Project 2

Shayne Cassidy, Sarah Nelson, Sarah Winston Nathan

12-6-2019

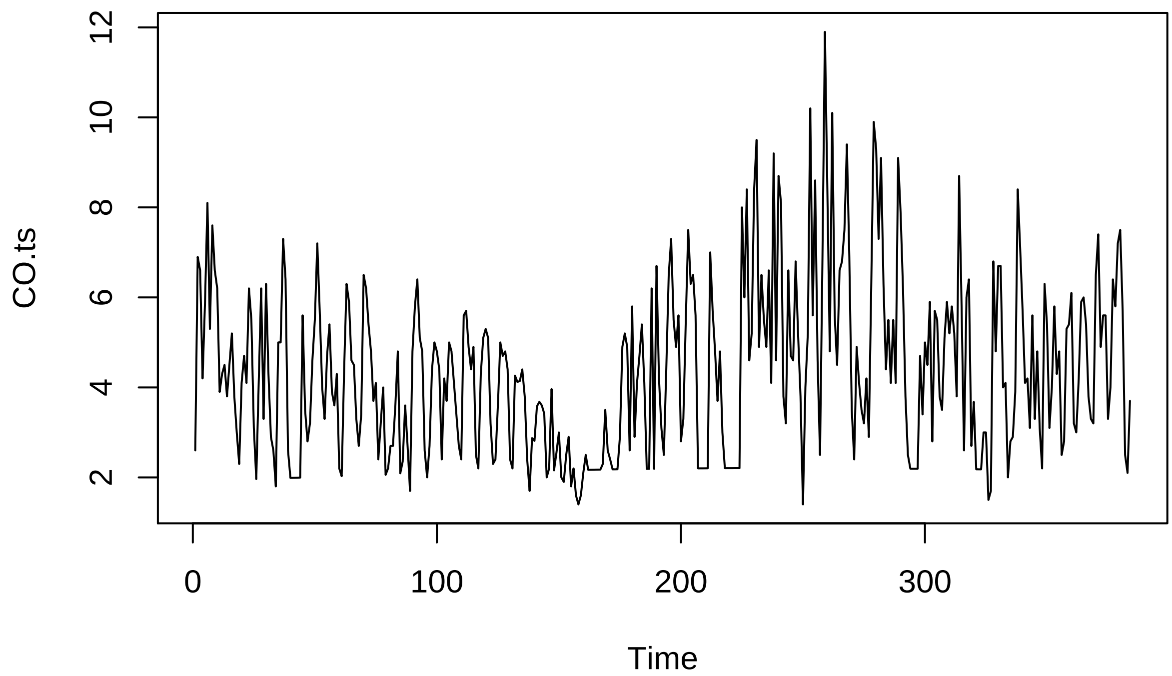
Load data and impute missing values

```
setwd(datadir)
airquality = read.csv('AirQualityUCI.csv')

# replace -200 with NA
airquality[airquality == -200] <- NA

# convert integer type to numeric
intcols = c(4,5,7,8,9,10,11,12)
for(i in 1:length(intcols)){
  airquality[,intcols[i]] <- as.numeric(airquality[,intcols[i]])
}

setwd(sourcedir)

# create new data frame with just CO and NO2
AQdata = airquality[,c(3,10)]

# impute missing air quality data
f <- ~ CO.GT. + NO2.GT.
t <- c(seq(1,dim(AQdata)[1],1))
i <- mnimput(f, AQdata, eps=1e-3, ts=TRUE, method='gam',
             ga.control=list(formula=paste(names(AQdata)[c(1:3)],'~ns(t,2)')))

# set airquality to imputed data
AQdata <- i$filled.dataset

# aggregate to daily maxima for model building
dailyAQ <- aggregate(AQdata, by=list(as.Date(airquality[,1],"%m/%d/%Y")), FUN=max)

# remove last 7 days
dailyAQ <- dailyAQ[1:(dim(dailyAQ)[1]-7),]
```

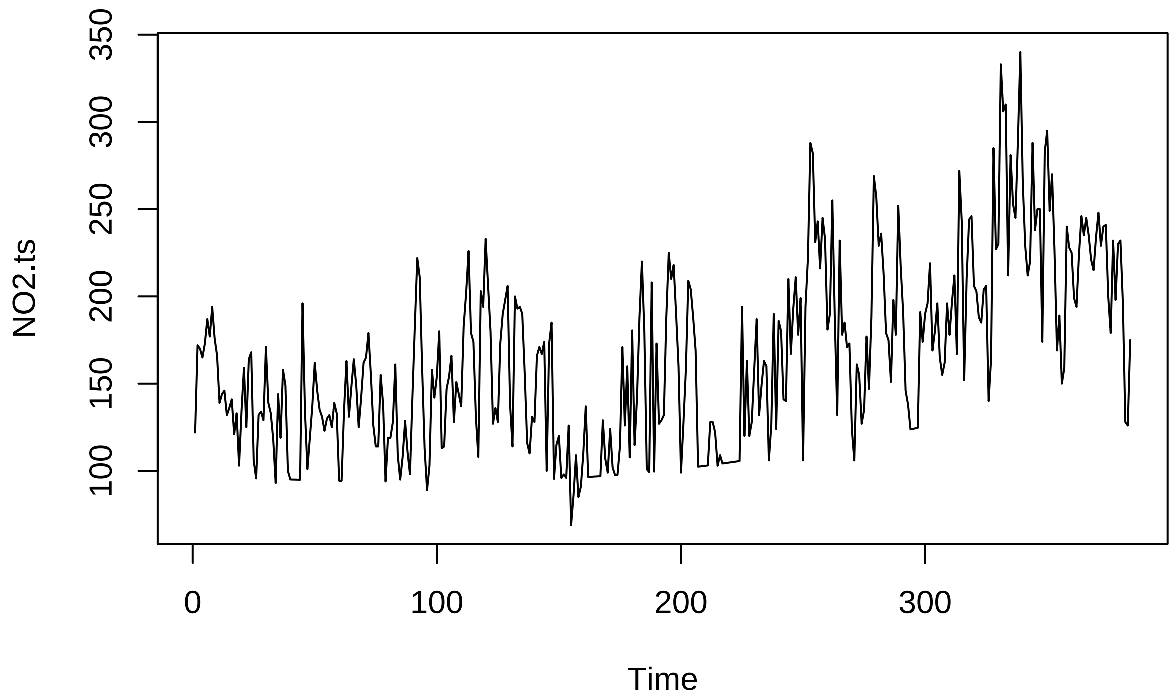
Part 1: Building Univariate Time Series Models

```
AQ.CO <- dailyAQ$CO.GT.
#AQ.CO <- AQdata$CO.GT.
AQ.NO2 <- dailyAQ$NO2.GT.
#AQ.NO2<-AQdata$NO2.GT.

CO.ts<-ts(AQ.CO)
NO2.ts<-ts(AQ.NO2)
```

```r
plot(CO.ts)
```

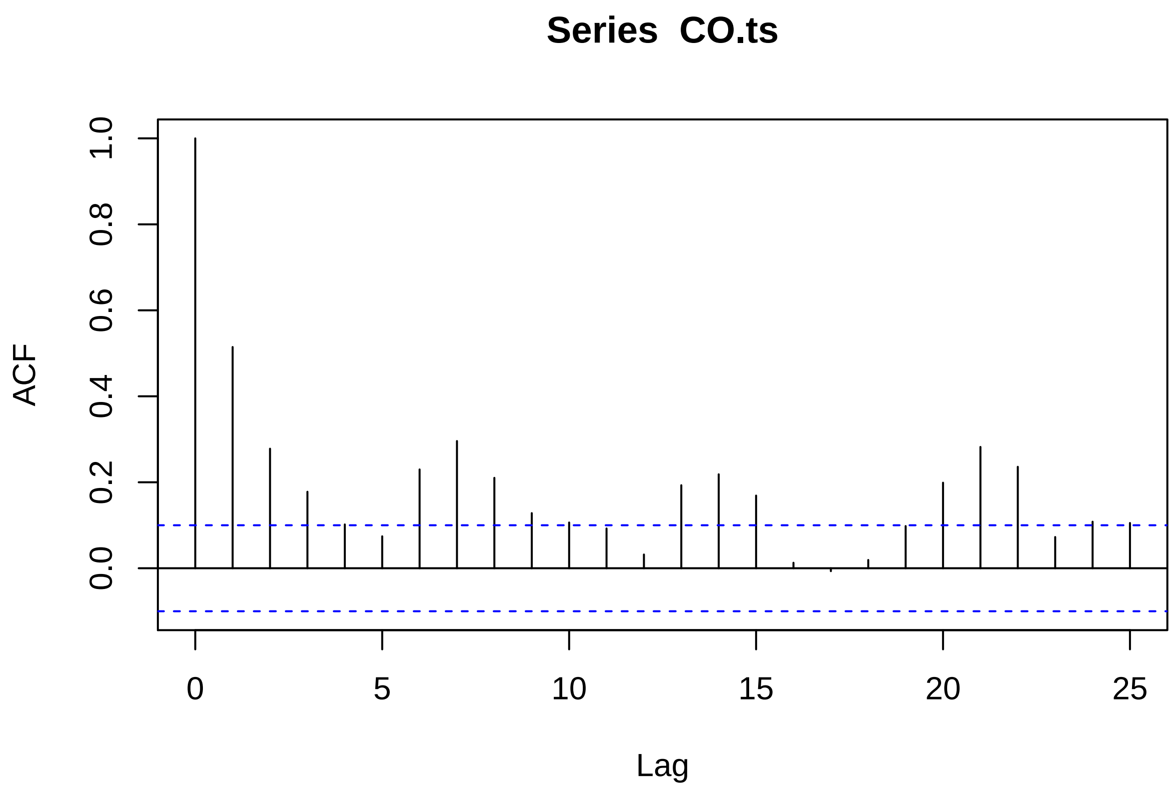



```r
plot(NO2.ts)
```

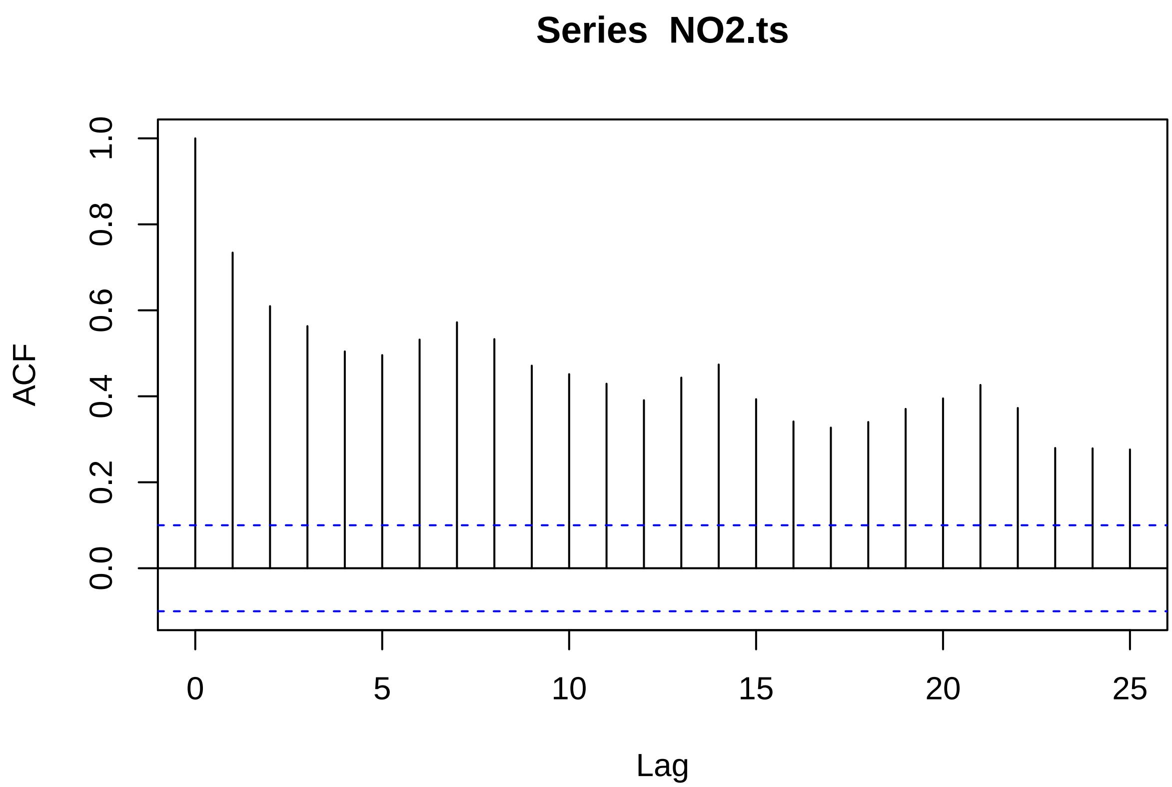


Part A: Seasonality

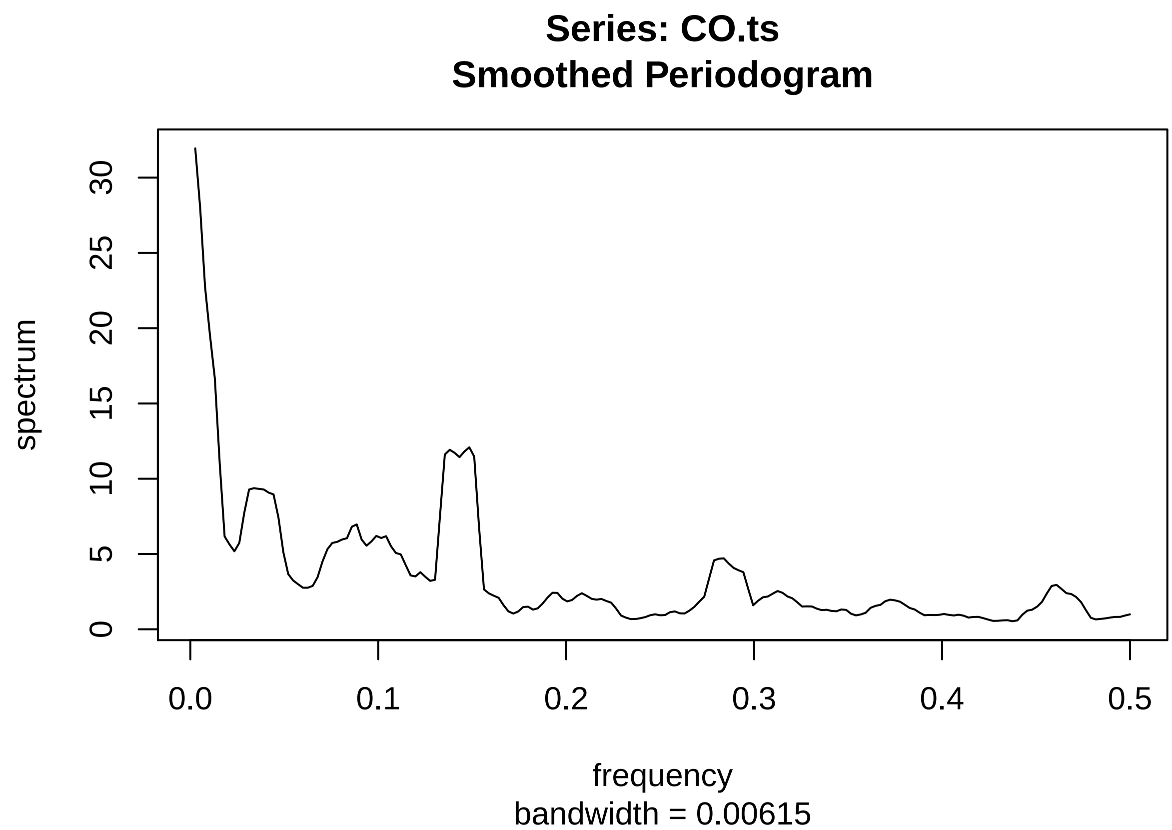
```
acf(CO.ts)
```




```r
acf(NO2.ts)
```




```r
# both show sinusoidal exponential decay --> AR model

pg.CO <- spec.pgram(CO.ts,spans=9,demean=T,log='no')
```



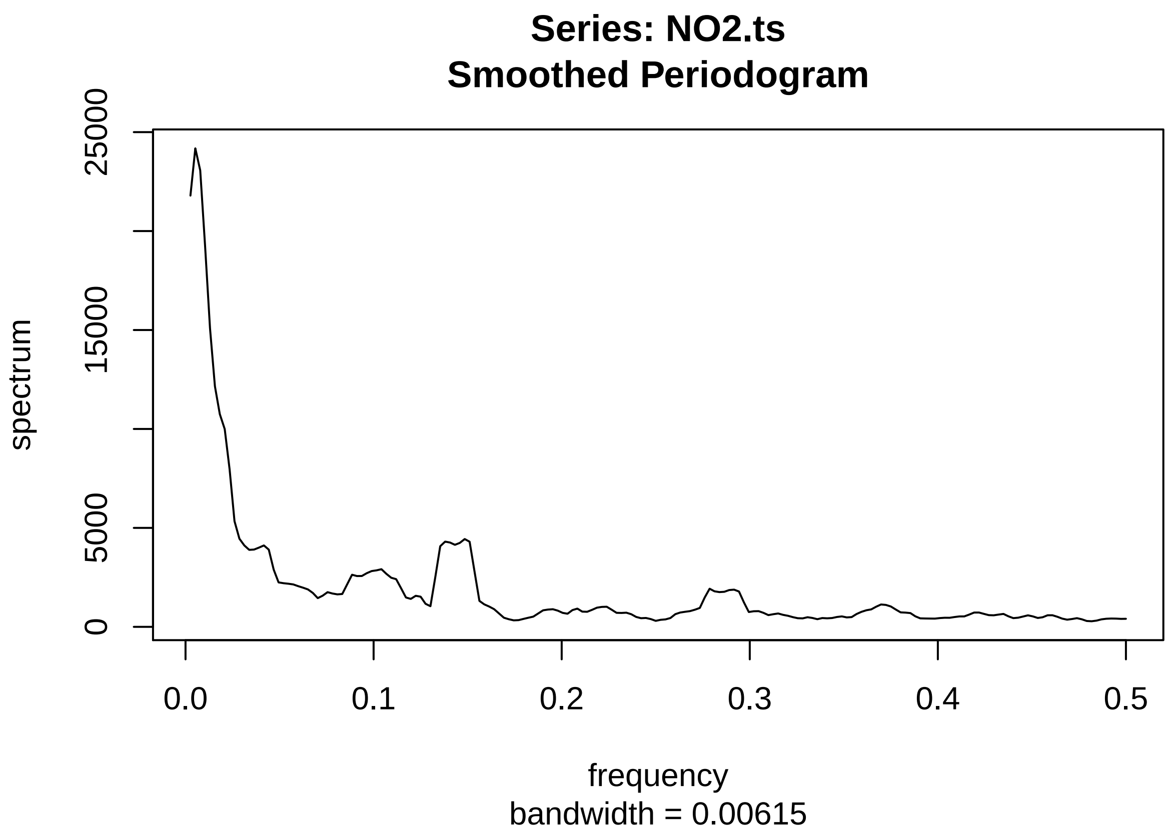

```
# spikes in periodagram at repeated frequencies --> indicates seasonality present
max.pg.CO<-pg.CO$freq[which(pg.CO$spec==max(pg.CO$spec))]

# Where is the peak? -->0.002604167
max.pg.CO
```

```
## [1] 0.002604167
```

```
# What is the period? -->384
1/max.pg.CO
```

```
## [1] 384
```

```
pg.NO2 <- spec.pgram(NO2.ts,spans=9,demean=T,log='no')
```




```
# spikes in periodagram at repeated frequencies --> indicates seasonality present
max.pg.NO2<-pg.CO$freq[which(pg.NO2$spec==max(pg.NO2$spec))]

# Where is the peak? -->0.00520833
max.pg.NO2
```

```
## [1] 0.005208333
```

```
# What is the period? -->192
1/max.pg.NO2
```

```
## [1] 192
```

```
# What are the periods of the next biggest peaks?
# sort spectrum from largest to smallest and find index
sorted.spec <- sort(pg.CO$spec, decreasing=T, index.return=T)
names(sorted.spec)
```

```
## [1] "x"  "ix"
```

```
# corresponding periods
sorted.omegas <- pg.CO$freq[sorted.spec$ix]
sorted.Ts <- 1/pg.CO$freq[sorted.spec$ix]
```

```r
# look at first 20
sorted.omegas[1:20]
```

```
##  [1] 0.002604167 0.005208333 0.007812500 0.010416667 0.013020833
##  [6] 0.148437500 0.138020833 0.145833333 0.140625000 0.135416667
## [11] 0.151041667 0.143229167 0.015625000 0.033854167 0.036458333
## [16] 0.039062500 0.031250000 0.041666667 0.044270833 0.028645833
```

```r
sorted.Ts[1:20]
```

```
##  [1] 384.000000 192.000000 128.000000  96.000000  76.800000   6.736842
##  [7]   7.245283   6.857143   7.111111   7.384615   6.620690   6.981818
## [13]  64.000000  29.538462  27.428571  25.600000  32.000000  24.000000
## [19]  22.588235  34.909091
```

```r
# evens around 7
period<-7
```

Part B: Trends

```r
# Build a new model, CO.trend which predicts CO.ts based on the time variable
time<-c(1:(length(CO.ts)))
CO.trend<-lm(CO.ts ~ time)
NO2.trend<-lm(NO2.ts ~ time)

summary(CO.trend)
```

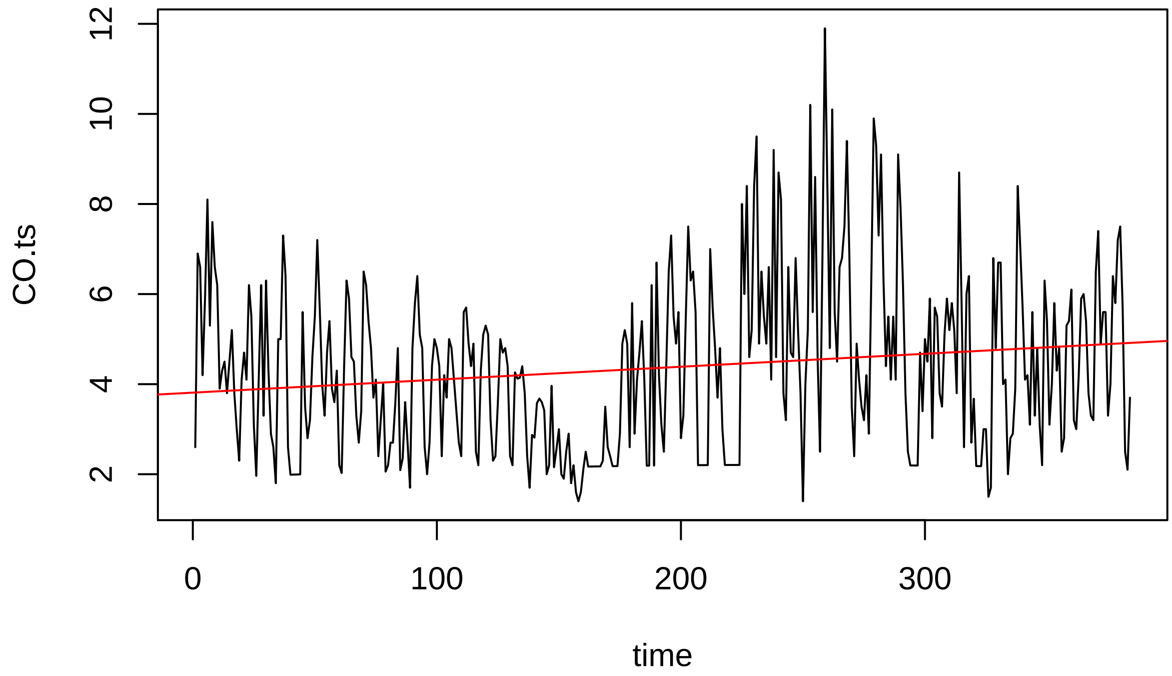
```
##
## Call:
## lm(formula = CO.ts ~ time)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2485 -1.6980 -0.0525  1.0863  7.3442
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.810929   0.192140  19.834  < 2e-16 ***
## time        0.002876   0.000865   3.325  0.00097 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.879 on 382 degrees of freedom
## Multiple R-squared:  0.02813,    Adjusted R-squared:  0.02558
## F-statistic: 11.06 on 1 and 382 DF,  p-value: 0.0009695
```

```r
summary(NO2.trend)
```

```
##
## Call:
## lm(formula = NO2.ts ~ time)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -87.389 -34.365   2.159  27.847 137.895
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 115.16473    4.40111   26.17   <2e-16 ***
## time          0.25646    0.01981   12.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.04 on 382 degrees of freedom
## Multiple R-squared:  0.3049, Adjusted R-squared:  0.3031
## F-statistic: 167.6 on 1 and 382 DF,  p-value: < 2.2e-16
```

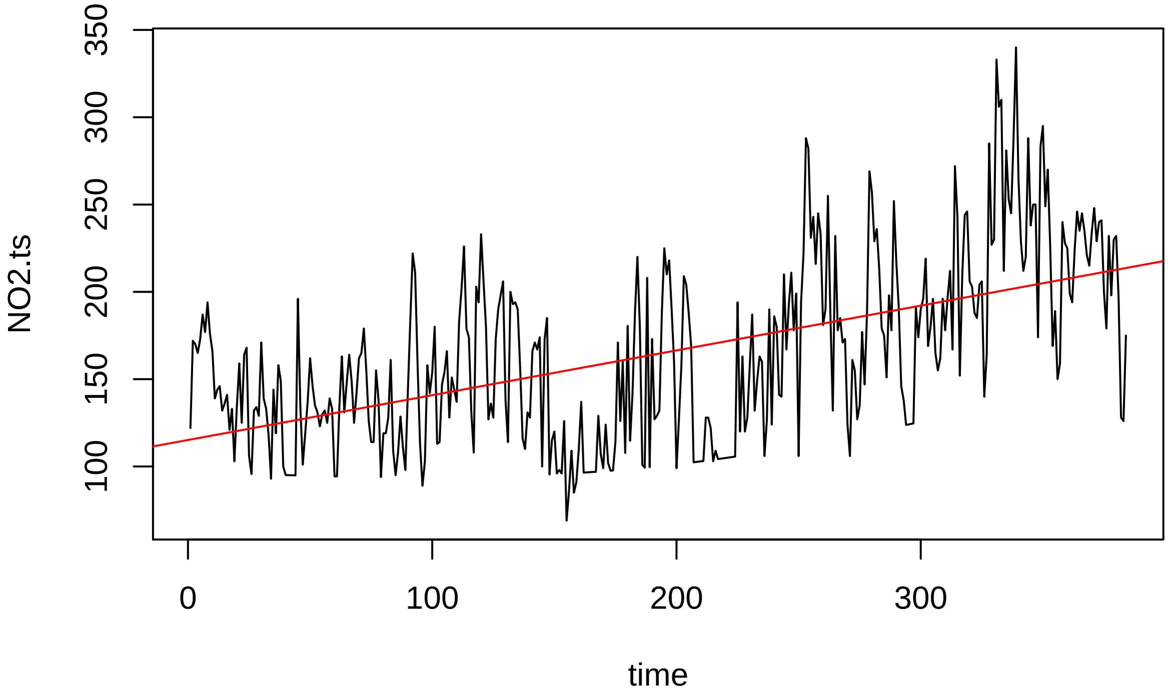
Plot CO.trend model

```r
{plot(time, CO.ts, type = "l")
abline(CO.trend, col = "red")}
```



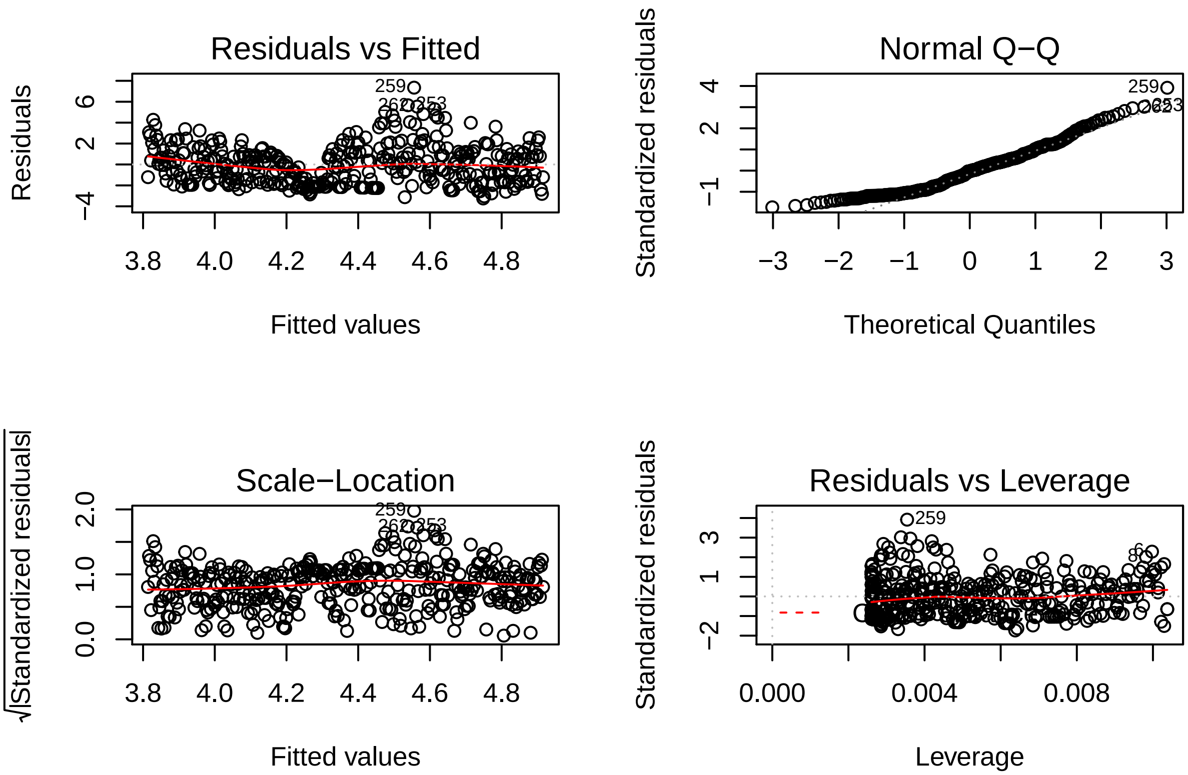
Plot NO2.trend model

```
{plot(time, NO2.ts, type = "l")
abline(NO2.trend, col = "red")}
```



Model diagnostics for CO.trend

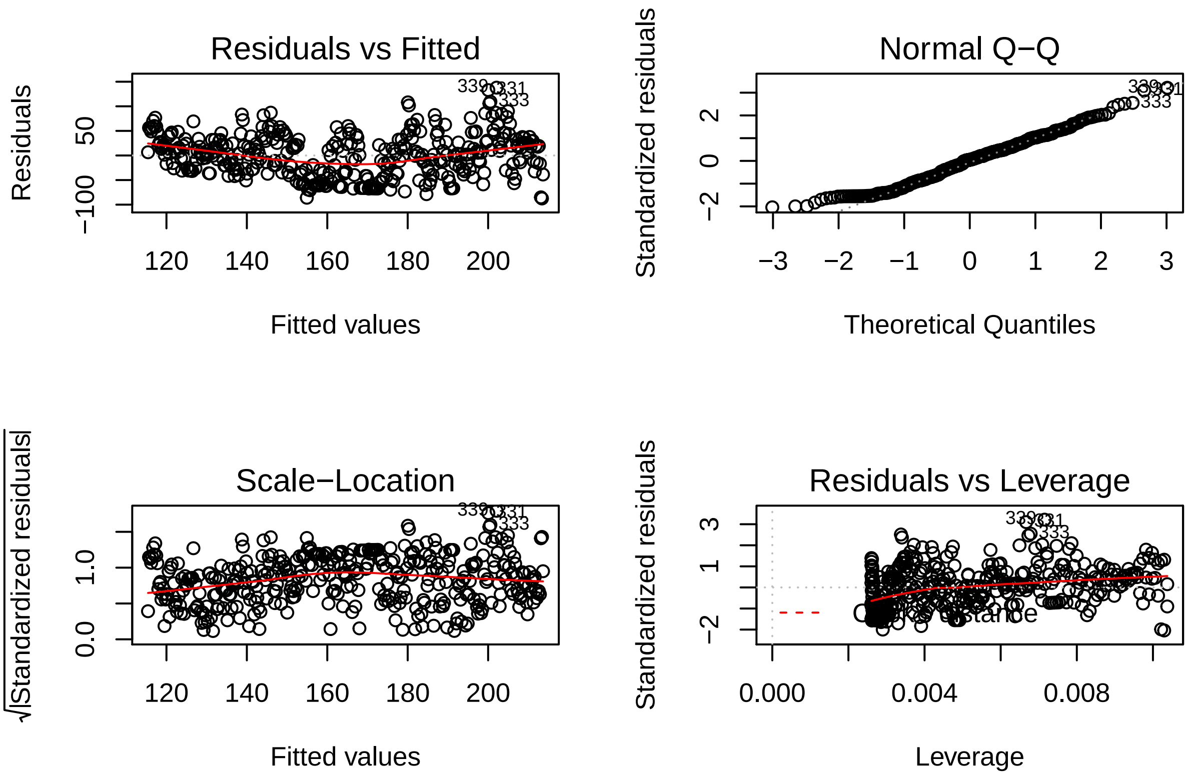
```
par(mfrow=c(2,2))
plot(CO.trend, labels.id = NULL)
```




```
par(mfrow=c(1,1))
```

Model diagnostics for NO2.trend

```
par(mfrow=c(2,2))
plot(NO2.trend, labels.id = NULL)
```




```r
par(mfrow=c(1,1))
```

add seasonality component to CO.trend

```r
CO.trend.seasonal <- lm(CO.ts[time] ~ time + sin(2*pi*time/365) + cos(2*pi*time/365))
summary(CO.trend.seasonal)
```

```
##
## Call:
## lm(formula = CO.ts[time] ~ time + sin(2 * pi * time/365) + cos(2 *
##     pi * time/365))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6868 -1.4345 -0.1749  1.3084  6.7229
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.778564   0.244485  19.545  < 2e-16 ***
## time                   -0.002197   0.001177  -1.866  0.06276 .
## sin(2 * pi * time/365) -1.096781   0.187652  -5.845 1.09e-08 ***
## cos(2 * pi * time/365)  0.374537   0.129044   2.902  0.00392 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.795 on 380 degrees of freedom
## Multiple R-squared:  0.1179, Adjusted R-squared:  0.1109
## F-statistic: 16.92 on 3 and 380 DF,  p-value: 2.449e-10
```

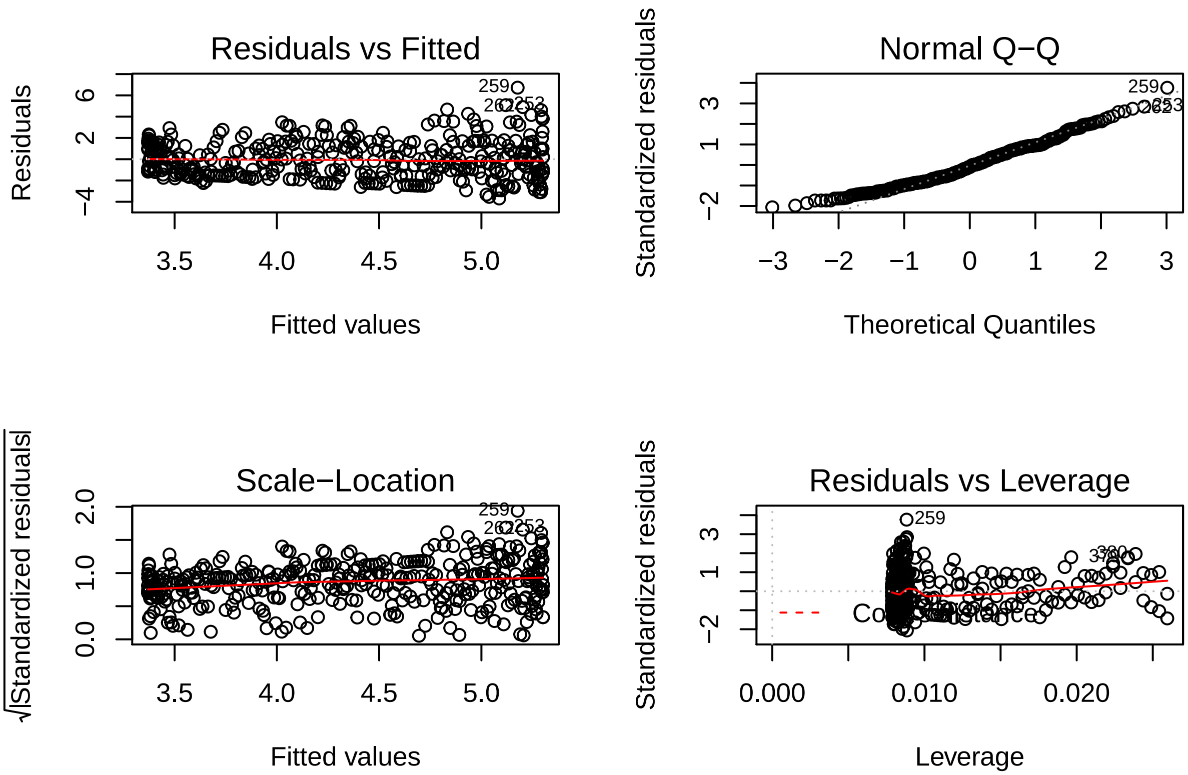
add seasonality component to NO2.trend

```r
NO2.trend.seasonal <- lm(NO2.ts[time] ~ time + sin(2*pi*time/365) + cos(2*pi*time/365))
summary(NO2.trend.seasonal)
```

```
##
## Call:
## lm(formula = NO2.ts[time] ~ time + sin(2 * pi * time/365) + cos(2 *
##     pi * time/365))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -103.73  -31.18   -5.36   26.57  122.08
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            117.10474    5.49391  21.315  < 2e-16 ***
## time                     0.24101    0.02645   9.111  < 2e-16 ***
## sin(2 * pi * time/365)   0.16176    4.21681   0.038    0.969
## cos(2 * pi * time/365)  21.28474    2.89980   7.340 1.31e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.33 on 380 degrees of freedom
## Multiple R-squared:  0.3928, Adjusted R-squared:  0.388
## F-statistic: 81.93 on 3 and 380 DF,  p-value: < 2.2e-16
```

Model diagnostics for CO.trend.seasonal

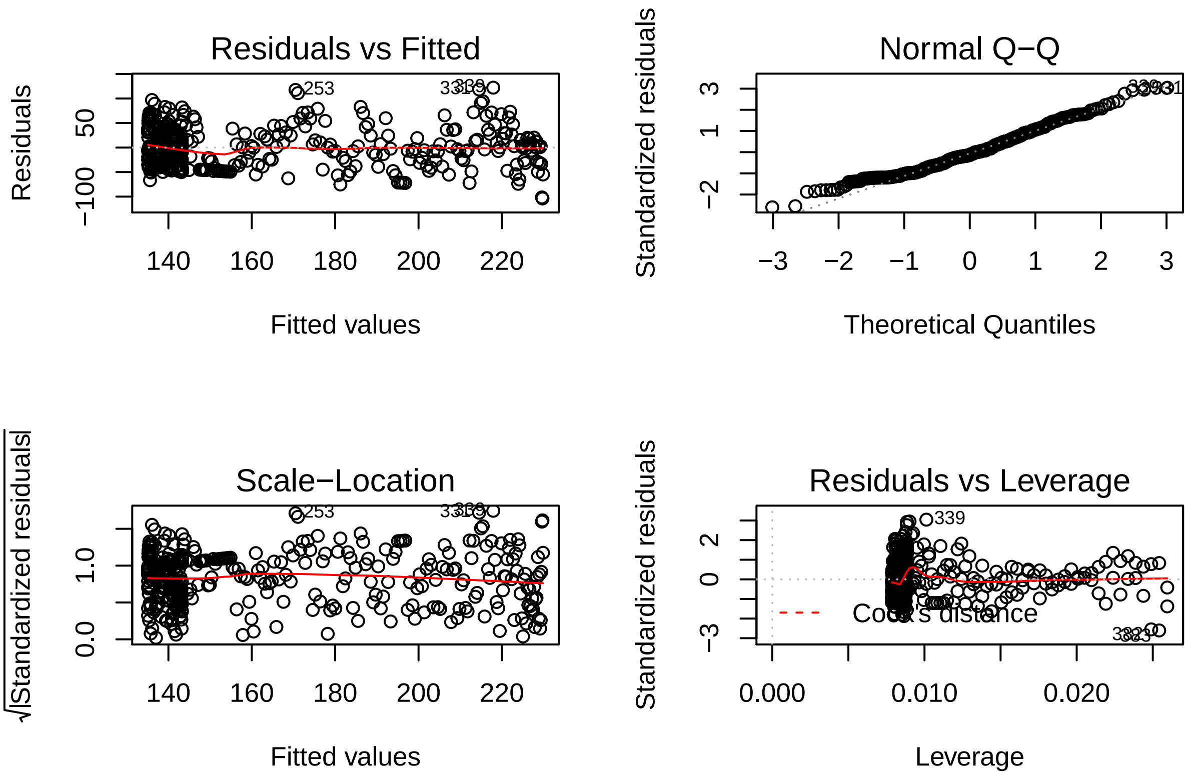
```r
par(mfrow=c(2,2))
plot(CO.trend.seasonal, labels.id = NULL)
```




```r
par(mfrow=c(1,1))
```

Model diagnostics for NO2.trend.seasonal

```r
par(mfrow=c(2,2))
plot(NO2.trend.seasonal, labels.id = NULL)
```




```r
par(mfrow=c(1,1))
```

Part C: Auto-Regressive and Moving Average

#Get the residuals from the CO.trend.seasonal model above and store in e.ts:

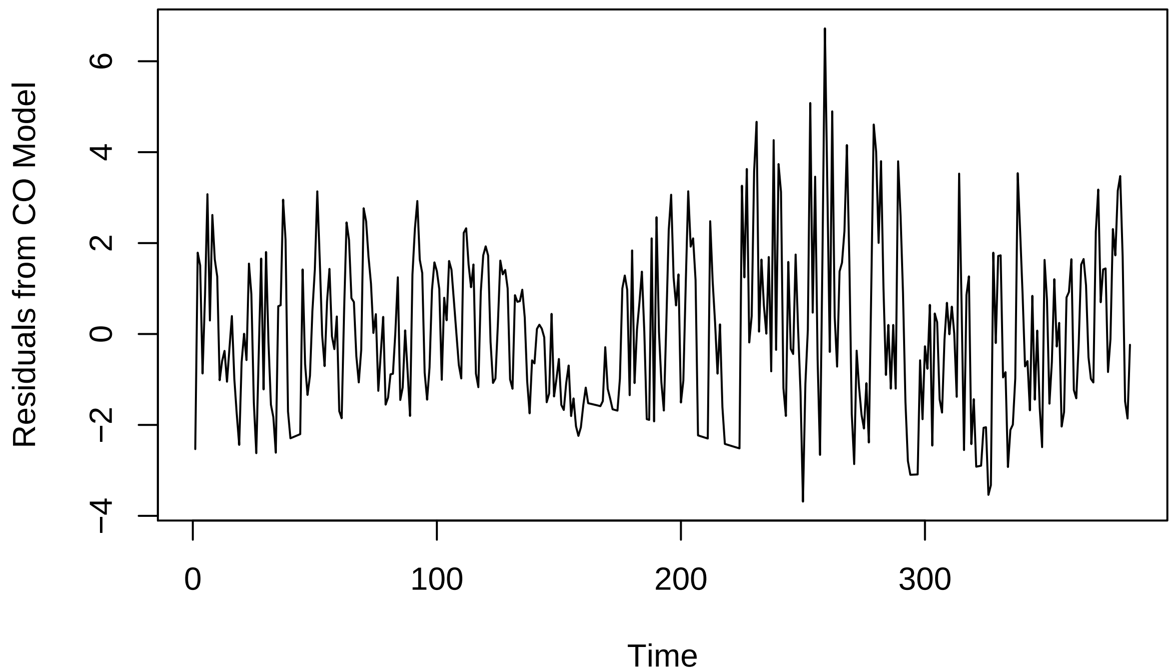
```r
e.ts.CO<-ts(CO.trend.seasonal$residuals)
```

#Get the residuals from the NO2.trend.seasonal model above and store in e.ts:

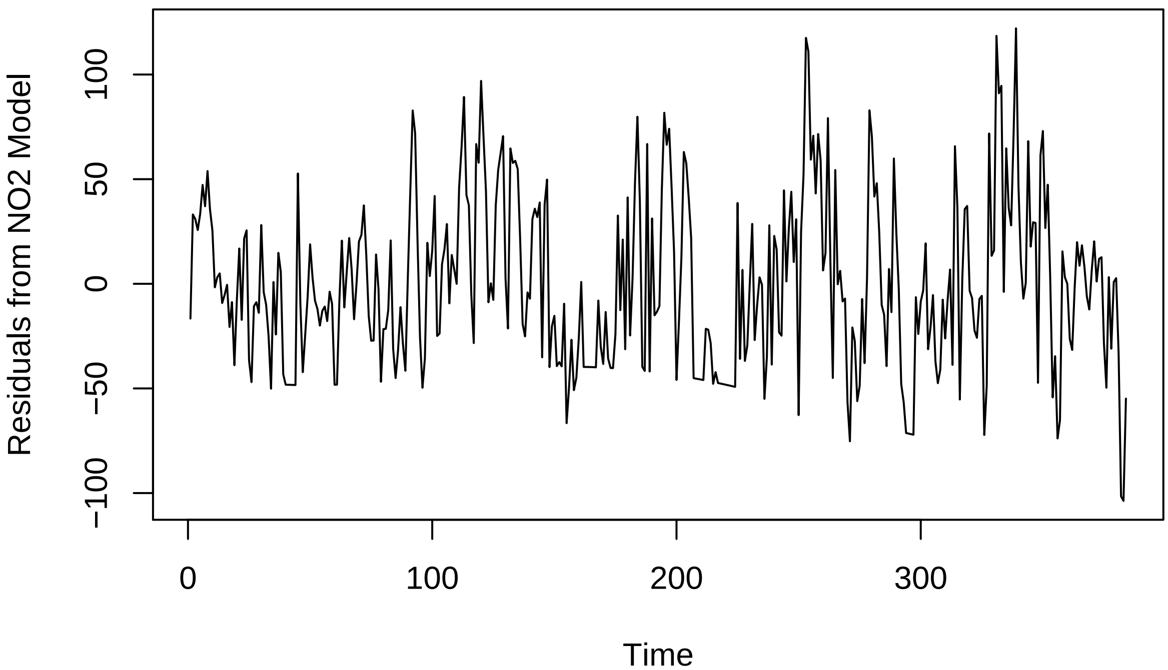
```r
e.ts.NO2<-ts(NO2.trend.seasonal$residuals)
```

#Plot the residuals for the CO.trend.seasonal model NO2.trend.seasonal

```r
plot(e.ts.CO, ylab = "Residuals from CO Model")
```

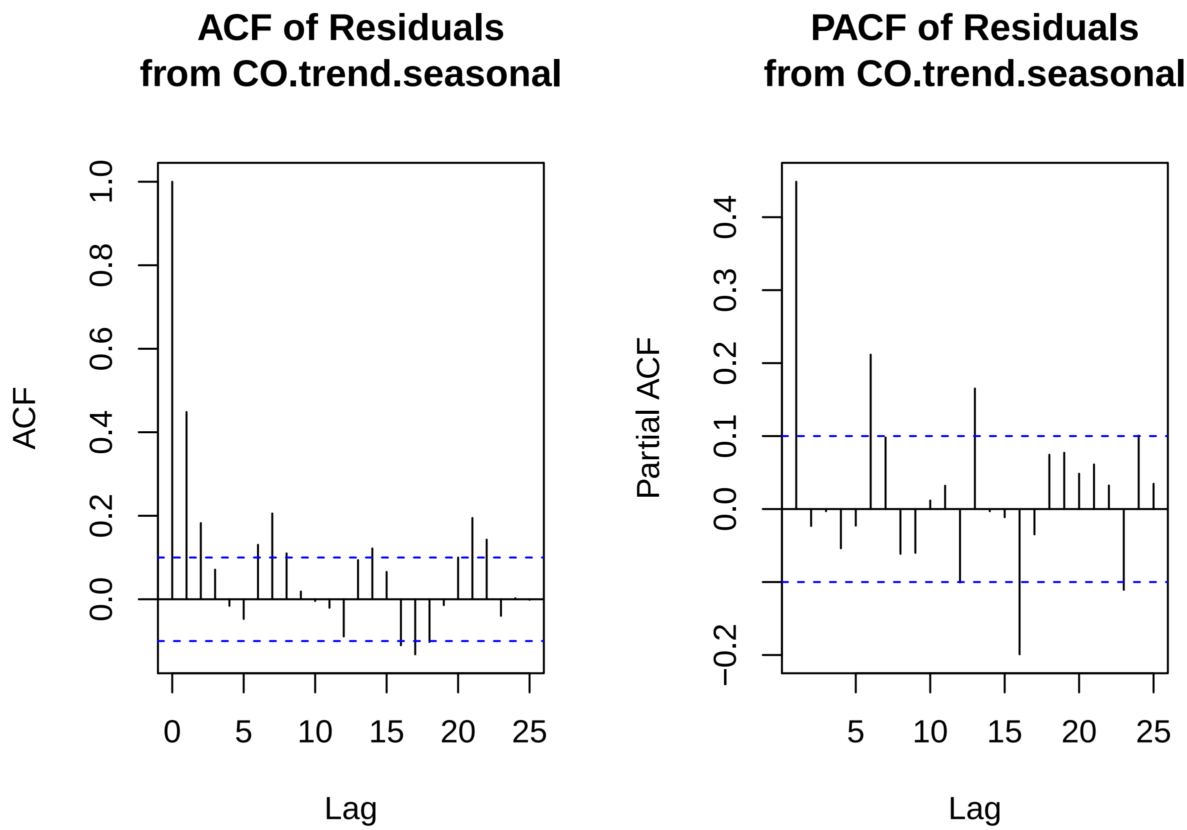



```r
plot(e.ts.NO2, ylab = "Residuals from NO2 Model")
```



Plot the autocorrelation (ACF) and partial autocorrelation (PACF) of the residuals of CO.trend.seasonal

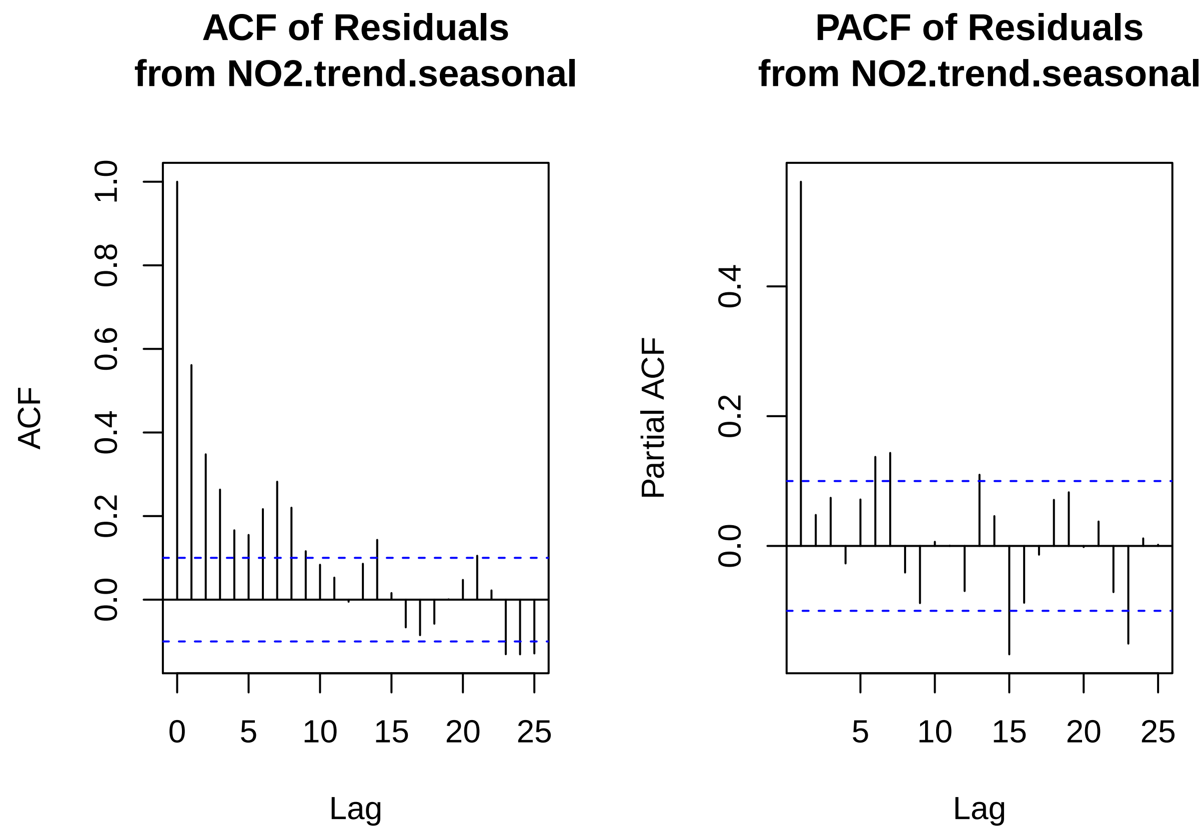
```
par(mfrow=c(1,2))
acf(e.ts.CO, main="ACF of Residuals\nfrom CO.trend.seasonal")
pacf(e.ts.CO,main="PACF of Residuals\nfrom CO.trend.seasonal")
```




```r
par(mfrow=c(1,1))
```

Plot the autocorrelation (ACF) and partial autocorrelation (PACF) of the residuals of NO2.trend.seasonal

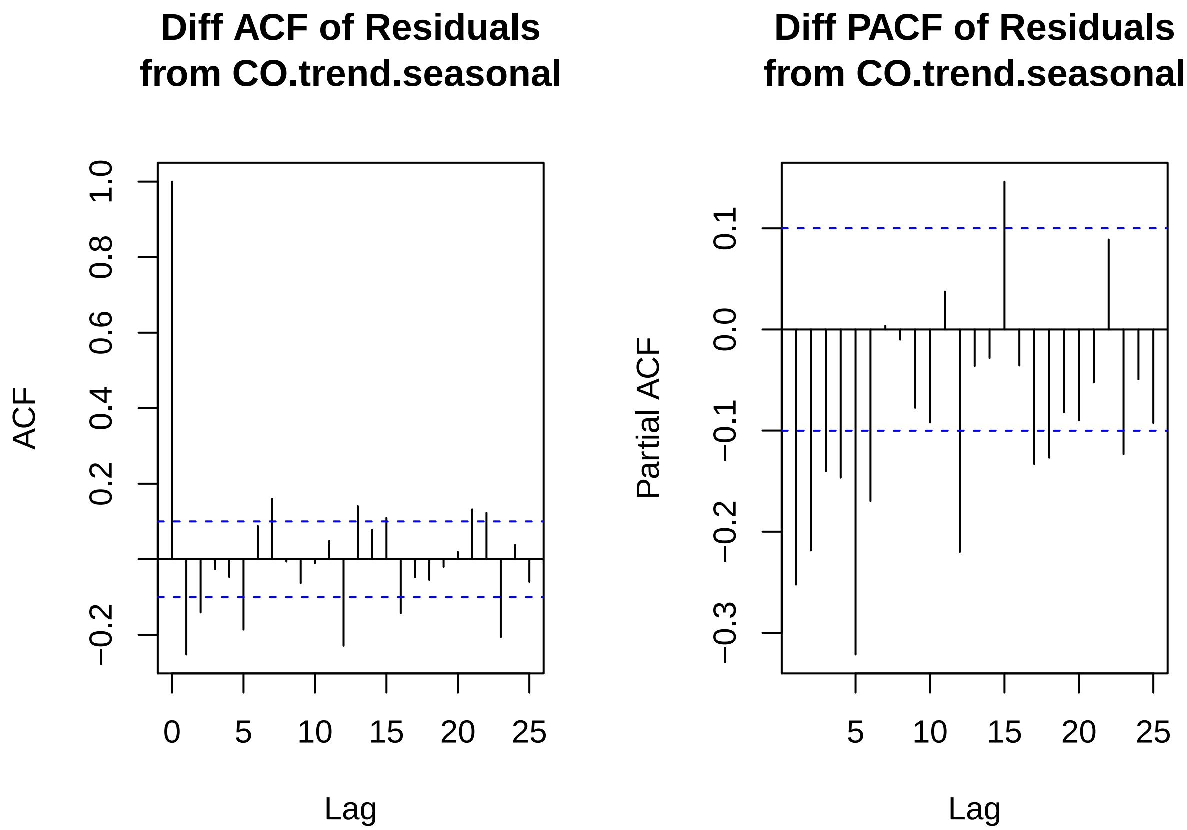
```r
par(mfrow=c(1,2))
acf(e.ts.NO2, main="ACF of Residuals\nfrom NO2.trend.seasonal")
pacf(e.ts.NO2,main="PACF of Residuals\nfrom NO2.trend.seasonal")
```




```
par(mfrow=c(1,1))
```

Do we need to consider a first order difference of our residuals?

```
par(mfrow=c(1,2))
acf(diff(e.ts.CO), main="Diff ACF of Residuals\nfrom CO.trend.seasonal")
pacf(diff(e.ts.CO),main="Diff PACF of Residuals\nfrom CO.trend.seasonal")
```




```
par(mfrow=c(1,1))
```

Choose p and q terms for e.ts.CO based on the acf and pacf

ar(1) p=1

```
CO.ar1 <- arima(e.ts.CO, order=c(1,0,0), include.mean=FALSE)
summary(CO.ar1)
```

```
##
## Call:
## arima(x = e.ts.CO, order = c(1, 0, 0), include.mean = FALSE)
##
## Coefficients:
##          ar1
##       0.4498
## s.e.  0.0456
##
## sigma^2 estimated as 2.543:  log likelihood = -724.18,  aic = 1452.36
##
## Training set error measures:
```

```
##                        ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.000426654 1.594642 1.283419 84.16402 197.3428 0.9209473
##                     ACF1
## Training set 0.009997023
```

ma(3) p=0, q=3

```
CO.ma3 <- arima(e.ts.CO, order=c(0,0,3), include.mean=FALSE)
summary(CO.ma3)
```

```
##
## Call:
## arima(x = e.ts.CO, order = c(0, 0, 3), include.mean = FALSE)
##
## Coefficients:
##          ma1     ma2     ma3
##       0.4482  0.1959  0.1077
## s.e.  0.0504  0.0576  0.0657
##
## sigma^2 estimated as 2.539:  log likelihood = -723.85,  aic = 1455.71
##
## Training set error measures:
##                         ME     RMSE     MAE      MPE     MAPE      MASE
## Training set 0.0004926393 1.593276 1.28069 90.86154 189.2615 0.9189885
##                     ACF1
## Training set 0.005819724
```

arma(1,3) p=1, q=3

```
CO.arma13 <- arima(e.ts.CO, order=c(1,0,3), include.mean=FALSE)
summary(CO.arma13)
```

```
##
## Call:
## arima(x = e.ts.CO, order = c(1, 0, 3), include.mean = FALSE)
##
## Coefficients:
##          ar1     ma1     ma2     ma3
##       0.1585  0.2976  0.1327  0.0857
## s.e.  0.2374  0.2315  0.1223  0.0793
##
## sigma^2 estimated as 2.535:  log likelihood = -723.62,  aic = 1457.24
##
## Training set error measures:
##                         ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.0004404388 1.592298 1.278017 89.87746 193.7119 0.9170703
##                      ACF1
## Training set 0.0005259324
```

use the auto.arima function on e.ts.CO

```
CO.auto <- auto.arima(e.ts.CO,approximation=FALSE)
summary(CO.auto)
```

```
## Series: e.ts.CO
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.4498
## s.e.  0.0456
##
## sigma^2 estimated as 2.55:  log likelihood=-724.18
## AIC=1452.36   AICc=1452.39   BIC=1460.26
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.000426654 1.594642 1.283419 84.16402 197.3428 0.9209473
##                     ACF1
## Training set 0.009997023
```

Choose p and q terms for e.ts.NO2 based on the acf and pacf

ar(1) p=1

```
NO2.ar1 <- arima(e.ts.NO2, order=c(1,0,0), include.mean=FALSE)
summary(NO2.ar1)
```

```
##
## Call:
## arima(x = e.ts.NO2, order = c(1, 0, 0), include.mean = FALSE)
##
## Coefficients:
##          ar1
##       0.5628
## s.e.  0.0422
##
## sigma^2 estimated as 1100:  log likelihood = -1889.63,  aic = 3783.26
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.07289483 33.16431 26.05539 240.1173 306.1687 0.9353814
##                     ACF1
## Training set -0.02845741
```

ma(10) p=0, q=10

```r
NO2.ma10 <- arima(e.ts.NO2, order=c(0,0,10), include.mean=FALSE)
summary(NO2.ma10)
```

```
##
## Call:
## arima(x = e.ts.NO2, order = c(0, 0, 10), include.mean = FALSE)
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5     ma6     ma7     ma8
##        0.480  0.2775  0.2560  0.0939  0.0496  0.0628  0.1842  0.1849
## s.e.   0.054  0.0586  0.0662  0.0621  0.0613  0.0662  0.0567  0.0585
##          ma9    ma10
##       0.1294  0.1198
## s.e.  0.0640  0.0715
##
## sigma^2 estimated as 1043:  log likelihood = -1879.57,  aic = 3781.14
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE    MAPE      MASE
## Training set -0.1553885 32.29358 25.28569 130.4682 270.227 0.9077494
##                    ACF1
## Training set 0.01867175
```

arma(1,10) p=1, q=10

```r
NO2.arma110 <- arima(e.ts.NO2, order=c(1,0,10), include.mean=FALSE)
summary(NO2.arma110)
```

```
##
## Call:
## arima(x = e.ts.NO2, order = c(1, 0, 10), include.mean = FALSE)
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4     ma5     ma6     ma7
##       0.3193  0.1730  0.1445  0.1787  0.0169  0.0171  0.0419  0.1716
## s.e.  0.1952  0.1867  0.1148  0.0889  0.0709  0.0593  0.0664  0.0504
##          ma8     ma9    ma10
##       0.1499  0.1052  0.1254
## s.e.  0.0598  0.0767  0.0792
##
## sigma^2 estimated as 1035:  log likelihood = -1878.26,  aic = 3780.52
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.1653649 32.17822 25.09854 128.7412 297.5734 0.9010309
##                     ACF1
## Training set 0.008108944
```

use the auto.arima function on e.ts.NO@

```
NO2.auto <- auto.arima(e.ts.NO2,approximation=FALSE)
summary(NO2.auto)
```

```
## Series: e.ts.NO2
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1
##       1.2966  -0.3601  -0.7884
## s.e.  0.1141   0.0837   0.0969
##
## sigma^2 estimated as 1092:  log likelihood=-1886.76
## AIC=3781.53   AICc=3781.63   BIC=3797.33
##
## Training set error measures:
##                      ME    RMSE      MAE      MPE     MAPE      MASE
## Training set -0.2025828 32.9152 25.89461 231.4739 324.1834 0.9296093
##                    ACF1
## Training set 0.01187437
```

Part D: Assessment of Models

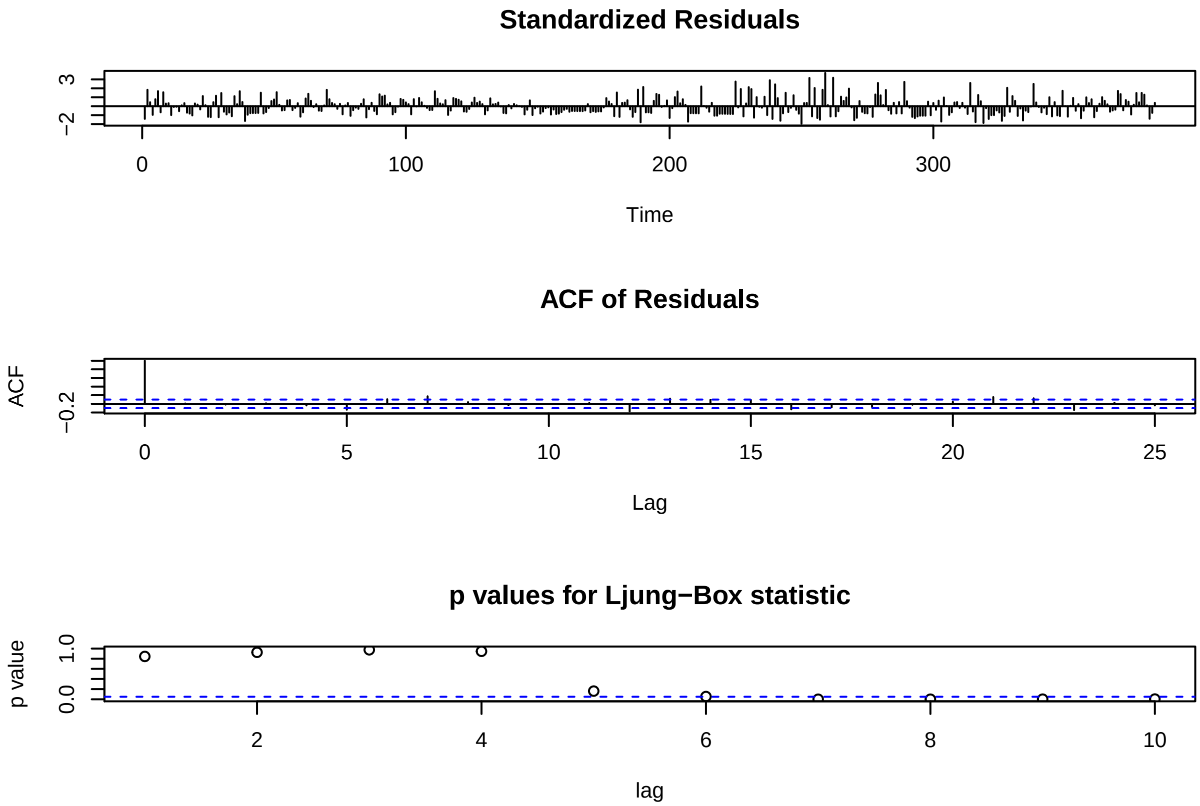
```
AIC(CO.ar1)
```

```
## [1] 1452.358
```

We used AIC and diagnostics to assess the models . . . .

Part E: Diagnostics

```
tsdiag(CO.ar1, lag = 30)
```


Standardized Residuals
3
−2
0
100
200
300
Time
ACF of Residuals
ACF
−0.2
0
5
10
15
20
25
Lag
p values for Ljung−Box statistic
p value
1.0
0.0
2
4
6
8
10
lag

Part 2: Building Multivariate Time Series Models

Part A: Seasonality

Part B: Trends

Part C: Auto-Regressive and Moving Average

Part D: Assessment of Models

Part E: Diagnostics

Part 3: Simulating from Univariate and Multivariate Time Series Models

Part A: Ability to reproduce appearance

Part B: Ability to reproduce observed trends

Part C: Ability to reproduce seasonality

Part D: Ability to reproduce observed mean and variance

Part E: Ability to reproduce auto-correlation

Part F: Ability to reproduce observed cross-correlation