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

Due Monday October 25 2021 11:59pm

Instructions (Please Read Carefully):

- Submit by the due date. **Late submissions will not be accepted**
- No page limit, but be reasonable
- Do not modify fontsize, margin or line-spacing settings
- One student from each group should submit the lab to their student github repo by the deadline
- Submit two files:
  1. A pdf file that details your answers. Include all R code used to produce the answers
  2. The R markdown (Rmd) file used to produce the pdf file

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

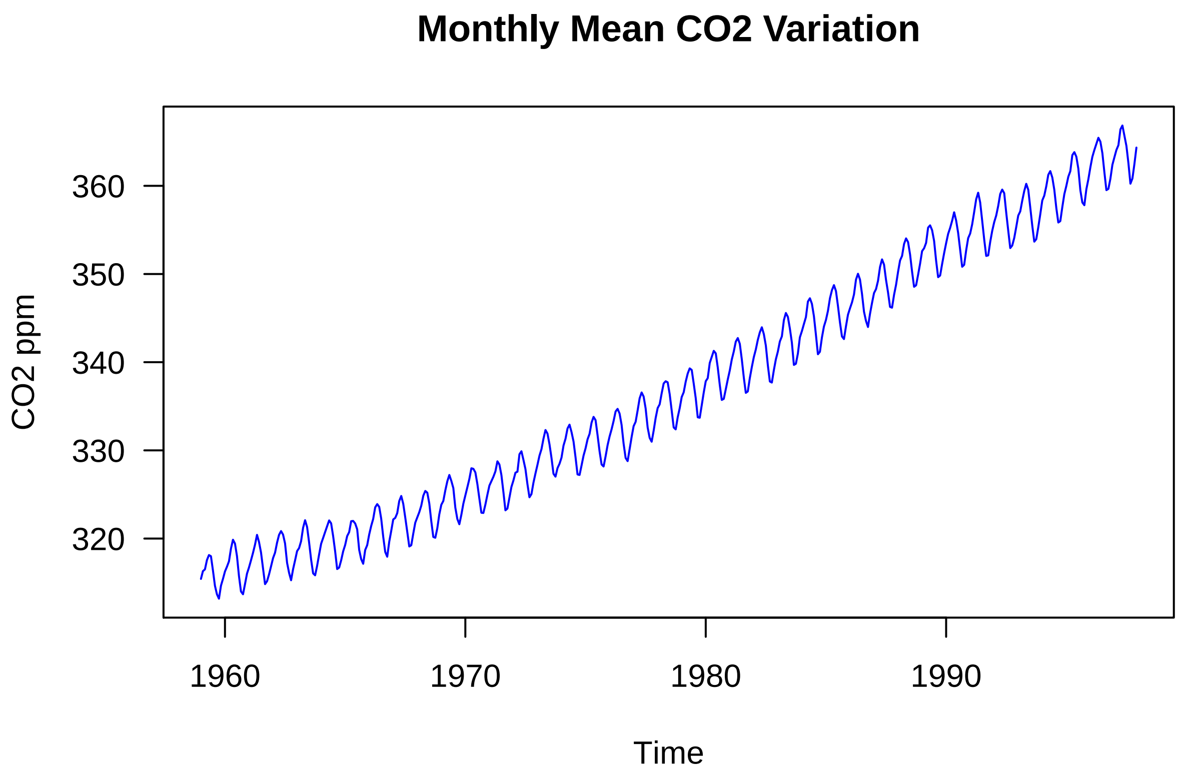
The Keeling Curve

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

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

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

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

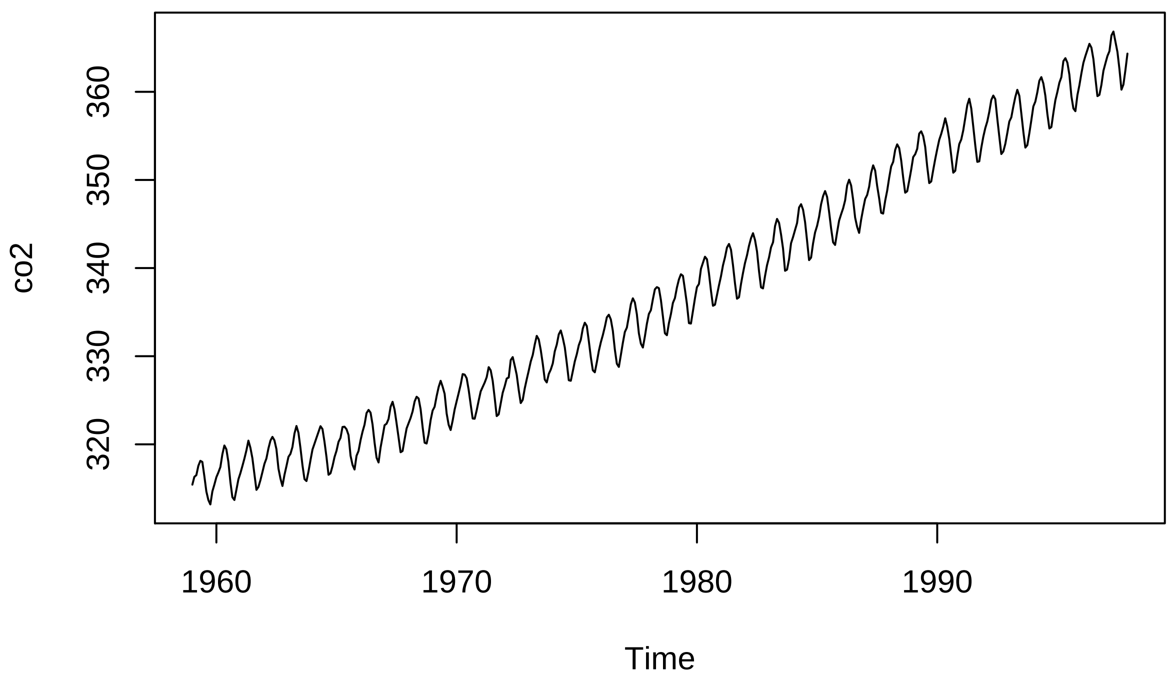


```
#apt-get install texlive-xetex
```

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.

```
#https://github.com/rstudio/bookdown/issues/292
plot(co2)
```



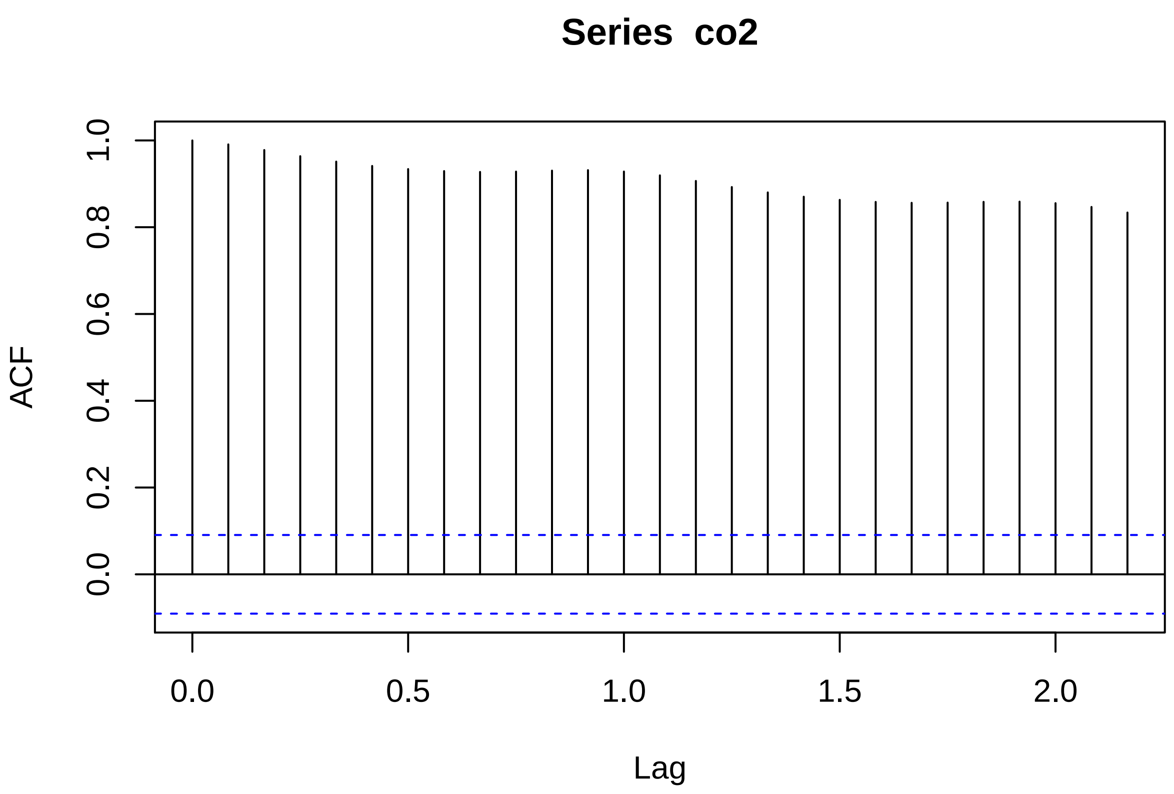
```
summary(co2)
```

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

```
str(co2)
```

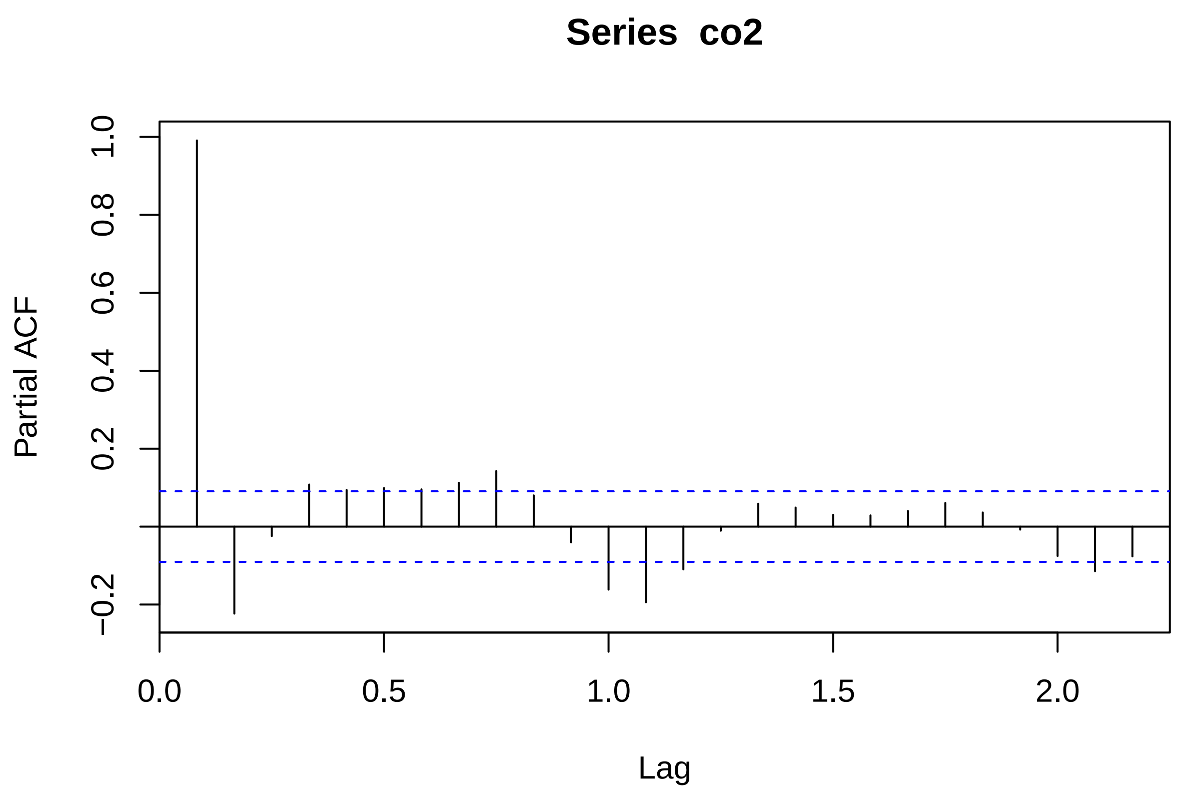
```
##  Time-Series [1:468] from 1959 to 1998: 315 316 316 318 318 ...
```

```
acf(co2)
```



```
pacf(co2)
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

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

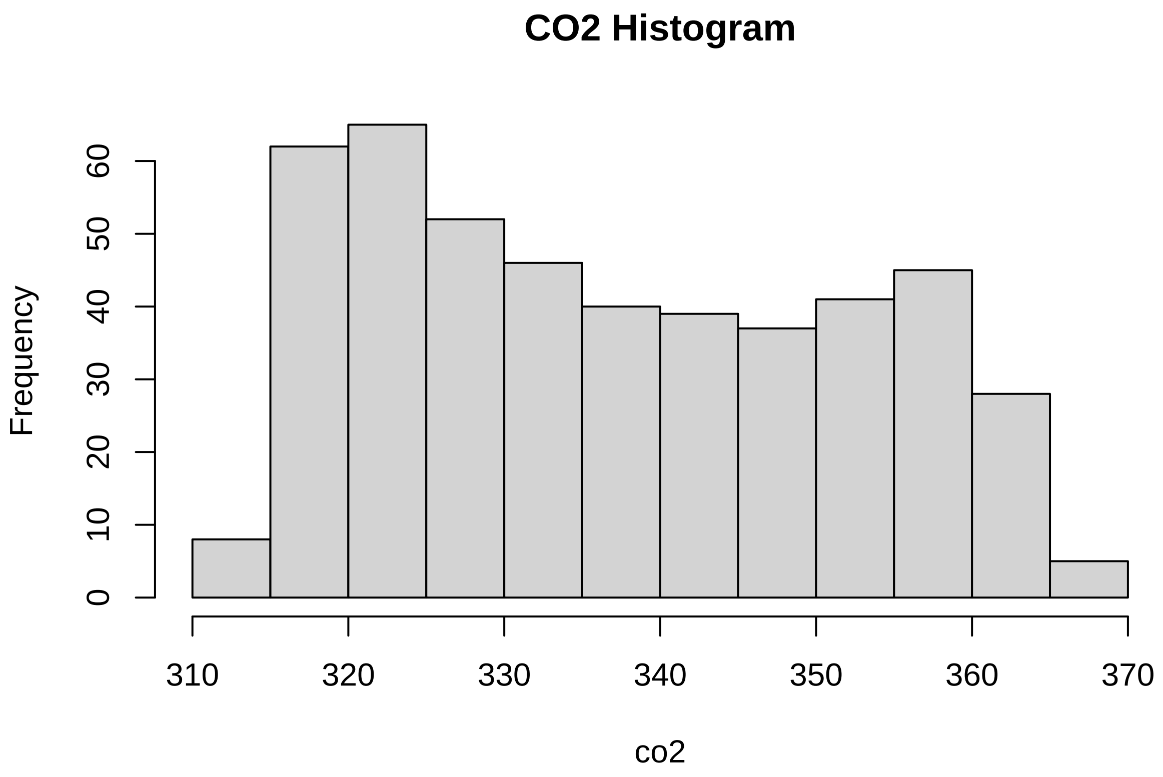


```
co2df=data.frame(
  co2data=as.vector(co2),
  co2time=time(co2)
  )
describe(co2df)
```

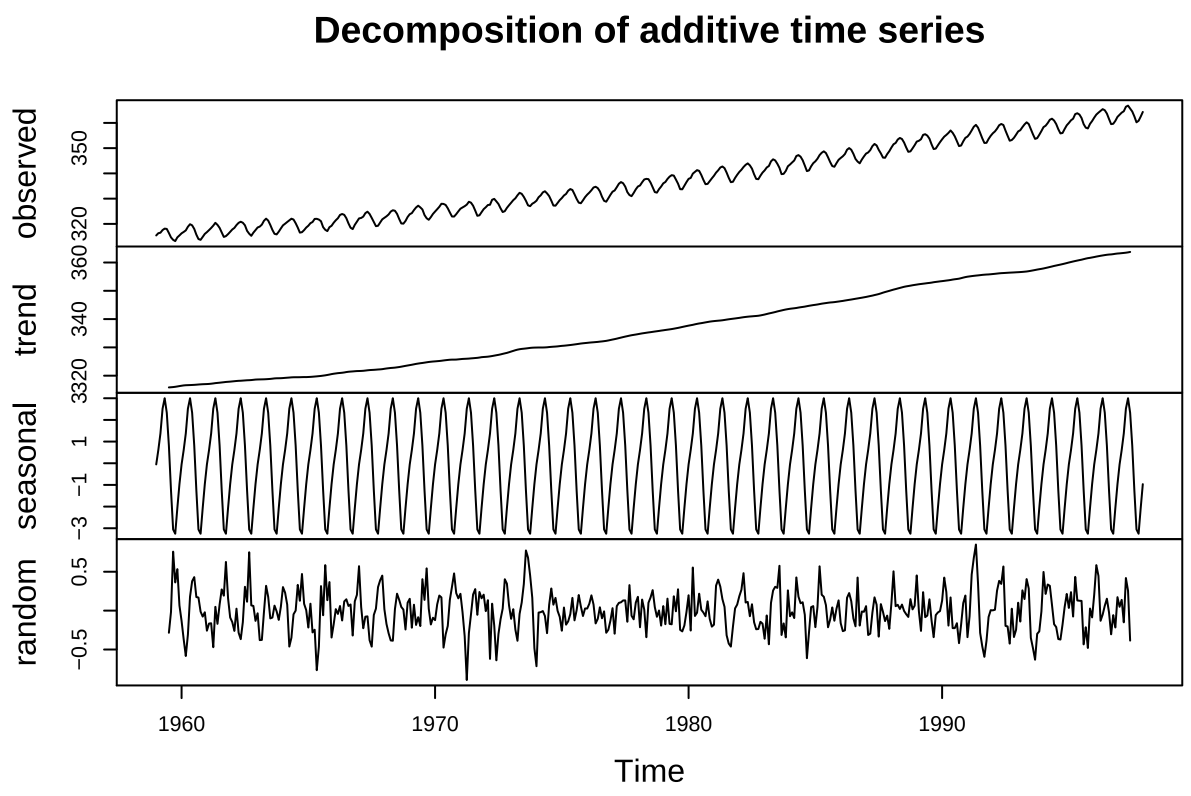
```
## co2df
##
##  2  Variables      468  Observations
## --------------------------------------------------------------------------------
## co2data
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      468        0      451        1    337.1    17.21    316.6    318.4
##      .25      .50      .75      .90      .95
##    323.5    335.2    350.3    358.9    361.6
##
## lowest : 313.18 313.68 314.00 314.65 314.66, highest: 365.01 365.45 365.68 366.40 366.84
## --------------------------------------------------------------------------------
## co2time
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      468        0      468        1     1978    13.03     1961     1963
##      .25      .50      .75      .90      .95
##     1969     1978     1988     1994     1996
```

```
##
## lowest : 1959.000 1959.083 1959.167 1959.250 1959.333
## highest: 1997.583 1997.667 1997.750 1997.833 1997.917
## --------------------------------------------------------------------------------
```

```
hist(co2,main="CO2 Histogram")
```



```
co2_decomp=decompose(co2)
plot(co2_decomp)
```



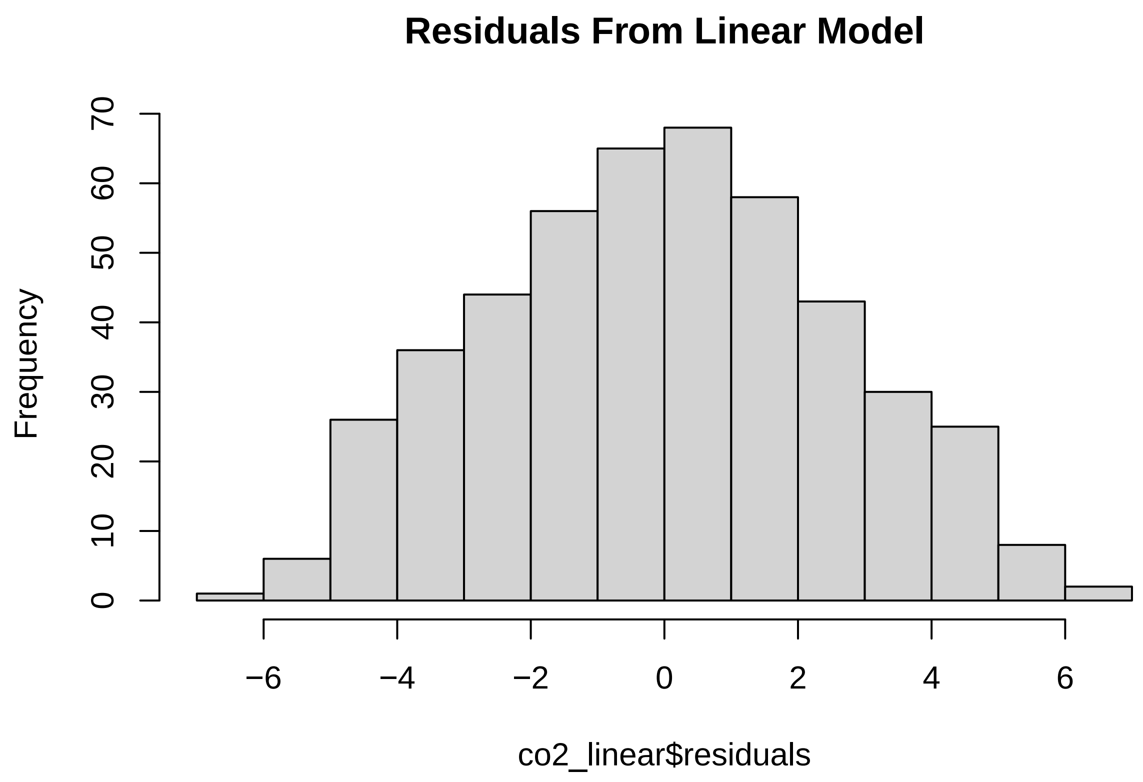
Part 2 (3 points)

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

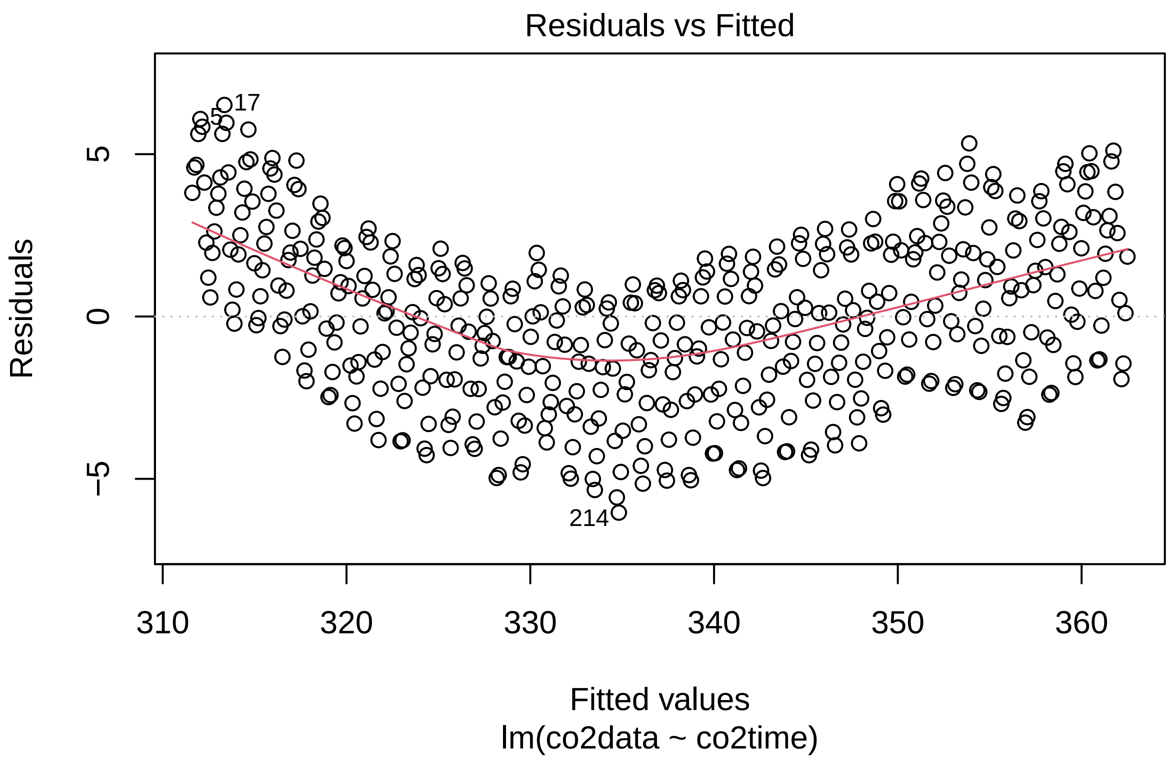
```
########################################
#first linear
co2_linear=lm(co2data~co2time,data=co2df)
summary(co2_linear)
```

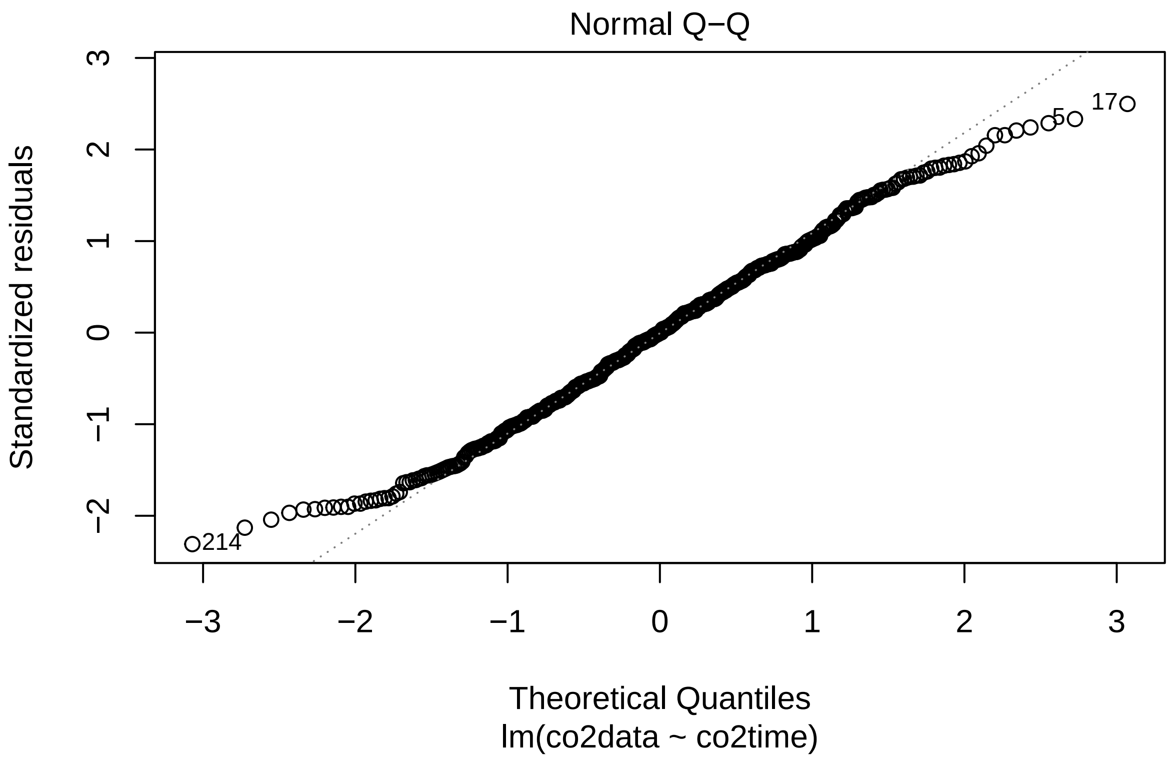
```
##
## Call:
## lm(formula = co2data ~ co2time, data = co2df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0399 -1.9476 -0.0017  1.9113  6.5149
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.250e+03  2.127e+01  -105.8   <2e-16 ***
## co2time      1.308e+00  1.075e-02   121.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.618 on 466 degrees of freedom
## Multiple R-squared:  0.9695, Adjusted R-squared:  0.9694
## F-statistic: 1.479e+04 on 1 and 466 DF,  p-value: < 2.2e-16
```

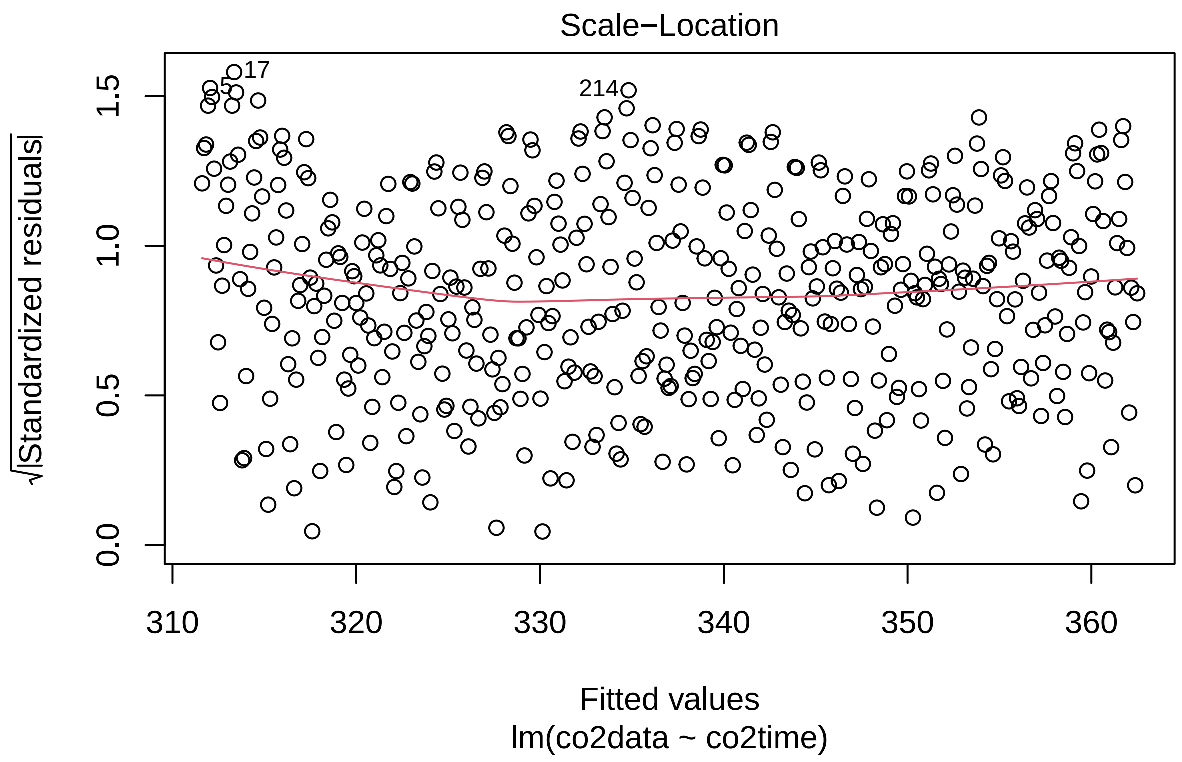
```
hist(co2_linear$residuals,main="Residuals From Linear Model")
```

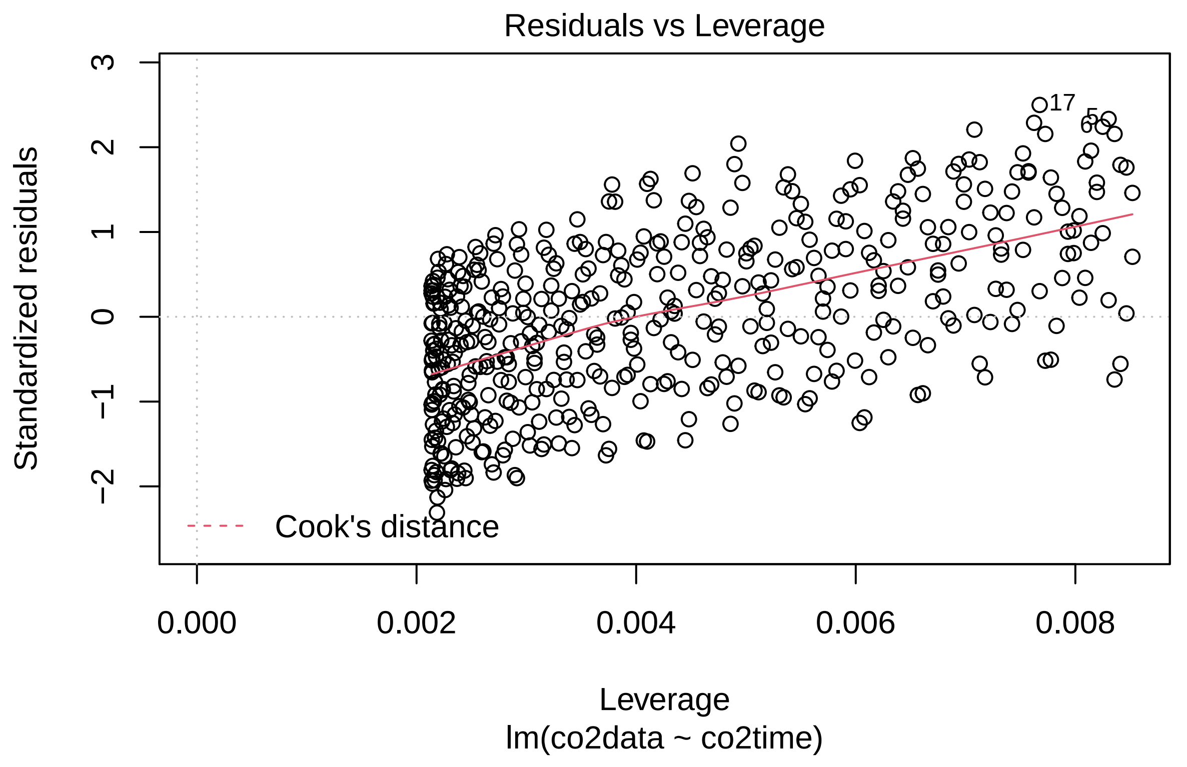


```
plot(co2_linear)
```

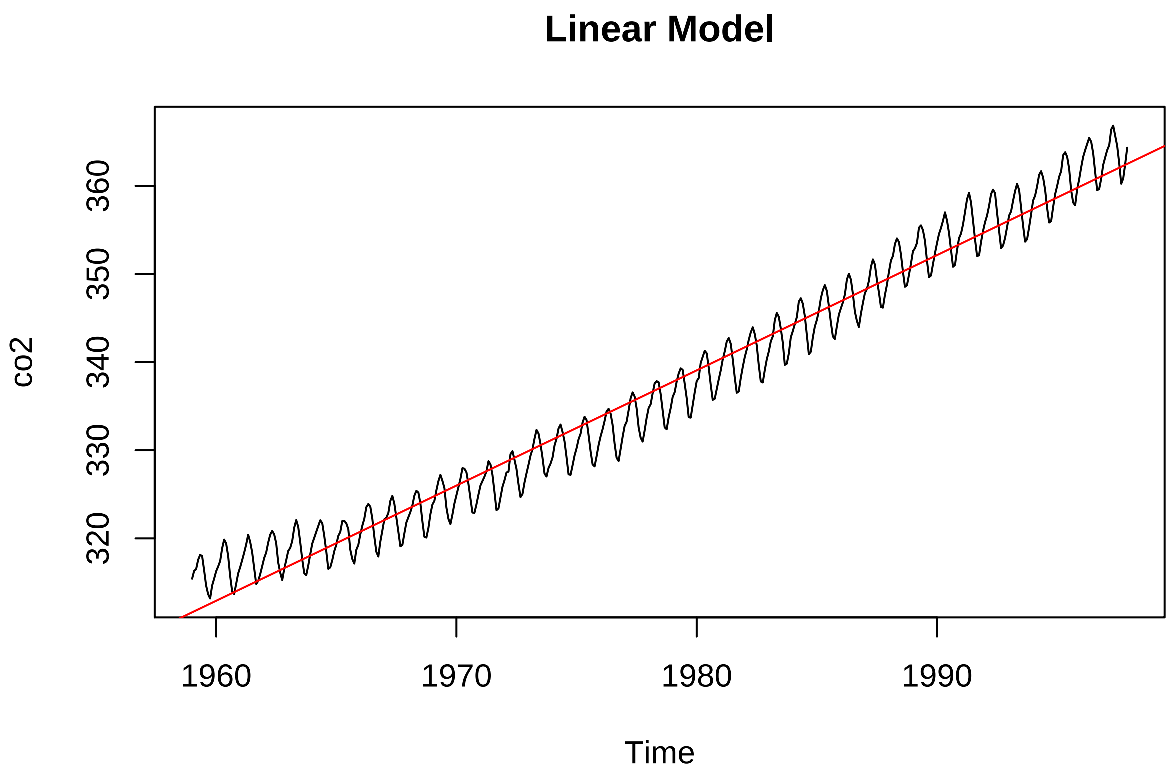

Residuals vs Fitted
Residuals
5
0
−5
17
5
214
310
320
330
340
350
360
Fitted values
lm(co2data ~ co2time)


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(co2data ~ co2time)
214
5
17

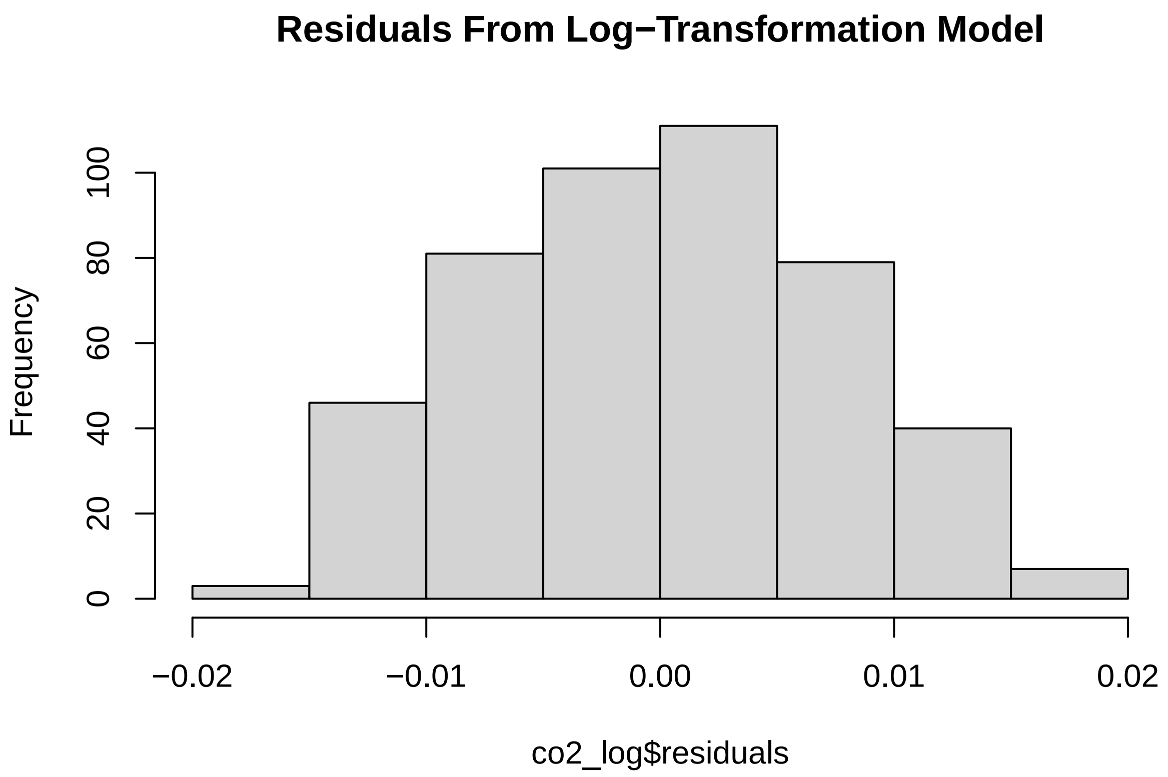

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
310
320
330
340
350
360
17
5
214
Fitted values
lm(co2data ~ co2time)



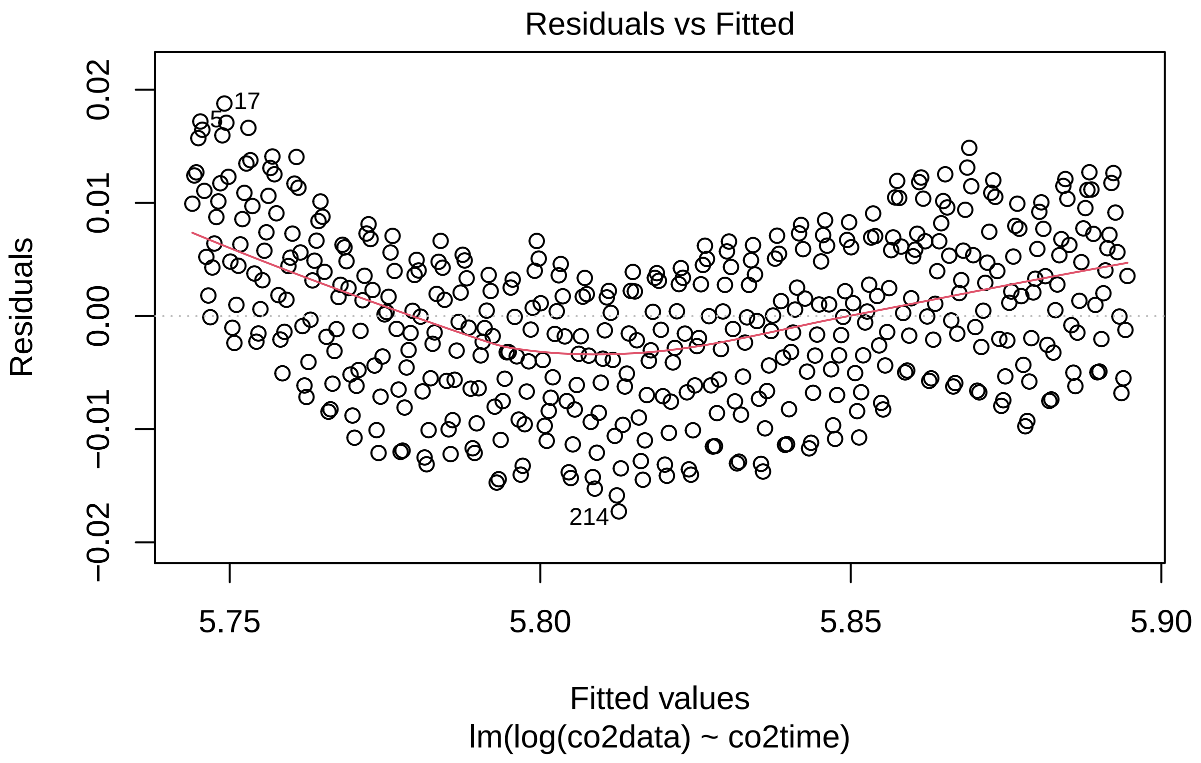
```
plot(co2,main="Linear Model")
abline(co2_linear,col="red")
```

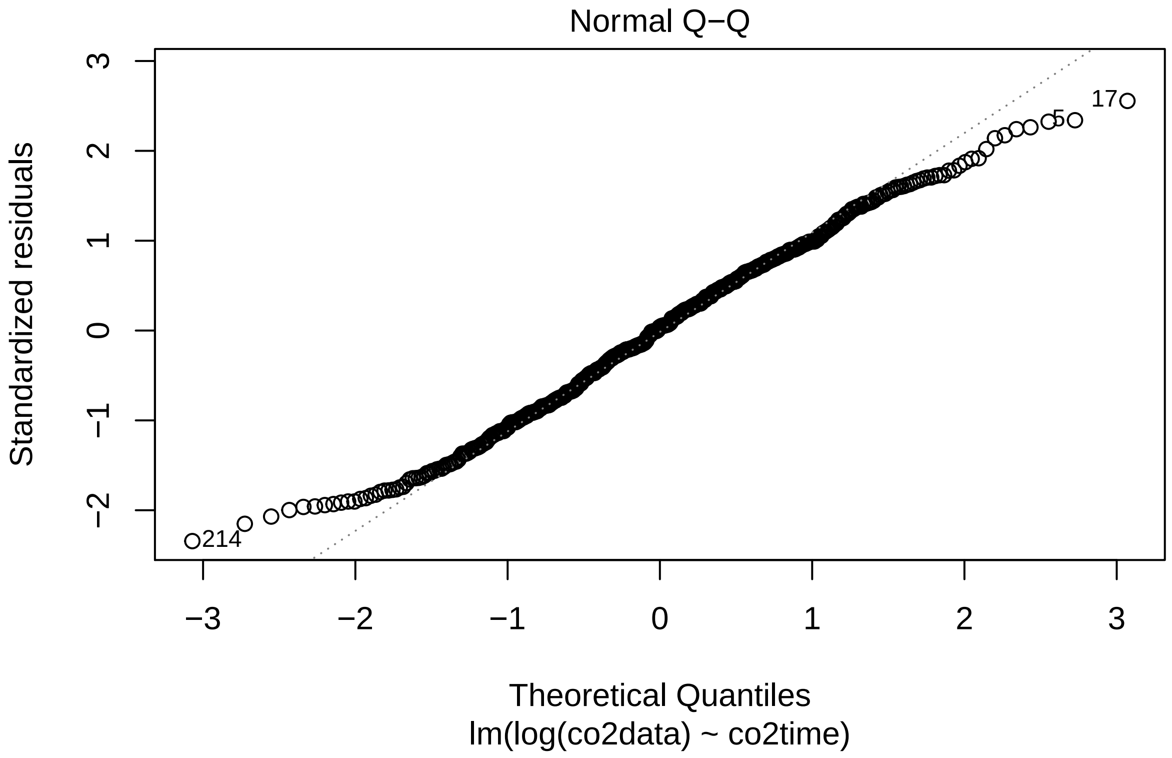


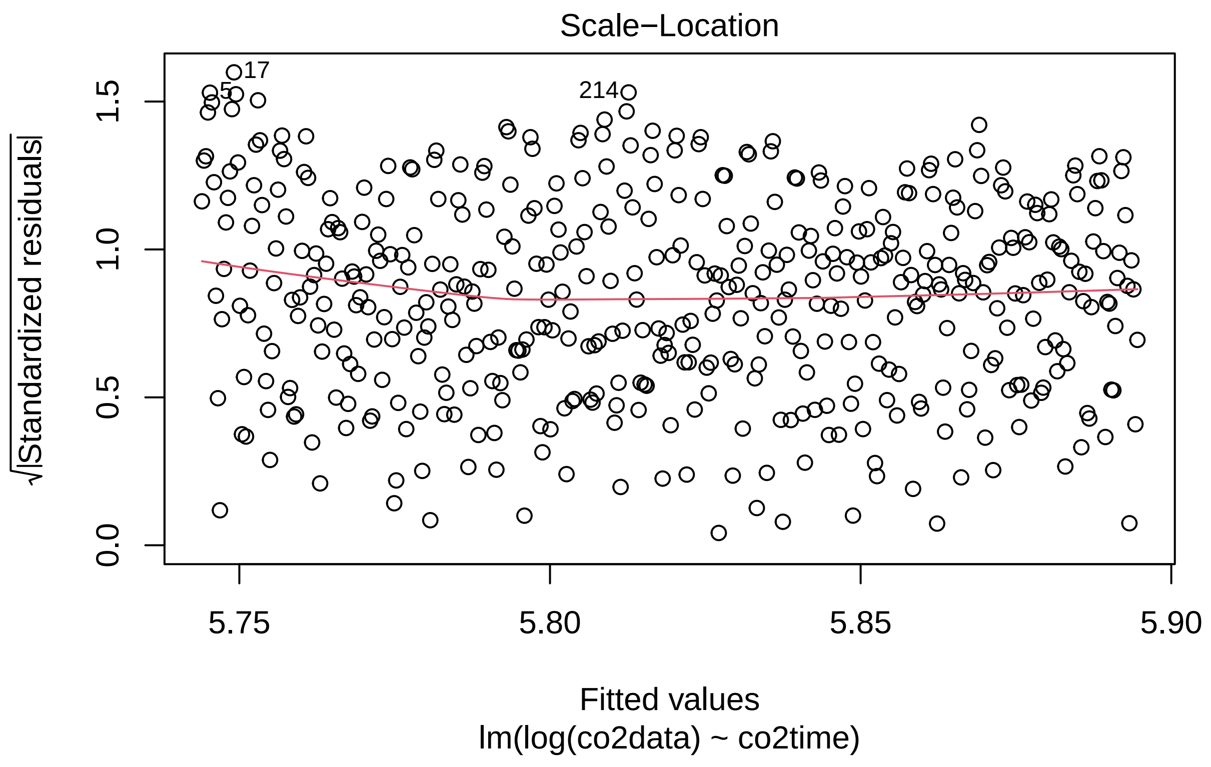
```r
###############################3
# explore about log transform
co2_log=lm(log(co2data)~co2time,data=co2df)
hist(co2_log$residuals,main="Residuals From Log-Transformation Model")
```

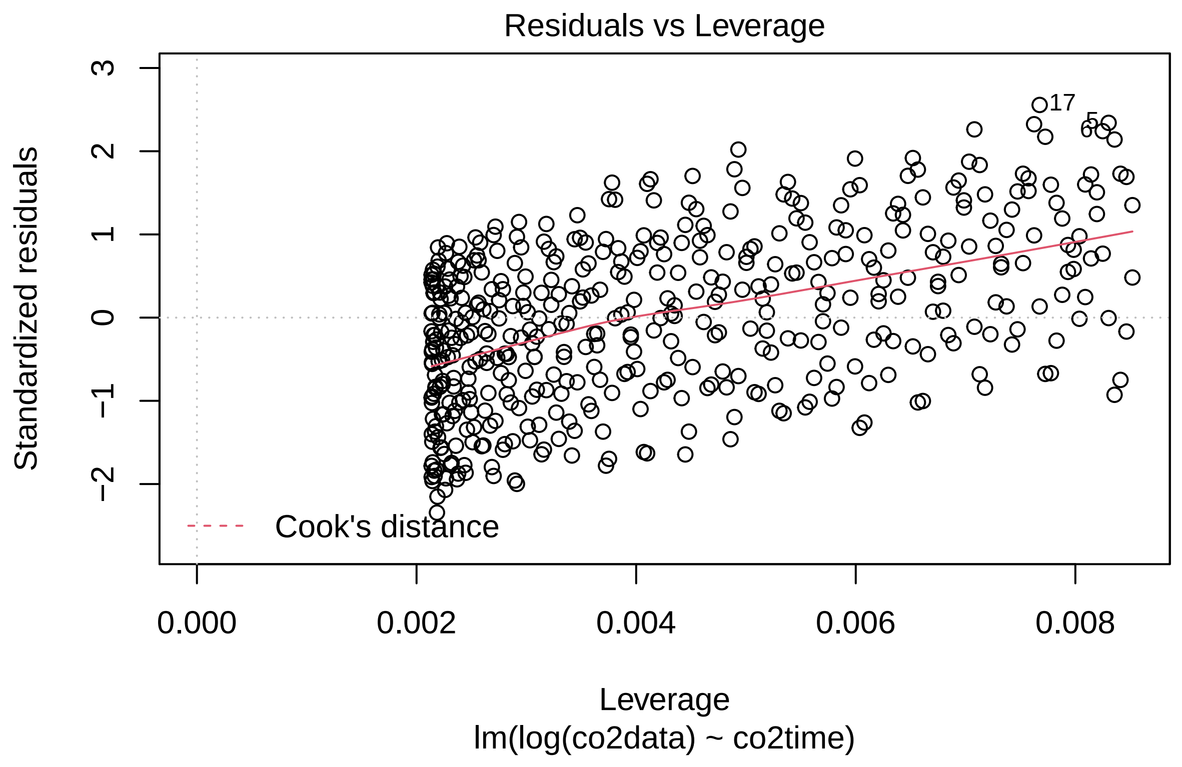

Residuals From Log−Transformation Model
Frequency
0
20
40
60
80
100
−0.02
−0.01
0.00
0.01
0.02
co2_log$residuals

```
plot(co2_log)
```

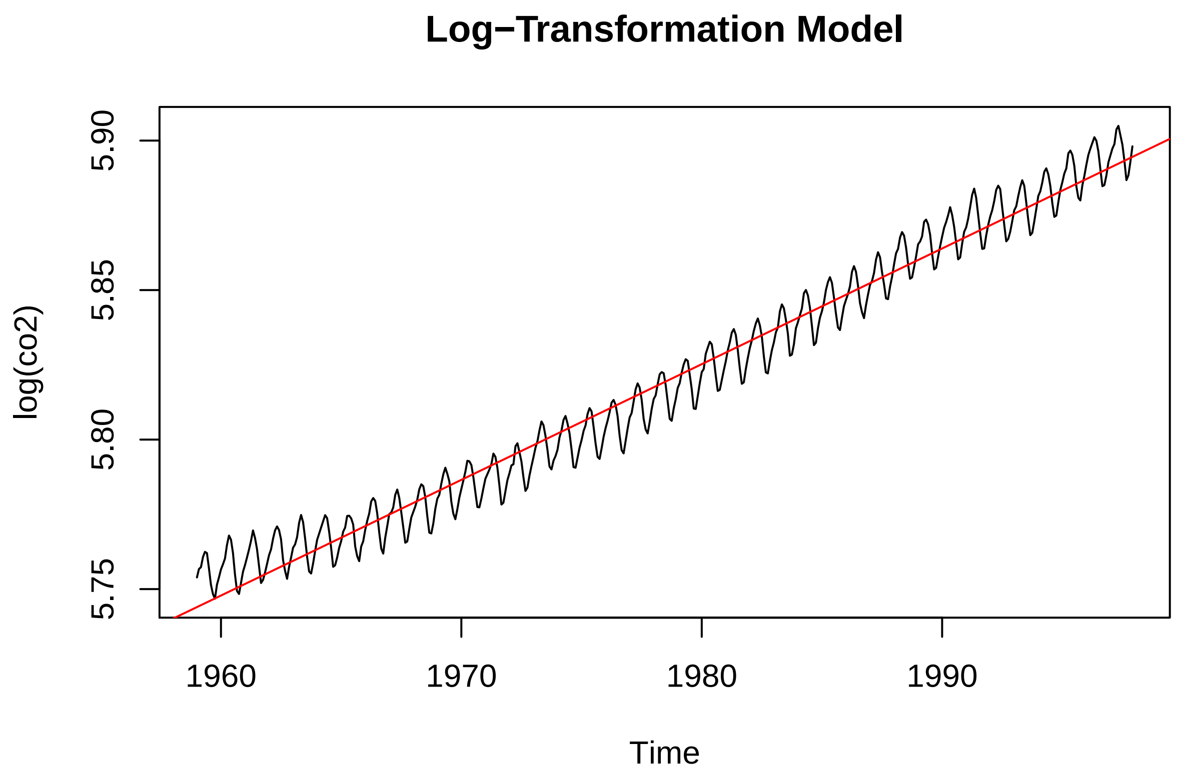

Residuals vs Fitted
Residuals
0.02
0.01
0.00
−0.01
−0.02
5.75
5.80
5.85
5.90
17
5
214
Fitted values
lm(log(co2data) ~ co2time)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log(co2data) ~ co2time)
214
5
17
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
5.75
5.80
5.85
5.90
Fitted values
lm(log(co2data) ~ co2time)
17
5
214



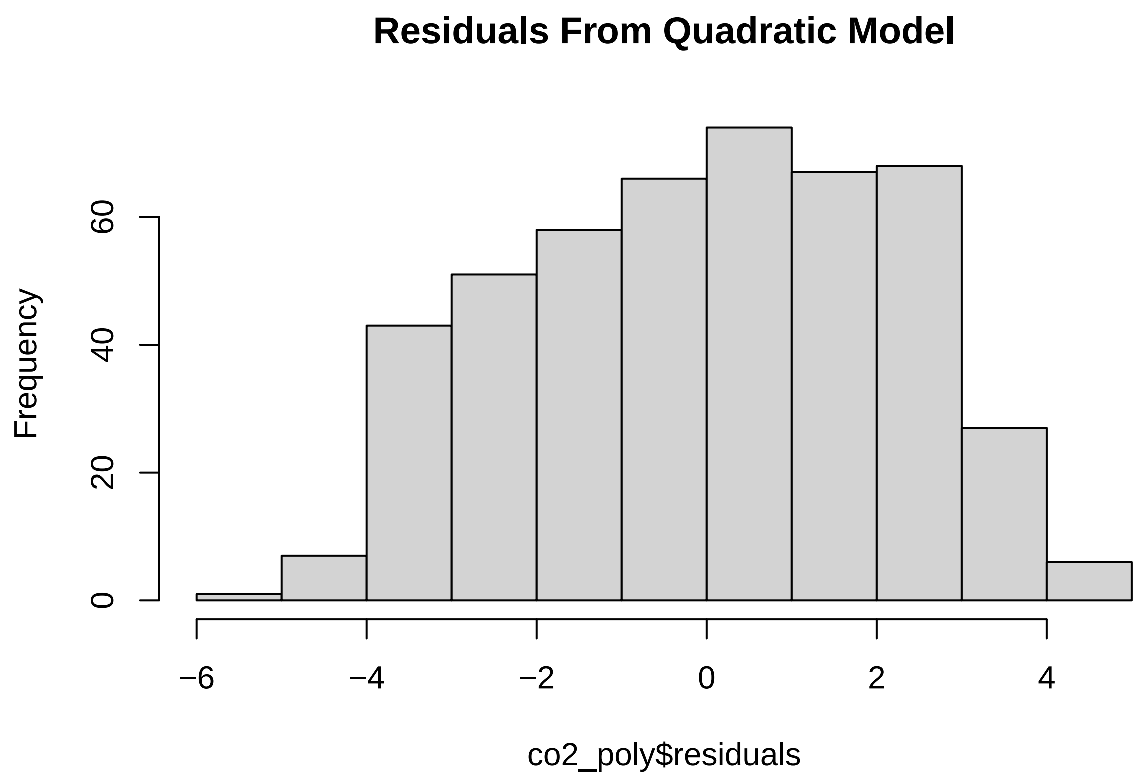
```
plot(log(co2),main="Log-Transformation Model")
abline(co2_log,col="red")
```



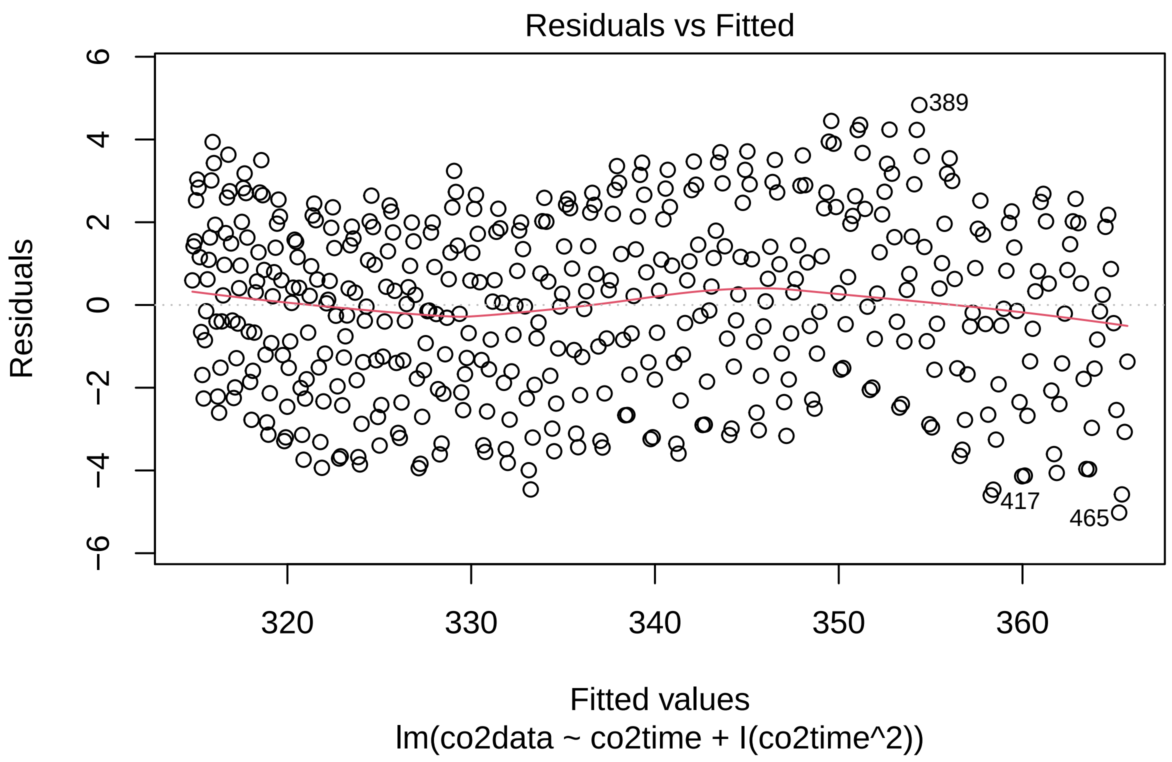
```
###############################################
#second polynomial
co2_poly=lm(co2data~co2time+I(co2time^2),data=co2df)
summary(co2_poly)
```

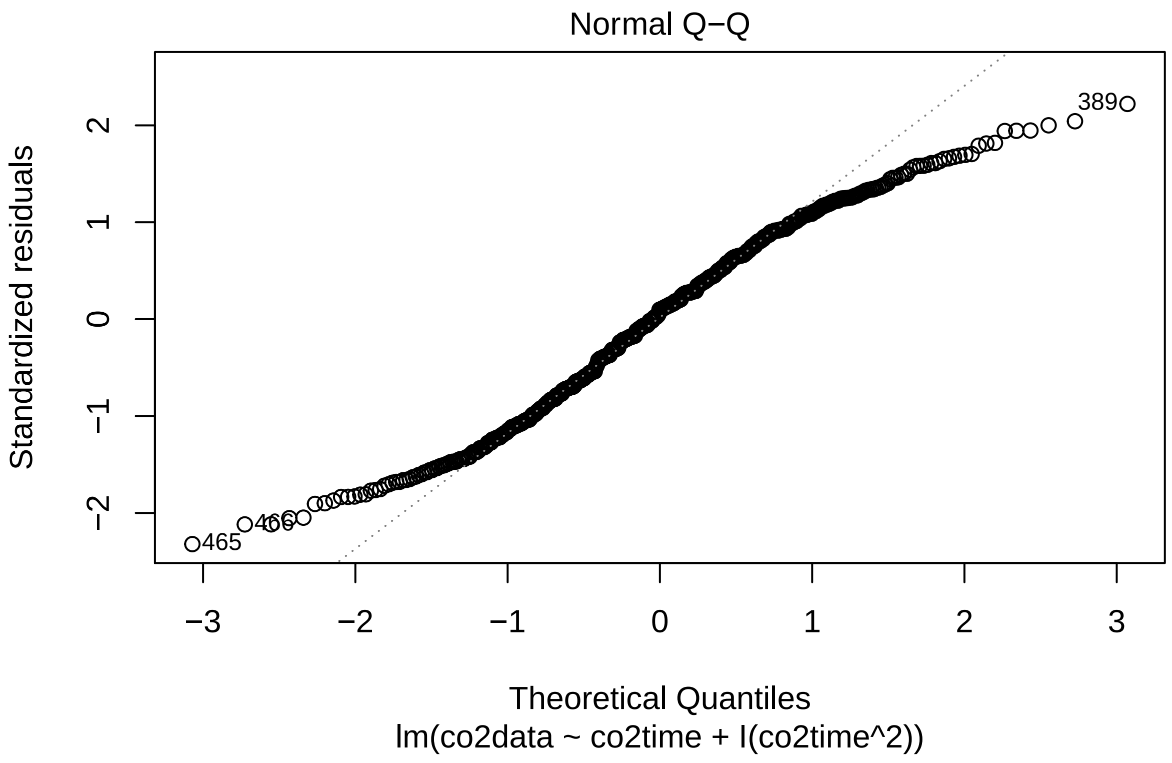
```
##
## Call:
## lm(formula = co2data ~ co2time + I(co2time^2), data = co2df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0195 -1.7120  0.2144  1.7957  4.8345
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.770e+04  3.483e+03   13.70   <2e-16 ***
## co2time      -4.919e+01  3.521e+00  -13.97   <2e-16 ***
## I(co2time^2)  1.276e-02  8.898e-04   14.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.182 on 465 degrees of freedom
## Multiple R-squared:  0.9788, Adjusted R-squared:  0.9787
## F-statistic: 1.075e+04 on 2 and 465 DF,  p-value: < 2.2e-16
```

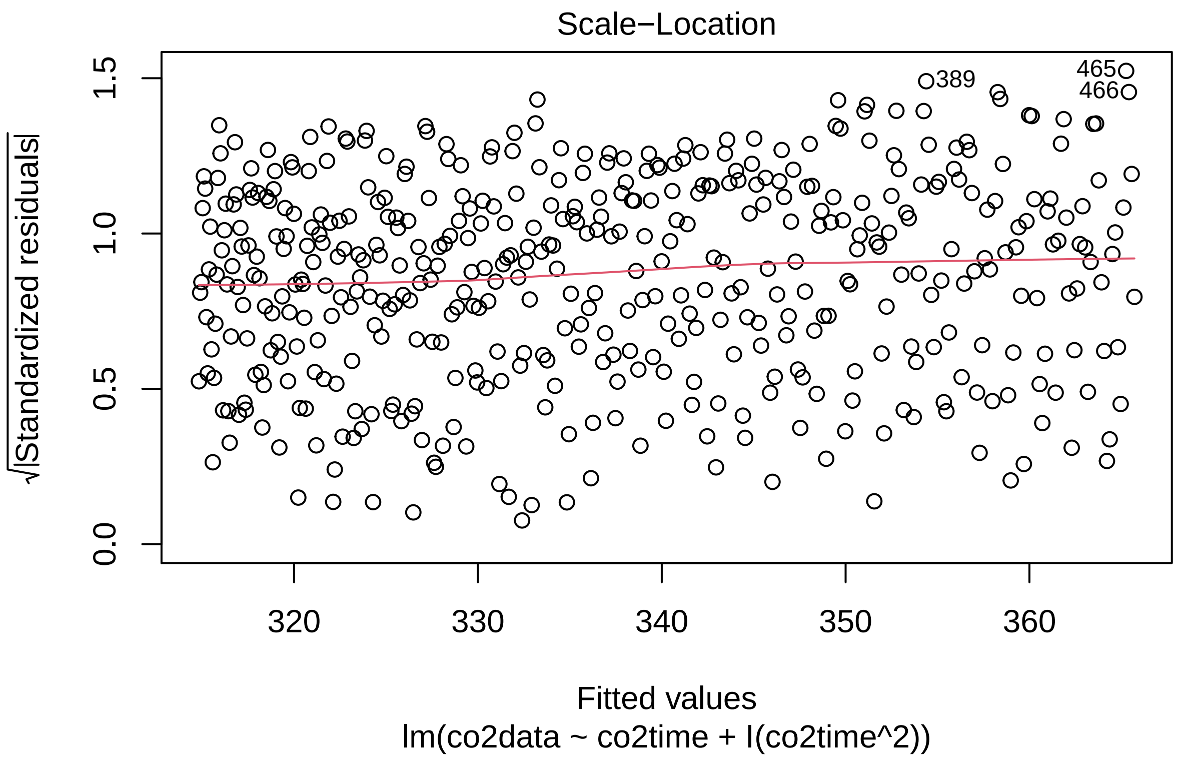
```
hist(co2_poly$residuals,main="Residuals From Quadratic Model")
```

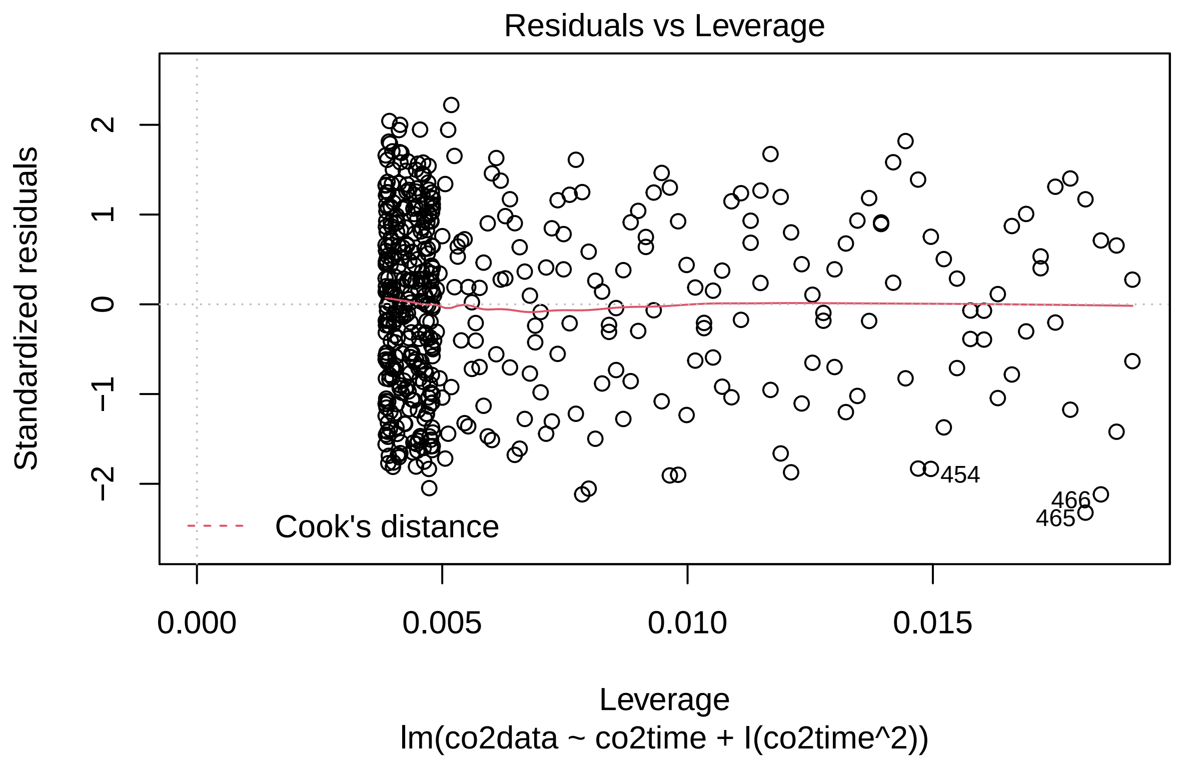


```
plot(co2_poly)
```

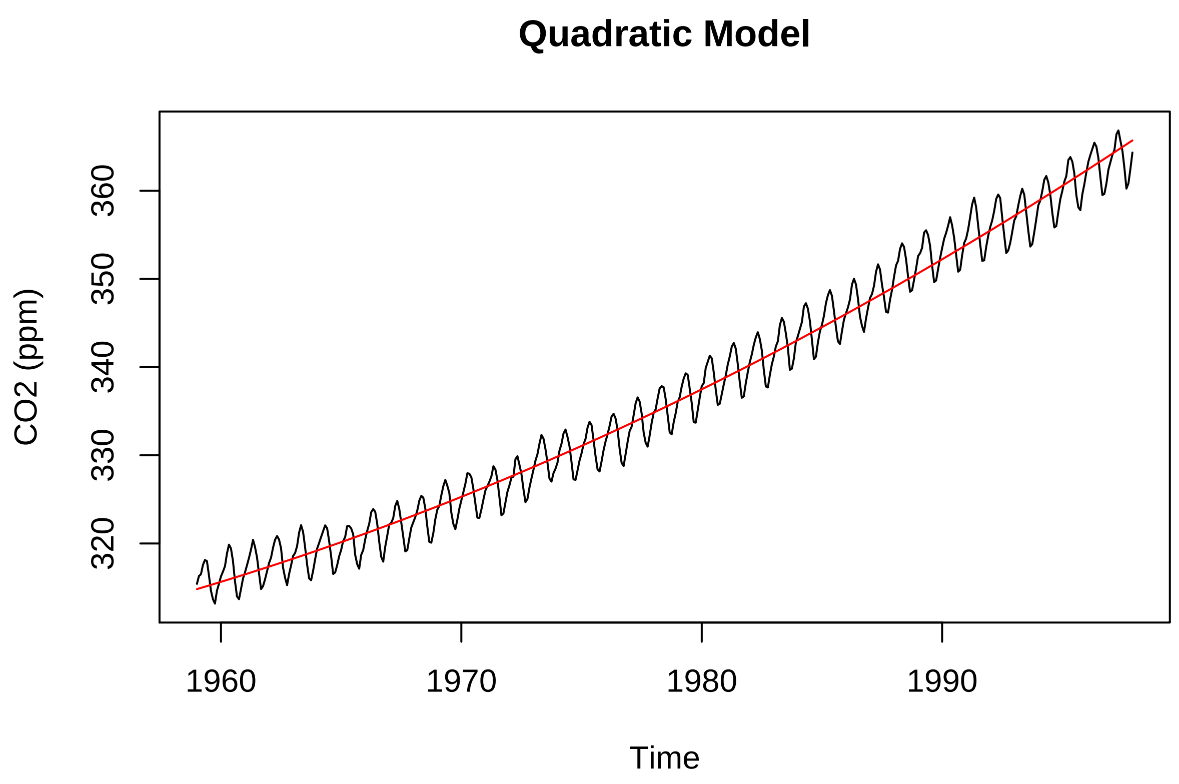

Residuals vs Fitted
Residuals
Fitted values
lm(co2data ~ co2time + I(co2time^2))
389
417
465
6
4
2
0
−2
−4
−6
320
330
340
350
360


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(co2data ~ co2time + I(co2time^2))
389
465
466
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
320
330
340
350
360
389
465
466
Fitted values
lm(co2data ~ co2time + I(co2time^2))



```r
plot(co2df$co2time,co2df$co2data,type = "l",
     main="Quadratic Model",xlab="Time",ylab="CO2 (ppm)")
predicted_co2_poly <- predict(co2_poly,list(co2time=co2df$co2time))
x_for_plot=as.vector(co2df$co2time)
y_for_plot=as.vector(predicted_co2_poly)
lines(x_for_plot,y_for_plot,col="red")
```



```
########################################
#try to understand seasons
seasonMaker=function(d) {
  int_floor=floor(d)
  int_ceil=ceiling(d)
  dpart=d-int_floor
  twelths=seq(from=0,to=1,by=1/12)
  abs_dists=abs(twelths-dpart)
  #print(abs_dists)
  idx_min=which.min(abs_dists)
  if(idx_min==length(abs_dists)) {
    return(1)
  }
  return(idx_min)
}
seasonMakerIndicator=function(idx,s) {
  if(s==idx) {
    return(1)
  } else {
    return(0)
  }
}
seasonData=sapply(co2df$co2time,seasonMaker)
```

```r
co2Sdf=data.frame(
  co2time=co2df$co2time,
  co2data=co2df$co2data
)
for(sname in seq(from=1,to=11)) {
  col_name=paste("season_",sname,sep="")
  season_indicators=c()
  for(t_idx in 1:length(seasonData)) {
    season_idx=seasonData[t_idx]
    season_indicator=seasonMakerIndicator(season_idx,sname)
    season_indicators=c(season_indicators,season_indicator)
  }
  co2Sdf[,col_name]=season_indicators
}
head(co2Sdf)
```

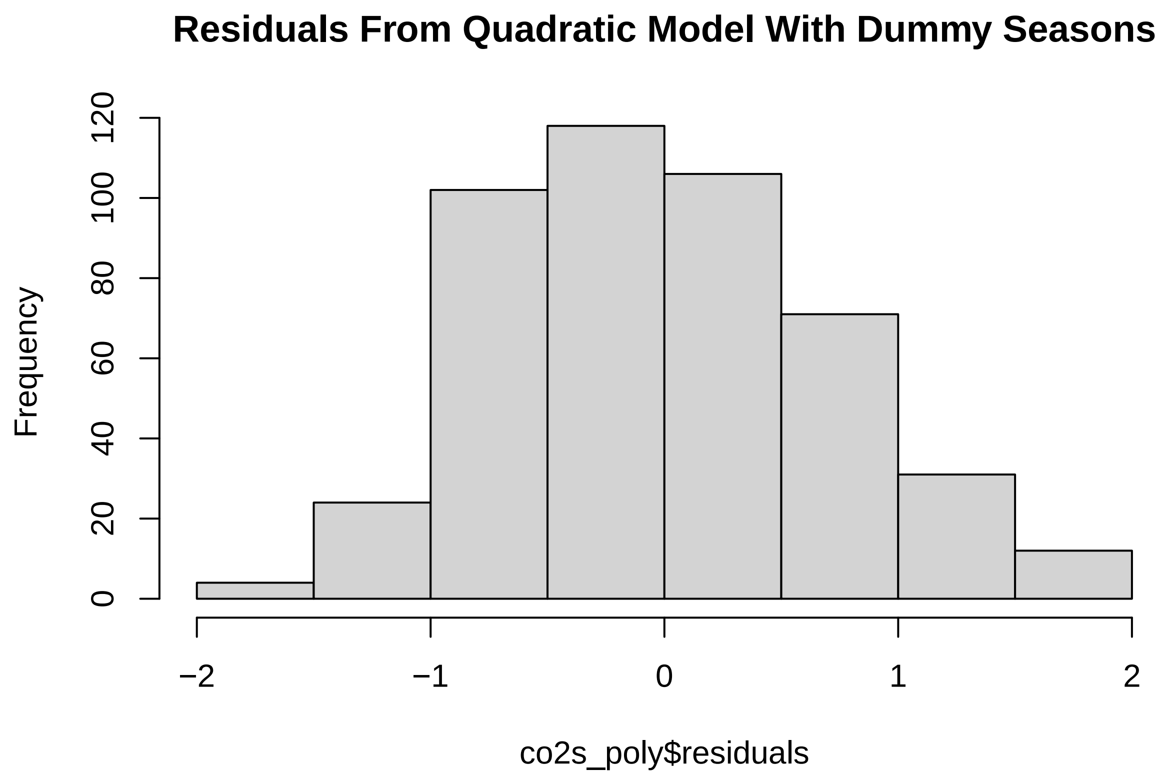
```
##    co2time co2data season_1 season_2 season_3 season_4 season_5 season_6
## 1 1959.000  315.42        1        0        0        0        0        0
## 2 1959.083  316.31        0        1        0        0        0        0
## 3 1959.167  316.50        0        0        1        0        0        0
## 4 1959.250  317.56        0        0        0        1        0        0
## 5 1959.333  318.13        0        0        0        0        1        0
## 6 1959.417  318.00        0        0        0        0        0        1
##   season_7 season_8 season_9 season_10 season_11
## 1        0        0        0         0         0
## 2        0        0        0         0         0
## 3        0        0        0         0         0
## 4        0        0        0         0         0
## 5        0        0        0         0         0
## 6        0        0        0         0         0
```

```r
########################################
# polynomial model using dummy seasons
co2s_poly=lm(co2data~co2time+I(co2time^2)+season_1+season_2+
               season_3+season_4+season_5+season_6+season_7+
               season_8+season_9+season_10+season_11,data=co2Sdf)
summary(co2s_poly)
```

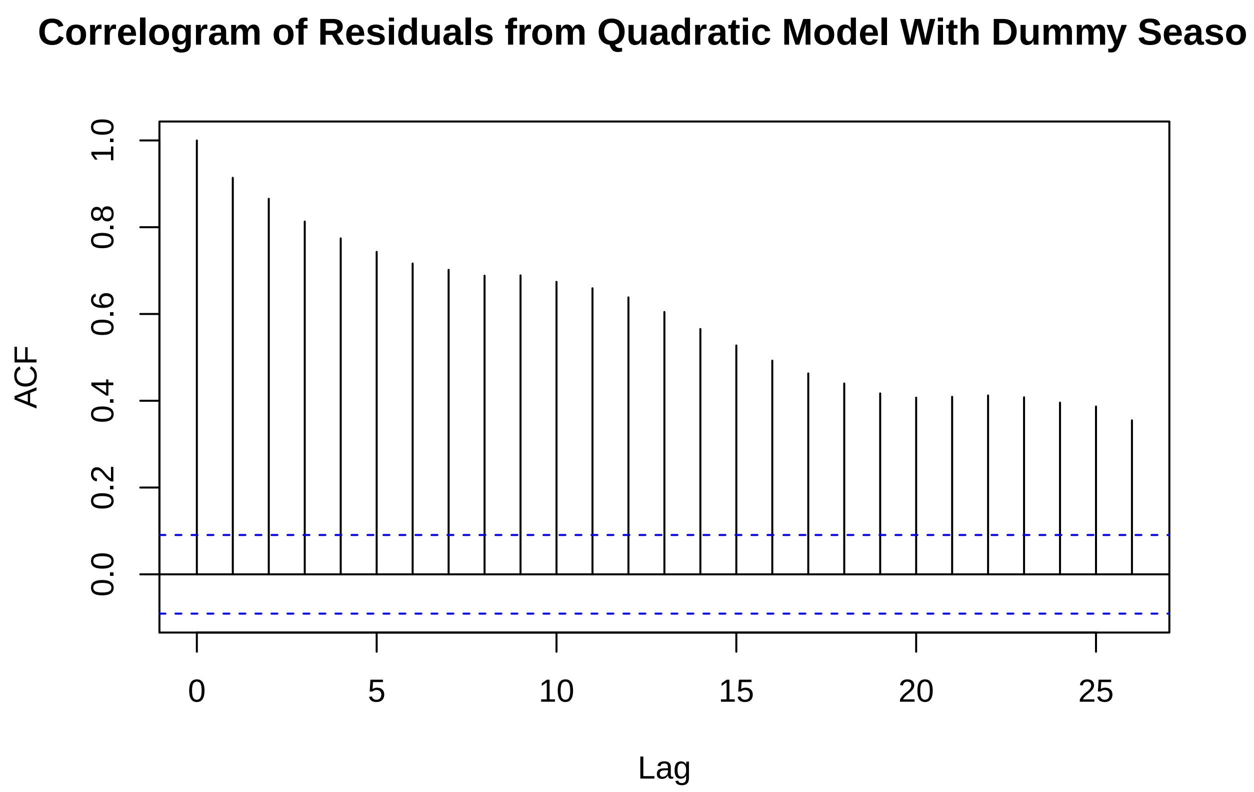
```
##
## Call:
## lm(formula = co2data ~ co2time + I(co2time^2) + season_1 + season_2 +
##     season_3 + season_4 + season_5 + season_6 + season_7 + season_8 +
##     season_9 + season_10 + season_11, data = co2Sdf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.99478 -0.54468 -0.06017  0.47265  1.95480
##
```

```
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    4.771e+04  1.156e+03  41.288  < 2e-16 ***
## co2time       -4.920e+01  1.168e+00 -42.120  < 2e-16 ***
## I(co2time^2)   1.277e-02  2.952e-04  43.242  < 2e-16 ***
## season_1       9.374e-01  1.640e-01   5.717 1.97e-08 ***
## season_2       1.602e+00  1.640e-01   9.768  < 2e-16 ***
## season_3       2.344e+00  1.640e-01  14.298  < 2e-16 ***
## season_4       3.476e+00  1.640e-01  21.196  < 2e-16 ***
## season_5       3.954e+00  1.640e-01  24.117  < 2e-16 ***
## season_6       3.291e+00  1.640e-01  20.074  < 2e-16 ***
## season_7       1.770e+00  1.640e-01  10.798  < 2e-16 ***
## season_8      -2.974e-01  1.640e-01  -1.814   0.0704 .
## season_9      -2.122e+00  1.640e-01 -12.942  < 2e-16 ***
## season_10     -2.305e+00  1.640e-01 -14.061  < 2e-16 ***
## season_11     -1.117e+00  1.640e-01  -6.810 3.10e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.724 on 454 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9977
## F-statistic: 1.531e+04 on 13 and 454 DF,  p-value: < 2.2e-16
```

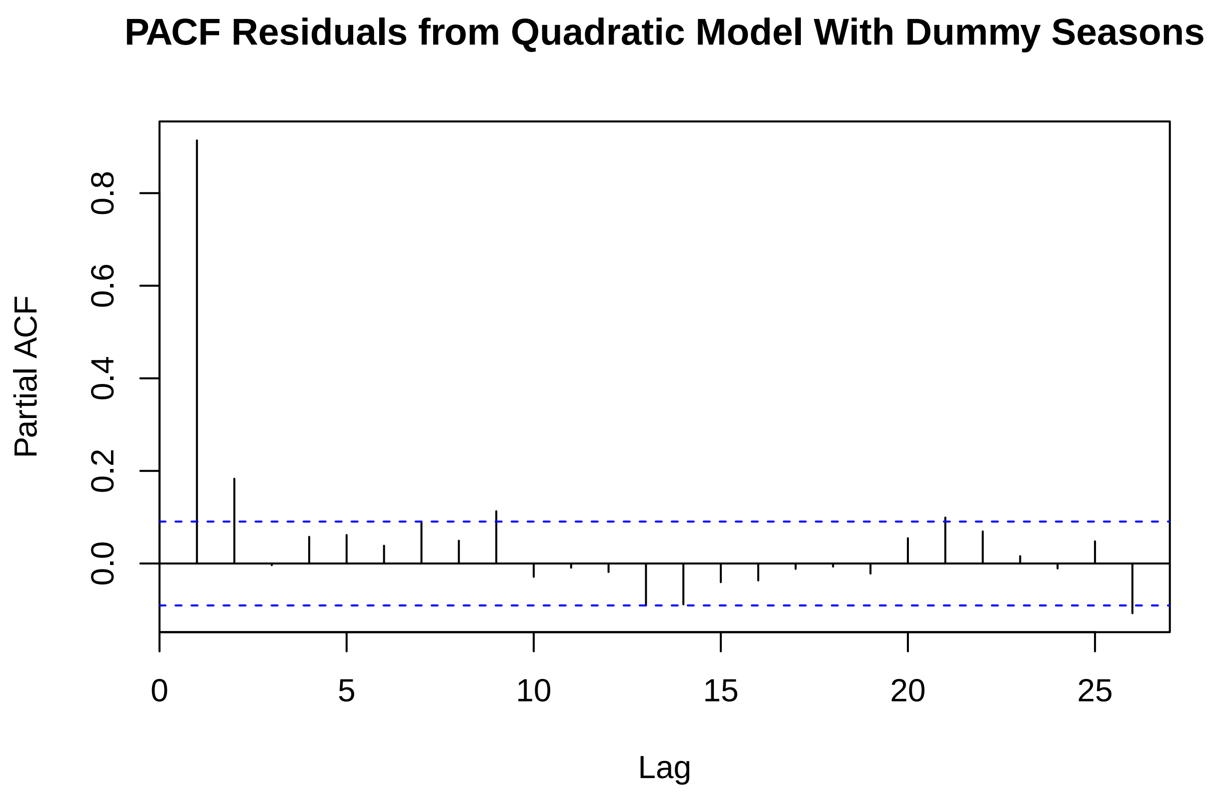
```
hist(co2s_poly$residuals,
     main="Residuals From Quadratic Model With Dummy Seasons")
```



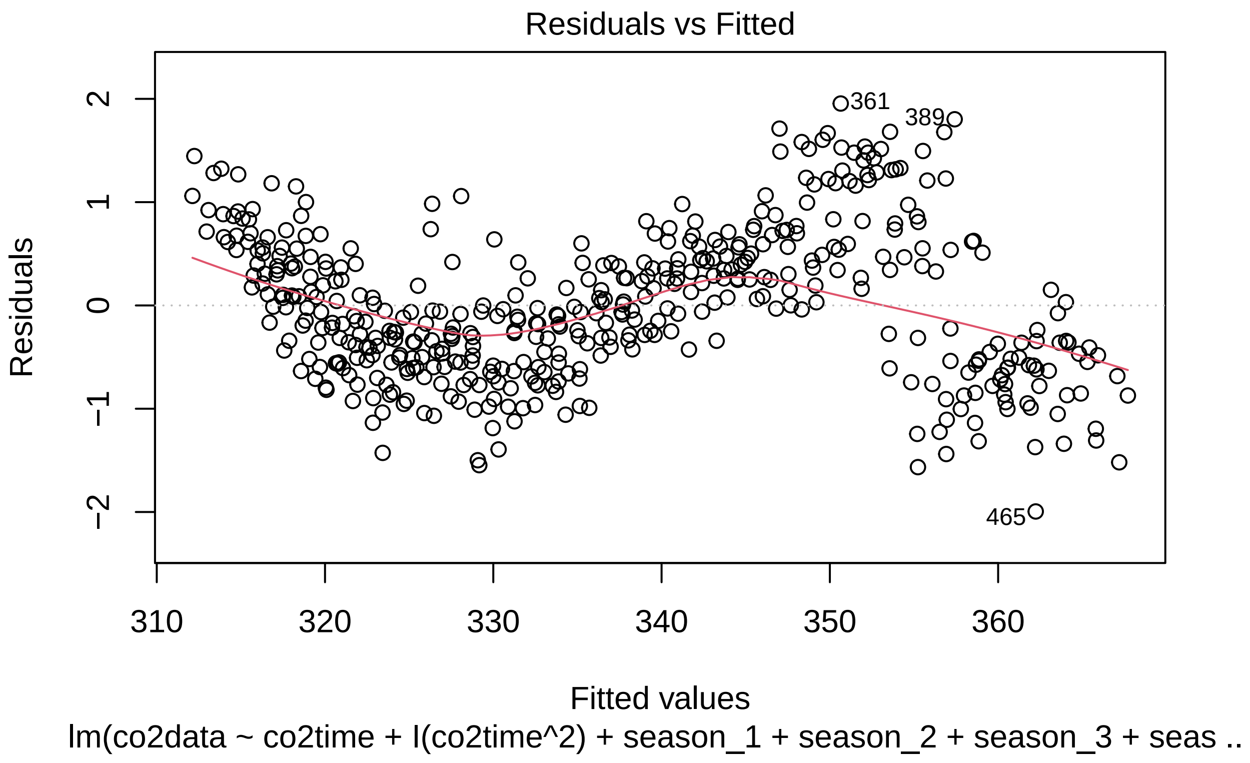
```
acf(co2s_poly$residuals,
    main="Correlogram of Residuals from Quadratic Model With Dummy Seasons")
```

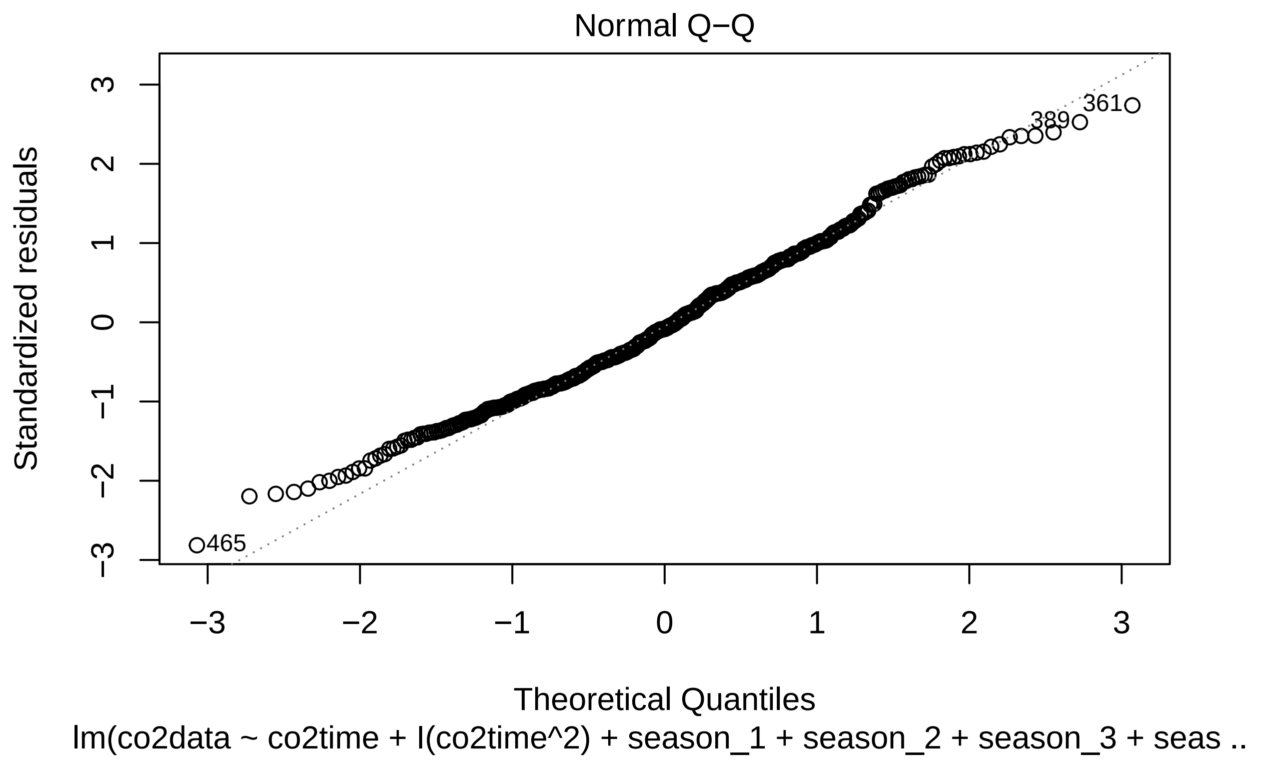


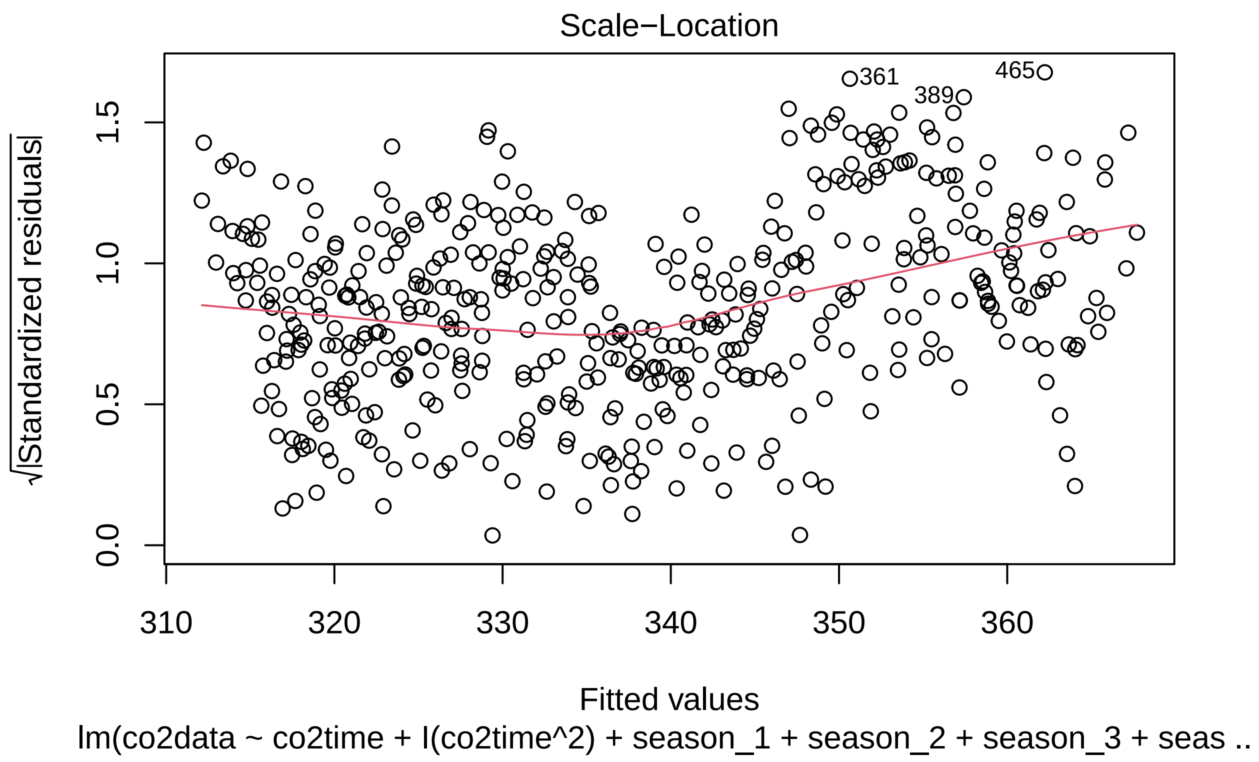
```
pacf(co2s_poly$residuals,
     main="PACF Residuals from Quadratic Model With Dummy Seasons")
```



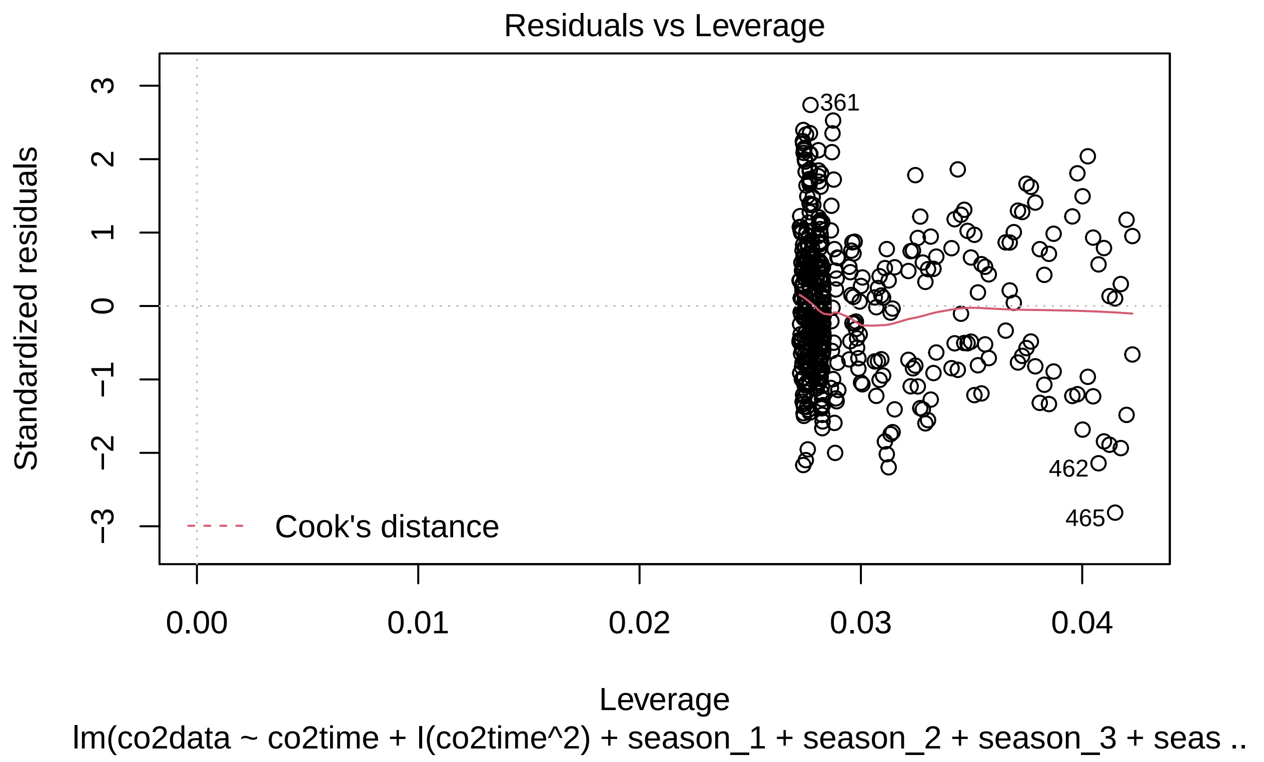
```
plot(co2s_poly)
```


Residuals vs Fitted
Residuals
Fitted values
lm(co2data ~ co2time + I(co2time^2) + season_1 + season_2 + season_3 + seas ..
361
389
465
2
1
0
−1
−2
310
320
330
340
350
360


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(co2data ~ co2time + I(co2time^2) + season_1 + season_2 + season_3 + seas ..
465
389
361
3
2
1
0
−1
−2
−3

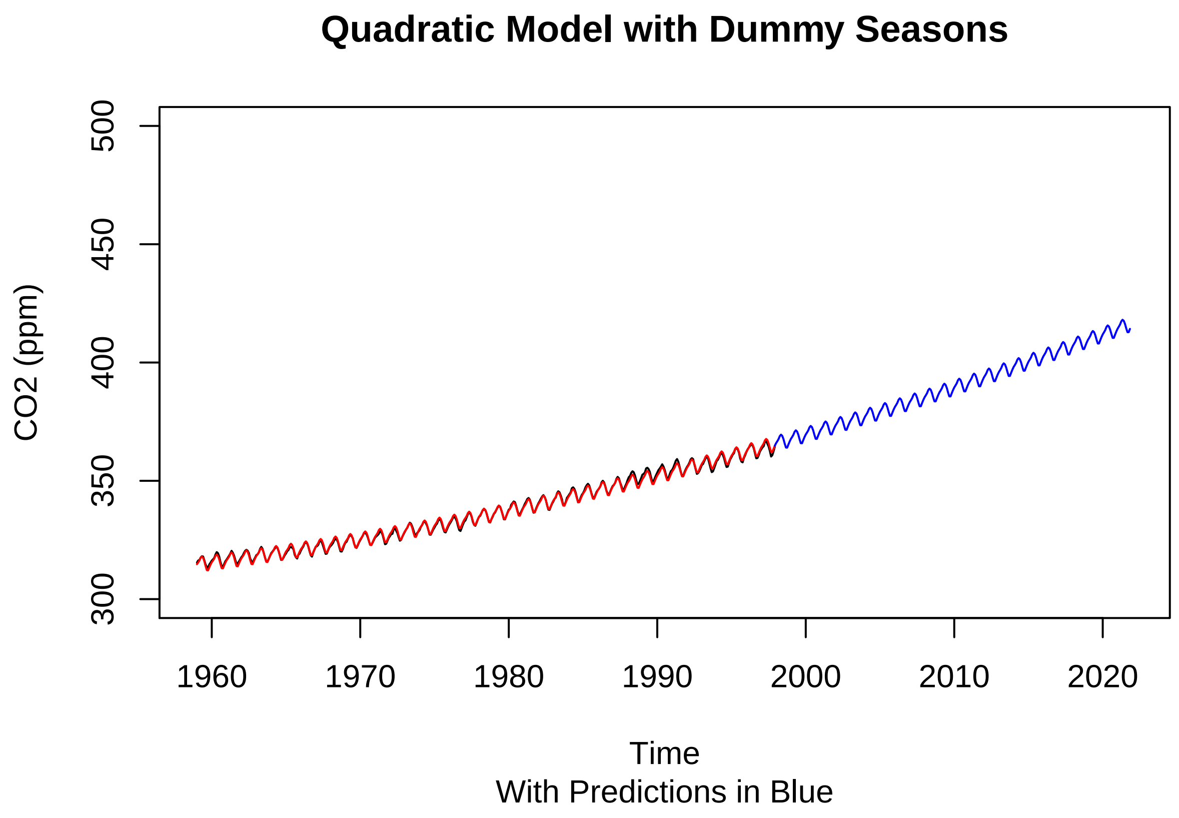


Scale−Location
361
389
465
√|Standardized residuals|
0.0
0.5
1.0
1.5
310
320
330
340
350
360
Fitted values
lm(co2data ~ co2time + I(co2time^2) + season_1 + season_2 + season_3 + seas ..



```
plot(co2Sdf$co2time,co2Sdf$co2data,type = "l",
     main="Quadratic Model with Dummy Seasons",sub="With Predictions in Blue",xlim=c(1959,2022)
predicted_co2s_poly <- predict(co2s_poly,list(co2time=co2Sdf$co2time,
                                              season_1=co2Sdf$season_1,
                                              season_2=co2Sdf$season_2,
                                              season_3=co2Sdf$season_3,
                                              season_4=co2Sdf$season_4,
                                              season_5=co2Sdf$season_5,
                                              season_6=co2Sdf$season_6,
                                              season_7=co2Sdf$season_7,
                                              season_8=co2Sdf$season_8,
                                              season_9=co2Sdf$season_9,
                                              season_10=co2Sdf$season_10,
                                              season_11=co2Sdf$season_11))
x_for_plot=as.vector(co2Sdf$co2time)
y_for_plot=as.vector(predicted_co2s_poly)
lines(x_for_plot,y_for_plot,col="red")

###########################
# now generate time data from 1998 to present so that the predictor
# can generate predictions for it
from_1998_to_present=seq(from=max(co2Sdf$co2time),to=2021+10.5/12,by=1/12)
from_1998_to_present_season_idx=sapply(from_1998_to_present,seasonMaker)
```

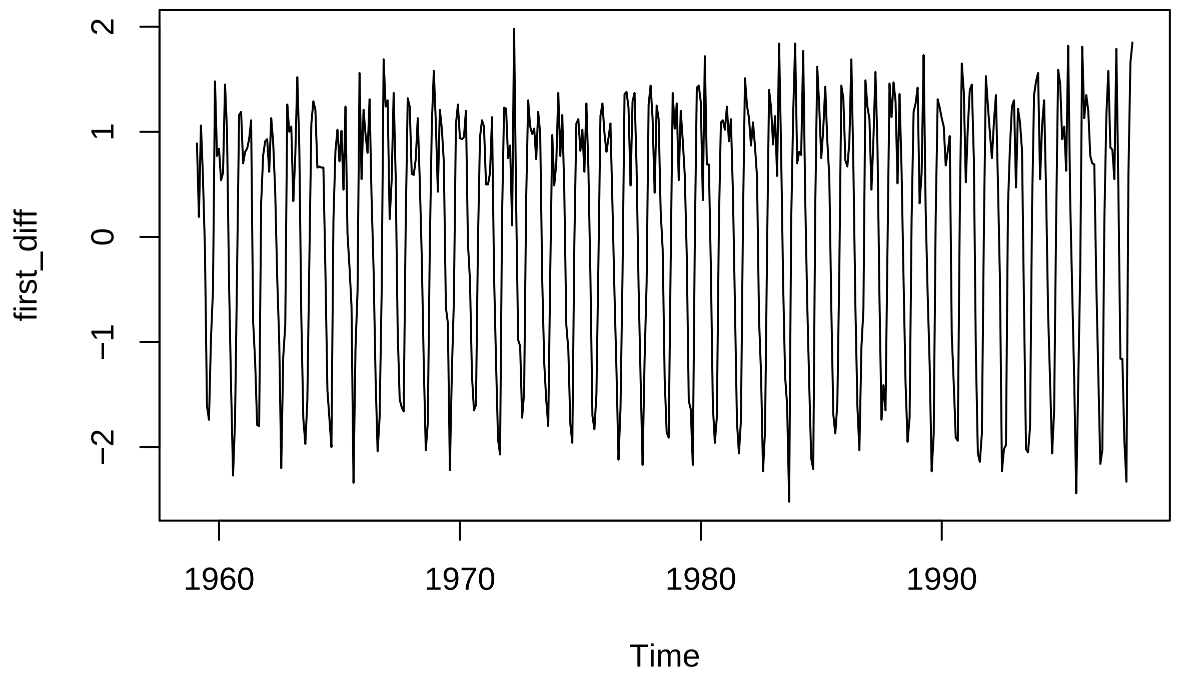
```r
then_to_now_df=data.frame(
  co2time=from_1998_to_present
)
for(sname in seq(from=1,to=11)) {
  col_name=paste("season_",sname,sep="")
  season_indicators=c()
  for(t_idx in 1:length(from_1998_to_present_season_idx)) {
    season_idx=from_1998_to_present_season_idx[t_idx]
    season_indicator=seasonMakerIndicator(season_idx,sname)
    season_indicators=c(season_indicators,season_indicator)
  }
  then_to_now_df[,col_name]=season_indicators
}
predicted_FUTURE_co2s_poly <- predict(co2s_poly,
                                      list(co2time=then_to_now_df$co2time,
                                           season_1=then_to_now_df$season_1,
                                           season_2=then_to_now_df$season_2,
                                           season_3=then_to_now_df$season_3,
                                           season_4=then_to_now_df$season_4,
                                           season_5=then_to_now_df$season_5,
                                           season_6=then_to_now_df$season_6,
                                           season_7=then_to_now_df$season_7,
                                           season_8=then_to_now_df$season_8,
                                           season_9=then_to_now_df$season_9,
                                           season_10=then_to_now_df$season_10,
                                         season_11=then_to_now_df$season_11))
x_for_plot=as.vector(then_to_now_df$co2time)
y_for_plot=as.vector(predicted_FUTURE_co2s_poly)
lines(x_for_plot,y_for_plot,col="blue")
```



Part 3 (4 points)

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

```
first_diff=diff(co2)
plot(first_diff)
```



```
diffed_co2=co2-first_diff
head(first_diff)
```

```
## [1]  0.89  0.19  1.06  0.57 -0.13 -1.61
```

```
arima_matrix=matrix(c(0,0,0,0),nrow=1,ncol=4)
for(p in 1:3) {
  for(i in 1:3) {
    for(q in 1:3) {
      print(paste(p,i,q))
      my_arima=arima(co2,order=c(p,i,q),optim.control = list(maxit=500))
      arima_matrix=rbind(arima_matrix,c(p,i,q,my_arima$aic))
    }
  }
}
```

```
## [1] "1 1 1"
## [1] "1 1 2"
## [1] "1 1 3"
## [1] "1 2 1"
## [1] "1 2 2"
## [1] "1 2 3"
## [1] "1 3 1"
## [1] "1 3 2"
## [1] "1 3 3"
## [1] "2 1 1"
## [1] "2 1 2"
## [1] "2 1 3"
## [1] "2 2 1"
## [1] "2 2 2"

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## [1] "2 2 3"
## [1] "2 3 1"
## [1] "2 3 2"
## [1] "2 3 3"
## [1] "3 1 1"
## [1] "3 1 2"
## [1] "3 1 3"
## [1] "3 2 1"
## [1] "3 2 2"
## [1] "3 2 3"
## [1] "3 3 1"
## [1] "3 3 2"
## [1] "3 3 3"
```

```
arima_df=data.frame(
  p=arima_matrix[,1],
  i=arima_matrix[,2],
  q=arima_matrix[,3],
  a=arima_matrix[,4]
)
arima_df=arima_df[2:dim(arima_df)[1],]
rownames(arima_df) <- 1:nrow(arima_df)
head(arima_df)
```

```
##   p i q        a
## 1 1 1 1 1115.097
## 2 1 1 2 1077.678
## 3 1 1 3 1052.075
## 4 1 2 1 1213.404
## 5 1 2 2 1206.798
## 6 1 2 3 1081.226
```

```
best_arima_idx=which.min(arima_df$a)
print(best_arima_idx)
```

```
## [1] 23
```

```
print(arima_df[best_arima_idx,])
```

```
##    p i q        a
## 23 3 2 2 879.7772
```

```
best_arima=arima(co2,c(arima_df[best_arima_idx,1],arima_df[best_arima_idx,2],arima_df[best_arin
print(best_arima)
```

```
##
## Call:
## arima(x = co2, order = c(arima_df[best_arima_idx, 1], arima_df[best_arima_idx,
##     2], arima_df[best_arima_idx, 3]), optim.control = list(maxit = 500))
##
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2
##       1.5311  -0.8160  -0.0258  -1.9498  0.9515
## s.e.  0.0478   0.0777   0.0478   0.0126  0.0126
##
## sigma^2 estimated as 0.3688:  log likelihood = -433.89,  aic = 879.78
```

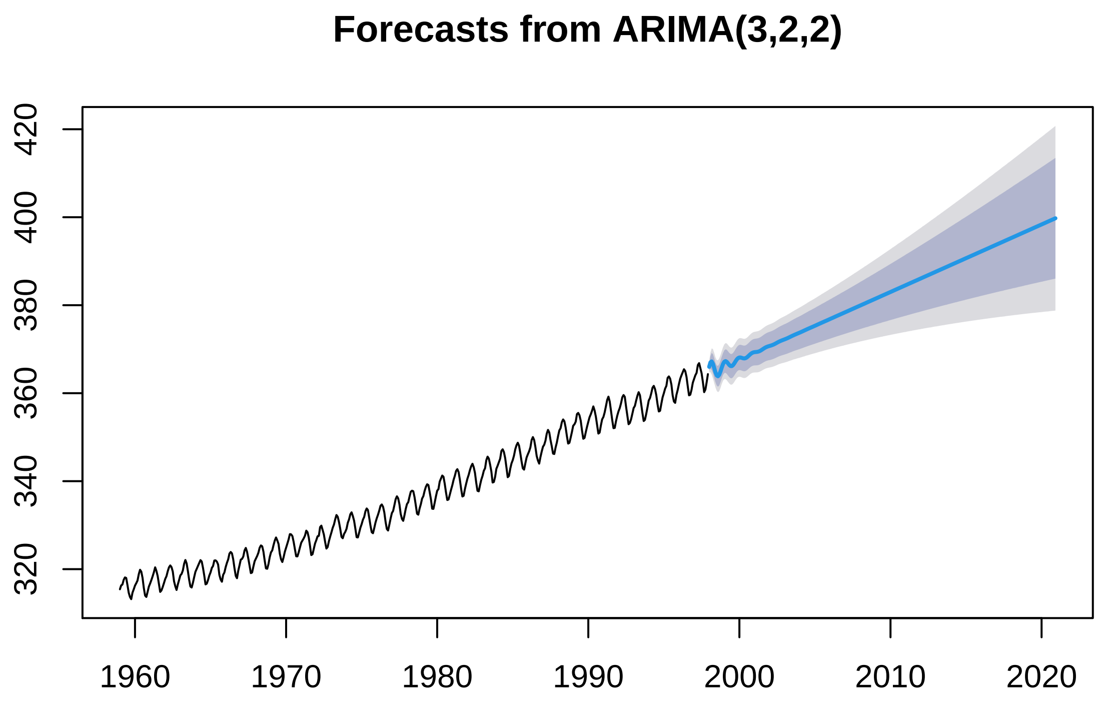
```
#install.packages("forecast")
library("forecast")
```

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

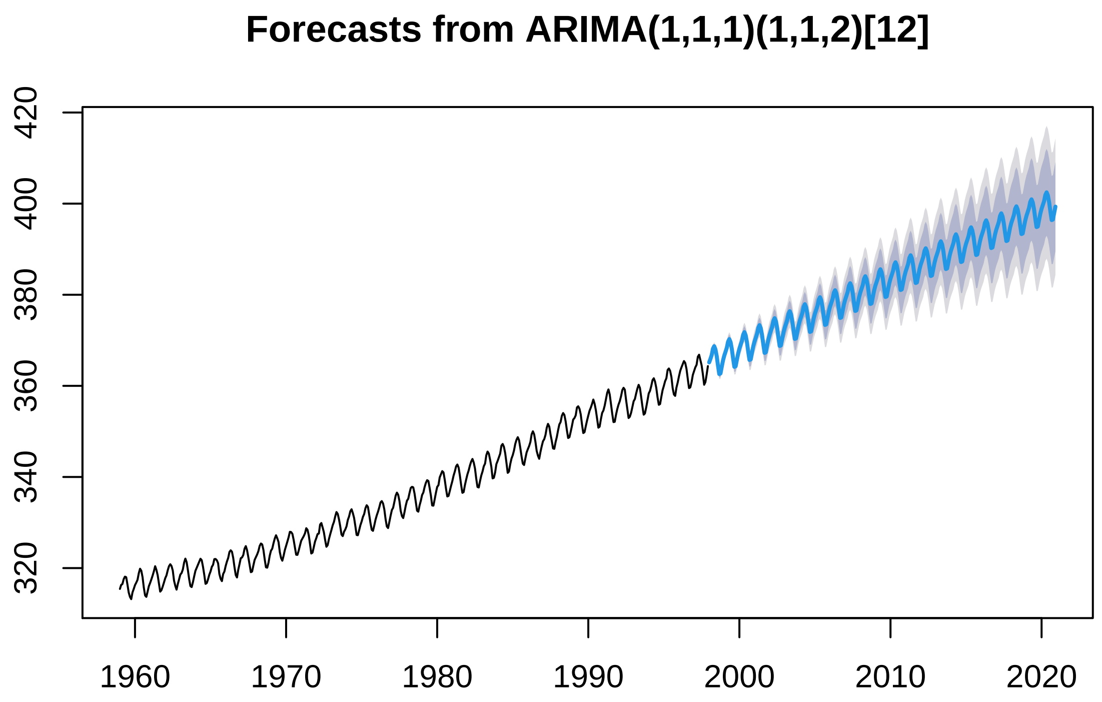
```
print(auto.arima(co2))
```

```
## Series: co2
## ARIMA(1,1,1)(1,1,2)[12]
##
## Coefficients:
##          ar1      ma1     sar1     sma1     sma2
##       0.2569  -0.5847  -0.5489  -0.2620  -0.5123
## s.e.  0.1406   0.1204   0.5881   0.5703   0.4820
##
## sigma^2 estimated as 0.08576:  log likelihood=-84.39
## AIC=180.78   AICc=180.97   BIC=205.5
```

```
plot(forecast(best_arima,(2021-1998)*12))
```



```
plot(forecast(auto.arima(co2),(2021-1998)*12))
```



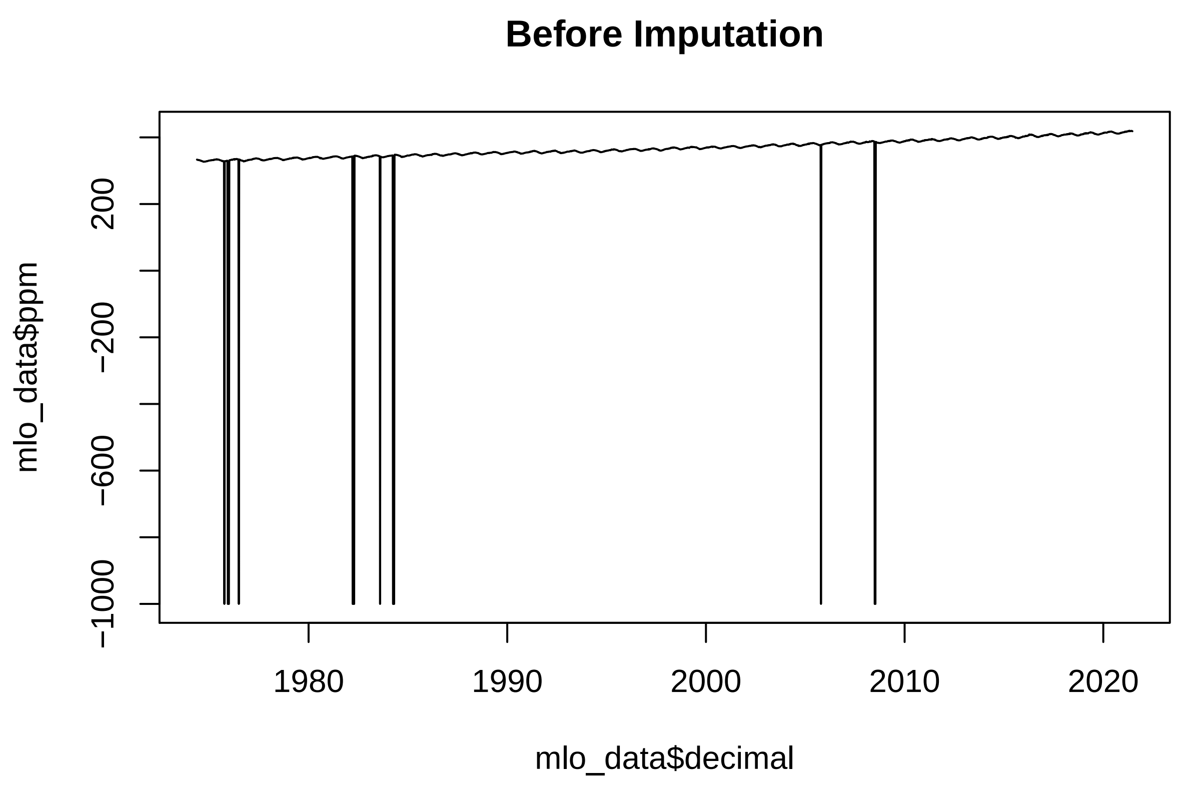
Part 4 (5 points)

The file `co2_weekly_mlo.txt` contains weekly observations of atmospheric carbon dioxide concentrations measured at the Mauna Loa Observatory from 1974 to 2020, published by the National Oceanic and Atmospheric Administration (NOAA). Convert these data into a suitable time series object, conduct a thorough EDA on the data, addressing the problem of missing observations and comparing the Keeling Curve's development to your predictions from Parts 2 and 3. Use the weekly data to generate a month-average series from 1997 to the present and use this to generate accuracy metrics for the forecasts generated by your models from Parts 2 and 3.

```
mlo_data=read.table("co2_weekly_mlo.txt")
head(mlo_data)
```

```
##     V1 V2 V3       V4     V5 V6      V7      V8    V9
## 1 1974  5 19 1974.380 333.37  5 -999.99 -999.99 50.40
## 2 1974  5 26 1974.399 332.95  6 -999.99 -999.99 50.06
## 3 1974  6  2 1974.418 332.35  5 -999.99 -999.99 49.60
## 4 1974  6  9 1974.437 332.20  7 -999.99 -999.99 49.65
## 5 1974  6 16 1974.456 332.37  7 -999.99 -999.99 50.06
## 6 1974  6 23 1974.475 331.73  5 -999.99 -999.99 49.72
```

```
colnames(mlo_data)=c("yr","mon","day","decimal","ppm","ya","tya","s8")



get_num_missing=function(v) {
  neg_ind=v<0
  num_neg=sum(neg_ind)
  return(num_neg)
}

getIndexOfFirstMissing=function(v) {
  f_res=which(v<=0)
  #head(f_res)
  f_res=as.vector(f_res)
  return(f_res[1])

}
library("forecast")
#use auto.arima to fill in missing values
num_missing=get_num_missing(mlo_data$ppm)
loop_count=0
plot(mlo_data$decimal,mlo_data$ppm,main="Before Imputation",type="l")
```



```
while(num_missing>0) {
 print(paste("Loop count is ",loop_count))
 print(paste("num missing is ",num_missing))
 idx_first_missing=getIndexOfFirstMissing(mlo_data$ppm)
 time_at_missing_idx=mlo_data[idx_first_missing-1,"decimal"]
 print(paste("idx_first_missing is ",idx_first_missing))
 input_arima=mlo_data[1:(idx_first_missing-1),]
 print(dim(input_arima))
 input_arima=input_arima[,c("decimal","ppm")]
 print(dim(input_arima))
 #print(input_arima$ppm)
 input_arima_ts=ts(input_arima$ppm,frequency=52,start=c(1974,20))
 #the_model=auto.arima(input_arima_ts)
 the_model=arima(input_arima_ts,order=c(2,1,2))
 print(the_model)
 predicted_obj=predict(the_model)
 predicted_value=as.numeric(predicted_obj[1])
 print(paste("The pred value is",predicted_value))
 mlo_data[idx_first_missing,"ppm"]=predicted_value
 loop_count=loop_count+1
 num_missing=get_num_missing(mlo_data$ppm)
 print(paste("at end of loop, num missing is ",num_missing))
}
```

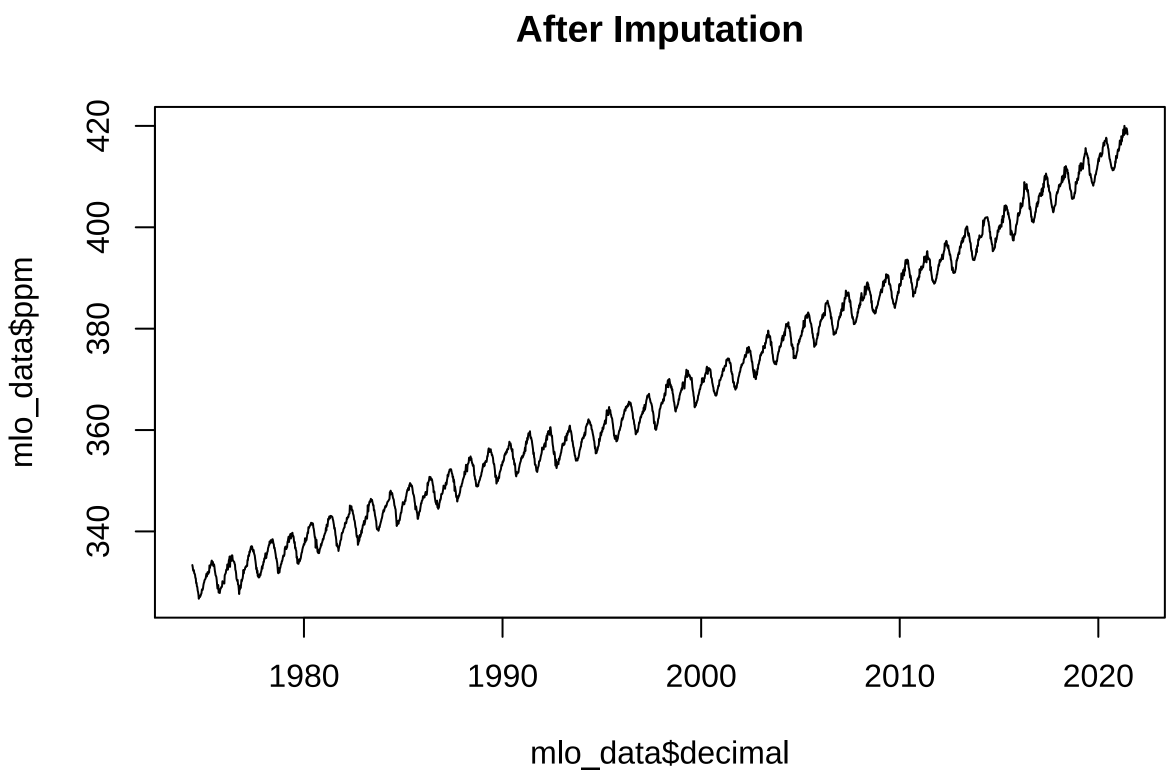
```
## [1] "Loop count is  0"
## [1] "num missing is  18"
## [1] "idx_first_missing is  73"
## [1] 72  9
## [1] 72  2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2373  -0.3766  -1.6309  1.0000
## s.e.  0.1149   0.1136   0.0797  0.0941
##
## sigma^2 estimated as 0.1475:  log likelihood = -36,  aic = 82
## [1] "The pred value is 327.842025144728"
## [1] "at end of loop, num missing is  17"
## [1] "Loop count is  1"
## [1] "num missing is  17"
## [1] "idx_first_missing is  82"
## [1] 81  9
## [1] 81  2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2816  -0.4487  -1.6206  1.0000
## s.e.  0.1066   0.1046   0.0914  0.1099
##
## sigma^2 estimated as 0.1555:  log likelihood = -42.18,  aic = 94.35
## [1] "The pred value is 329.971385458005"
## [1] "at end of loop, num missing is  16"
## [1] "Loop count is  2"
## [1] "num missing is  16"
## [1] "idx_first_missing is  83"
## [1] 82  9
## [1] 82  2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2815  -0.4484  -1.6206  1.0000
## s.e.  0.1053   0.1031   0.0877  0.1052
##
## sigma^2 estimated as 0.1536:  log likelihood = -42.17,  aic = 94.34
## [1] "The pred value is 329.971873260087"
## [1] "at end of loop, num missing is  15"
## [1] "Loop count is  3"
## [1] "num missing is  15"
## [1] "idx_first_missing is  84"
## [1] 83  9
## [1] 83  2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2815  -0.4483  -1.6206  1.0000
## s.e.  0.1046   0.1025   0.0844  0.1011
##
## sigma^2 estimated as 0.1517:  log likelihood = -42.16,  aic = 94.33
## [1] "The pred value is 329.859591339373"
## [1] "at end of loop, num missing is  14"
## [1] "Loop count is  4"
## [1] "num missing is  14"
## [1] "idx_first_missing is  85"
## [1] 84  9
## [1] 84  2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
```

```
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2813  -0.4480  -1.6206  1.0000
## s.e.  0.1040   0.1018   0.0815  0.0974
##
## sigma^2 estimated as 0.1499:  log likelihood = -42.15,  aic = 94.3
## [1] "The pred value is 329.715452327518"
## [1] "at end of loop, num missing is  13"
## [1] "Loop count is  5"
## [1] "num missing is  13"
## [1] "idx_first_missing is  111"
## [1] 110   9
## [1] 110   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2647  -0.3572  -1.5350  0.7401
## s.e.  0.1703   0.1527   0.1206  0.0915
##
## sigma^2 estimated as 0.2623:  log likelihood = -82.24,  aic = 174.49
## [1] "The pred value is 333.983016741049"
## [1] "at end of loop, num missing is  12"
## [1] "Loop count is  6"
## [1] "num missing is  12"
## [1] "idx_first_missing is  410"
## [1] 409   9
## [1] 409   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4944  -0.5660  -1.6623  0.8442
## s.e.  0.0887   0.0855   0.0615  0.0499
##
## sigma^2 estimated as 0.2249:  log likelihood = -275.22,  aic = 560.44
## [1] "The pred value is 342.918224731656"
## [1] "at end of loop, num missing is  11"
## [1] "Loop count is  7"
## [1] "num missing is  11"
## [1] "idx_first_missing is  413"
## [1] 412   9
## [1] 412   2
```

```
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4931  -0.5646  -1.6617  0.8435
## s.e.  0.0885   0.0854   0.0612  0.0497
##
## sigma^2 estimated as 0.2237:  log likelihood = -276.17,  aic = 562.34
## [1] "The pred value is 343.443046049082"
## [1] "at end of loop, num missing is  10"
## [1] "Loop count is  8"
## [1] "num missing is  10"
## [1] "idx_first_missing is  414"
## [1] 413   9
## [1] 413   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4931  -0.5646  -1.6617  0.8436
## s.e.  0.0883   0.0852   0.0611  0.0497
##
## sigma^2 estimated as 0.2232:  log likelihood = -276.34,  aic = 562.68
## [1] "The pred value is 343.520671195378"
## [1] "at end of loop, num missing is  9"
## [1] "Loop count is  9"
## [1] "num missing is  9"
## [1] "idx_first_missing is  482"
## [1] 481   9
## [1] 481   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4624  -0.5331  -1.6329  0.8171
## s.e.  0.1331   0.1256   0.0993  0.0751
##
## sigma^2 estimated as 0.2331:  log likelihood = -332.24,  aic = 674.48
## [1] "The pred value is 342.847152088293"
## [1] "at end of loop, num missing is  8"
## [1] "Loop count is  10"
## [1] "num missing is  8"
## [1] "idx_first_missing is  516"
## [1] 515   9
## [1] 515   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4689  -0.5392  -1.6302  0.8149
## s.e.  0.1329   0.1253   0.0996  0.0746
##
## sigma^2 estimated as 0.2235:  log likelihood = -344.92,  aic = 699.83
## [1] "The pred value is 346.021929721749"
## [1] "at end of loop, num missing is  7"
## [1] "Loop count is  11"
## [1] "num missing is  7"
## [1] "idx_first_missing is  517"
## [1] 516   9
## [1] 516   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4689  -0.5392  -1.6302  0.8149
## s.e.  0.1328   0.1251   0.0995  0.0745
##
## sigma^2 estimated as 0.2231:  log likelihood = -345.08,  aic = 700.17
## [1] "The pred value is 346.155496582587"
## [1] "at end of loop, num missing is  6"
## [1] "Loop count is  12"
## [1] "num missing is  6"
## [1] "idx_first_missing is  518"
## [1] 517   9
## [1] 517   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4690  -0.5392  -1.6303  0.8149
## s.e.  0.1326   0.1249   0.0994  0.0744
##
## sigma^2 estimated as 0.2227:  log likelihood = -345.25,  aic = 700.51
## [1] "The pred value is 346.29134588307"
## [1] "at end of loop, num missing is  5"
## [1] "Loop count is  13"
## [1] "num missing is  5"
## [1] "idx_first_missing is  519"
## [1] 518   9
## [1] 518   2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.4690  -0.5392  -1.6303  0.8150
## s.e.  0.1324   0.1248   0.0992  0.0743
##
## sigma^2 estimated as 0.2222:  log likelihood = -345.42,  aic = 700.84
## [1] "The pred value is 346.418884401108"
## [1] "at end of loop, num missing is  4"
## [1] "Loop count is  14"
## [1] "num missing is  4"
## [1] "idx_first_missing is  1640"
## [1] 1639    9
## [1] 1639    2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5109  -0.5820  -1.6287  0.8109
## s.e.  0.0710   0.0663   0.0526  0.0381
##
## sigma^2 estimated as 0.2242:  log likelihood = -1100.28,  aic = 2210.56
## [1] "The pred value is 376.84541414599"
## [1] "at end of loop, num missing is  3"
## [1] "Loop count is  15"
## [1] "num missing is  3"
## [1] "idx_first_missing is  1781"
## [1] 1780    9
## [1] 1780    2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5563  -0.6251  -1.6617  0.8336
## s.e.  0.0605   0.0571   0.0442  0.0328
```

```
##
## sigma^2 estimated as 0.2296:  log likelihood = -1216.07,  aic = 2442.15
## [1] "The pred value is 387.37608547821"
## [1] "at end of loop, num missing is  2"
## [1] "Loop count is  16"
## [1] "num missing is  2"
## [1] "idx_first_missing is  1782"
## [1] 1781    9
## [1] 1781    2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5563  -0.6251  -1.6617  0.8336
## s.e.  0.0605   0.0571   0.0441  0.0328
##
## sigma^2 estimated as 0.2295:  log likelihood = -1216.26,  aic = 2442.51
## [1] "The pred value is 387.042772755756"
## [1] "at end of loop, num missing is  1"
## [1] "Loop count is  17"
## [1] "num missing is  1"
## [1] "idx_first_missing is  1783"
## [1] 1782    9
## [1] 1782    2
##
## Call:
## arima(x = input_arima_ts, order = c(2, 1, 2))
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5563  -0.6251  -1.6617  0.8336
## s.e.  0.0605   0.0570   0.0441  0.0327
##
## sigma^2 estimated as 0.2293:  log likelihood = -1216.44,  aic = 2442.88
## [1] "The pred value is 386.620252075717"
## [1] "at end of loop, num missing is  0"
```

```
plot(mlo_data$decimal,mlo_data$ppm,main="After Imputation",type="l")
```



```
#library(Hmisc)
#describe(mlodf)
# get regular data (from yr dec)

#mlodf_impute=data.frame(
#  yr_dec=mlodf$YRDEC,
#  co2=mlodf$CO2
#)
#mlodf_impute=mlodf_impute[mlodf_impute$co2>0,]
#
#print("dim before")
#tail(mlodf_impute)
#print(dim(mlodf_impute))
#for(i in 1:nrow(mlodf)) {
#  temp_row=mlodf[i,]
#  if(temp_row$YRAGO>0) {
#    new_row=c(temp_row[1,"YRDEC"]-1,temp_row[1,"YRAGO"])
#    mlodf_impute=rbind(mlodf_impute,new_row)
#    if(i==1) {
#     print('tail')
#      print(tail(mlodf_impute))
#    }
#  }
#  if(temp_row$TENYRAGO>0) {
#    new_row=c(temp_row[1,"YRDEC"]-10,temp_row[1,"TENYRAGO"])
#    mlodf_impute=rbind(mlodf_impute,new_row)
#  }
#}
#print("dim after")
#print(dim(mlodf_impute))
#mlodf_impute=mlodf_impute[order(mlodf_impute$yr_dec),]
#plot(mlodf_impute[,1],mlodf_impute[,2],type="l")
#write.csv(mlodf_impute,file="eddie.dat",row.names = FALSE)
#mlodf_ts=as.ts(mlodf_impute$co2)
#plot(mlodf_ts)
#write.csv(mlodf_ts,file="eddie.dat",sep="\t")
```

Part 5 (5 points)

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

```
#noaa_train=mlodf_impute[mlodf_impute$yr_dec<2020,]
#noaa_test=mlodf_impute[mlodf_impute$yr_dec>=2020,]
```