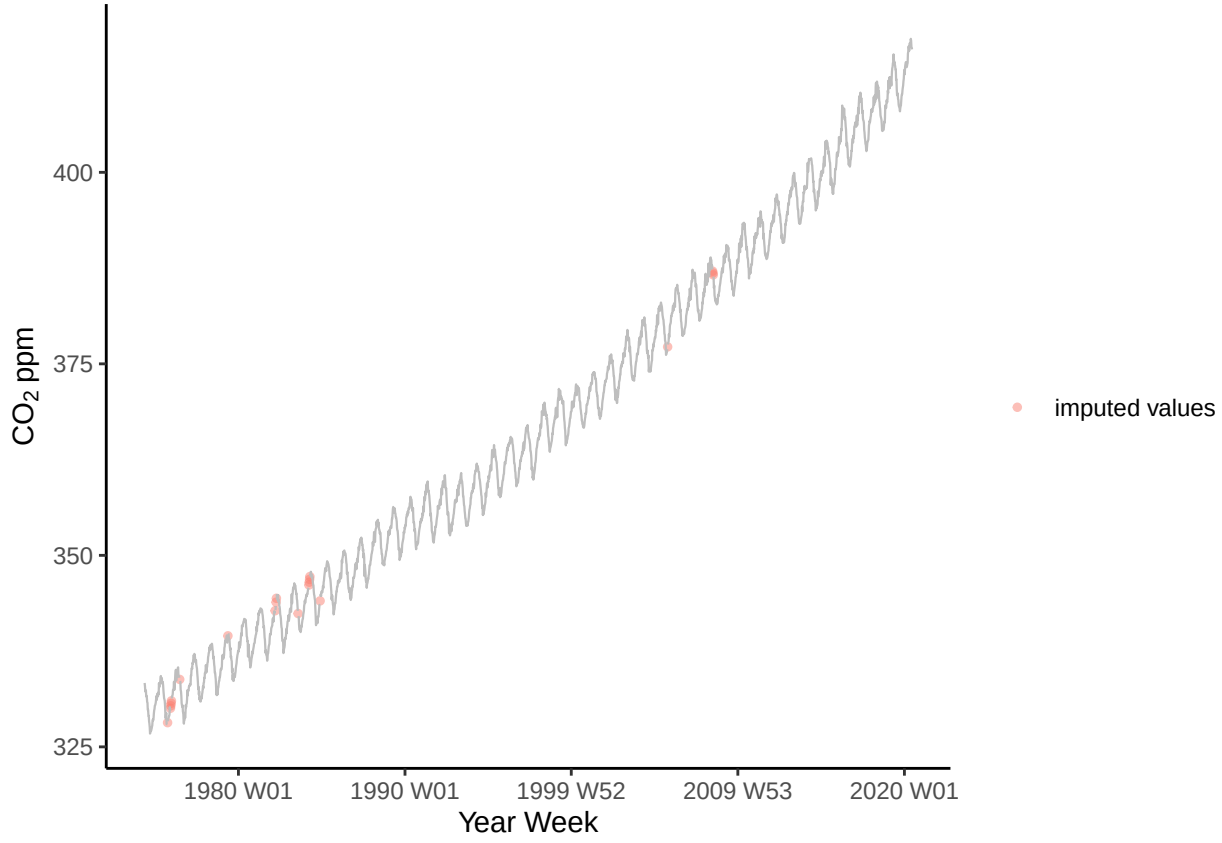


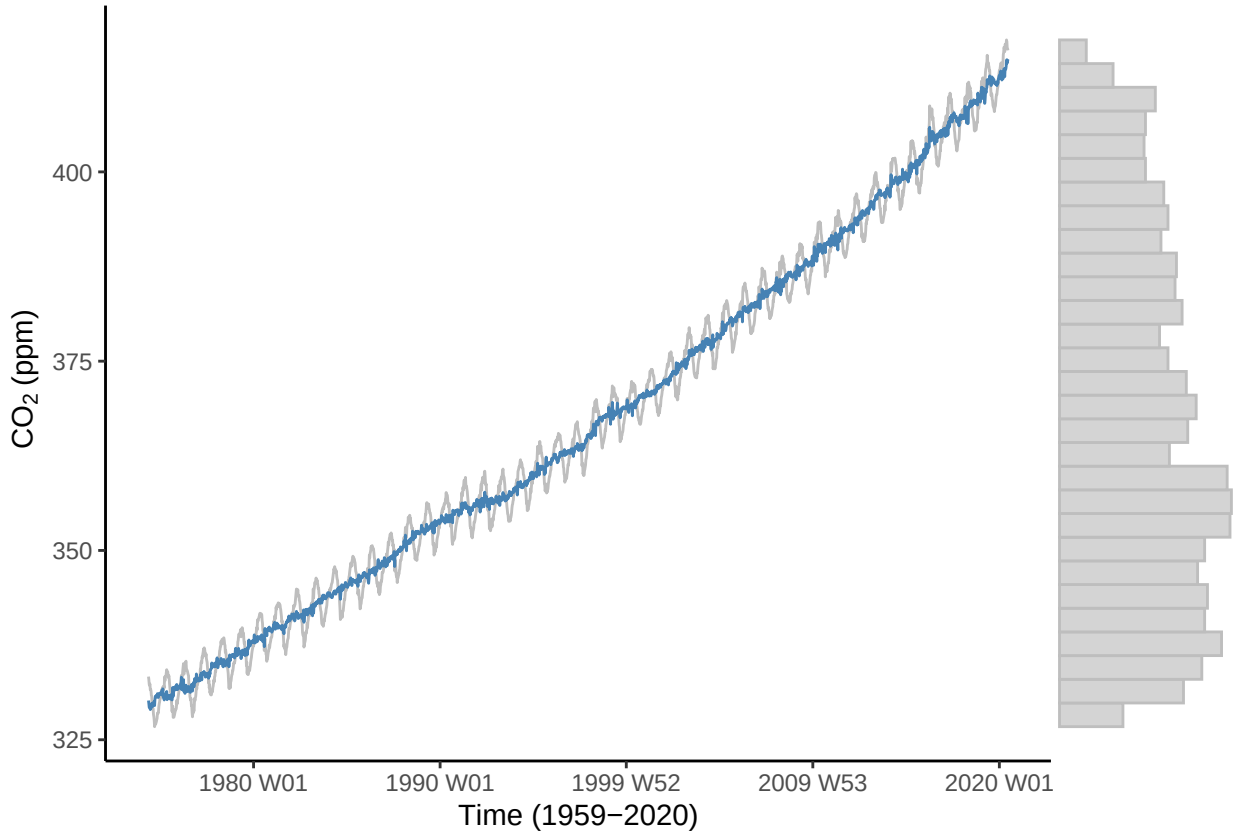

Figure 30: Imputed co2_weekly_mlo.txt weekly data

```
# impute
co2.weekly.imputed.ts <- co2.weekly %>%
               mutate(co2ppm = zoo::na.approx(co2ppm))
```

The missing values in the data are replaced by approximated values using the `zoo` package. The figure above depicts the full data set including the approximated values. The imputation looks appropriate as it does not visibly disrupt the overall trend and seasonality.

### Exploratory Data Analysis - `co2_weekly_mlo.txt` series

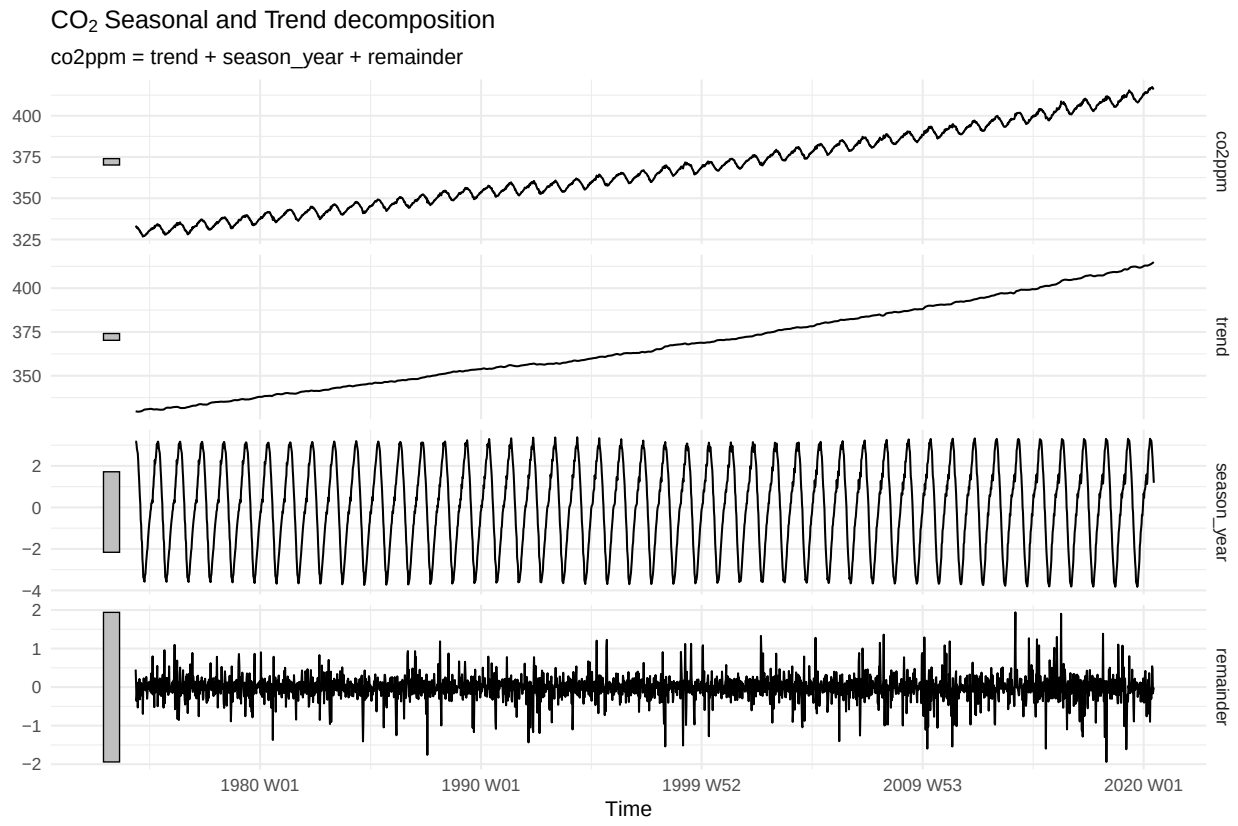
```
# build STL model
co2.weekly.stl <- co2.weekly.imputed.ts %>%
  model(STL(co2ppm ~ trend(window=13) + season(), robust = TRUE))

# Plot yearly data
p=co2.weekly.imputed.ts %>%
  autoplot(co2ppm, color='gray') +
  autolayer(components(co2.weekly.stl),
            season_adjust, color='steelblue') +
  ylab(expression("CO"[2]~ " (ppm)")) +
  xlab("Time (1959-2020)") +
    scale_y_continuous(name = expression(paste(" CO"[2]~"(ppm)")))+
  theme_classic() + geom_point(col="transparent")
ggMarginal(p, type="histogram",
           margins = "y",
           col="grey",
           fill = "grey83")
```




The overall distribution, trend and seasonality observes in the co2 data is consistent with the distribution, trend and seasonality obeserved in the `co2_weekly_mlo.txt` data.

To analyse the time series components, the series is decomposed into the seasonal, trend, and remaining components:

```
# Seasonal and Trend decomposition using Loess (STL)
co2.weekly.stl %>%
  components() %>%
  autoplot() +
  labs(title=expression("CO"[2]~
                          "Seasonal and Trend decomposition"),
       x="Time (1959-1997)") +
  xlab('Time') +
  theme_minimal()
```




Leveraging STL decomposition, once the trend and seasonality are removed the remaining series resembles white noise.

### Trend

```
# Overall trend by fitting a straight line to the data
co2.weekly.imputed.ts %>%
ggplot(aes(x = date, y = co2ppm)) +
  geom_line(color = 'grey') +
  theme_minimal() +
```

```
  stat_smooth(method = "lm", se = FALSE, aes(col = "red")) +
  stat_smooth(method = "lm",formula = y ~ x + I(x^2), se = FALSE, aes(col = "blue")) +
  stat_smooth(method = "lm",formula = y ~ x + I(x^4), se = FALSE, aes(col = "green")) +
  scale_color_identity(name = "Model fit",
                       breaks = c("red", "blue", "green"),
                       labels = c("Linear", "Quadratic", "Cubic"),
                       guide = "legend")
```

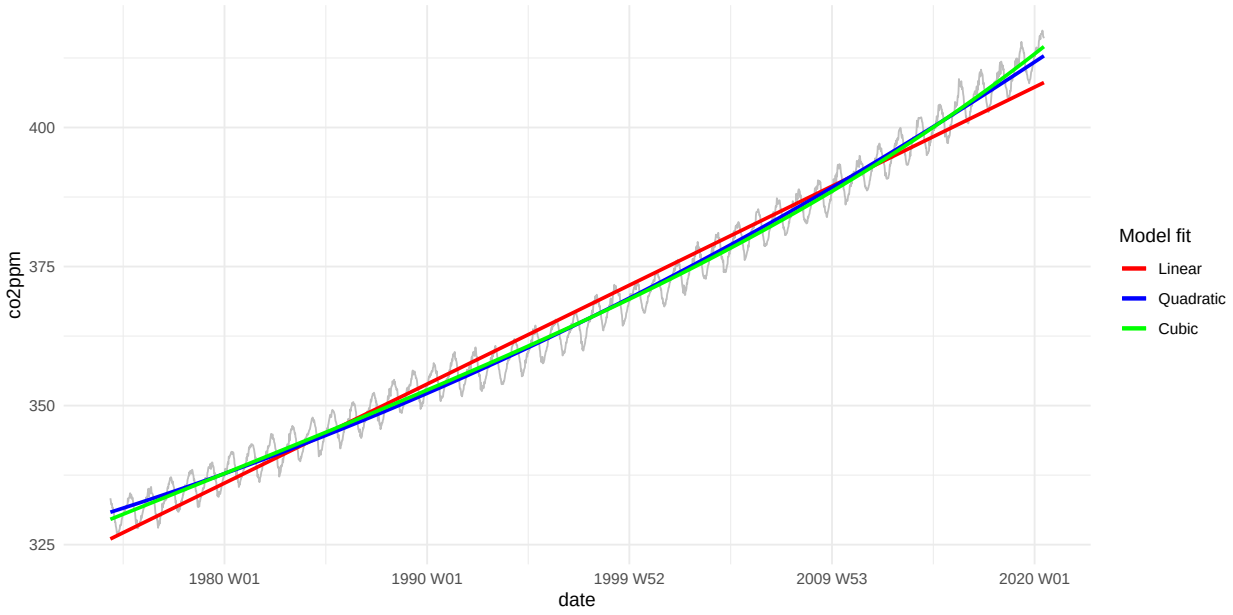



Figure 31: Weekly - CO2 Trend Analysis

Same as in the `co2` series, the quadratic and cubic trend are a better fit to the series than a simple linear trend.

### Seasonality

```
ggplot(data = co2.weekly.imputed.ts %>% subset(year(date) == 1985), aes(date, co2ppm)) +
  geom_line() +
  theme_minimal() +
  xlab('Time') +
  ylab('CO2 Concentration PPM') +
  ggtitle('Carbon Dioxide Concentration in 1985')
```

The seasonality pattern observed in the `co2` series can be observed in the `co2_weekly_mlo.txt` series as well.

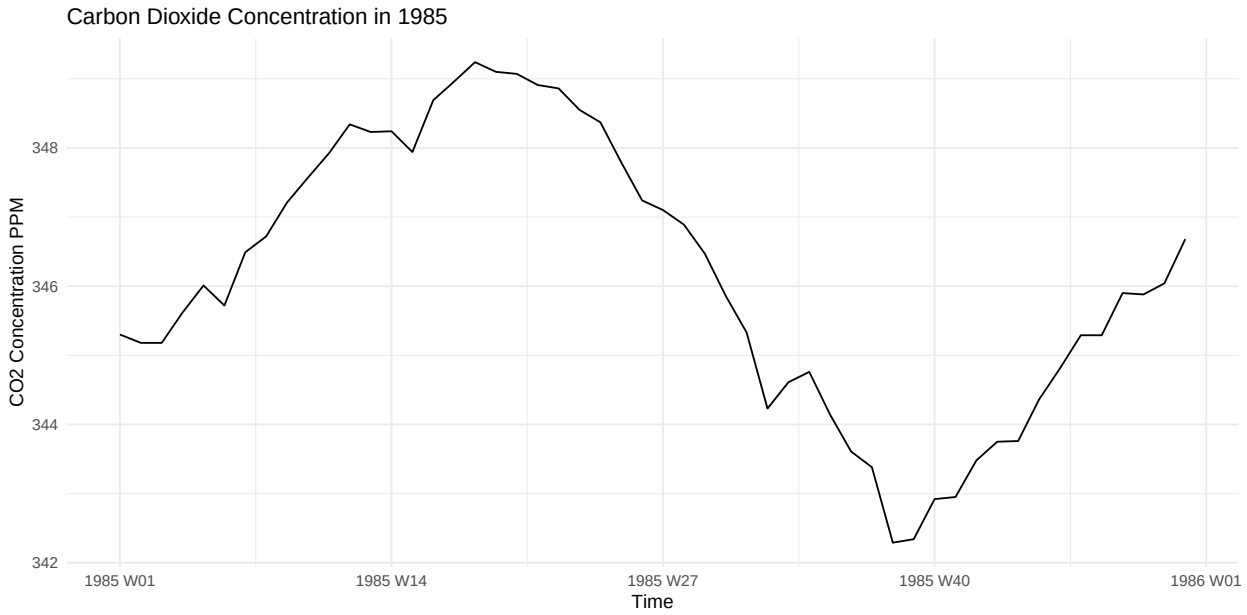



Figure 32: Weekly CO2 Seasonality Component

```
# Plot time series
co2.weekly.imputed.ts %>% gg_season(co2ppm, period = "year") +
  theme_minimal() +
  scale_y_continuous(name = expression(paste("CO"[2]~"(ppm)")))+
  labs(title=expression("CO"[2]~"as a function of time by season"),
       x="Time (1959 - 2020)")
```

This seasonal cycle repeats each year with a long term increasing trend. In recent years, this seasonal cycle of CO2 concentration appears to be accelerating.

```
# Annual average data
co2wAA <- co2.weekly.imputed.ts %>% index_by(Year = ~ year(.)) %>%
   summarise(AnnualAverage = mean(co2ppm))

# Plot annualized average trend
co2wAA %>% autoplot(AnnualAverage) +
  theme_classic() +
  scale_y_continuous(name = expression(paste("Annualized average CO"[2]~"(ppm)")))+
  xlab('Time')
```

The minor acceleration in `co2ppm` can also be observed from an annualised growth rate figure above between 2010 and 2020.

```
# Plot lag plot
co2.weekly.imputed.ts %>% gg_lag(co2ppm, geom = "point", size = .1) +
 theme_minimal() +
 scale_y_continuous(name = expression(paste("CO"[2]~"(ppm)")))
```

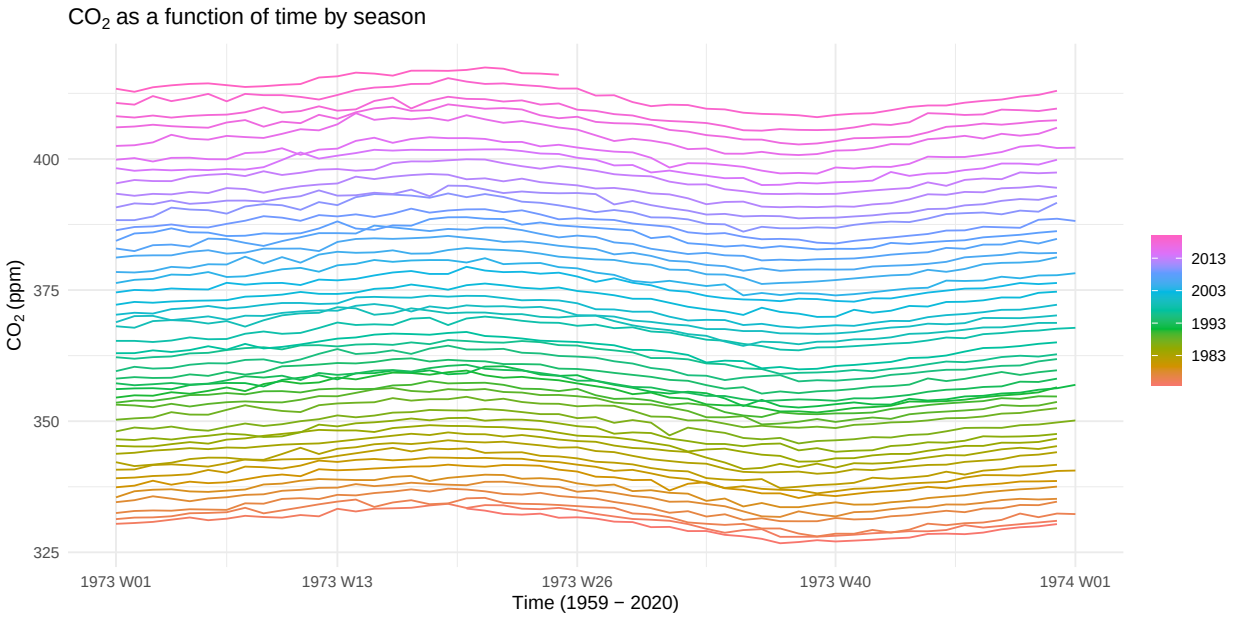



Figure 33: Weekly - CO2 Seasonality Analysis

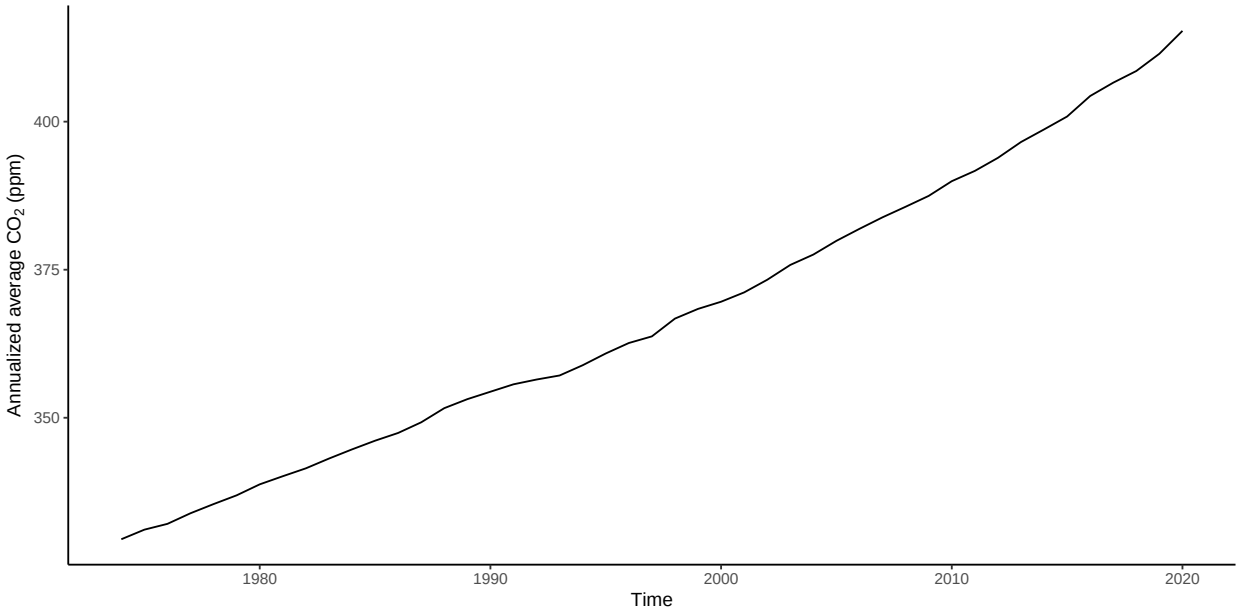
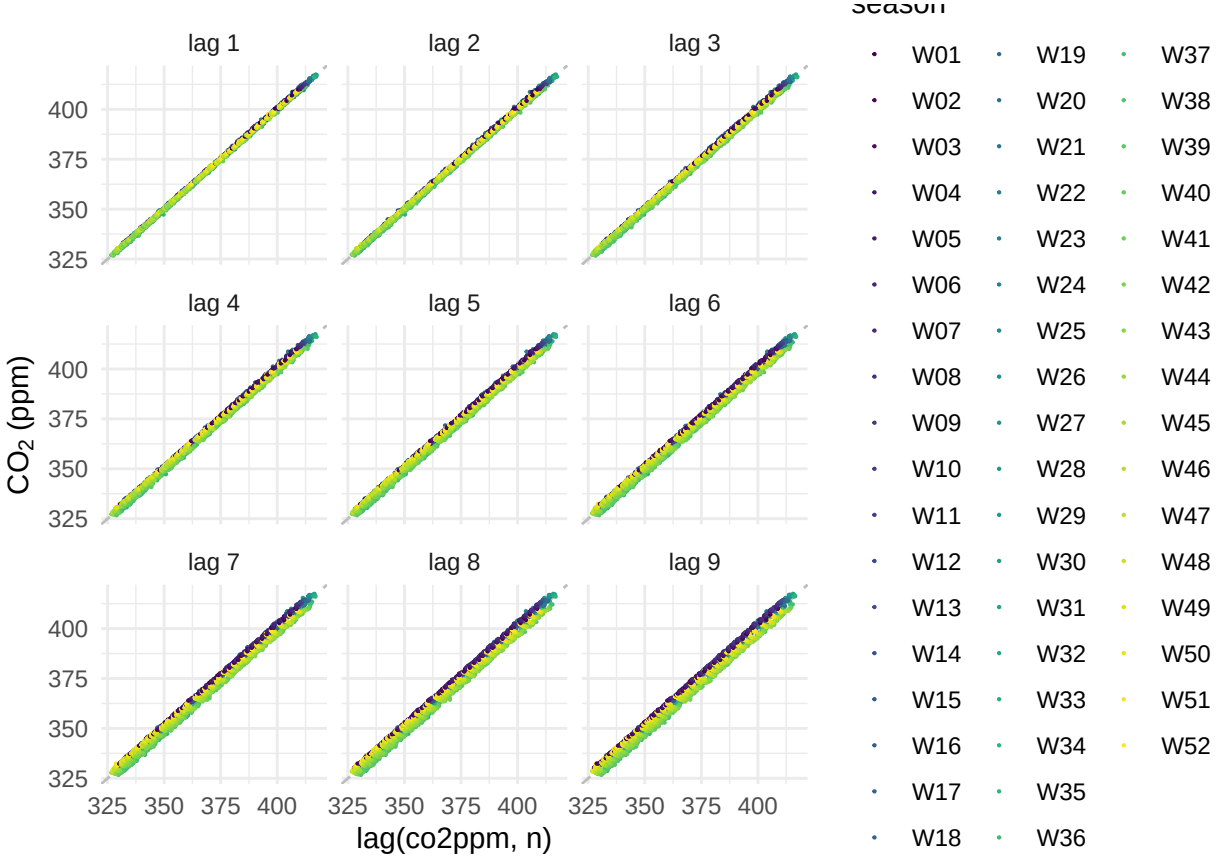
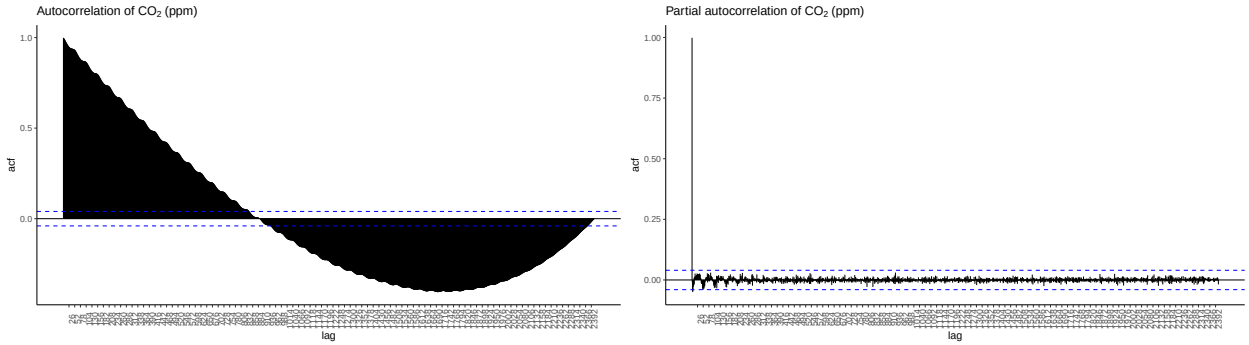



Figure 34: Weekly - Average CO2 concentrations




As per original series, the `co2_weekly_mlo.txt` shows a linear pattern, a positive linear trend (i.e. going upwards from left to right) which is suggestive of positive autocorrelation.

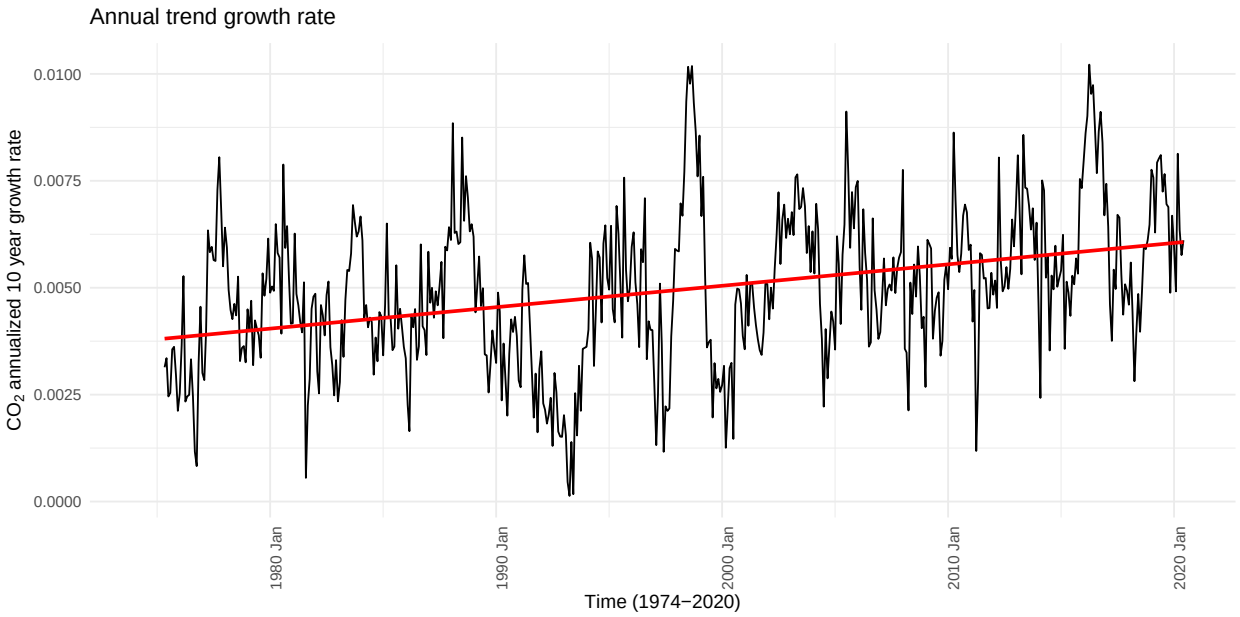
```
# Plot ACF
acf <- co2.weekly.imputed.ts %>% ACF(co2ppm, lag_max = nrow(co2.weekly.imputed.ts)) %>%
  autoplot() +
  theme_classic() +
  theme(axis.text.x = element_text(angle=90))+
  scale_y_continuous(name = expression(paste("acf")))+
  labs(title=expression("Autocorrelation of CO"[2]~"(ppm)"),
     x="lag")

# Plot PACF
pacf <-co2.weekly.imputed.ts %>% PACF(co2ppm, lag_max = nrow(co2.weekly.imputed.ts)) %>%
  autoplot() +
  theme_classic() +
  theme(axis.text.x = element_text(angle=90))+
  scale_y_continuous(name = expression(paste("acf")))+
  labs(title=expression("Partial autocorrelation of CO"[2]~"(ppm)"),
     x="lag")

# All together
grid.arrange(acf, pacf, ncol= 2)
```




The ACF and the PACF are largely consistent with the co2 series, with perhaps smaller partial correlation of the stationary series with its own lagged values.

### Keeling Curve evolution / growth rate

```
co2.weekly.imputed.ts %>%
  index_by(Year = ~ yearmonth(.)) %>%
  summarise(MonthlyAverage = mean(co2ppm)) %>%

  # define yt_1 as yt - yt-1
  dplyr::mutate(Yt_1 = MonthlyAverage/lag(MonthlyAverage, 12)-1) %>%

  # Plot
  autoplot(Yt_1) +
  stat_smooth(method = "lm", se = FALSE, span = 0.1 , col='red',
                method.args = list(family = "symmetric", degree = 2)) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=90))+
  scale_y_continuous(name = expression(paste("CO"[2]~"annualized 10 year growth rate")))+
  labs(title=expression("Annual trend growth rate"),
     x="Time (1974-2020)")
```



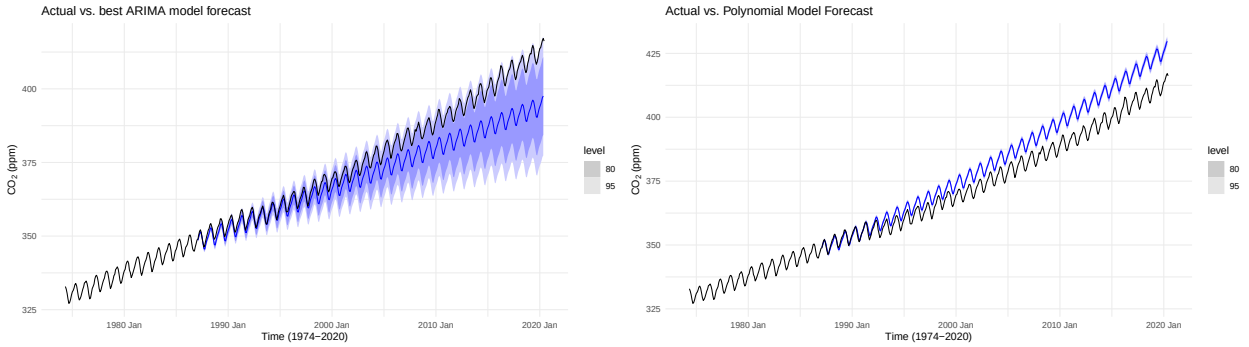

In line with the data observed in the seasonality and the $CO_2$ average concentration plots, the annual growth rate plot further confirms the accelerating trend of increasing $CO_2$ concentrations.

## Compare overall forecasting performance of models

### Using data split into train and test

```
p1 <- sarimaES1 %>%
  forecast(h = 400) %>%
  autoplot() +
  autolayer(
    co2.weekly.imputed.ts %>%
      index_by(Month = ~ yearmonth(.)) %>%
      summarise(MonthlyAverage = mean(co2ppm)),
    MonthlyAverage
  ) +
  theme_minimal() +
  labs(title= 'Actual vs. best ARIMA model forecast', x = "Time (1974-2020)",
       y = expression(paste("CO"[2]~"(ppm)")))

# retrain poly model with co2_train to enable forecast precision comparison
poly.model.quad.v2 <- co2_train %>%
  model(TSLM(value ~ trend() + I(trend()^2) + season() ))

p2 <-poly.model.quad.v2 %>%
  forecast(h = 400) %>%
  autoplot() +
  autolayer(
    co2.weekly.imputed.ts %>%
```

```
      index_by(Month = ~ yearmonth(.)) %>%
      summarise(MonthlyAverage = mean(co2ppm)),
    MonthlyAverage
  ) +
  theme_minimal() +
  labs(title='Actual vs. Polynomial Model Forecast', x = "Time (1974-2020)",
       y = expression(paste("CO"[2]~"(ppm)")))

grid.arrange(p1,p2, ncol=2)
```



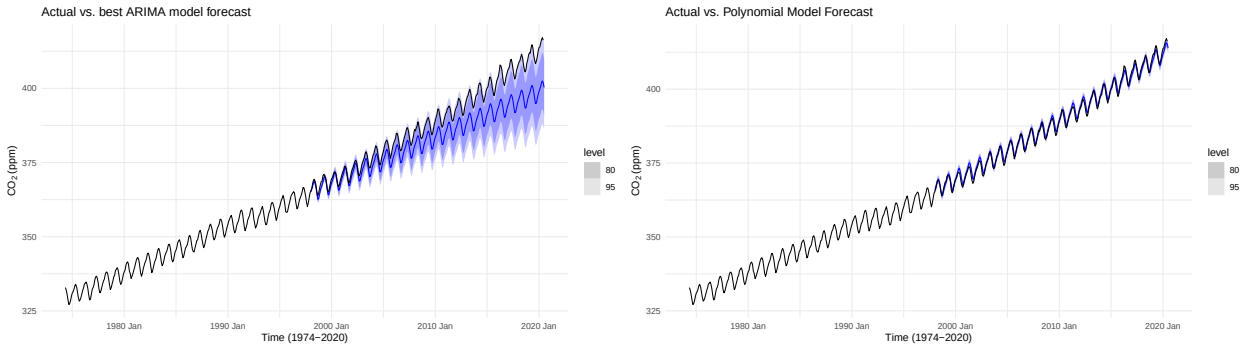

If the co2 data set is split into training and test data the ARIMA model underestimates the 2020 forecast and the polynomial model overestimates the 2020 forecast.

### Using entire co2 dataset to train models

The following comparison is carried out by comparing models that were *trained using the entire co2 dataset*:

```
sarimaES1.alldata <- co2 %>%
  model(ARIMA(value ~ 0 + pdq(3,1,3) + PDQ(0,1,1),
                        stepwise=FALSE))

p1 <- sarimaES1.alldata %>%
  forecast(h = 271) %>%
  autoplot() +
  autolayer(
    co2.weekly.imputed.ts %>%
      index_by(Month = ~ yearmonth(.)) %>%
      summarise(MonthlyAverage = mean(co2ppm)),
    MonthlyAverage
  ) +
  theme_minimal() +
  labs(title= 'Actual vs. best ARIMA model forecast', x = "Time (1974-2020)",
       y = expression(paste("CO"[2]~"(ppm)")))


p2 <-poly.model.quad %>%
```

```
  forecast(h = 271) %>%
  autoplot() +
  autolayer(
    co2.weekly.imputed.ts %>%
      index_by(Month = ~ yearmonth(.)) %>%
      summarise(MonthlyAverage = mean(co2ppm)),
    MonthlyAverage
  ) +
  theme_minimal() +
  labs(title='Actual vs. Polynomial Model Forecast', x = "Time (1974-2020)",
       y = expression(paste("CO"[2]~"(ppm)")))

grid.arrange(p1,p2, ncol=2)
```




However, if the entire `co2` data set is used to train the models, while the performance of the ARIMA model is worse than previously the polynomial model forecast performance overlaps entirely with the actual data from `co2_weekly_mlo.txt`.

### Generate Month Average Series

The weekly data is converted into month average series by calculating the number of days pertaining to each month of a weekly observation and then calculating the weighted average for the month based on the number of days in each week.

```
#  create monthly dataset from weekly data
co2.monthly <- co2.weekly.imputed.ts %>%
  index_by(index = yearmonth(date)) %>%
  summarise(co2ppm = mean(co2ppm))
```

### Part 5 (4 points)

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

**Seasonally adjusted the weekly NOAA data and split test data**

```
# Train/Test Split
co2.weekly.train <- components(co2.weekly.stl) %>%
  filter_index(~ "2018-01-01")

co2.weekly.test <- components(co2.weekly.stl) %>%
  filter_index("2018-01-01" ~ .)

# plot training data
co2.weekly.train %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = co2ppm), color = "grey") +
  geom_line(aes(y = season_adjust), color = "lightcoral") +
  theme_classic() +
  labs(y=expression(paste("CO"[2]~"(ppm)")),
       x="Time (1974-2020)")
```

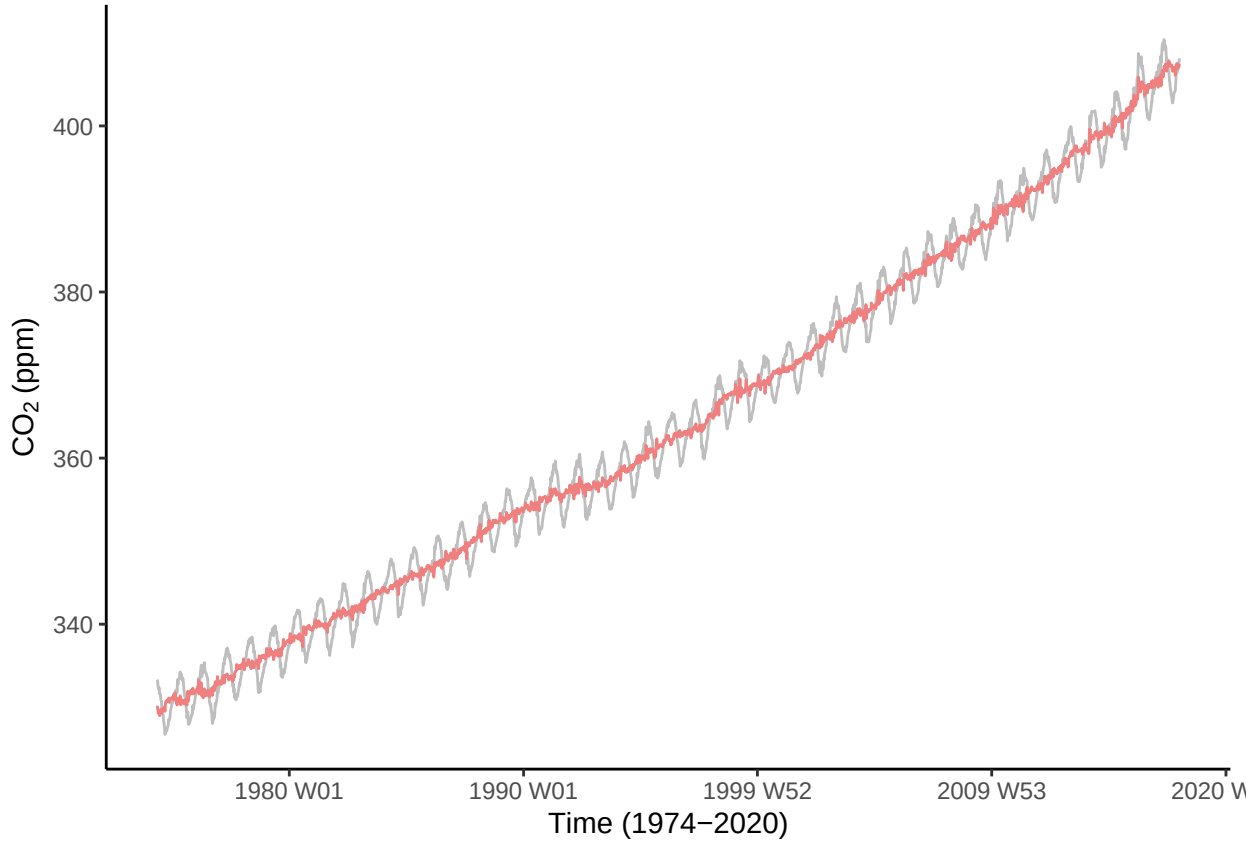



Figure 35: Seasonally adjusted and non-seasonally adjusted CO2 ppm

### Examine Differences to Judge Stationarity

```
# differenced data
co2.weekly.train %>%
  transmute(
    `co2ppm` = co2ppm, # raw co2
    `sa` = season_adjust,# log co2
    `co2ppm.d1` = difference(co2ppm), # first diff
    `sa.d1` = difference(season_adjust), # first log diff
    `co2ppm.d2` = difference(difference(co2ppm)), # second diff
    `sa.d2` = difference(difference(season_adjust)), # second log diff
    `co2ppm.D1` = difference(co2ppm, 12), # seasonal first diff co2
    `sa.D1` = difference(season_adjust, 12), # seasonal first diff in log co2
    `co2ppm.D1d1` = difference(difference(co2ppm, 12), 1), # D=1,d=1
    `sa.D1d1` = difference(difference(season_adjust, 12), 1))  %>%#  D=1,d=1
    gather("series","value",-date) %>%
    mutate(series = factor(series,
                           levels = c('co2ppm', 'sa',
                                      'co2ppm.d1','sa.d1',
                                      'co2ppm.d2', 'sa.d2',
                                      'co2ppm.D1', 'sa.D1',
                                      'co2ppm.D1d1','sa.D1d1'))) %>%
    ggplot(aes(x = date, y = value)) +
    geom_line(colour = "grey", size = 1) +
    facet_wrap(~ series, ncol = 2, scales = "free") +
    theme_classic() + theme(legend.position = "none")  +
    scale_y_continuous(name = expression(paste("CO"[2])))+
   labs(title=expression(~" "),
       x=" ")+
    theme(
    axis.title.x = element_blank(),
    axis.text.x = element_blank(),
    axis.title.y = element_text(size = 5),
    axis.text.y = element_text(size = 4))
```

The first difference of both the seasonally adjusted and non-seasonally adjusted series appear to be stationary.

### Testing for Stationarity

```
# stationarity testing using kpss
stationary.tests <- co2.weekly.train %>%

  # taking first differences for testing
   mutate(nsa.d1 = difference(co2ppm, 1),
```

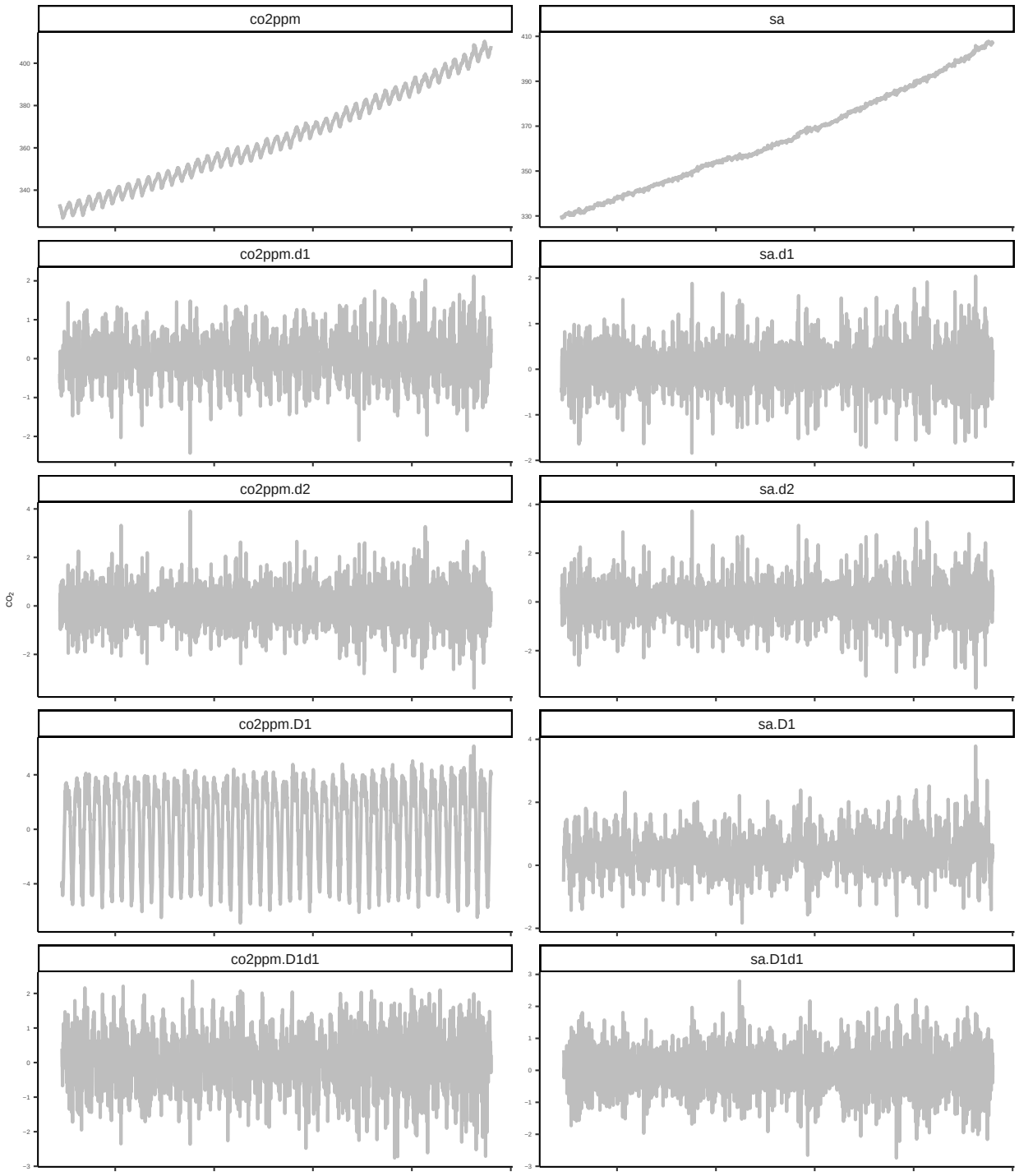



Figure 36: Differences used to obtain stationarity

```
          sa.d1 = difference(season_adjust, 1)) %>%

  # selecting columns for tests
  select(co2ppm, season_adjust, nsa.d1, sa.d1) %>%

  # applying stationarity and differencing tests
  features_all(features = list(unitroot_ndiffs,
                               unitroot_nsdiffs,
                               unitroot_kpss))
```

Testing for stationarity of the original and first differences of each series produce the following results.

The original non-seasonally adjusted series produces a KPSS statistic of 24.9124666 with a p-value of 0.01, which confirms that the series is non-stationary. The unit root test indicates we need to take the the first difference of the series, but no seasonal difference, which produces a new KPSS statistic or 0.0298948 with a corresponding p-value of 0.1 failing to reject the null hypothesis of a stationary series. The seasonally adjusted series produces similar results with an initial KPSS stat of 25.1265162 and p-value 0.01 with the same differencing scheme with new KPSS stat of 0.2550768 with p-value 0.1. Our results indicate that taking differences $d = 1$ for both produce stationary series. We will fix these value while executing our algorithm for the autoregressive and moving average parameters in our search, allowing us to directly compare models goodness of fit via information criterion measures.

## Fit ARIMA models

### Seasonally adjusted ARIMA(p,d,q)

```
# # Subset date and data
# co2.weekly.train.sa = co2.weekly.train[,c("date","season_adjust")]
# co2.weekly.test.sa = co2.weekly.test[,c("date","co2ppm")]
# names(co2.weekly.test.sa) = c("date","season_adjust")
# # Define parameter space
# ARIMA.df.sa <- data.frame(index=rep(0, 256),
#                           p=rep(0, 256),
#                           q=rep(0, 256),
#                           P=rep(0, 256),
#                           Q=rep(0, 256),
#                           AIC=rep(0, 256),
#                           BIC=rep(0, 256),
#                           RMSE=rep(0, 256))
# # Index
# idx <- 1
# # For each unique combination of p,q, and P,Q
# for (p in 0:3){
#     for (q in 0:3){
```

```
#       for (P in 0:3){
#           for (Q in 0:3){
#           # Estimate ARIMA (pdq)(PDQ)m
#           mod <- co2.weekly.train.sa %>% model(ARIMA(season_adjust ~ 0 + pdq(p,1,q) + PDQ(P,
#           # forecast the future!
#           f <- mod %>% forecast(h=129)
#           # populate dataframe with parameters and RMSE
#          ARIMA.df.nsa[idx,] <- c(idx, p, q, P, Q,
#                                  try(select(glance(mod), AIC),silent=TRUE),
#                                  try(select(glance(mod), BIC), silent=TRUE),
#                                  sqrt(mean((f$season_adjust - co2.weekly.test.nsa$season_adjust)^2,
#           # Plus one
#           idx=idx+1
#           }
#         }
#     }
# }
# # Coerce characters to numerics
# ARIMA.df.sa$AIC = as.numeric(ARIMA.df.nsa$AIC)
# ARIMA.df.sa$BIC = as.numeric(ARIMA.df.nsa$BIC)
# ARIMA.df.sa$RMSE = as.numeric(ARIMA.df.nsa$RMSE)
# # Order by IC and RMSE
# head(ARIMA.df.nsa[order(ARIMA.df.sa$AIC,
#                         ARIMA.df.sa$BIC,
#                         ARIMA.df.sa$RMSE),])
#
# # Ordered df
# odf5sa = ARIMA.df.sa[order(ARIMA.df.sa$AIC,
#                         ARIMA.df.sa$BIC,
#                         ARIMA.df.sa$RMSE),]
#
# # Save top 100 models in df
# write.csv(odf5sa[c(1:100),],
#           "ARIMAdf5SA.csv")

# Open & view df
#head(read.csv( "ARIMAdf5SA.csv"))
```

**Non seasonally adjusted ARIMA(p,d,q)(P,D,Q)s**

```
# # Subset date and data
# co2.weekly.train.nsa = co2.weekly.train[,c("date","co2ppm")]
# co2.weekly.test.nsa = co2.weekly.test[,c("date","co2ppm")]
#
# # Define parameter space
```

```
# ARIMA.df.nsa <- data.frame(index=rep(0, 256),
#                            p=rep(0, 256),
#                            q=rep(0, 256),
#                            P=rep(0, 256),
#                            Q=rep(0, 256),
#                            AIC=rep(0, 256),
#                            BIC=rep(0, 256),
#                            RMSE =rep(0, 256))
# # Index
# idx <- 1
# # For each unique combination of p,q, and P,Q
# for (p in 0:3){
#     for (q in 0:3){
#       for (P in 0:3){
#           for (Q in 0:3){
#
#           # Estimate ARIMA (pdq)(PDQ)m
#           mod <- co2.weekly.train.nsa %>% model(ARIMA(co2ppm ~ 0 + pdq(p,1,q) + PDQ(P,1,Q)))
#           # forecast the future!
#           f <- mod %>% forecast(h=129)
#
#           # populate dataframe with parameters and RMSE
#          ARIMA.df.nsa[idx,] <- c(idx, p, q, P, Q,
#                                  try(select(glance(mod), AIC),silent=TRUE),
#                                  try(select(glance(mod), BIC), silent=TRUE),
#                                  sqrt(mean((f$co2ppm - co2.weekly.test.nsa$co2ppm)^2)))
#           # Plus one
#           idx=idx+1
#           }
#         }
#     }
# }
#
# # Coerce characters to numerics
# ARIMA.df.nsa$AIC = as.numeric(ARIMA.df.nsa$AIC)
# ARIMA.df.nsa$BIC = as.numeric(ARIMA.df.nsa$BIC)
# ARIMA.df.nsa$RMSE = as.numeric(ARIMA.df.nsa$RMSE)
#
# # Order by IC and RMSE
# head(ARIMA.df.nsa[order(ARIMA.df.nsa$AIC,
#                         ARIMA.df.nsa$BIC,
#                         ARIMA.df.nsa$RMSE),])
#
# # Ordered df
# odf5nsa = ARIMA.df.nsa[order(ARIMA.df.nsa$AIC,
#                         ARIMA.df.nsa$BIC,
#                         ARIMA.df.nsa$RMSE),]
```

```
#
# # Save top 100 models in df
# write.csv(odf5nsa[c(1:100),],
#           "ARIMAdf5NSA.csv")

# Open & view df
#head(read.csv("ARIMAdf5NSA.csv"))
```

### Models performance in-sample and (psuedo-) out-of-sample

#### Seasonally adjusted data

```
# Order by IC and RMSE
head(read.csv("ARIMAdf5SA.csv"))
```

```
##     X index p q P Q      AIC      BIC     RMSE
## 1 161   161 2 2 0 0 2244.051 2272.702 2.570165
## 2 162   162 2 2 0 1 2244.051 2272.702 2.570165
## 3 163   163 2 2 0 2 2244.051 2272.702 2.570165
## 4 164   164 2 2 0 3 2244.051 2272.702 2.570165
## 5 165   165 2 2 1 0 2244.051 2272.702 2.570165
## 6 166   166 2 2 1 1 2244.051 2272.702 2.570165
```

#### Non-seasonally adjusted data

```
# Order by IC and RMSE
head(read.csv("ARIMAdf5NSA.csv"))
```

```
##     X index p q P Q      AIC      BIC     RMSE
## 1 241   241 3 3 0 0 3272.644 3312.755 5.927339
## 2 242   242 3 3 0 1 3272.644 3312.755 5.927339
## 3 243   243 3 3 0 2 3272.644 3312.755 5.927339
## 4 244   244 3 3 0 3 3272.644 3312.755 5.927339
## 5 245   245 3 3 1 0 3272.644 3312.755 5.927339
## 6 246   246 3 3 1 1 3272.644 3312.755 5.927339
```

### Fit Polynomial Time-Trend to NOAA Weekly SA data

```
linear.sa.fit <- co2.weekly.train %>%
  select(date, season_adjust) %>%
  model(TSLM(season_adjust ~ trend() ))
```

```r
quadratic.sa.fit <- co2.weekly.train %>%
  select(date, season_adjust) %>%
  model(TSLM(season_adjust ~ trend() + I(trend()^2)))

cubic.sa.fit <- co2.weekly.train %>%
  select(date, season_adjust) %>%
  model(TSLM(season_adjust ~ trend() + I(trend()^2) + I(trend()^3)))

colors = c("Actuals" = "grey", "Linear" = "lightcoral",
           "Quadratic" = "cornflowerblue", "Cubic" = "lightgreen")

# plot fit
co2.weekly.train %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = season_adjust), color = "grey") +
  geom_line(data = augment(linear.sa.fit),
            aes(x = date, y = .fitted, color = "Linear"),
            size = .3) +
  geom_line(data = augment(quadratic.sa.fit),
            aes(x = date, y = .fitted, color = "Quadratic"),
            size = .3) +
  geom_line(data = augment(cubic.sa.fit),
            aes(x = date, y = .fitted, color = "Cubic"),
            size = .3) +
  theme_minimal() +
  labs(y = expression("CO"[2]~"ppm"),
  x = "Year Week",
  color = "Guide") +
  scale_color_manual(values = colors)
```

### Linear Residuals

```r
linear.sa.fit %>% gg_tsresiduals()
```

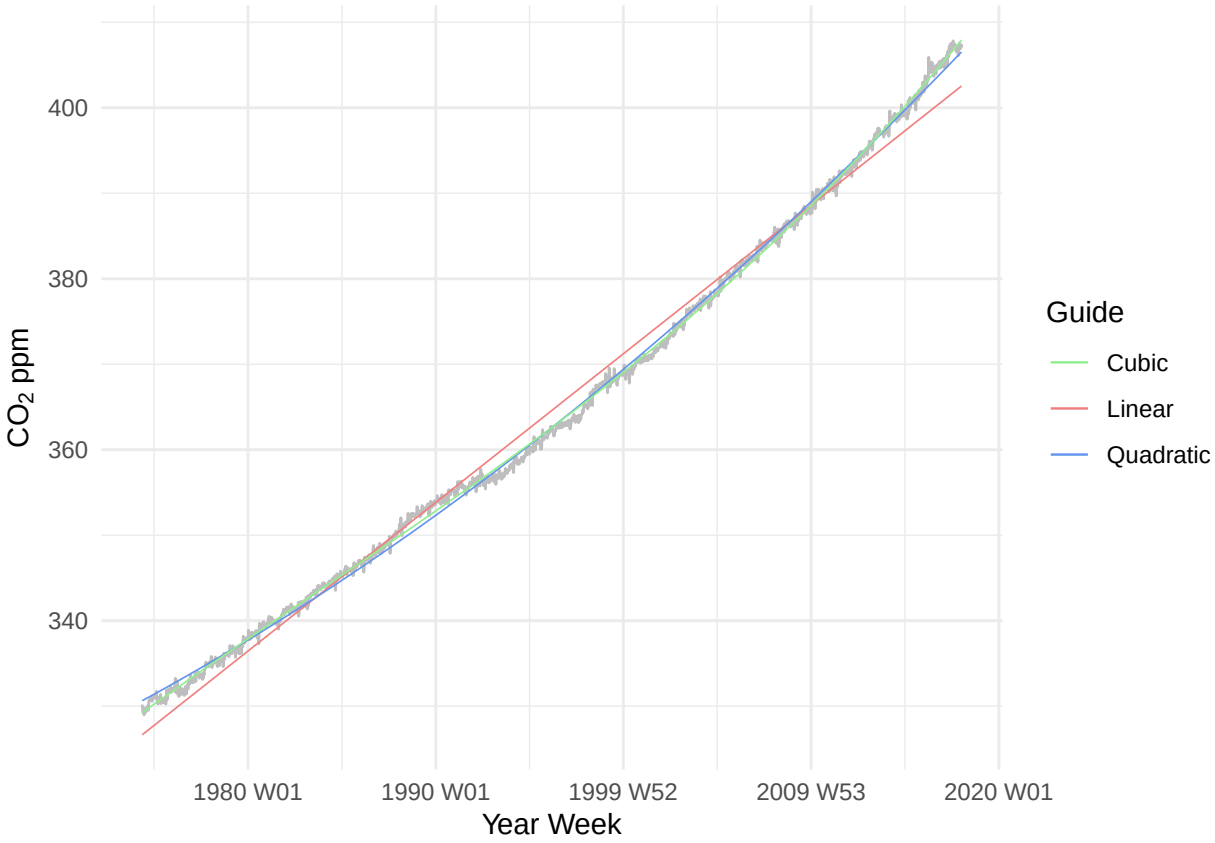
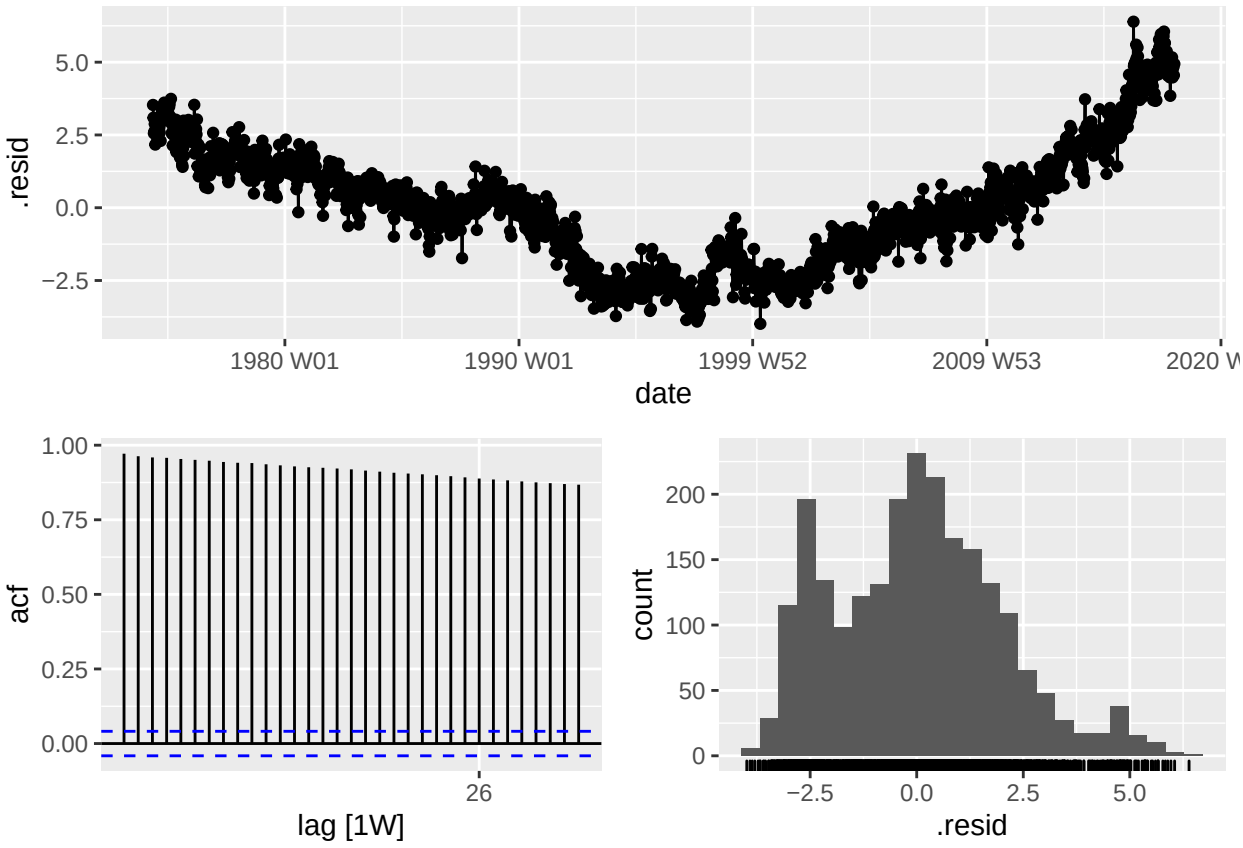


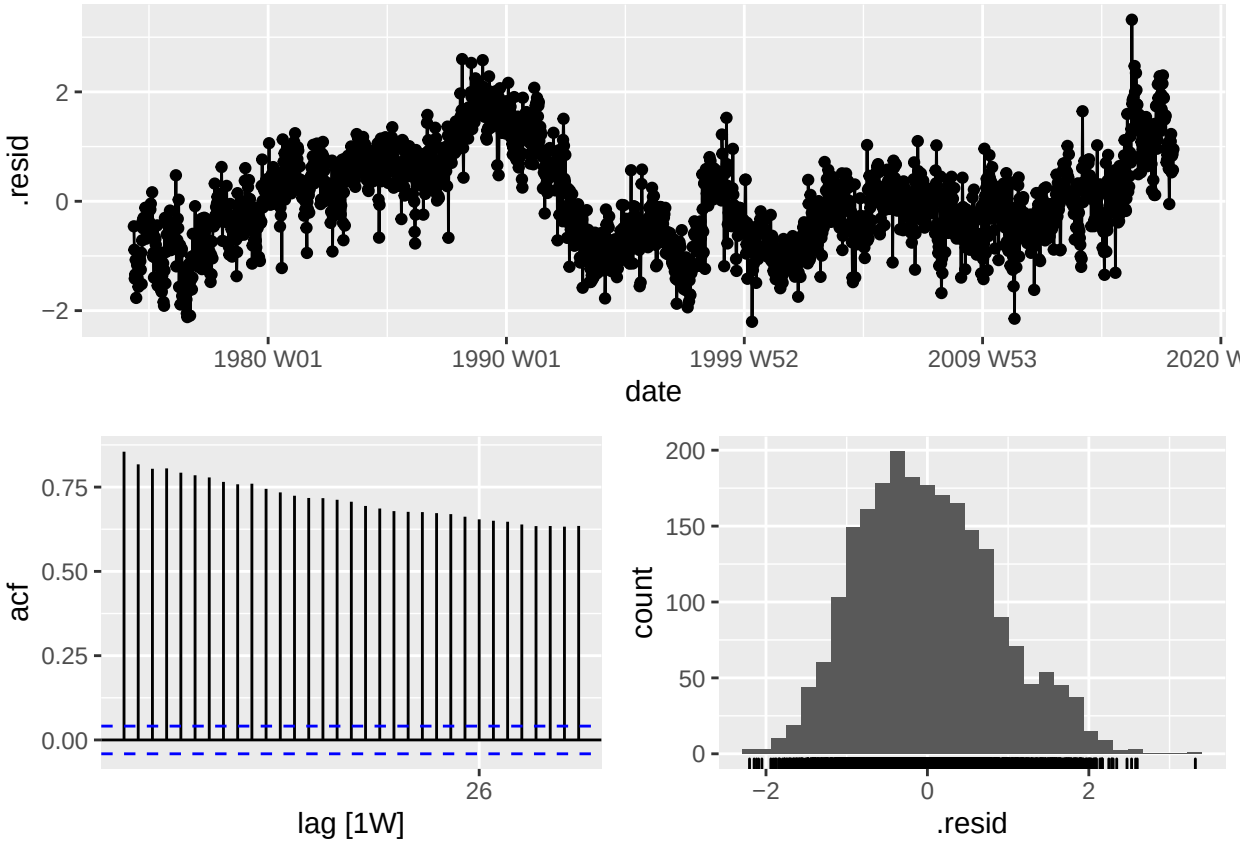

Figure 37: Fitted time trend models




- Clearly non-white noise residuals
- Nont normally distribted residuals
- Residuals high autocorrelation

### Quadratic Residuals

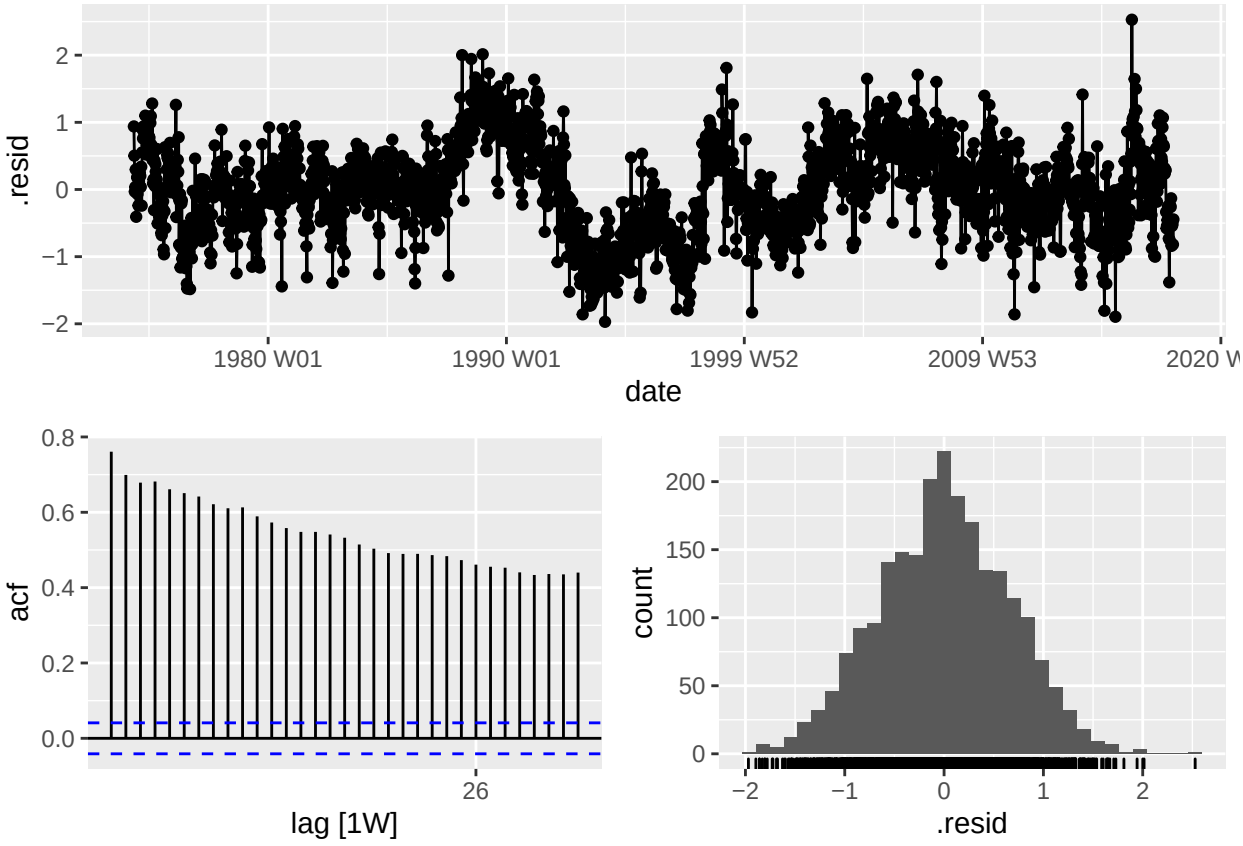
```
quadratic.sa.fit %>% gg_tsresiduals()
```




- Still non-white noise residuals
- More normally distribted residuals
- Residuals high autocorrelation

### Cubic Residuals

```
cubic.sa.fit %>% gg_tsresiduals()
```




- Less non-white noise residuals
- Plausibly normally distribted residuals
- Residuals high autocorrelation

### Polynomial Forecasts

```
lin.forecast <- linear.sa.fit %>% forecast(h = 134)
lin.forecast.m <- lin.forecast %>%
  index_by(yrmonth = yearmonth(date)) %>%
  summarise(season_adjust = mean(.mean))

qud.forecast <- quadratic.sa.fit %>% forecast(h = 134)
qud.forecast.m <- qud.forecast %>%
  index_by(yrmonth = yearmonth(date)) %>%
  summarise(season_adjust = mean(.mean))

cub.forecast <- cubic.sa.fit %>% forecast(h = 134)
cub.forecast.m <- cub.forecast %>%
  index_by(yrmonth = yearmonth(date)) %>%
  summarise(season_adjust = mean(.mean))
```

```r
# arim.forecast <- NA

colors = c("Linear" = "lightcoral",
           "Quadratic" = "cornflowerblue", "Cubic" = "lightgreen")

co2.weekly.test %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = season_adjust), color = 'grey') +
  geom_line(data = lin.forecast.m, aes(x = yrmonth, y = season_adjust, colour = "Linear")) +
  geom_line(data = qud.forecast.m, aes(x = yrmonth, y = season_adjust, colour = "Quadratic")) +
  geom_line(data = cub.forecast.m, aes(x = yrmonth, y = season_adjust, colour = "Cubic")) +
  theme_minimal() +
  labs(x = "Time",
       y = expression("CO"[2] ~ "ppm"),
       color = "Forecast") +
  scale_color_manual(values = colors)
```


Figure 38: Polynomial forecasts

The polynomials don't fit their forecasts particularly well, even against the seasonally adjusted data. The linear model performs the worst while the quadratic and cubic either seem to over or undershoot the actuals.

## Compare ARIMA model and polynomial time trend model

### Part 6 (3 points)

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

### 420 ppm and 500 ppm levels using point and interval estimates

```
# Data
head(co2.monthly)
```

```
## # A tsibble: 6 x 2 [1M]
##      index co2ppm
##      <mth>  <dbl>
## 1 1974 May   333.
## 2 1974 Jun   332.
## 3 1974 Jul   331.
## 4 1974 Aug   329.
## 5 1974 Sep   327.
## 6 1974 Oct   328.
```

```
# Coerce to ts
co2.monthly.ts <- ts(co2.monthly,
          start=c(1974,5),
          frequency = 12)
# remove extraneous column
co2.monthly.ts <- co2.monthly.ts[,2]
```

Above, we identified the $ARIMA(2,1,3)(2,1,2)_{12}$ as the *best* ARIMA model, determined by AIC. Several of the polynomial seemed to perform better, but we will proceed with the ARIMA model for part 6.

### Estimate model and confidence interval

```
# Refit best model to extract predictions, se and compute PI
mod6 <- sarima.for(co2.monthly.ts,
                   n.ahead=12*80,
                   p=2,d=1,q=3,
                   P=2,D=1,Q=2,
                   S=12, plot.all = FALSE)
```

```
# Refit model for plotting
mod6p <- co2.monthly %>%
  model(ARIMA(co2ppm ~ 0 + pdq(2,1,3) + PDQ(2,1,2)))
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
# inverse roots within unit circle
gg_arma(mod6p)
```

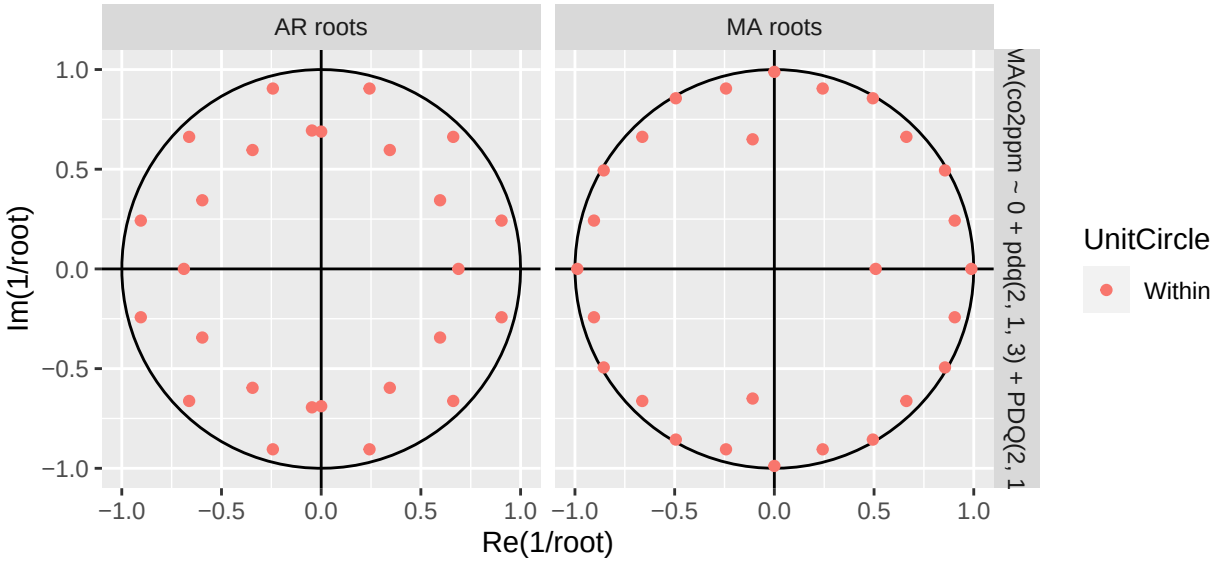



Figure 39: check unit roots

```
# Build dataframe for results
df6 = data.frame("Time" = tk_index(mod6$pred),
                 "FittedValues"=mod6$pred,
                 "StandardErrors"=mod6$se,
                 "Lower"= mod6$pred - 1.96 * mod6$se,
                 "Upper"= mod6$pred + 1.96 * mod6$se)

# Generate forecasts
fc2100 <- mod6p %>% forecast(h=960)
```

**First and last time 420ppm and 500ppm are detected & 2100 forecast:**

```
### First time 420 and 500ppm are detected
## Point estimates and fitted values
# 420 ppm
fc2100$index[min(which(round(df6$Lower,0) >= 420))] # "2022 Apr"
```

```
## <yearmonth[1]>
## [1] "2022 Apr"
```

```
fc2100$index[min(which(round(fc2100$.mean,0) >= 420))] # "2021 May"
```

```
## <yearmonth[1]>
## [1] "2021 May"
```

```
fc2100$index[min(which(round(df6$Upper,0) >= 420))] # "2021 Apr"
```

```
## <yearmonth[1]>
## [1] "2021 Apr"
```

```
# 500 ppm
fc2100$index[min(which(round(df6$Lower,0) >= 500))] # "2071 May"
```

```
## <yearmonth[1]>
## [1] "2071 May"
```

```
fc2100$index[min(which(round(fc2100$.mean,0) >= 500))] # "2055 Apr"
```

```
## <yearmonth[1]>
## [1] "2055 Apr"
```

```
fc2100$index[min(which(round(df6$Upper,0) >= 500))] # "2048 Apr"
```

```
## <yearmonth[1]>
## [1] "2048 Apr"
```

```
### Last time 420 and 500ppm are detected
## Point estimates and fitted values
# 420 ppm
fc2100$index[max(which(round(df6$Lower,0) <= 420))] # "2025 Oct"
```

```
## <yearmonth[1]>
## [1] "2025 Oct"
```

```r
fc2100$index[max(which(round(fc2100$.mean,0) <= 420))] # "2023 Oct"
```

```
## <yearmonth[1]>
## [1] "2023 Oct"
```

```r
fc2100$index[max(which(round(df6$Upper,0) <= 420))] # "2022 Nov"
```

```
## <yearmonth[1]>
## [1] "2022 Nov"
```

```r
# 500 ppm
fc2100$index[max(which(round(df6$Lower,0) <= 500))] # ""2076 Sep"
```

```
## <yearmonth[1]>
## [1] "2076 Sep"
```

```r
fc2100$index[max(which(round(fc2100$.mean,0) <= 500))] # "2057 Oct"
```

```
## <yearmonth[1]>
## [1] "2057 Oct"
```

```r
fc2100$index[max(which(round(df6$Upper,0) <= 500))] # "2049 Oct"
```

```
## <yearmonth[1]>
## [1] "2049 Oct"
```

```r
# Build data frame
data.frame("Predictions" = c(rep("420ppm",2),
                             rep("500ppm",2),
                             "2100"),
           "FirstOrLastTime" = c(rep(c("first", "last"),2), NaN),
           "PointEstimates" = c(toString(fc2100$index[min(which(round(fc2100$.mean,0) >= 420))]
                                toString(fc2100$index[max(which(round(fc2100$.mean,0) <= 420))]
                                toString(fc2100$index[min(which(round(fc2100$.mean,0) >= 500))]
                                toString(fc2100$index[max(which(round(fc2100$.mean,0) <= 500))]
                                round(mean(df6$FittedValues[df6$Time > 2100]),2)),
           "LowerPI"= c(toString(fc2100$index[min(which(round(df6$Lower,0) >= 420))]),
                        toString(fc2100$index[max(which(round(df6$Lower,0) <= 420))]),
                        toString(fc2100$index[min(which(round(df6$Lower,0) >= 500))]),
                        toString(fc2100$index[max(which(round(df6$Lower,0) <= 500))]),
                        round(min(df6$Lower[df6$Time > 2100]),2)),
           "UpperPI"= c(toString(fc2100$index[min(which(round(df6$Upper,0) >= 420))]),
                        toString(fc2100$index[max(which(round(df6$Upper,0) <= 420))]),
                        toString(fc2100$index[min(which(round(df6$Upper,0) >= 500))]),
                        toString(fc2100$index[max(which(round(df6$Upper,0) <= 500))]),
                        round(min(df6$Upper[df6$Time > 2100]),2)))
```

```
##   Predictions FirstOrLastTime PointEstimates  LowerPI  UpperPI
## 1      420ppm           first       2021 May 2022 Apr 2021 Apr
## 2      420ppm            last       2023 Oct 2025 Oct 2022 Nov
## 3      500ppm           first       2055 Apr 2071 May 2048 Apr
## 4      500ppm            last       2057 Oct 2076 Sep 2049 Oct
## 5        2100             NaN         605.49   533.89   673.91
```

```
# Plot forecasts
fc2100 %>%
  autoplot(co2.monthly) +
  theme_classic()+
  theme(legend.position = "none")+
  labs(x="Time (1974-2100)",
       y=expression("CO"[2]~"(ppm)"))+
  # First time a point estimate for CO2 is >= 420
  geom_point(aes(x=fc2100$index[min(which(round(df6$FittedValues,0) >= 420))],
                 y=df6$FittedValues[min(which(round(df6$FittedValues,0) >= 420))]),
                 colour="orangered", shape=1) +
  # Add label
  annotate(geom="text",
           x=fc2100$index[min(which(round(df6$FittedValues,0) >= 420))]+200,
           y=df6$FittedValues[min(which(round(df6$FittedValues,0) >= 420))]-10,
           label=expression(~bold('2021:')~"CO"[2]~"> 420ppm (418-421)"),
              color="black",size=2) +
  # First time a point estimate for CO2 is >= 500
  geom_point(aes(x=fc2100$index[min(which(round(df6$FittedValues,0) >= 500))],
                 y=df6$FittedValues[min(which(round(df6$FittedValues,0) >= 500))]),
                 colour="orangered", shape=1) +
    # Add label
  annotate(geom="text",
           x=fc2100$index[min(which(round(df6$FittedValues,0) >= 500))]+200,
           y=df6$FittedValues[min(which(round(df6$FittedValues,0) >= 500))]-10,
           label=expression(~bold('2055:')~"CO"[2]~"> 500 ppm (475-525)"),
              color="black",size=2) +
  # Estimates in 2100
  geom_point(aes(x=fc2100$index[[955]],
                 y=min(df6$FittedValues[df6$Time > 2100])),
                 colour="orangered", shape=1 )+
  # Add label
  annotate(geom="text",
           x=fc2100$index[[955]]-200,
           y=min(df6$FittedValues[df6$Time > 2100])+10,
           label=expression(~bold('2100:')~"CO"[2]~"~ 606 ppm (528-680)"),
              color="black",size=2)
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'

## Warning in is.na(x): is.na() applied to non-(list or vector) of type
## 'expression'
```

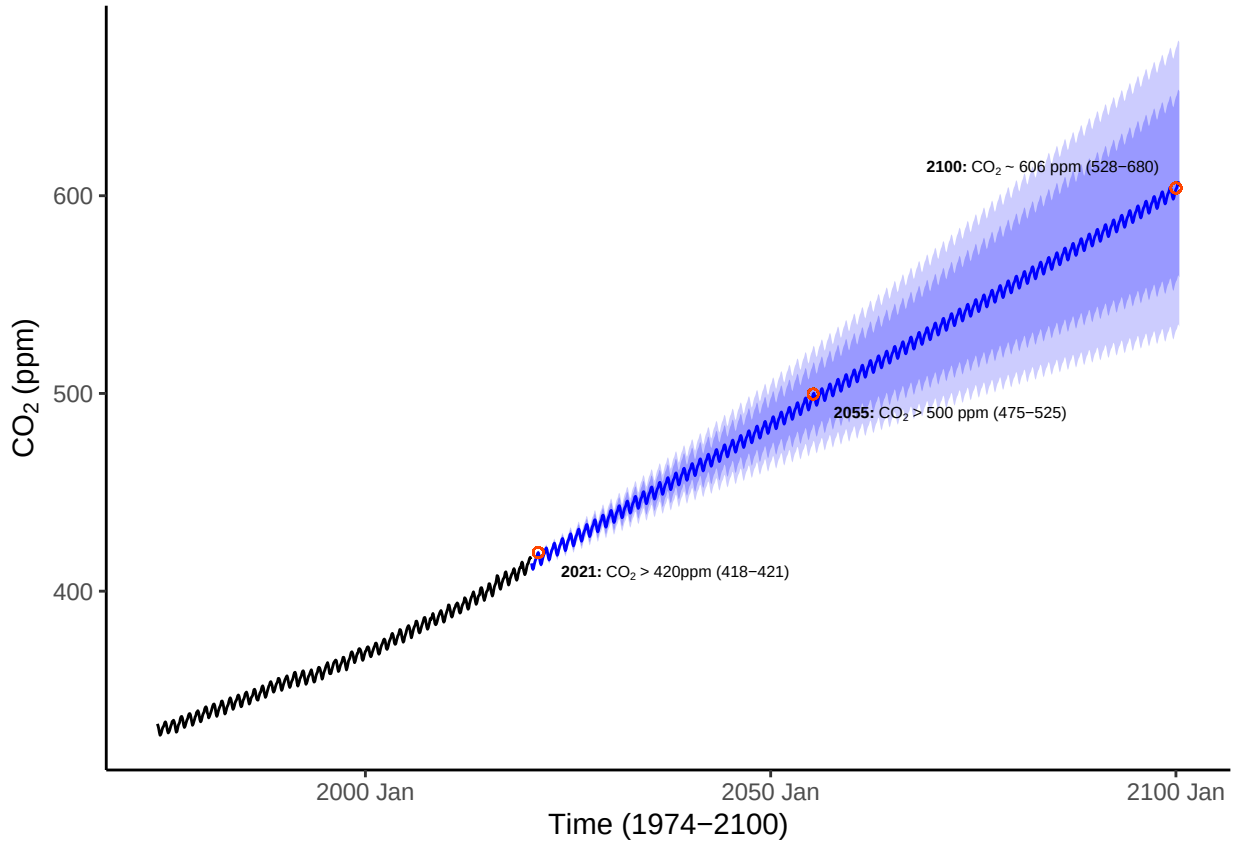



Figure 40: Forecasts for 2100 & first time points where CO2 levels reach 420 and 500ppm

## Confidence in predictions

**Our confidence in the predictions will be expressed and discussed both quantitatively and qualitatively.**

First, we recognize the quantitative uncertainty around the above predictions. This uncertainty is expressed as prediction intervals and grows over time (see figure X). Thus, our confidence is inversely related to the horizon of the forecast- we are more confident about the predictions around $CO_2$ levels than we are about the distant 2100 forecast.

Second, the uncertainty represented in the prediction intervals do not capture exogenous sources of variation that will undoubtedly affect $CO_2$ levels. To introduce any of the other enumerable sources of variation is to further erode our already dismal confidence. For example, consider the

case of mundane source or sink of $CO_2$, like car emissions or the ocean. The current forecasts assume these sources and sinks will behave the same over time. Will they? Electric and or hybrid cars will violate this assumption, as will something like a global pandemic where normal driving patterns are disturbed. Thus, even if we do not invoke complicated biogeochemical phenomenon, we can still see that $CO_2$ levels are subject to mundane to exotic sources of variation.

Thus, we are not confident in these predictions. If we had to bet, we would bet more capital on the closer predictions (e.g. the first and last instances of 420ppm of $C0_2$) than on the long term predictions (e.g. $CO_2$ levels in 2100).