Project 2

Canyu Lei, Ryan Gao, Conor Moore, and Shay Ladd

12/10/23

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=lis
t(formula=paste(names(AQdata)[c(1:2)],'~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)
```

#Building Univariate Time Series Models

a) How you discovered and modeled any seasonal components, if applicable. (5 points)

Seasonal components for CO and NO2 were discovered and modeled using sine and cosine functions within linear regression models. This suggests distinct seasonal behaviors in CO and NO2 concentrations, effectively captured by the frequencies chosen for the sine and cosine terms in the models.

b) How you discovered and modeled any trends, if applicable. (5 points)

Trends in CO and NO2 data were discovered and modeled using linear regression. These trends were modeled as part of the linear regression equations, which included both the time index (representing linear trend) and sine and cosine functions (capturing seasonal variations).

c) How you determined autoregressive and moving average components, if applicable. Compare at least two models. (5 points)

Shown as the 1c) part below

d) How you assessed your models (e.g. adjusted R2, AIC, diagnostics, etc.) to select one model for each pollutant. Assessments should discuss diagnostics and at least one metric. Show and discuss diagnostics of both the linear models of trends and seasonality, and the ARIMA models of the residuals. (15 points)

Shown as the 1d) part below

e) What problems, if any, remain in the diagnostics of the selected models. (5 points)

Shown as the 1e) part below

Part 1

```
# Create time series for CO and NO2
co.ts <- ts(dailyAQ$CO.GT.)
no2.ts <- ts(dailyAQ$NO2.GT.)

# Build univariate time series
time.index <- c(1:length(co.ts))
co.lm <- lm(co.ts[time.index] ~ time.index + sin(2*pi*time.index/400) +

                cos(2*pi*time.index/400))
no2.lm <- lm(no2.ts[time.index] ~ time.index + sin(2*pi*time.index/200)
 +
                cos(2*pi*time.index/200))

summary(co.lm)

##
## Call:
## lm(formula = co.ts[time.index] ~ time.index + sin(2 * pi * time.inde
x/400) +
##     cos(2 * pi * time.index/400))
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6606 -1.4324 -0.1678  1.2835  6.7503
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)

## (Intercept)                     5.222420   0.279942  18.655  < 2e-16 *
**
## time.index                     -0.004439   0.001351  -3.285  0.00111 *
*
## sin(2 * pi * time.index/400) -1.365226   0.213025  -6.409 4.27e-10 *
**
## cos(2 * pi * time.index/400)  0.024715   0.130034   0.190  0.84935

## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.79 on 387 degrees of freedom
## Multiple R-squared:  0.1168, Adjusted R-squared:  0.1099
## F-statistic: 17.06 on 3 and 387 DF,  p-value: 2.01e-10
```

```
summary(no2.lm)
```

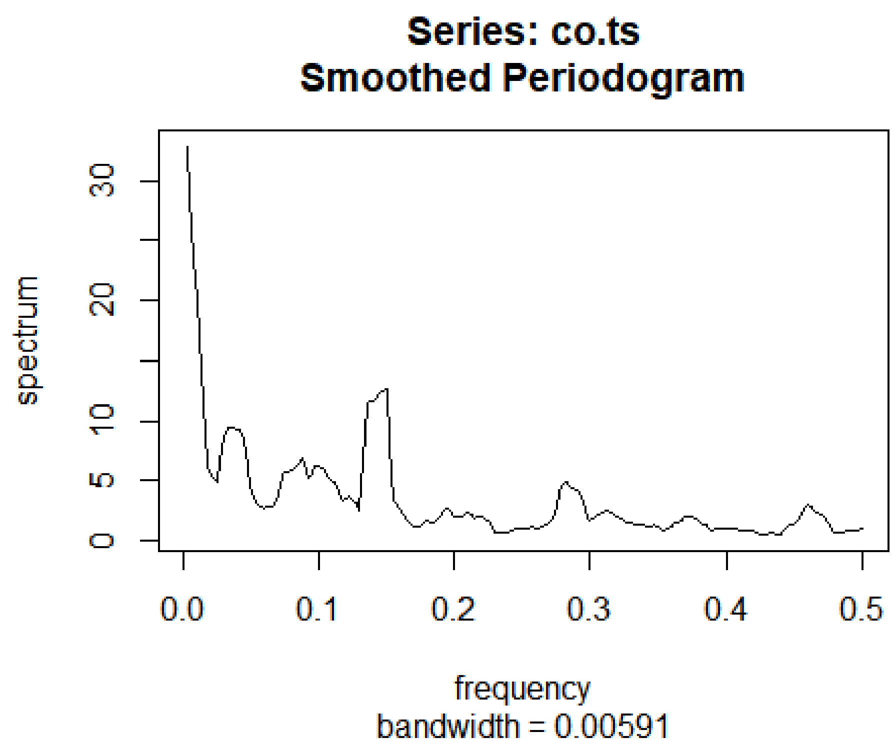
```
##
## Call:
## lm(formula = no2.ts[time.index] ~ time.index + sin(2 * pi * time.ind
ex/200) +
##     cos(2 * pi * time.index/200))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -87.794 -32.118  -1.004  29.638 134.827
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)

## (Intercept)                    118.61624    4.58043  25.896   <2e-16 *
**
## time.index                       0.23440    0.02066  11.343   <2e-16 *
**
## sin(2 * pi * time.index/200)  -3.60165    3.25858  -1.105   0.2697

## cos(2 * pi * time.index/200) -10.94162    3.06368  -3.571   0.0004 *
**
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.28 on 387 degrees of freedom
```

```
## Multiple R-squared:  0.3219, Adjusted R-squared:  0.3167
## F-statistic: 61.25 on 3 and 387 DF,  p-value: < 2.2e-16
```

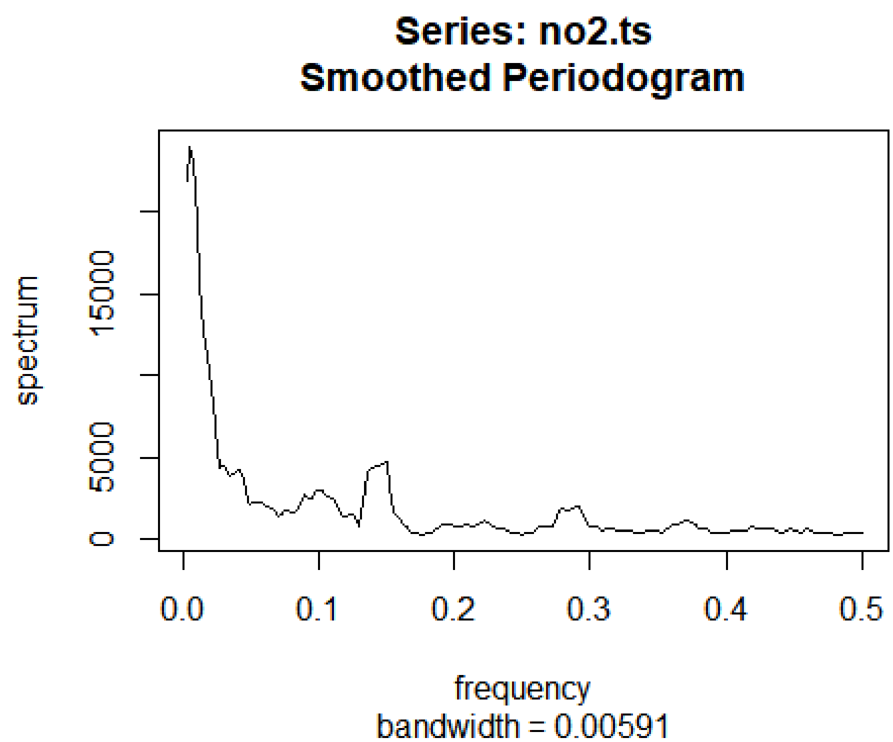
a) Seasonality

Get periods and peak of both co and no2 model, the find the seasonality.

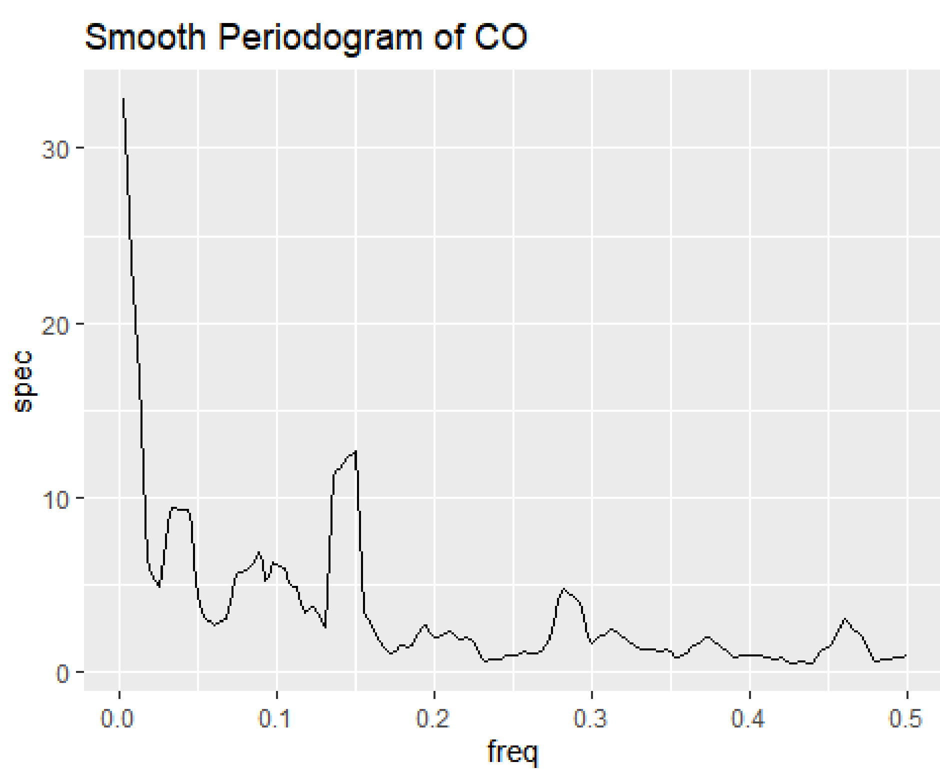
```
# Get the periodogram for co.ts and no2.ts
pg.co <- spec.pgram(co.ts,spans=9,demean=T,log='no')
```



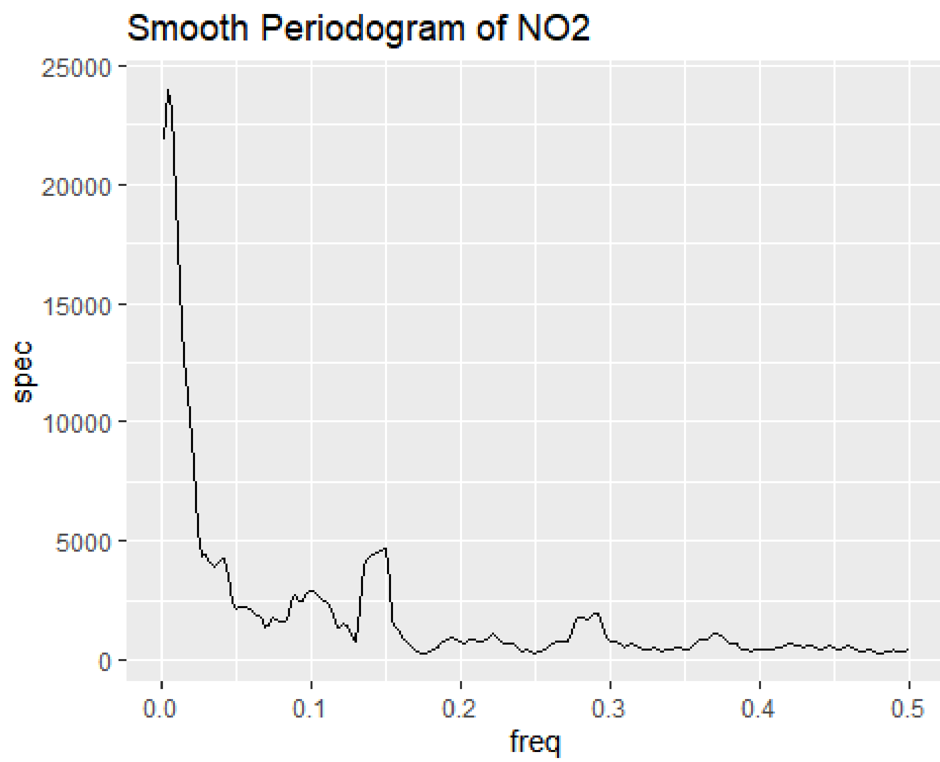

```
pg.no2 <- spec.pgram(no2.ts,spans=9,demean=T,log='no')
```




```
spec.co <- data.frame(freq=pg.co$freq, spec=pg.co$spec)
ggplot(spec.co) + geom_line(aes(x=freq,y=spec)) +
  ggtitle("Smooth Periodogram of CO")
```



```r
spec.no2 <- data.frame(freq=pg.no2$freq, spec=pg.no2$spec)
ggplot(spec.no2) + geom_line(aes(x=freq,y=spec)) +
  ggtitle("Smooth Periodogram of NO2")
```



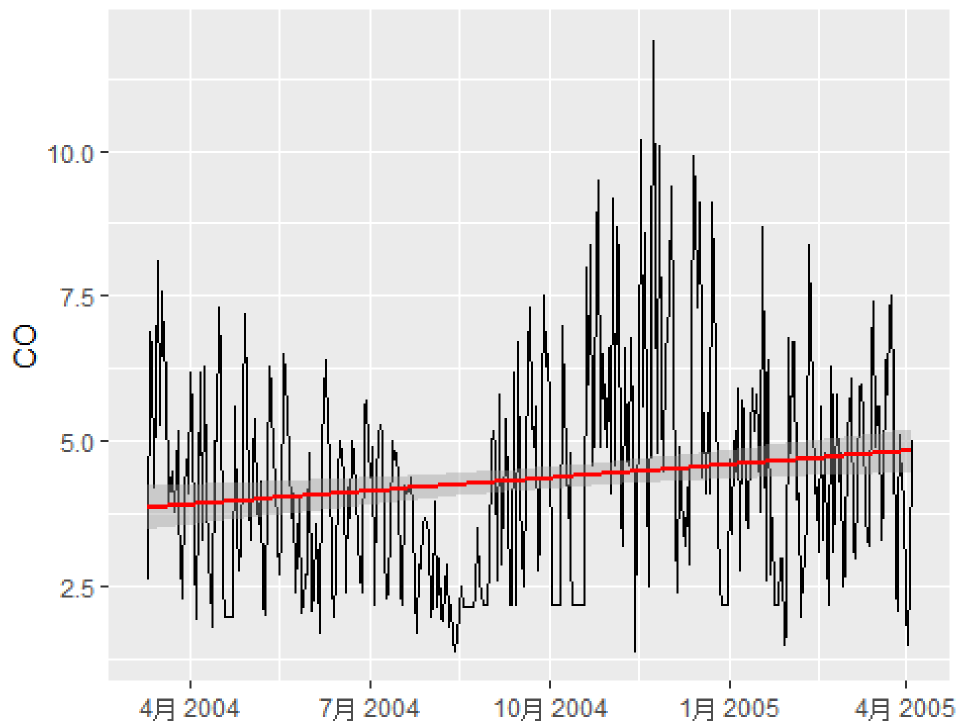

```r
# What are the periods of the next biggest peaks?
# sort spectrum from largest to smallest and find index
sorted.spec <- sort(pg.co$spec, decreasing=T, index.return=T)
names(sorted.spec)

## [1] "x"  "ix"

# corresponding periods (omegas = frequences, Ts = periods)
sorted.omegas <- pg.co$freq[sorted.spec$ix]
sorted.Ts <- 1/pg.co$freq[sorted.spec$ix]

# use next biggest peaks?
sorted.spec <- sort(pg.no2$spec, decreasing=T, index.return=T)
names(sorted.spec)

## [1] "x"  "ix"

# corresponding periods (omegas = frequences, Ts = periods)
sorted.omegas <- pg.no2$freq[sorted.spec$ix]
sorted.Ts <- 1/pg.no2$freq[sorted.spec$ix]
```

The peak is approximately 0.15 The period of seasonality is ~7 days (1 / 0.15) for both CO and NO2

1b) Trends

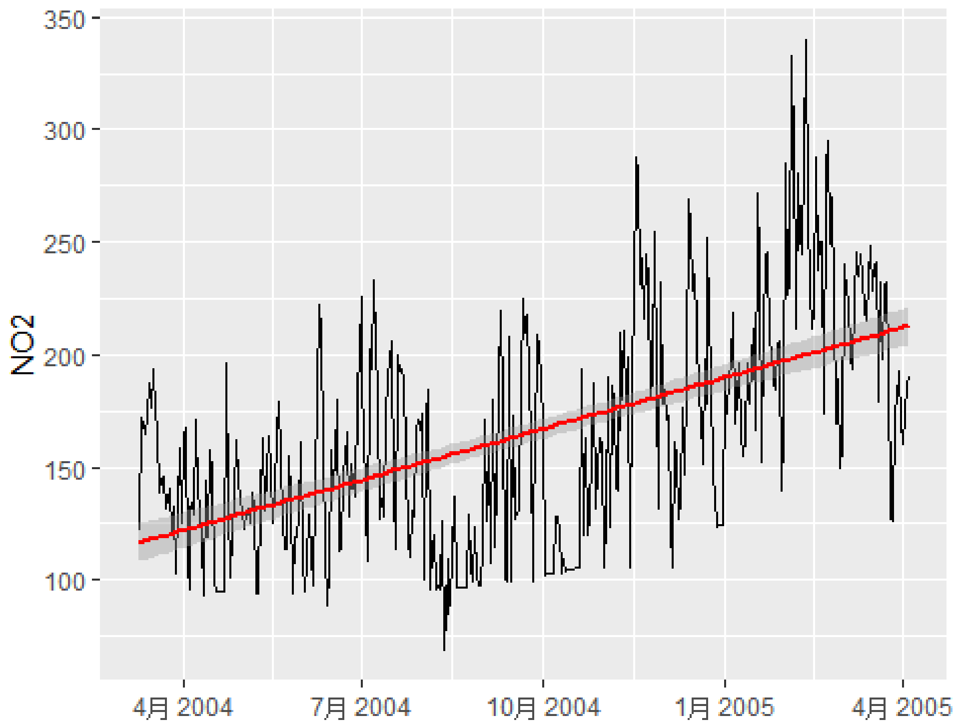
To model the trend, we use time as a predictor.

```
# co trend

co.trend<-lm(co.ts ~ time.index)
summary(co.trend)

##
## Call:
## lm(formula = co.ts ~ time.index)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3326 -1.6511 -0.0703  1.0938  7.3925
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.8595756  0.1903894  20.272  < 2e-16 ***
## time.index  0.0025015  0.0008418   2.972  0.00315 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.879 on 389 degrees of freedom
## Multiple R-squared:  0.0222, Adjusted R-squared:  0.01968
## F-statistic: 8.831 on 1 and 389 DF,  p-value: 0.003146

# time is significant in predicting

# Plot co.trend model

ggplot(dailyAQ, aes(x=Group.1,y=CO.GT.)) + geom_line() +
  stat_smooth(method="lm",col="red") + xlab("") + ylab("CO")

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




```r
# no2 trend

no2.trend<-lm(no2.ts ~ time.index)
summary(no2.trend)

##
## Call:
## lm(formula = no2.ts ~ time.index)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -85.674 -34.168   1.771  28.437 139.893
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 116.40204    4.34936   26.76   <2e-16 ***
## time.index    0.24692    0.01923   12.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.92 on 389 degrees of freedom
## Multiple R-squared:  0.2977, Adjusted R-squared:  0.2959
## F-statistic: 164.9 on 1 and 389 DF,  p-value: < 2.2e-16

# time is significant in predicting

# Plot no2.trend model
```

```r
ggplot(dailyAQ, aes(x=Group.1,y=NO2.GT.)) + geom_line() +
  stat_smooth(method="lm",col="red") + xlab("") + ylab("NO2")

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



For co, if we use a cut off at time 190, the trend of co will be better modeled. For no2, if we use a cutoff point at timestep 147, the trend of no2 will be better modeled.

```r
# add new variable to time series reflecting cutoff

# co
x_1 <- c(1:length(time.index))
for (i in 1:190) {
  x_1[i] <- 1
}
for (i in 191:391) {
  x_1[i] <- 0
}

co.trend.cutoff <- lm(co.ts ~ time.index + x_1 + time.index:x_1)
summary(co.trend.cutoff)

##
## Call:
## lm(formula = co.ts ~ time.index + x_1 + time.index:x_1)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3.6401 -1.2614 -0.2778  1.2164  6.8964 
## 
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      6.054129   0.642761   9.419   <2e-16 ***
## time.index      -0.004056   0.002166  -1.872   0.0619 .  
## x_1             -1.359050   0.693196  -1.961   0.0506 .  
## time.index:x_1 -0.005363   0.003201  -1.675   0.0947 .  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.782 on 387 degrees of freedom
## Multiple R-squared:  0.1249, Adjusted R-squared:  0.1181 
## F-statistic: 18.41 on 3 and 387 DF,  p-value: 3.469e-11
```

```r
AIC(co.trend.cutoff)
```

```
## [1] 1567.345
```

```r
# no2

x_2 <- c(1:length(no2.ts))
for (i in 1:147) {
  x_2[i] <- 1
}
for (i in 148:391) {
  x_2[i] <- 0
}

no2.trend.cutoff <- lm(no2.ts ~ time.index + x_1 + time.index:x_1)
summary(no2.trend.cutoff)
```

```
## 
## Call:
## lm(formula = no2.ts ~ time.index + x_1 + time.index:x_1)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -103.84  -32.23   -2.48   25.11  129.72 
## 
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     59.61378   14.72216   4.049 6.21e-05 ***
## time.index       0.44445    0.04961   8.958  < 2e-16 ***
## x_1             82.93795   15.87733   5.224 2.87e-07 ***
## time.index:x_1 -0.47902    0.07332  -6.533 2.03e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## Residual standard error: 40.81 on 387 degrees of freedom
## Multiple R-squared:  0.3681, Adjusted R-squared:  0.3632
## F-statistic: 75.16 on 3 and 387 DF,  p-value: < 2.2e-16
```

```r
AIC(no2.trend.cutoff)
```

```
## [1] 4016.05
```

After cut off, the model shows better AIC and adj r^2

Combine trend predictors and seasonality predictors for co and no2

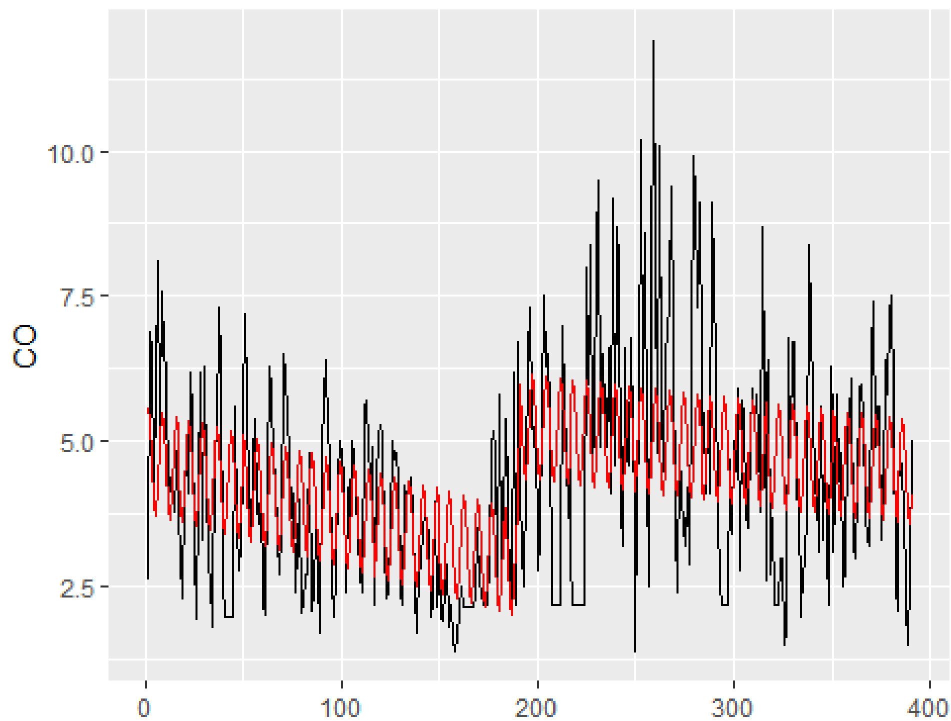
```r
# co trend.seasonality
co.trend.seasonal <- lm(co.ts ~ time.index + x_1 + time.index:x_1 + sin
(2*pi*time.index/7) + cos(2*pi*time.index/7))
summary(co.trend.seasonal)
```

```
##
## Call:
## lm(formula = co.ts ~ time.index + x_1 + time.index:x_1 + sin(2 *
##     pi * time.index/7) + cos(2 * pi * time.index/7))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8901 -0.9926 -0.0396  0.9700  6.4986
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 6.024139   0.599235  10.053  < 2e-16 ***
## time.index                 -0.003931   0.002019  -1.946 0.052323 .
## x_1                        -1.340679   0.646213  -2.075 0.038681 *
## sin(2 * pi * time.index/7)  0.833059   0.118669   7.020 1.01e-11 ***
## cos(2 * pi * time.index/7)  0.395380   0.118975   3.323 0.000975 ***
## time.index:x_1             -0.005416   0.002984  -1.815 0.070335 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.661 on 385 degrees of freedom
## Multiple R-squared:  0.2435, Adjusted R-squared:  0.2336
## F-statistic: 24.78 on 5 and 385 DF,  p-value: < 2.2e-16
```

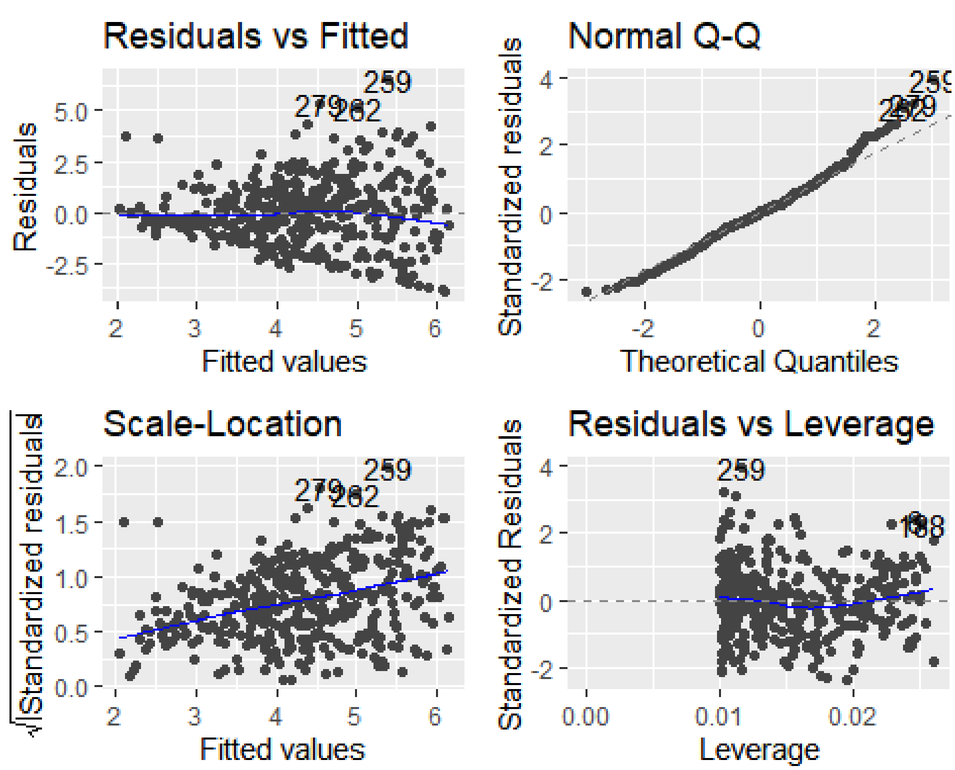
```r
AIC(co.trend.seasonal)
```

```
## [1] 1514.424
```

```r
# Plot co.trend.seasonal model
ggplot(dailyAQ, aes(x=time.index,y=CO.GT.)) + geom_line() +
  geom_line(aes(x=time.index,y=co.trend.seasonal$fitted.values),color="
red") +
  xlab("") + ylab("CO")
```




```r
# diagnostics
autoplot(co.trend.seasonal, labels.id = NULL)
```



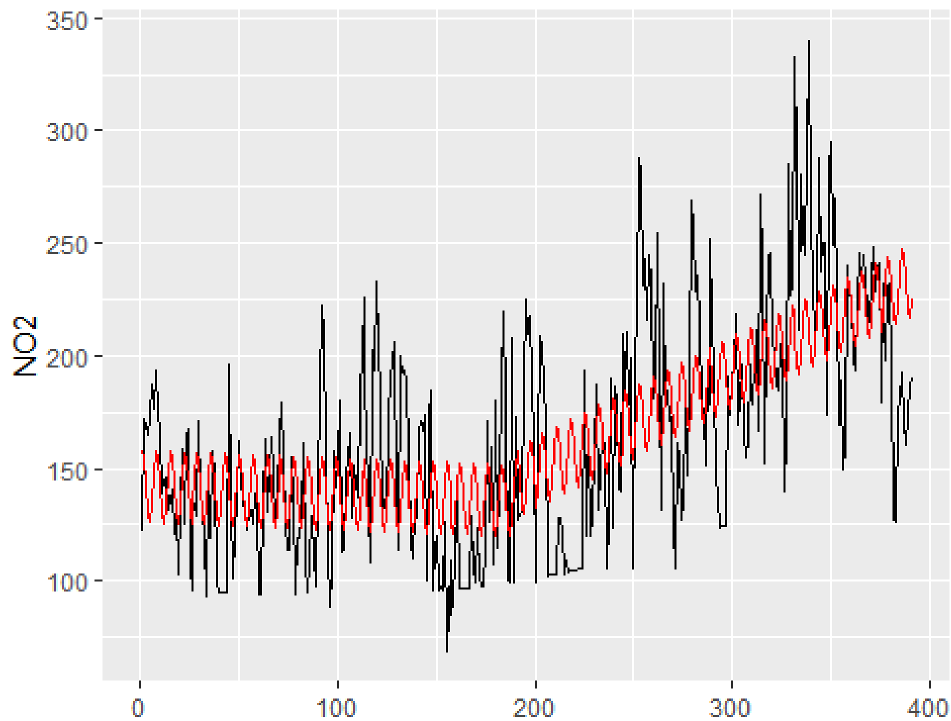
```r
# no2 trend seasonality
no2.trend.seasonal <- lm(no2.ts ~ time.index + x_1 + time.index:x_1 + s
in(2*pi*time.index/7) + cos(2*pi*time.index/7))
summary(no2.trend.seasonal)
```

```
##
## Call:
## lm(formula = no2.ts ~ time.index + x_1 + time.index:x_1 + sin(2 *
##     pi * time.index/7) + cos(2 * pi * time.index/7))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -89.859 -27.867  -2.021  20.546 128.056
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                59.00922   14.14690   4.171 3.75e-05 ***
## time.index                  0.44689    0.04768   9.373  < 2e-16 ***
## x_1                        83.29538   15.25598   5.460 8.56e-08 ***
## sin(2 * pi * time.index/7) 15.32820    2.80158   5.471 8.06e-08 ***
## cos(2 * pi * time.index/7)  5.78404    2.80878   2.059   0.0401 *
## time.index:x_1             -0.47974    0.07045  -6.809 3.78e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.22 on 385 degrees of freedom
## Multiple R-squared:  0.4197, Adjusted R-squared:  0.4121
## F-statistic: 55.68 on 5 and 385 DF,  p-value: < 2.2e-16
```

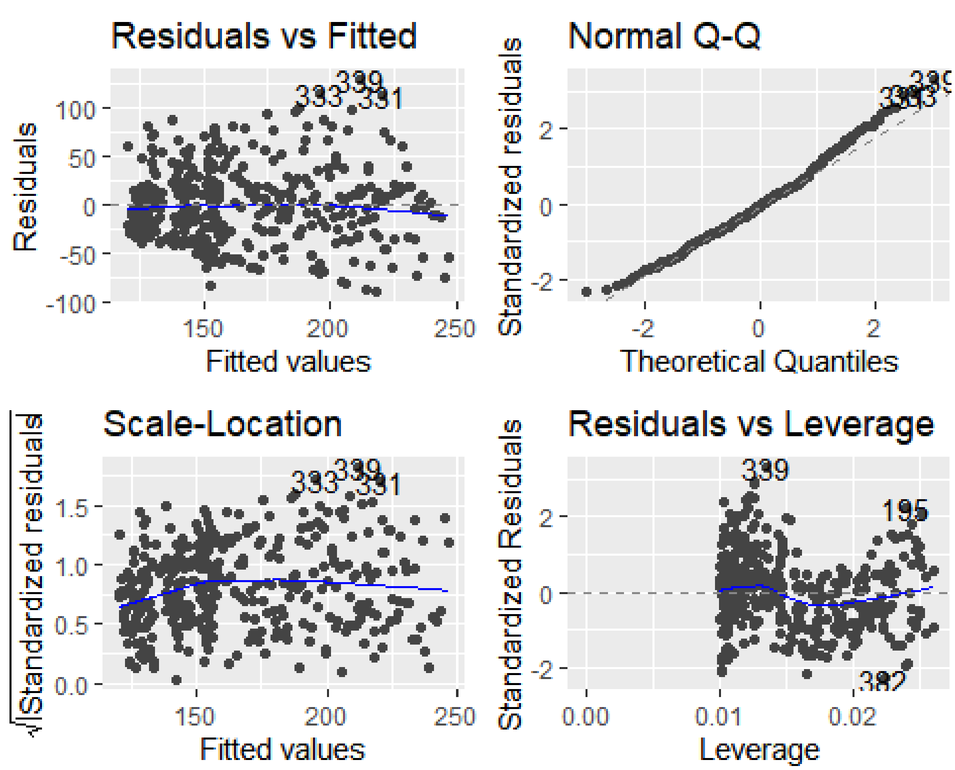
```r
AIC(no2.trend.seasonal)
```

```
## [1] 3986.793
```

```r
# Plot no2.trend.seasonal model
ggplot(dailyAQ, aes(x=time.index,y=NO2.GT.)) + geom_line() +
  geom_line(aes(x=time.index,y=no2.trend.seasonal$fitted.values),color=
"red") +
  xlab("") + ylab("NO2")
```



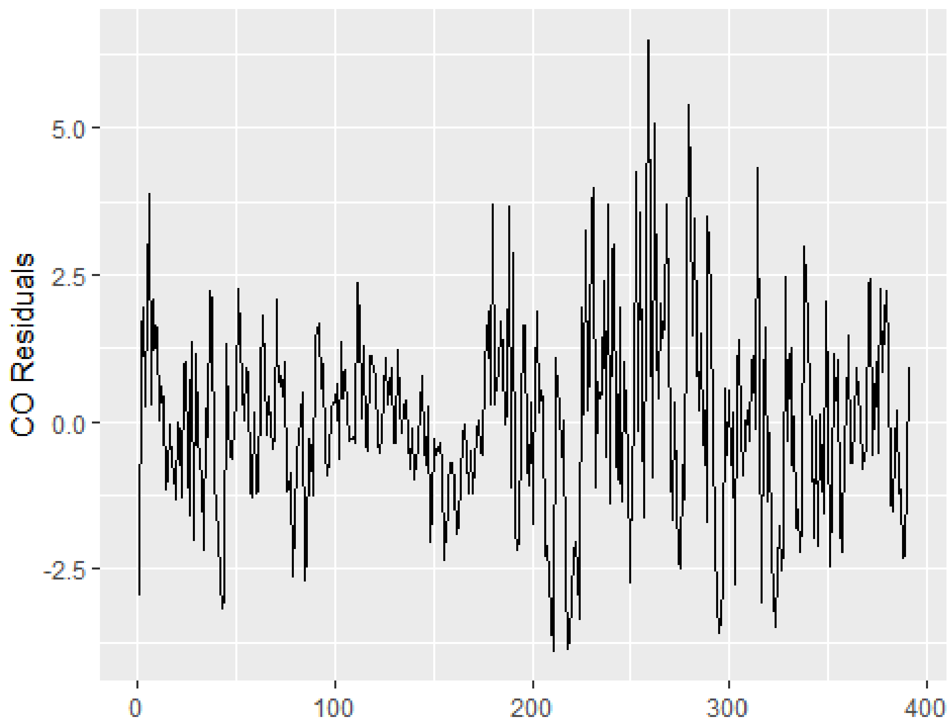

```r
# Model diagnostics for no2.trend.seasonal
autoplot(no2.trend.seasonal, labels.id = NULL)
```



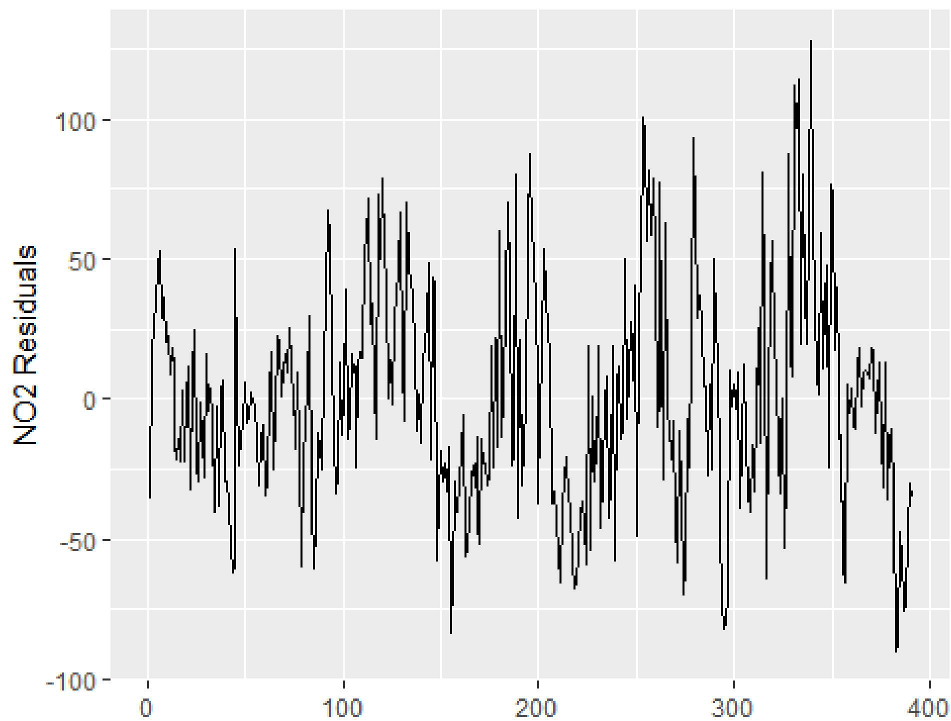
For both co and no2 trend.seasonal model, the diagnostic plot looks good, no pattern in the residuals vs. fitted and good qq(only slight upper tail and lower tail)

1c) AR and MA models

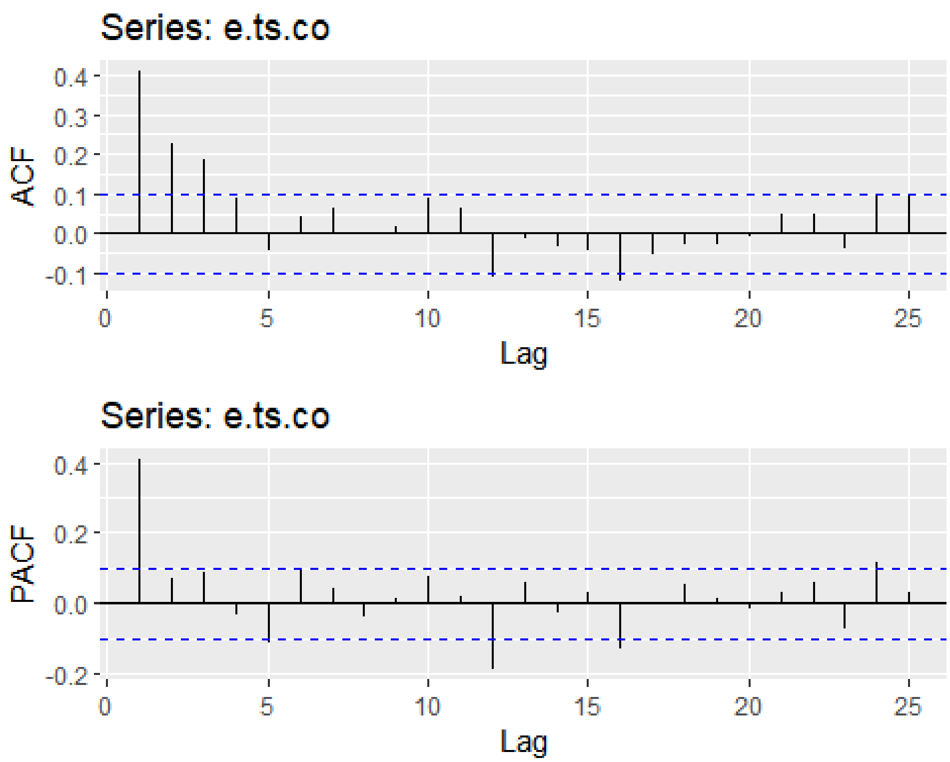
```
#AR MA ARIMA models
e.ts.co<-ts(co.trend.seasonal$residuals)
e.ts.no2<-ts(no2.trend.seasonal$residuals)

##Plot the residuals for the co.trend.seasonal model
autoplot(e.ts.co, ylab = "CO Residuals")
```



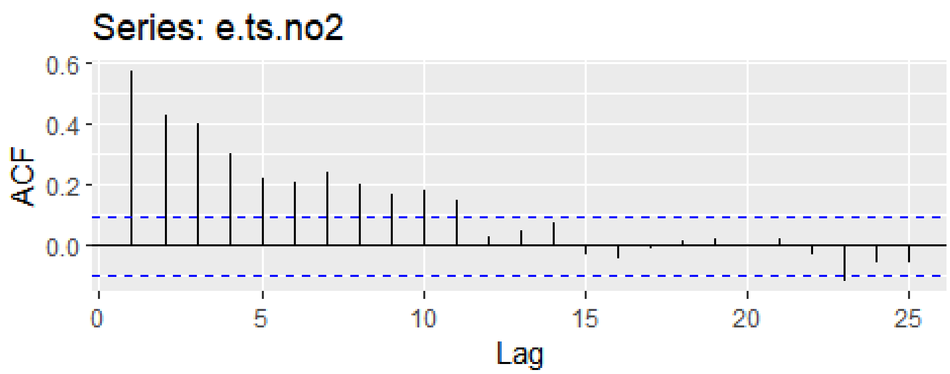

```
autoplot(e.ts.no2, ylab = "NO2 Residuals")
```

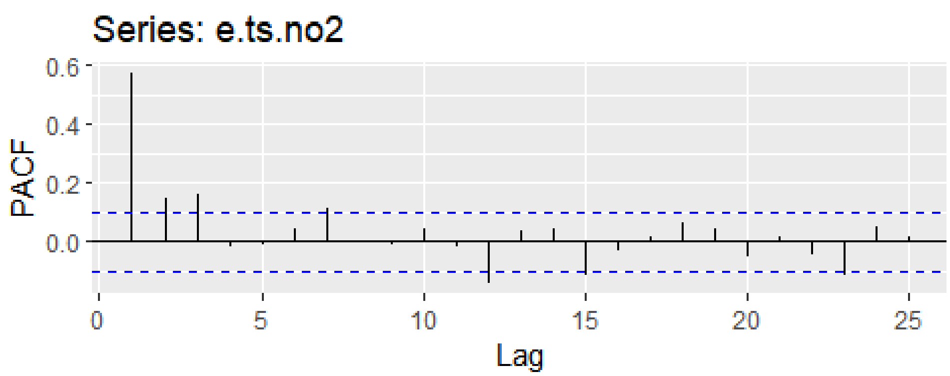



```
# plot ACF, PACF
co.acf <- ggAcf(e.ts.co)
co.pacf <- ggPacf(e.ts.co)
ggarrange(co.acf,co.pacf,nrow=2,ncol=1)
```




```
# from this plot choose p = 1, q = 3

# plot ACF PACF FOR NO2
no2.acf <- ggAcf(e.ts.no2)
no2.pacf <- ggPacf(e.ts.no2)
ggarrange(no2.acf,no2.pacf,nrow=2,ncol=1)
```

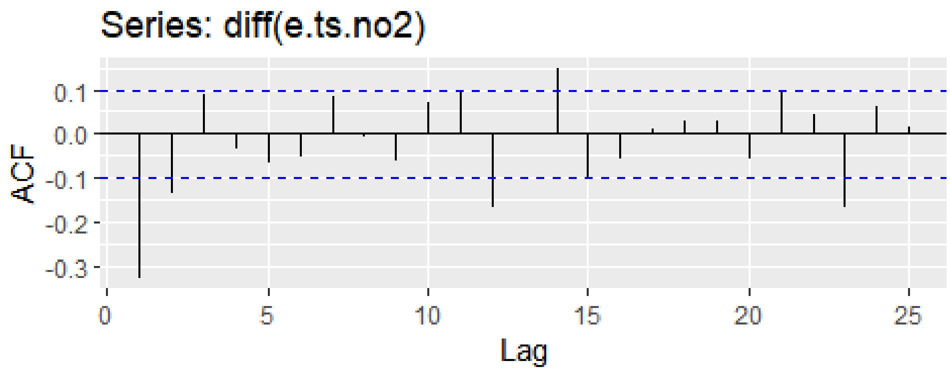


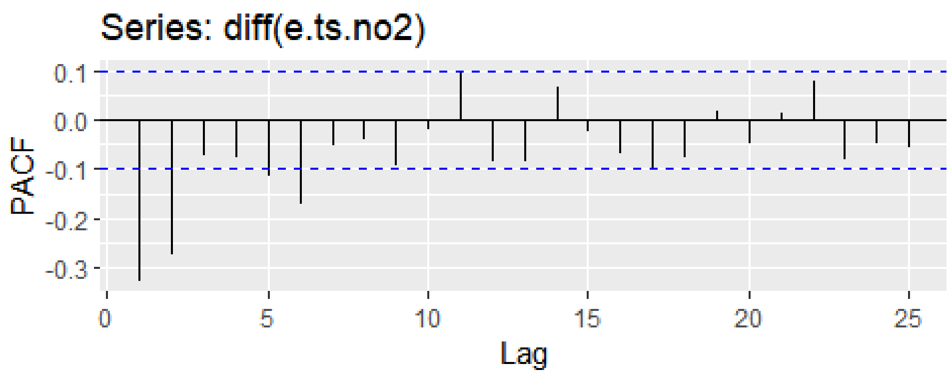


1.For co, based on acf and pacf plots. For MA model, we choose q = 3 For AR model, we choose p = 1 For ARMA model, we choose q = 3, p = 1.

2.For no2, based on acf and pacf plots. the acf plot shows slow, liner decay, indicating that the time series is nonstationary. we need to take first difference

```
# Check first order difference
no2.diff.acf <- ggAcf(diff(e.ts.no2))
no2.diff.pacf <- ggPacf(diff(e.ts.no2))
ggarrange(no2.diff.acf,no2.diff.pacf,nrow=2,ncol=1)
```





For no2, After take first difference, The ACF and PACF plots both appear to have sinusoidal decay, we choose p = 2, q = 2, d = 1. ARIMA(2,1,2) model.

```
# build the model
# co
# 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
##       0.4119
## s.e.  0.0462
##
## sigma^2 estimated as 2.258:  log likelihood = -714.12,  aic = 1432.2
3
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE       MA
SE
## Training set 0.001641399 1.502621 1.152933 125.7687 180.3853 0.88191
82
```

```
##                       ACF1
## Training set -0.03084152
```

```
AIC(co.ar1)
```

```
## [1] 1432.235
```

```
# 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
##       0.3499  0.1854  0.1638
## s.e.  0.0501  0.0494  0.0536
##
## sigma^2 estimated as 2.244:  log likelihood = -712.95,  aic = 1433.9
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE       MA
SE
## Training set 0.001926119 1.498103 1.155294 121.5587 186.7825 0.88372
46
##                      ACF1
## Training set 0.01865256
```

```
AIC(co.ma3)
```

```
## [1] 1433.898
```

```
# 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
##       0.3857  -0.0151  0.0598  0.1153
## s.e.  0.2011   0.1972  0.0905  0.0638
##
## sigma^2 estimated as 2.226:  log likelihood = -711.33,  aic = 1432.6
7
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE       MA
```

```
SE
## Training set 0.002054205 1.491879 1.145394 118.0271 185.3006 0.87615
16
##                       ACF1
## Training set 0.00210354

AIC(co.arma13)

## [1] 1432.667

# no2
# we need to use first difference model
no2.diff <- diff(e.ts.no2)

# ar(2) p = 2
no2.ar2 <- arima(no2.diff, order=c(2,0,0), include.mean=FALSE)
summary(no2.ar2)

##
## Call:
## arima(x = no2.diff, order = c(2, 0, 0), include.mean = FALSE)
##
## Coefficients:
##           ar1      ar2
##       -0.4221  -0.2778
## s.e.   0.0488   0.0487
##
## sigma^2 estimated as 1059:  log likelihood = -1911.65,  aic = 3829.3
1
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MA
SE
## Training set -0.02780589 32.53714 25.23126 352.3704 420.1196 0.57840
21
##                        ACF1
## Training set -0.01907731

# ma(2) p=0, q=2
no2.ma2 <- arima(no2.diff, order=c(0,0,2), include.mean=FALSE)
summary(no2.ma2)

##
## Call:
## arima(x = no2.diff, order = c(0, 0, 2), include.mean = FALSE)
##
## Coefficients:
##           ma1      ma2
##       -0.4994  -0.2063
## s.e.   0.0503   0.0551
##
```

```
## sigma^2 estimated as 1013:  log likelihood = -1903.14,  aic = 3812.2
8
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.2898495 31.8216 24.48985 319.5645 408.1458 0.5614058
##                     ACF1
## Training set 0.007123302
```

```
# arma(2,1,2)
no2.arma22 <- arima(no2.diff, order=c(2,1,2), include.mean=FALSE)
summary(no2.arma22)
```

```
##
## Call:
## arima(x = no2.diff, order = c(2, 1, 2), include.mean = FALSE)
##
## Coefficients:
##          ar1     ar2      ma1     ma2
##       0.3411  0.0336  -1.8432  0.8433
## s.e.  0.0784  0.0654   0.0616  0.0613
##
## sigma^2 estimated as 1010:  log likelihood = -1902.14,  aic = 3814.2
7
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MAS
E
## Training set -0.6769693 31.74191 24.43748 246.8975 342.2234 0.560205
4
##                       ACF1
## Training set -0.0003262038
```

based on best AIC, generate the auto model

```
co.auto <- auto.arima(e.ts.co,approximation=FALSE)
summary(co.auto) #(1, 0, 1)
```

```
## Series: e.ts.co
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##          ar1      ma1
##       0.5878  -0.2158
## s.e.  0.0999   0.1222
##
## sigma^2 = 2.255:  log likelihood = -712.83
## AIC=1431.66   AICc=1431.72   BIC=1443.57
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE       MA
```

```
SE
## Training set 0.001734336 1.497658 1.150629 121.6782 183.4999 0.88015
65
##                      ACF1
## Training set 0.00260201
```

```
no2.auto <- auto.arima(e.ts.no2,approximation=FALSE)
summary(no2.auto) #(1, 0, 2)
```

```
## Series: e.ts.no2
## ARIMA(1,0,2) with zero mean
##
## Coefficients:
##          ar1      ma1      ma2
##       0.8679  -0.4010  -0.1135
## s.e.  0.0507   0.0731   0.0688
##
## sigma^2 = 977.1:  log likelihood = -1899.5
## AIC=3807   AICc=3807.11   BIC=3822.88
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE       MAS
E
## Training set -0.1183307 31.13892 23.87463 80.62341 179.7894 0.881708
8
##                       ACF1
## Training set -0.004175665
```

#3 1d) assessment

```
# BIC

# co
BIC(co.ar1) # 1440

## [1] 1440.172

BIC(co.ma3) # 1449

## [1] 1449.773

BIC(co.arma13) # 1452

## [1] 1452.51

BIC(co.auto) # 1443

## [1] 1443.568

# no2
BIC(no2.ar2) # 3841

## [1] 3841.205
```

```
BIC(no2.ma2) # 3824

## [1] 3824.175

BIC(no2.arma22) # 3834

## [1] 3834.089

BIC(no2.auto) # 3822

## [1] 3822.877
```

Check the BIC for all models of co and no2

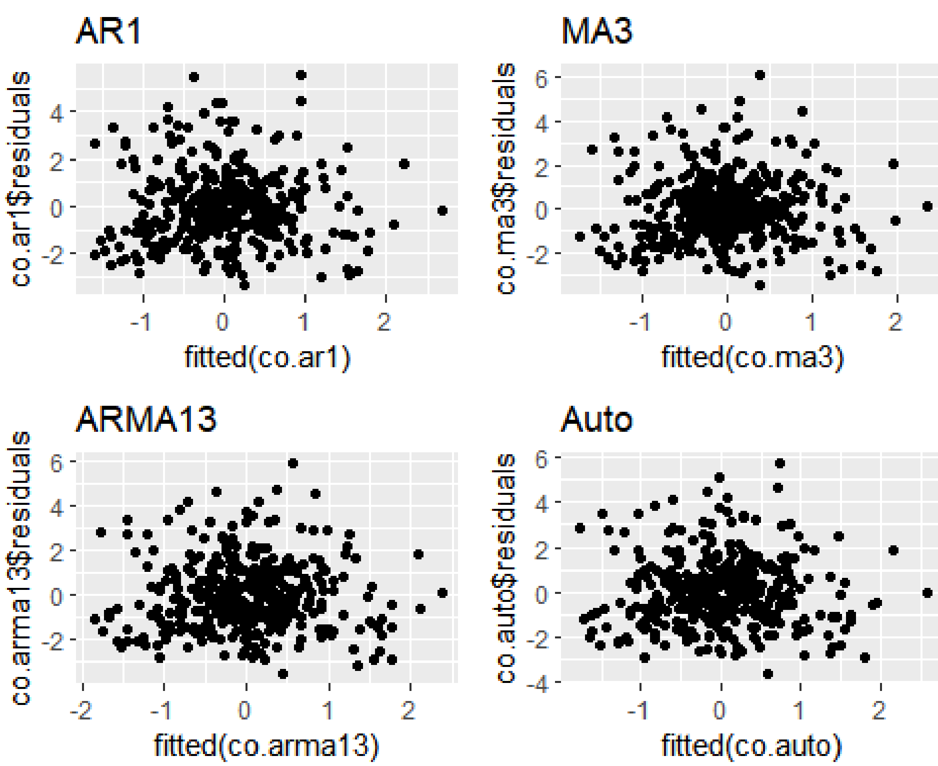
For co, the ar(1) model shows best BIC, but it;s very close to co.auto

For no2, the no2.auto model shows best BIC.

1d: Diagnostic plots

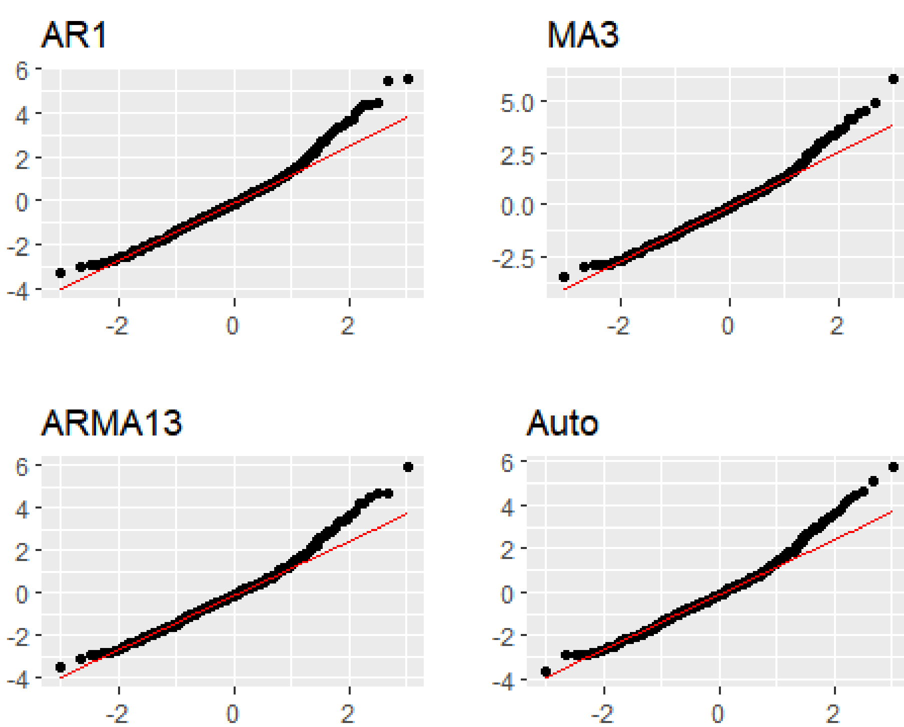
```
# co

# assess residuals vs. fitted
model1 = ggplot() + geom_point(aes(x=fitted(co.ar1), y=co.ar1$residual
s)) + ggtitle("AR1")
model2 = ggplot() + geom_point(aes(x=fitted(co.ma3), y=co.ma3$residual
s)) + ggtitle("MA3")
model3 = ggplot() + geom_point(aes(x=fitted(co.arma13), y=co.arma13$res
iduals)) + ggtitle("ARMA13")
model4 = ggplot() + geom_point(aes(x=fitted(co.auto), y=co.auto$residua
ls)) + ggtitle("Auto")

ggarrange(model1, model2, model3, model4, ncol=2, nrow=2)

## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
```

```
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
```

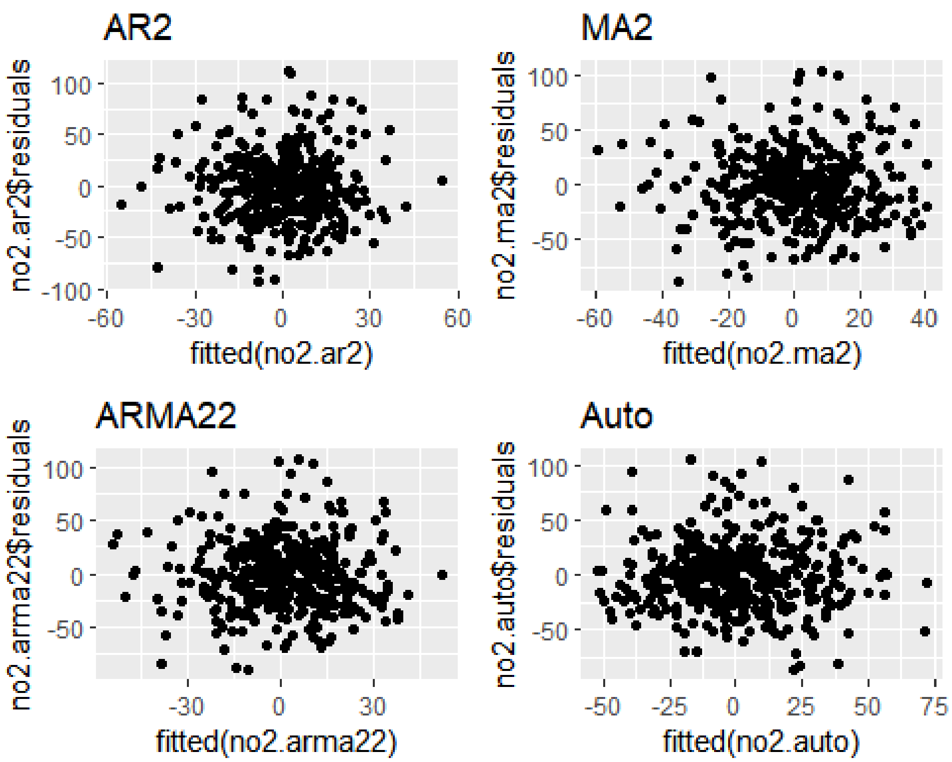



```
# qq

# assess normality of residuals
model1 = qplot(sample=co.ar1$residuals) + stat_qq_line(color="red") + g
gtitle("AR1")

## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warnin
g was
## generated.

model2 = qplot(sample=co.ma3$residuals) + stat_qq_line(color="red") + g
gtitle("MA3")
model3 = qplot(sample=co.arma13$residuals) + stat_qq_line(color="red")
+ ggtitle("ARMA13")
model4 = qplot(sample=co.auto$residuals) + stat_qq_line(color="red") +
ggtitle("Auto")
```

```r
ggarrange(model1, model2, model3, model4, ncol=2, nrow=2)

## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
```

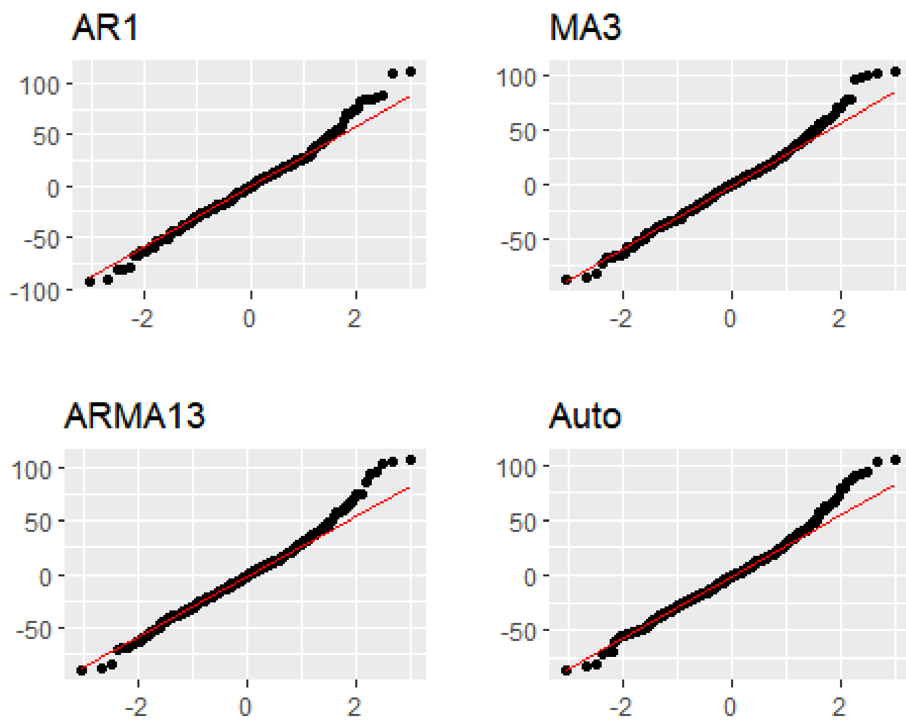


```
# no2

# assess residuals vs. fitted
model1 = ggplot() + geom_point(aes(x=fitted(no2.ar2), y=no2.ar2$residua
ls)) + ggtitle("AR2")
model2 = ggplot() + geom_point(aes(x=fitted(no2.ma2), y=no2.ma2$residua
ls)) + ggtitle("MA2")
model3 = ggplot() + geom_point(aes(x=fitted(no2.arma22), y=no2.arma22$r
esiduals)) + ggtitle("ARMA22")
model4 = ggplot() + geom_point(aes(x=fitted(no2.auto), y=no2.auto$resid
uals)) + ggtitle("Auto")

ggarrange(model1, model2, model3, model4, ncol=2, nrow=2)

## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
```

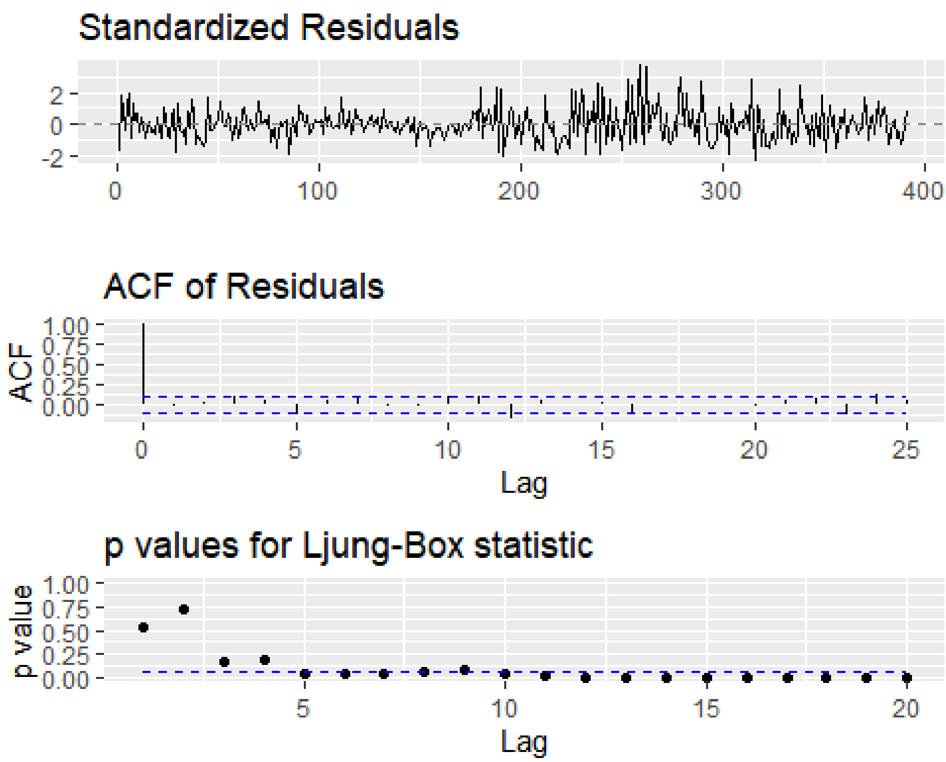



```
# assess normality of residuals
model1 = qplot(sample=no2.ar2$residuals) + stat_qq_line(color="red") +
ggtitle("AR1")
model2 = qplot(sample=no2.ma2$residuals) + stat_qq_line(color="red") +
ggtitle("MA3")
model3 = qplot(sample=no2.arma22$residuals) + stat_qq_line(color="red")
 + ggtitle("ARMA13")
model4 = qplot(sample=no2.auto$residuals) + stat_qq_line(color="red") +
 ggtitle("Auto")

ggarrange(model1, model2, model3, model4, ncol=2, nrow=2)

## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>.
Defaulting
## to continuous.
```

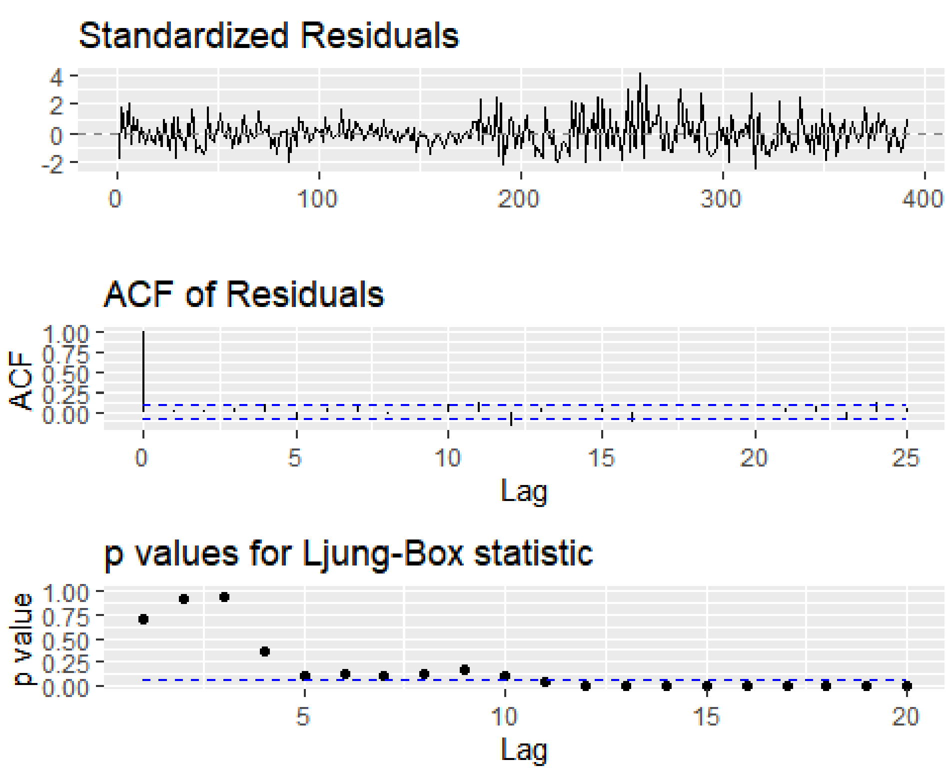
```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



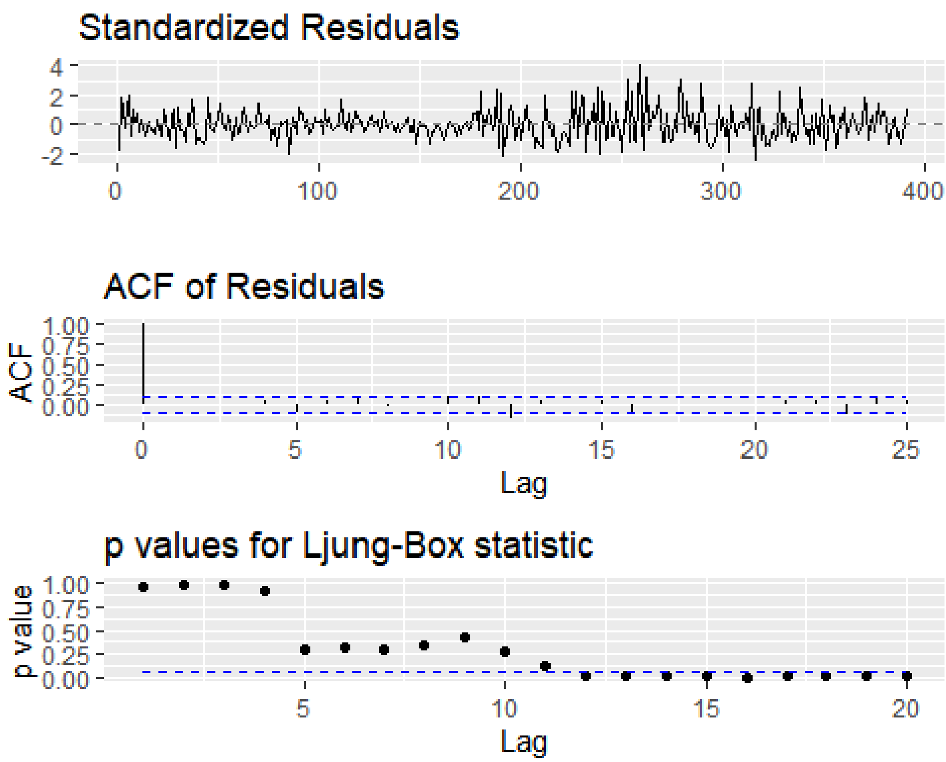

```
# Ljung Box

# co
# Ljung Box
ggtsdiag(co.ar1,gof.lag=20)
```



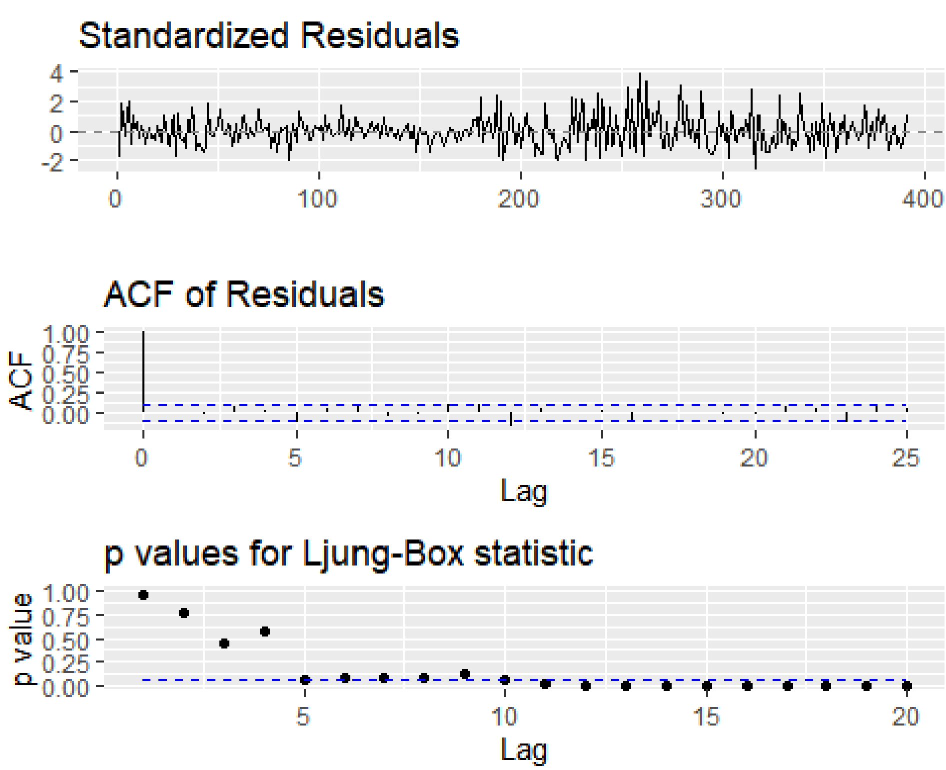

```
ggtsdiag(co.ma3,gof.lag=20)
```



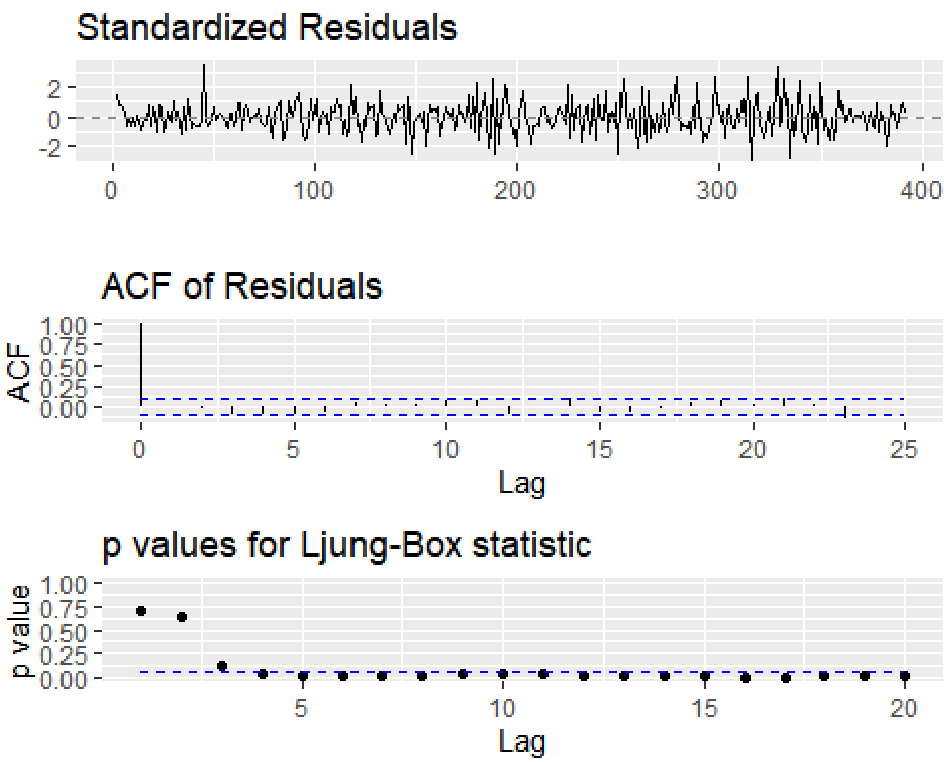

```
ggtsdiag(co.arma13,gof.lag=20)
```



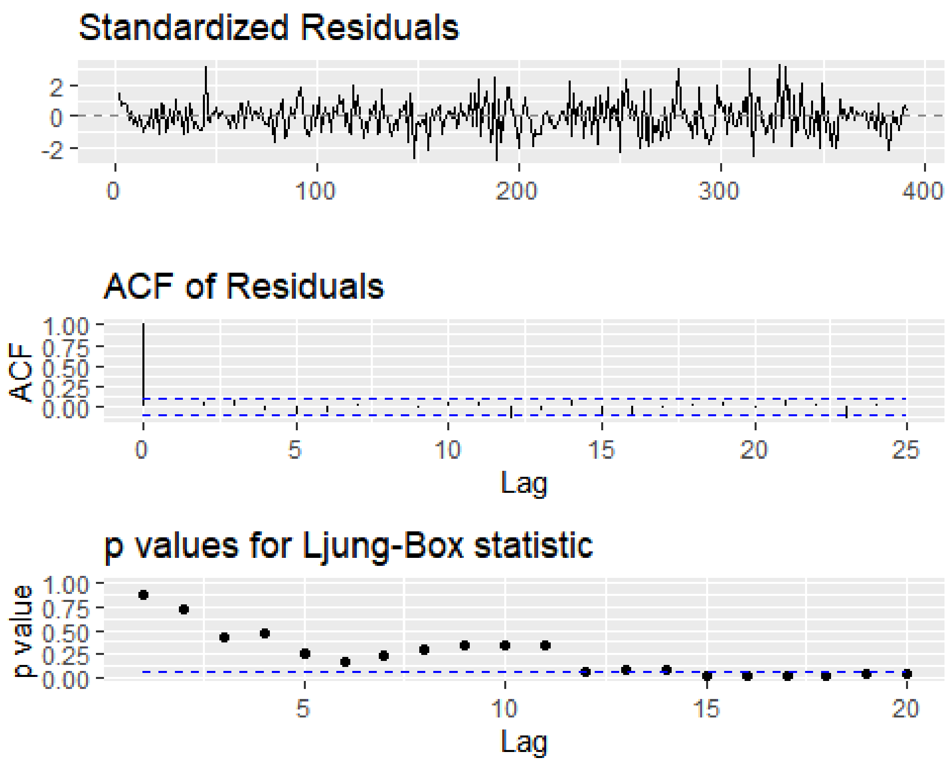

```r
ggtsdiag(co.auto,gof.lag=20)
```



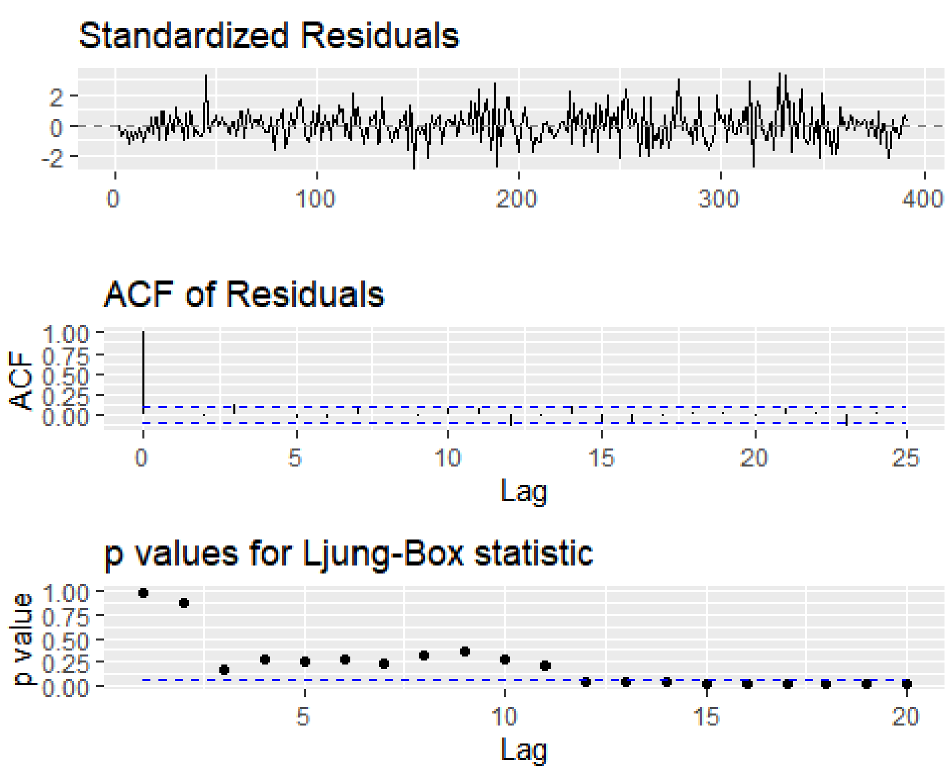
```r
# no2
ggtsdiag(no2.ar2,gof.lag=20)
```



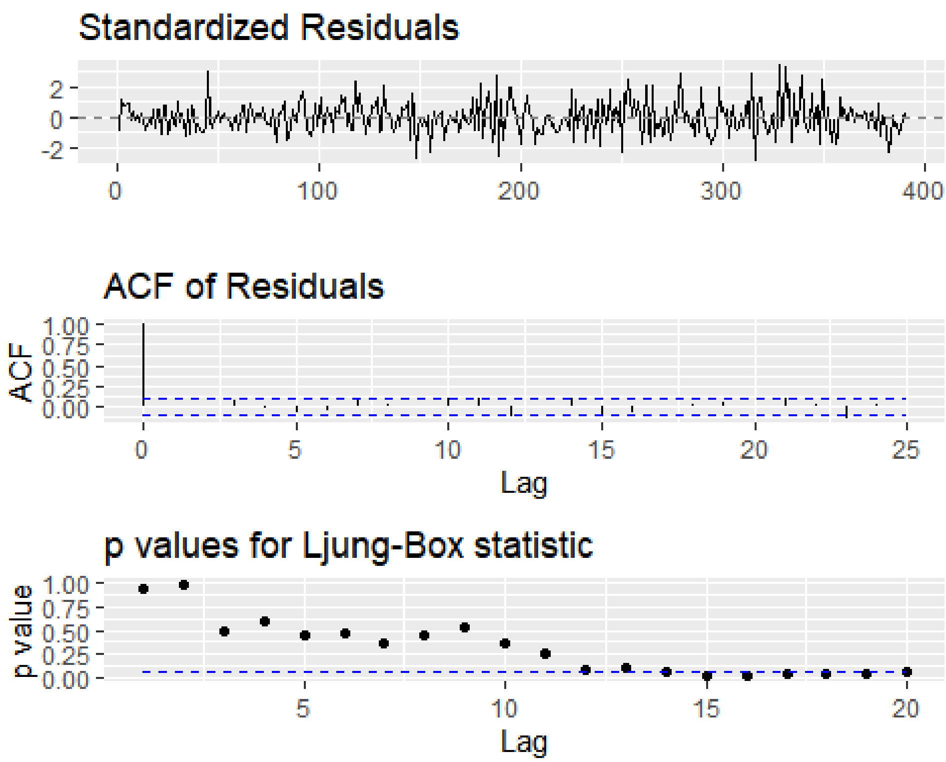

```r
ggtsdiag(no2.ma2,gof.lag=20)
```




```r
ggtsdiag(no2.arma22,gof.lag=20)
```




```r
ggtsdiag(no2.auto,gof.lag=20)
```

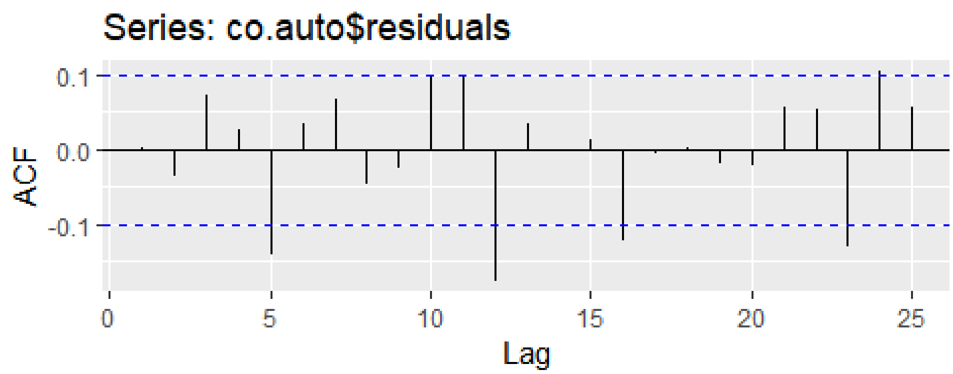


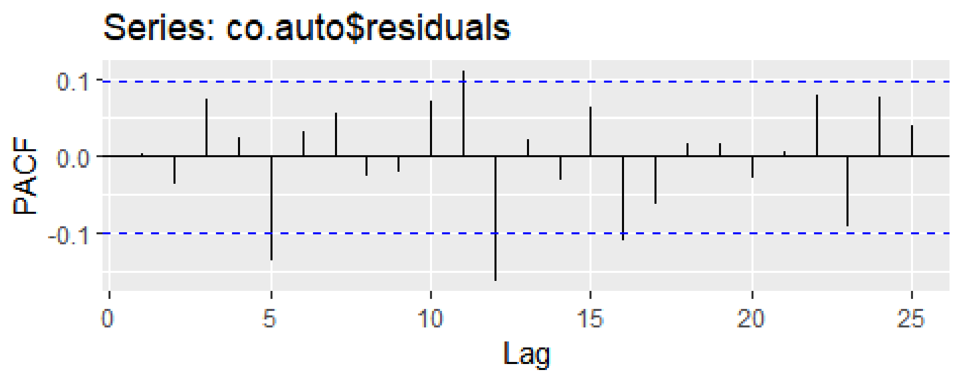
Based on AIC BIC, diagnose plots and Ljung Box plots above

For co, we choose auto ARIMA(1,0,1) as the best model, because it has best AIC and good BIC, and shows good results in diagnose plots and Ljung Box plots

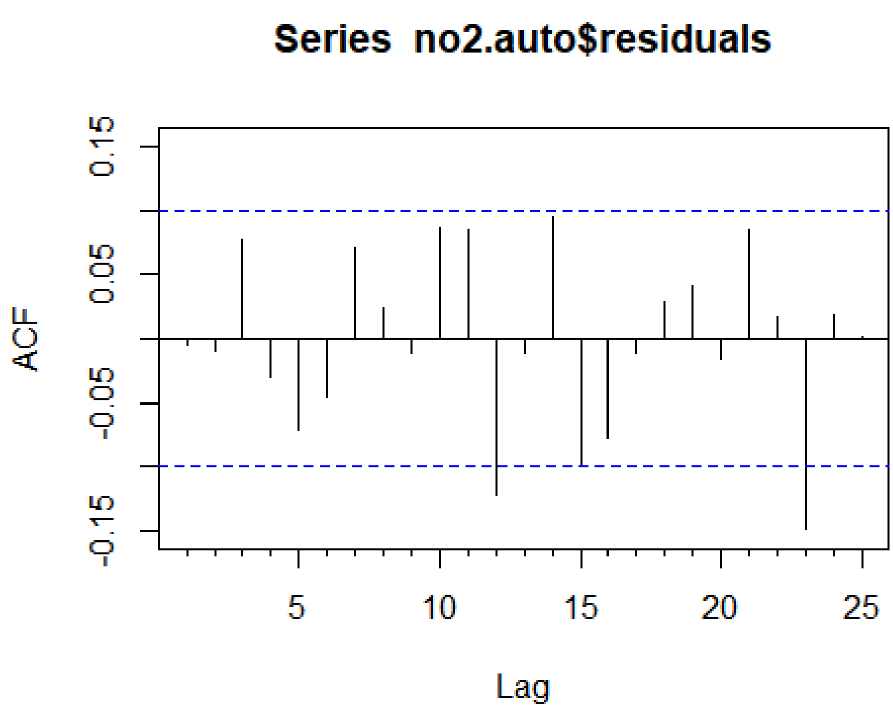
For no2, we choose auto ARIMA(1,0,2) as the best model,because it has best AIC and BIC, and shows good results in diagnose plots and Ljung Box plots

```r
# Best model

# co arima(1,0,2)

co_P1 <- ggAcf(co.auto$residuals)
co_P2 <- ggPacf(co.auto$residuals)
ggarrange(co_P1,co_P2,nrow=2,ncol=1)
```

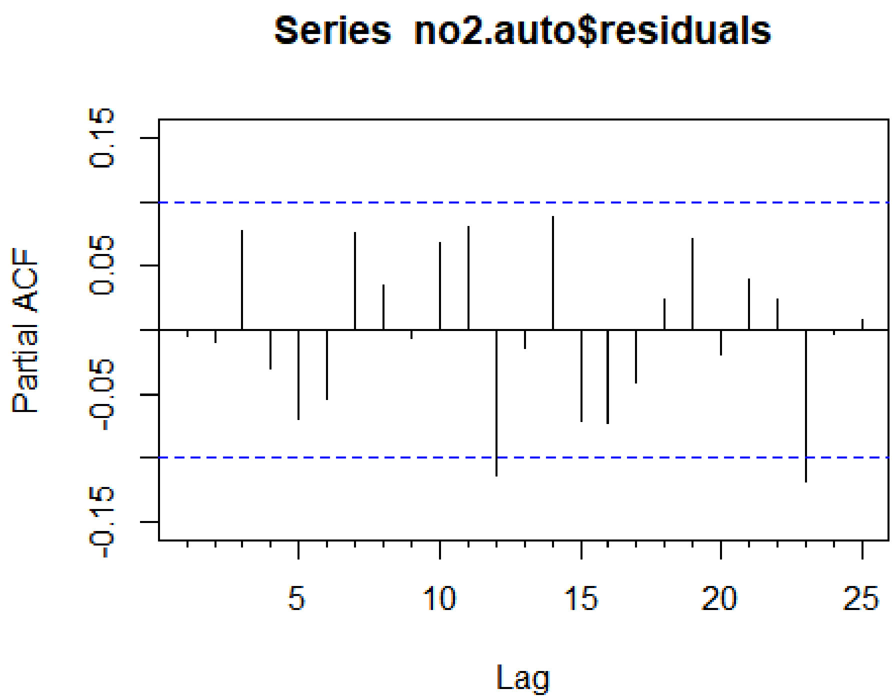





```
# no2 arima(1,0,2)

no2_P1 <- Acf(no2.auto$residuals)
```



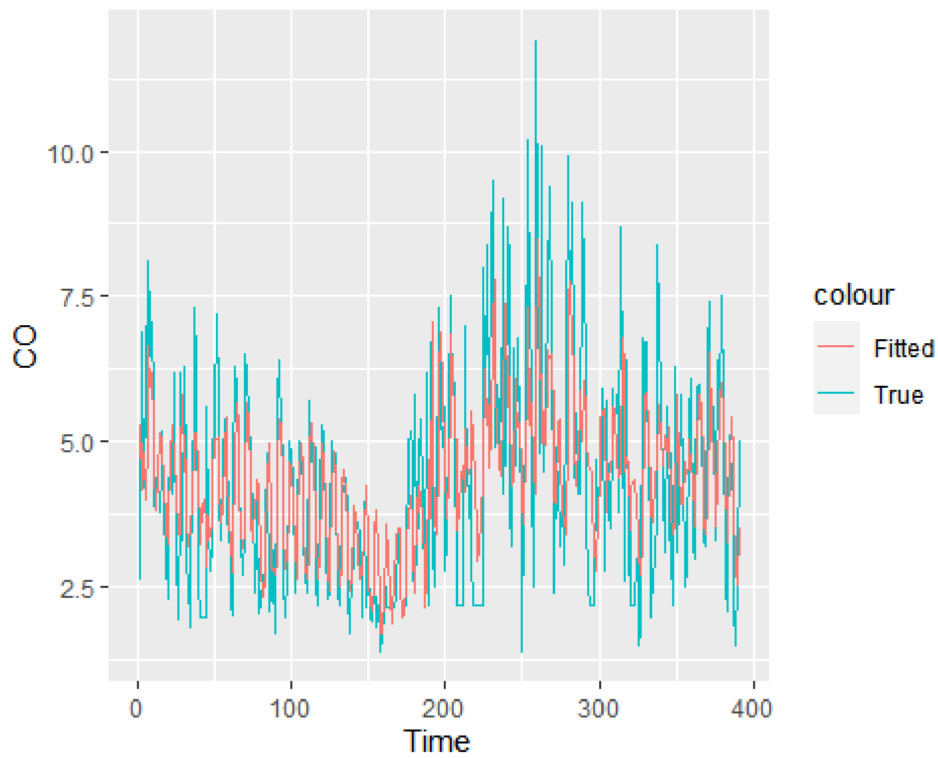
```r
no2_P2 <- Pacf(no2.auto$residuals)
```



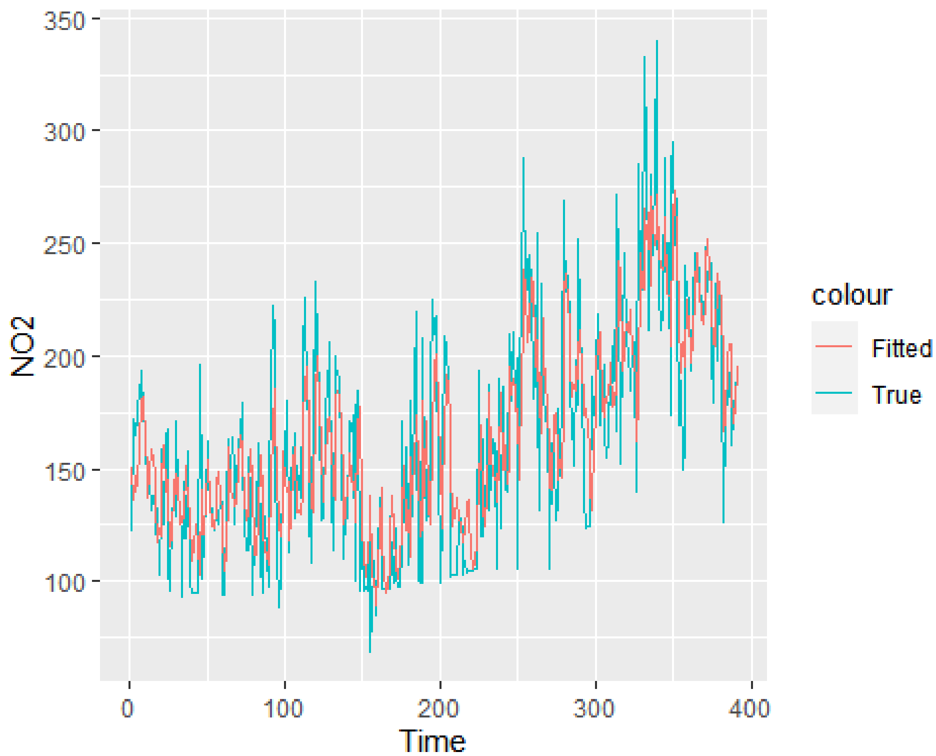

```r
ggarrange(no2_P1,no2_P2,nrow=2,ncol=1)

## Warning in as_grob.default(plot): Cannot convert object of class acf
 into a
## grob.

## Warning in as_grob.default(plot): Cannot convert object of class acf
 into a
## grob.
```

Plot the fitted values vs. true values of best model for co and no2

```
# co
co.fit <- co.trend.seasonal$fitted.values + fitted(co.auto)

ggplot() + geom_line(aes(x=time.index,y=co.ts[1:length(time.index)],col
or="True")) +
  geom_line(aes(x=time.index,y=co.fit,color="Fitted")) + xlab("Time") +

  ylab("CO")
```




```
# no2

no2.fit <- no2.trend.seasonal$fitted.values + fitted(no2.auto)

ggplot() + geom_line(aes(x=time.index,y=no2.ts[1:length(time.index)],co
lor="True")) +
  geom_line(aes(x=time.index,y=no2.fit,color="Fitted")) + xlab("Time")
+
  ylab("NO2")
```



Both plots shows good results.

1e)

For both co and no2, the qq plots show little problem only(slight upper tail and lower tail); the Ljung Box plots shows some significant points, which means the model can't reflect auto-correlation perfectly.

2: Multivariate Time Series Models

2a) & 2b)

For 2a and 2b, we used the same models and approach as in part 1.

2c)

```
e.co.lm <- auto.arima(co.lm$residuals,approximation=FALSE)
e.no2.lm <- auto.arima(no2.lm$residuals,approximation=FALSE)

summary(e.co.lm) # ARIMA(1,0,1)

## Series: co.lm$residuals
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
```

```
##       0.4513
## s.e.  0.0452
##
## sigma^2 = 2.533:  log likelihood = -736.15
## AIC=1476.3   AICc=1476.33   BIC=1484.24
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE       MA
SE
## Training set 0.002218077 1.589639 1.277884 -33.91342 252.268 0.91915
49
##                   ACF1
## Training set 0.0123567
```

```
summary(e.no2.lm) # ARIMA(1,0,2)
```

```
## Series: no2.lm$residuals
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1
##       1.3287  -0.3705  -0.8112
## s.e.  0.0935   0.0760   0.0732
##
## sigma^2 = 1086:  log likelihood = -1920.18
## AIC=3848.36   AICc=3848.46   BIC=3864.23
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE       MASE
      ACF1
## Training set -0.1626036 32.82693 25.85169 76.97745 189.326 0.9376907
 0.0145908
```

```
# See if the residuals are correlated
allResiduals <- data.frame(co.trend.seasonal$residuals, no2.trend.seaso
nal$residuals)
colnames(allResiduals) <- c("CO","NO2")
cor(allResiduals)
```

```
##            CO       NO2
## CO  1.0000000 0.6243741
## NO2 0.6243741 1.0000000
```

we can see the residuals are highly correlated

Examine a number of potential VARMA models with different p and q values

The model with lowest AIC is VARMA(2,3)

```
# Pick the model with the lowest AIC
AICmatrix
```

```
##          [,1]     [,2]       [,3]     [,4]
## [1,] 7.154400 7.173316   7.315219 7.118306
## [2,] 7.322403 8.205151   7.114808 7.115987
## [3,] 7.416771 8.457797 120.500278 7.128826
```

```
# Build the model with best AIC p=2,q=3
varma.model <- VARMACpp(allResiduals, p=2, q=3, include.mean=F) # aic =
 7.21
```

```
## Number of parameters:  20
## initial estimates:  0.4539 -0.0037 -0.3087 0.0072 1.9958 0.7533 -11.
8677 0.1692 -0.0754 0.0039 0.3161 -0.0032 0.2019 -0.0025 -3.5625 -0.268
8 9.3598 -0.1757 5.3921 -0.0218
## Par. lower-bounds:  -0.137 -0.0313 -0.7589 -0.0177 -10.2769 0.1796 -
21.2181 -0.3488 -0.6815 -0.0244 -0.0521 -0.018 0.0029 -0.0109 -16.1503
-0.8573 1.7136 -0.4819 1.2581 -0.1972
## Par. upper-bounds:  1.0448 0.0239 0.1415 0.0321 14.2684 1.3269 -2.51
73 0.6872 0.5307 0.0322 0.6842 0.0115 0.401 0.006 9.0253 0.3196 17.0061
 0.1304 9.5262 0.1535
## Final   Estimates:  0.4539208 -0.003692827 -0.3086913 0.007204856 1.
995758 0.7532815 -11.86771 0.1691811 -0.07538718 0.003903814 0.3160796
-0.003222334 0.201938 -0.002474412 -3.562457 -0.268843 9.359844 -0.1757
359 5.392146 -0.02182656
##
## Coefficient(s):
##        Estimate  Std. Error  t value Pr(>|t|)
## CO    0.453921         NaN      NaN      NaN
## NO2  -0.003693         NaN      NaN      NaN
## CO   -0.308691         NaN      NaN      NaN
## NO2   0.007205         NaN      NaN      NaN
## CO    1.995758         NaN      NaN      NaN
## NO2   0.753281         NaN      NaN      NaN
## CO   -11.867715        NaN      NaN      NaN
## NO2   0.169181         NaN      NaN      NaN
##      -0.075387         NaN      NaN      NaN
##       0.003904         NaN      NaN      NaN
##       0.316080         NaN      NaN      NaN
##      -0.003222         NaN      NaN      NaN
##       0.201938         NaN      NaN      NaN
##      -0.002474         NaN      NaN      NaN
##      -3.562457         NaN      NaN      NaN
##      -0.268843         NaN      NaN      NaN
##       9.359844         NaN      NaN      NaN
##      -0.175736         NaN      NaN      NaN
##       5.392146         NaN      NaN      NaN
##      -0.021827         NaN      NaN      NaN
## ---
## Estimates in matrix form:
## AR coefficient matrix
## AR( 1 )-matrix
```

```
##       [,1]     [,2]
## [1,] 0.454 -0.00369
## [2,] 1.996  0.75328
## AR( 2 )-matrix
##         [,1]   [,2]
## [1,]  -0.309 0.0072
## [2,] -11.868 0.1692
## MA coefficient matrix
## MA( 1 )-matrix
##        [,1]    [,2]
## [1,] 0.0754 -0.0039
## [2,] 3.5625  0.2688
## MA( 2 )-matrix
##        [,1]    [,2]
## [1,] -0.316 0.00322
## [2,] -9.360 0.17574
## MA( 3 )-matrix
##        [,1]    [,2]
## [1,] -0.202 0.00247
## [2,] -5.392 0.02183
##
## Residuals cov-matrix:
##           [,1]      [,2]
## [1,]  2.187621  30.28428
## [2,] 30.284284 927.43747
## ----
## aic=  7.115987
## bic=  7.318989
```

```
# Build another model (next best AIC) and compare diagnostics
varma.model2 <- VARMACpp(allResiduals, p=2, q=4, include.mean=F)
```

```
## Number of parameters:  24
## initial estimates:  0.2668 -0.0117 -0.6127 0.0198 1.1712 0.7362 -12.
9591 0.213 0.113 0.0119 0.6759 -0.0119 0.3568 -0.0057 0.2668 -0.0061 -2.
7334 -0.252 10.7264 -0.2109 6.0167 -0.0365 1.2044 -0.0387
## Par. lower-bounds:  -0.3337 -0.0419 -1.1344 -0.0076 -11.4335 0.1022
-23.9097 -0.3639 -0.5022 -0.019 0.2064 -0.0279 0.1273 -0.0143 0.0815 -0.
0136 -15.6457 -0.8993 0.8731 -0.5468 1.2008 -0.2178 -2.6851 -0.197
## Par. upper-bounds:  0.8674 0.0185 -0.091 0.0473 13.7758 1.3702 -2.00
85 0.7898 0.7282 0.0427 1.1453 0.0041 0.5862 0.003 0.4521 0.0015 10.178
8 0.3953 20.5796 0.125 10.8326 0.1448 5.0939 0.1195
## Final   Estimates:  0.2700046 -0.003596812 -0.5859131 -0.00761801 1.
546748 0.7815061 -13.19801 0.2031516 -0.4975751 -0.01267062 0.2362696 0.
004107754 0.2428389 -0.001100103 0.08698114 -0.006881578 -1.81065 -0.06
604994 10.90949 -0.5225777 1.200788 0.02287232 -0.9116242 0.1065261
```

```
## Warning in sqrt(diag(solve(Hessian))): 产生了 NaNs
```

```
##
## Coefficient(s):
```

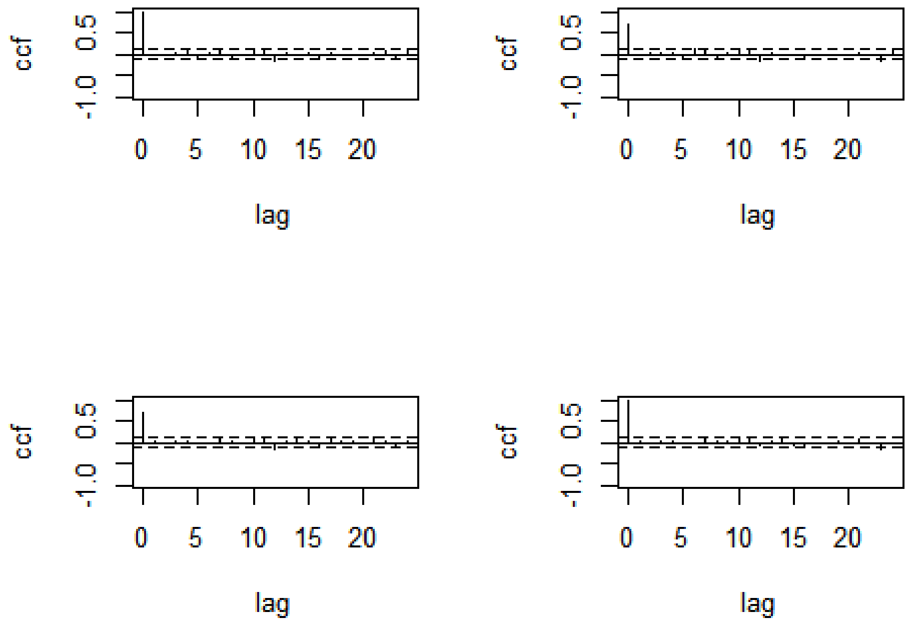
```
##        Estimate  Std. Error   t value Pr(>|t|)
## CO   2.700e-01   9.206e+00     0.029   0.9766
## NO2 -3.597e-03   2.053e-01    -0.018   0.9860
## CO  -5.859e-01         NaN       NaN      NaN
## NO2 -7.618e-03         NaN       NaN      NaN
## CO   1.547e+00   3.819e+01     0.041   0.9677
## NO2  7.815e-01   2.653e+00     0.295   0.7683
## CO  -1.320e+01         NaN       NaN      NaN
## NO2  2.032e-01         NaN       NaN      NaN
##     -4.976e-01   1.641e-02   -30.316  < 2e-16 ***
##     -1.267e-02   1.229e-03   -10.311  < 2e-16 ***
##      2.363e-01   2.018e-01     1.171   0.2417
##      4.108e-03   1.294e-02     0.317   0.7509
##      2.428e-01   5.965e-03    40.714  < 2e-16 ***
##     -1.100e-03   4.546e-04    -2.420   0.0155 *
##      8.698e-02   5.784e-02     1.504   0.1326
##     -6.882e-03   6.126e-03    -1.123   0.2613
##     -1.811e+00   1.198e-03 -1511.212  < 2e-16 ***
##     -6.605e-02   3.108e-04  -212.531  < 2e-16 ***
##      1.091e+01   5.992e-02   182.064  < 2e-16 ***
##     -5.226e-01   9.758e-03   -53.557  < 2e-16 ***
##      1.201e+00   1.554e-02    77.250  < 2e-16 ***
##      2.287e-02   1.308e-03    17.484  < 2e-16 ***
##     -9.116e-01   1.997e-01    -4.565 4.98e-06 ***
##      1.065e-01   1.706e-02     6.242 4.31e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## ---
## Estimates in matrix form:
## AR coefficient matrix
## AR( 1 )-matrix
##      [,1]    [,2]
## [1,] 0.27 -0.0036
## [2,] 1.55  0.7815
## AR( 2 )-matrix
##         [,1]     [,2]
## [1,]  -0.586 -0.00762
## [2,] -13.198  0.20315
## MA coefficient matrix
## MA( 1 )-matrix
##       [,1]   [,2]
## [1,] 0.498 0.0127
## [2,] 1.811 0.0660
## MA( 2 )-matrix
##         [,1]     [,2]
## [1,]  -0.236 -0.00411
## [2,] -10.909  0.52258
## MA( 3 )-matrix
##        [,1]    [,2]
## [1,] -0.243  0.0011
```

```
## [2,] -1.201 -0.0229
## MA( 4 )-matrix
##        [,1]     [,2]
## [1,] -0.087  0.00688
## [2,]  0.912 -0.10653
##
## Residuals cov-matrix:
##           [,1]       [,2]
## [1,]  7.416519   42.62121
## [2,] 42.621210 1782.84336
## ----
## aic=  9.46465
## bic=  9.708254
```

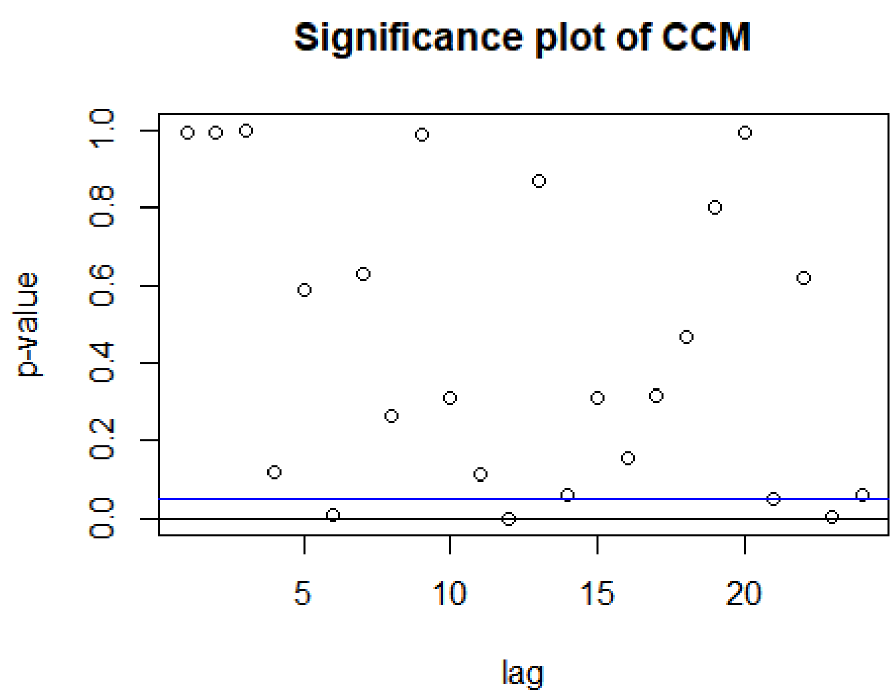
check the diagnostics

```
# independence of residuals
MTSdiag(varma.model)

## [1] "Covariance matrix:"
##        CO   NO2
## CO   2.19  30.4
## NO2 30.36 929.8
## CCM at lag:  0
##       [,1]  [,2]
## [1,] 1.000 0.672
## [2,] 0.672 1.000
## Simplified matrix:
## CCM at lag:  1
## . .
## . .
## CCM at lag:  2
## . .
## . .
## CCM at lag:  3
## . .
## . .
## CCM at lag:  4
## . .
## . .
## CCM at lag:  5
## . .
## . .
## CCM at lag:  6
## . +
## . .
## CCM at lag:  7
## . .
## . .
## CCM at lag:  8
```

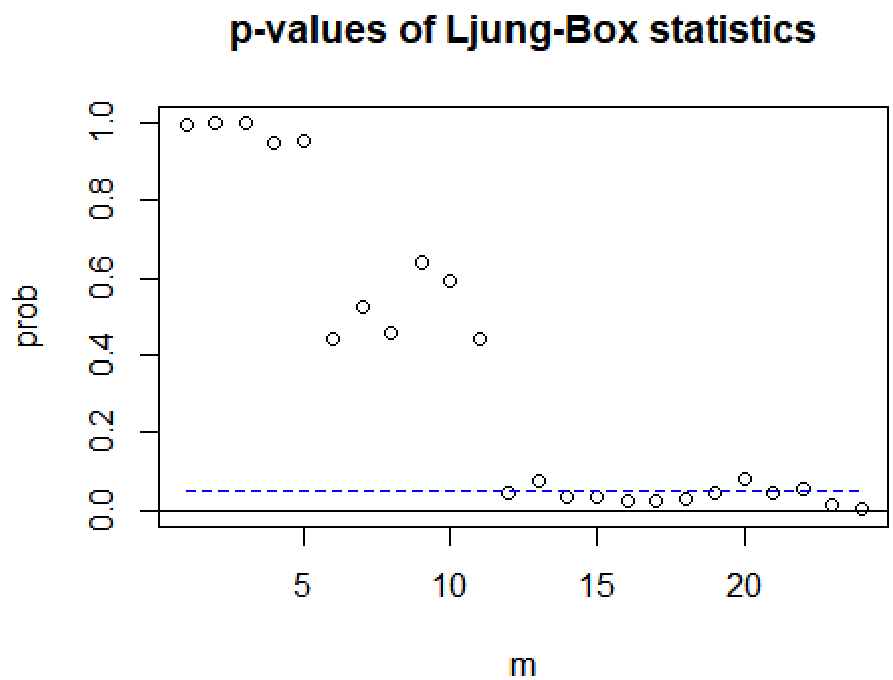
```
## . .
## . .
## CCM at lag:  9
## . .
## . .
## CCM at lag:  10
## . .
## . .
## CCM at lag:  11
## + +
## . .
## CCM at lag:  12
## - -
## - .
## CCM at lag:  13
## . .
## . .
## CCM at lag:  14
## . .
## . .
## CCM at lag:  15
## . .
## . .
## CCM at lag:  16
## - .
## . .
## CCM at lag:  17
## . .
## . .
## CCM at lag:  18
## . .
## . .
## CCM at lag:  19
## . .
## . .
## CCM at lag:  20
## . .
## . .
## CCM at lag:  21
## . .
## + .
## CCM at lag:  22
## . .
## . .
## CCM at lag:  23
## - -
## - -
## CCM at lag:  24
## + .
## . .
```



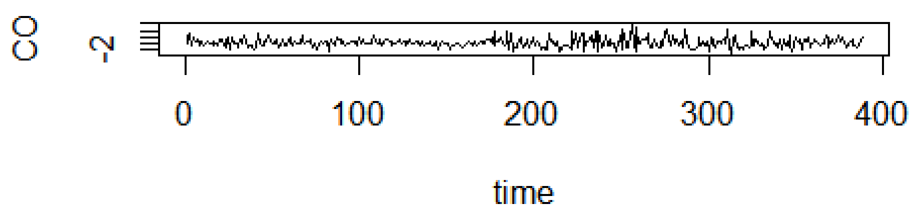

```
## Hit Enter for p-value plot of individual ccm:
```

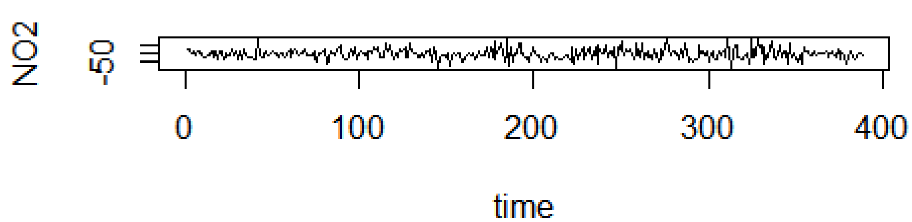


```
## Hit Enter to compute MQ-statistics:
##
## Ljung-Box Statistics:
##            m       Q(m)     df    p-value
##  [1,]   1.000     0.208   4.000     0.99
##  [2,]   2.000     0.472   8.000     1.00
##  [3,]   3.000     0.581  12.000     1.00
##  [4,]   4.000     7.996  16.000     0.95
##  [5,]   5.000    10.843  20.000     0.95
##  [6,]   6.000    24.307  24.000     0.44
##  [7,]   7.000    26.908  28.000     0.52
##  [8,]   8.000    32.158  32.000     0.46
##  [9,]   9.000    32.439  36.000     0.64
## [10,]  10.000    37.257  40.000     0.59
## [11,]  11.000    44.735  44.000     0.44
## [12,]  12.000    65.900  48.000     0.04
## [13,]  13.000    67.164  52.000     0.08
## [14,]  14.000    76.205  56.000     0.04
## [15,]  15.000    81.007  60.000     0.04
## [16,]  16.000    87.673  64.000     0.03
## [17,]  17.000    92.394  68.000     0.03
## [18,]  18.000    95.978  72.000     0.03
## [19,]  19.000    97.628  76.000     0.05
## [20,]  20.000    97.846  80.000     0.09
## [21,]  21.000   107.314  84.000     0.04
## [22,]  22.000   109.975  88.000     0.06
## [23,]  23.000   124.673  92.000     0.01
## [24,]  24.000   133.675  96.000     0.01
```




```
## Hit Enter to obtain residual plots:
```

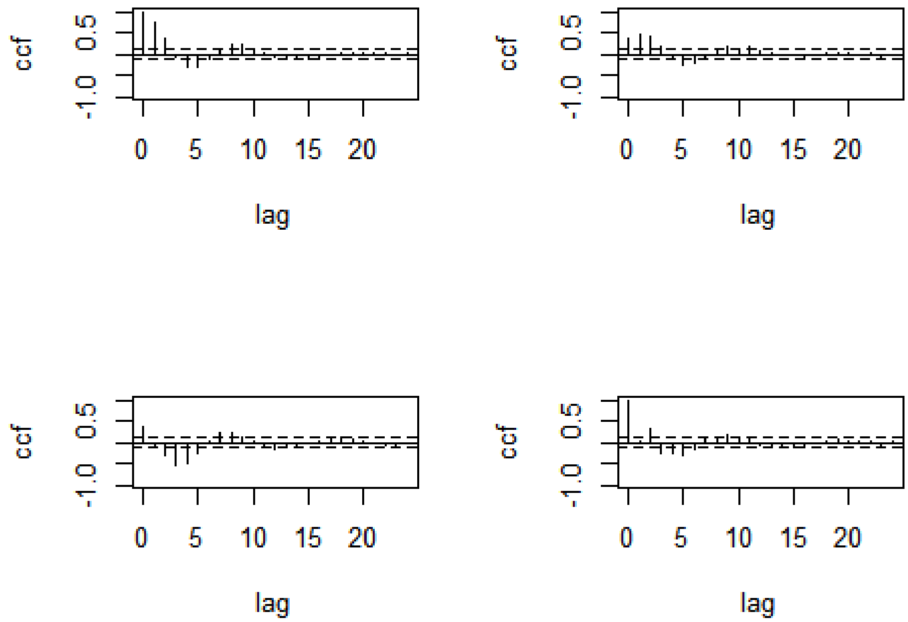




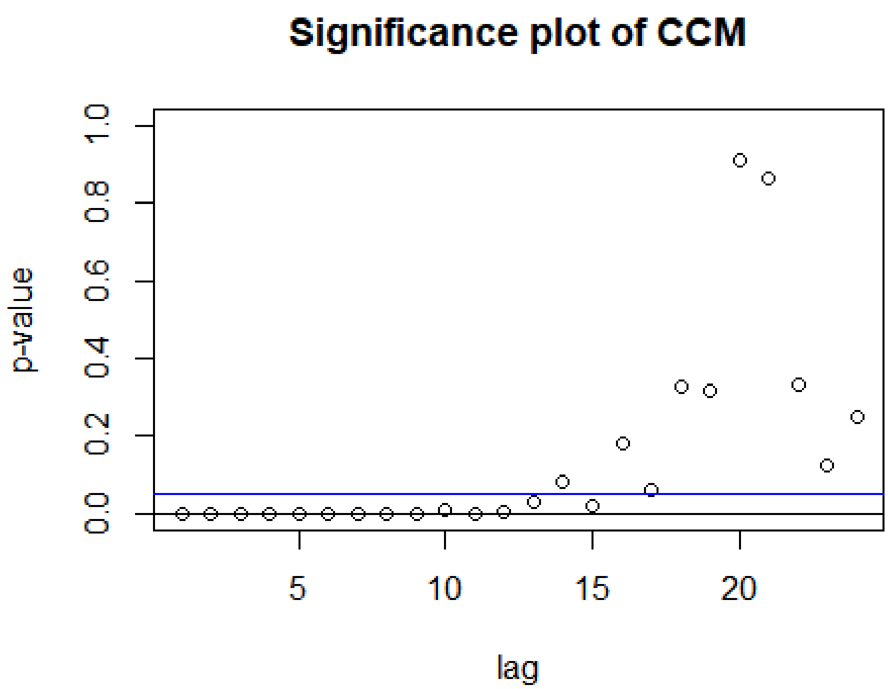

```
MTSdiag(varma.model2)
```

```
## [1] "Covariance matrix:"
##        CO    NO2
## CO   7.44   42.7
## NO2 42.73 1787.3
## CCM at lag:  0
##       [,1]  [,2]
## [1,] 1.000 0.371
## [2,] 0.371 1.000
## Simplified matrix:
## CCM at lag:  1
## + +
## - .
## CCM at lag:  2
## + +
## - +
## CCM at lag:  3
## . +
## - -
## CCM at lag:  4
## - .
## - -
## CCM at lag:  5
## - -
## - -
## CCM at lag:  6
## - -
## . -
## CCM at lag:  7
## . .
## + .
## CCM at lag:  8
## + .
## + +
## CCM at lag:  9
## + +
## + +
## CCM at lag:  10
## + +
## . .
## CCM at lag:  11
## . +
## . .
## CCM at lag:  12
## . .
## - .
## CCM at lag:  13
## - .
## . .
## CCM at lag:  14
## - .
```

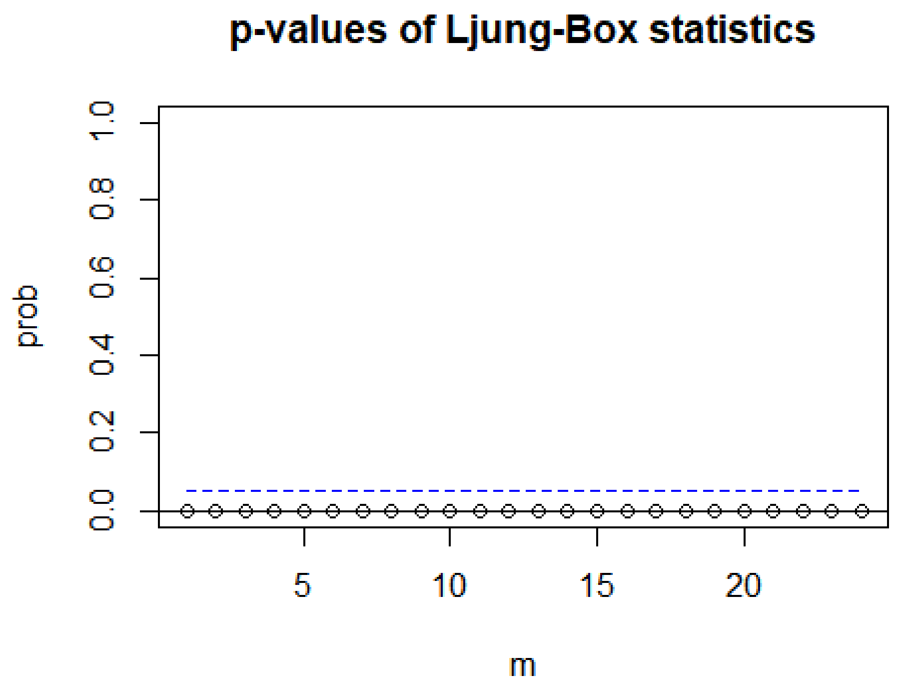
```
## . .
## CCM at lag:  15
## . .
## . -
## CCM at lag:  16
## . .
## . .
## CCM at lag:  17
## . .
## + .
## CCM at lag:  18
## . .
## . .
## CCM at lag:  19
## . .
## . .
## CCM at lag:  20
## . .
## . .
## CCM at lag:  21
## . .
## . .
## CCM at lag:  22
## . .
## . .
## CCM at lag:  23
## . .
## . -
## CCM at lag:  24
## . .
## . .
```



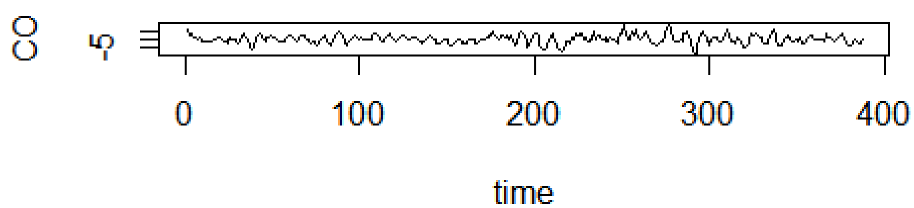

```
## Hit Enter for p-value plot of individual ccm:
```

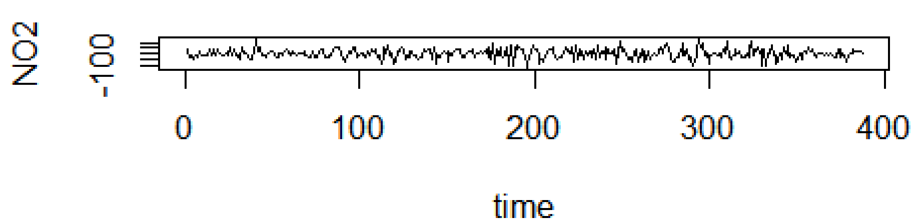


```
## Hit Enter to compute MQ-statistics:
##
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]     1       289       4         0
##  [2,]     2       506       8         0
##  [3,]     3       661      12         0
##  [4,]     4       768      16         0
##  [5,]     5       841      20         0
##  [6,]     6       872      24         0
##  [7,]     7       899      28         0
##  [8,]     8       928      32         0
##  [9,]     9       956      36         0
## [10,]    10       970      40         0
## [11,]    11       989      44         0
## [12,]    12      1004      48         0
## [13,]    13      1015      52         0
## [14,]    14      1023      56         0
## [15,]    15      1035      60         0
## [16,]    16      1042      64         0
## [17,]    17      1051      68         0
## [18,]    18      1055      72         0
## [19,]    19      1060      76         0
## [20,]    20      1061      80         0
## [21,]    21      1062      84         0
## [22,]    22      1067      88         0
## [23,]    23      1074      92         0
## [24,]    24      1080      96         0
```




```
## Hit Enter to obtain residual plots:
```





The CCFs are patternless, and the Ljung Box test indicates the model is adequate up to and including lag 11.

check the diagnostics of varma.model2. The CCFs show significance, indicating the model is not as good as the previous varma.model. The Ljung Box test shows no significant lags.

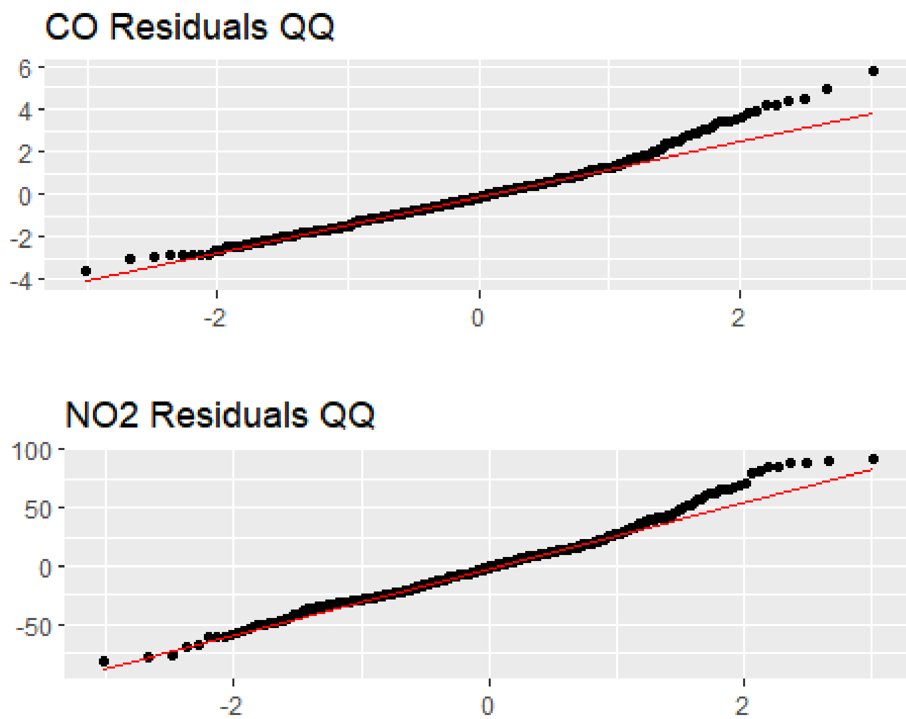
We choose VARMA(2,3) model as it has better diagnostics and AIC.

2d & 2e

As shown above, the VARMA(2,3) has better AIC and diagnose plots than VARMA(2,4) model We also check the QQ plots and Residuals vs Fitted for both CO and NO2 to enhance our conclusion

```
# Diagnostics

# compute fitted values (true - residual; lose 1st 2 observations becau
se p=2)
CO.fitted = allResiduals[3:dim(allResiduals)[1],1] - varma.model$residu
als[,1]

## Warning in allResiduals[3:dim(allResiduals)[1], 1] - varma.model$res
iduals[, :
## 长的对象长度不是短的对象长度的整倍数

NO2.fitted = allResiduals[3:dim(allResiduals)[1],2] - varma.model$resid
uals[,2]

## Warning in allResiduals[3:dim(allResiduals)[1], 2] - varma.model$res
iduals[, :
## 长的对象长度不是短的对象长度的整倍数

# Residuals vs Fitted
CO_resid_v_fitted = ggplot() + geom_point(aes(x=CO.fitted+co.lm$fitted.
values[3:length(co.lm$fitted.values)],
                              y=varma.model$residuals[,1])) +
                              xlab("CO Fitted Values") + ylab("CO Residuals
")

NO2_resid_v_fitted = ggplot() + geom_point(aes(x=NO2.fitted+no2.lm$fitt
ed.values[3:length(no2.lm$fitted.values)],
                              y=varma.model$residuals[,2]))  +
                              xlab("NO2 Fitted Values") + ylab("NO2 Residua
ls")



# QQ plot of residuals
coQQ = qplot(sample=varma.model$residuals[,1]) +
  stat_qq_line(color="red") + ggtitle("CO Residuals QQ")
```

```
no2QQ = qplot(sample=varma.model$residuals[,2]) +
  stat_qq_line(color="red") + ggtitle("NO2 Residuals QQ")

ggarrange(coQQ, no2QQ, nrow=2, ncol=1)
```



Diagnostics for model 1 look good, there is a slight tail on the QQ plots, but nothing too strong. Residual vs Fitted looks good, with even spread and ~0 slope. actually the diagnose for both model looks seem

3: Simulating from Univariate and Multivariate Models

1. simulate 1 year data for univariate models

```
set.seed(1)
e.CO.sim <- arima.sim(n=365, list(n=365, list(ar=c(co.auto$coef[1]),
                                       ma=c(co.auto$coef[2])),
                        sd=sqrt(co.auto$sigma2)))



e.NO2.sim <- arima.sim(n=365, list(n=365, list(ar=c(no2.auto$coef[1]),
                                       ma=c(no2.auto$coef[2]), no2.auto
$coef[3]),
                        sd=sqrt(no2.auto$sigma2)))

# Add mean predictions and plot simulation of Tmin
next.yr.time <- c(1:365)
```

```r
next.yr <- data.frame(time.index = next.yr.time)

# remove x-1 from model for simulating as x=0
co.trend.seasonal.predict <- lm(co.ts ~ time.index + sin(2*pi*time.inde
x/7) + cos(2*pi*time.index/7))
next.yr.co.predictions <- predict(co.trend.seasonal.predict, newdata=ne
xt.yr)
next.yr.co.predictions
```

```
##        1        2        3        4        5        6        7
  8
## 4.745610 4.588922 3.877465 3.148916 2.953823 3.441031 4.245597 4.763
602
##        9       10       11       12       13       14       15
 16
## 4.606914 3.895457 3.166908 2.971815 3.459023 4.263589 4.781594 4.624
906
##       17       18       19       20       21       22       23
 24
## 3.913449 3.184900 2.989808 3.477015 4.281581 4.799587 4.642898 3.931
441
##       25       26       27       28       29       30       31
 32
## 3.202892 3.007800 3.495007 4.299573 4.817579 4.660890 3.949433 3.220
885
##       33       34       35       36       37       38       39
 40
## 3.025792 3.512999 4.317565 4.835571 4.678882 3.967425 3.238877 3.043
784
##       41       42       43       44       45       46       47
 48
## 3.530991 4.335557 4.853563 4.696875 3.985418 3.256869 3.061776 3.548
984
##       49       50       51       52       53       54       55
 56
## 4.353550 4.871555 4.714867 4.003410 3.274861 3.079768 3.566976 4.371
542
##       57       58       59       60       61       62       63
 64
## 4.889547 4.732859 4.021402 3.292853 3.097760 3.584968 4.389534 4.907
539
##       65       66       67       68       69       70       71
 72
## 4.750851 4.039394 3.310845 3.115752 3.602960 4.407526 4.925531 4.768
843
##       73       74       75       76       77       78       79
 80
## 4.057386 3.328837 3.133745 3.620952 4.425518 4.943523 4.786835 4.075
378
##       81       82       83       84       85       86       87
```

```
 88
## 3.346829 3.151737 3.638944 4.443510 4.961516 4.804827 4.093370 3.364
822
##       89       90       91       92       93       94       95
 96
## 3.169729 3.656936 4.461502 4.979508 4.822819 4.111362 3.382814 3.187
721
##       97       98       99      100      101      102      103
104
## 3.674928 4.479494 4.997500 4.840812 4.129354 3.400806 3.205713 3.692
920
##      105      106      107      108      109      110      111
112
## 4.497487 5.015492 4.858804 4.147347 3.418798 3.223705 3.710913 4.515
479
##      113      114      115      116      117      118      119
120
## 5.033484 4.876796 4.165339 3.436790 3.241697 3.728905 4.533471 5.051
476
##      121      122      123      124      125      126      127
128
## 4.894788 4.183331 3.454782 3.259689 3.746897 4.551463 5.069468 4.912
780
##      129      130      131      132      133      134      135
136
## 4.201323 3.472774 3.277682 3.764889 4.569455 5.087460 4.930772 4.219
315
##      137      138      139      140      141      142      143
144
## 3.490766 3.295674 3.782881 4.587447 5.105453 4.948764 4.237307 3.508
759
##      145      146      147      148      149      150      151
152
## 3.313666 3.800873 4.605439 5.123445 4.966756 4.255299 3.526751 3.331
658
##      153      154      155      156      157      158      159
160
## 3.818865 4.623431 5.141437 4.984748 4.273291 3.544743 3.349650 3.836
857
##      161      162      163      164      165      166      167
168
## 4.641423 5.159429 5.002741 4.291284 3.562735 3.367642 3.854850 4.659
416
##      169      170      171      172      173      174      175
176
## 5.177421 5.020733 4.309276 3.580727 3.385634 3.872842 4.677408 5.195
413
##      177      178      179      180      181      182      183
184
## 5.038725 4.327268 3.598719 3.403626 3.890834 4.695400 5.213405 5.056
```

```
717
##       185       186       187       188       189       190       191
192
## 4.345260 3.616711 3.421618 3.908826 4.713392 5.231397 5.074709 4.363
252
##       193       194       195       196       197       198       199
200
## 3.634703 3.439611 3.926818 4.731384 5.249390 5.092701 4.381244 3.652
695
##       201       202       203       204       205       206       207
208
## 3.457603 3.944810 4.749376 5.267382 5.110693 4.399236 3.670688 3.475
595
##       209       210       211       212       213       214       215
216
## 3.962802 4.767368 5.285374 5.128685 4.417228 3.688680 3.493587 3.980
794
##       217       218       219       220       221       222       223
224
## 4.785360 5.303366 5.146678 4.435220 3.706672 3.511579 3.998787 4.803
353
##       225       226       227       228       229       230       231
232
## 5.321358 5.164670 4.453213 3.724664 3.529571 4.016779 4.821345 5.339
350
##       233       234       235       236       237       238       239
240
## 5.182662 4.471205 3.742656 3.547563 4.034771 4.839337 5.357342 5.200
654
##       241       242       243       244       245       246       247
248
## 4.489197 3.760648 3.565555 4.052763 4.857329 5.375334 5.218646 4.507
189
##       249       250       251       252       253       254       255
256
## 3.778640 3.583548 4.070755 4.875321 5.393326 5.236638 4.525181 3.796
632
##       257       258       259       260       261       262       263
264
## 3.601540 4.088747 4.893313 5.411319 5.254630 4.543173 3.814625 3.619
532
##       265       266       267       268       269       270       271
272
## 4.106739 4.911305 5.429311 5.272622 4.561165 3.832617 3.637524 4.124
731
##       273       274       275       276       277       278       279
280
## 4.929297 5.447303 5.290615 4.579157 3.850609 3.655516 4.142723 4.947
290
##       281       282       283       284       285       286       287
```

```
288
## 5.465295 5.308607 4.597150 3.868601 3.673508 4.160716 4.965282 5.483
287
##      289      290      291      292      293      294      295      
296
## 5.326599 4.615142 3.886593 3.691500 4.178708 4.983274 5.501279 5.344
591
##      297      298      299      300      301      302      303      
304
## 4.633134 3.904585 3.709492 4.196700 5.001266 5.519271 5.362583 4.651
126
##      305      306      307      308      309      310      311      
312
## 3.922577 3.727485 4.214692 5.019258 5.537263 5.380575 4.669118 3.940
569
##      313      314      315      316      317      318      319      
320
## 3.745477 4.232684 5.037250 5.555256 5.398567 4.687110 3.958562 3.763
469
##      321      322      323      324      325      326      327      
328
## 4.250676 5.055242 5.573248 5.416559 4.705102 3.976554 3.781461 4.268
668
##      329      330      331      332      333      334      335      
336
## 5.073234 5.591240 5.434551 4.723094 3.994546 3.799453 4.286660 5.091
226
##      337      338      339      340      341      342      343      
344
## 5.609232 5.452544 4.741087 4.012538 3.817445 4.304653 5.109219 5.627
224
##      345      346      347      348      349      350      351      
352
## 5.470536 4.759079 4.030530 3.835437 4.322645 5.127211 5.645216 5.488
528
##      353      354      355      356      357      358      359      
360
## 4.777071 4.048522 3.853429 4.340637 5.145203 5.663208 5.506520 4.795
063
##      361      362      363      364      365 
## 4.066514 3.871421 4.358629 5.163195 5.681200

# remove x-2 from model for simulating as x=0
no2.trend.seasonal.predict <- lm(no2.ts ~ time.index + sin(2*pi*time.in
dex/7) + cos(2*pi*time.index/7))
next.yr.no2.predictions <- predict(no2.trend.seasonal.predict, newdata=
next.yr)
next.yr.no2.predictions
```

```
##        1        2        3        4        5        6        7        
 8 
## 131.9349 130.3955 118.5180 105.4333 101.1814 109.1508 123.5274 133.6
721 
##        9       10       11       12       13       14       15       
 16 
## 132.1327 120.2552 107.1705 102.9186 110.8880 125.2646 135.4093 133.8
699 
##       17       18       19       20       21       22       23       
 24 
## 121.9924 108.9077 104.6558 112.6252 127.0018 137.1465 135.6071 123.7
296 
##       25       26       27       28       29       30       31       
 32 
## 110.6449 106.3930 114.3624 128.7390 138.8837 137.3443 125.4668 112.3
821 
##       33       34       35       36       37       38       39       
 40 
## 108.1302 116.0996 130.4762 140.6209 139.0815 127.2040 114.1193 109.8
674 
##       41       42       43       44       45       46       47       
 48 
## 117.8368 132.2134 142.3581 140.8187 128.9412 115.8565 111.6046 119.5
740 
##       49       50       51       52       53       54       55       
 56 
## 133.9506 144.0953 142.5559 130.6784 117.5937 113.3418 121.3112 135.6
878 
##       57       58       59       60       61       62       63       
 64 
## 145.8325 144.2931 132.4156 119.3309 115.0790 123.0484 137.4250 147.5
697 
##       65       66       67       68       69       70       71       
 72 
## 146.0303 134.1528 121.0682 116.8162 124.7856 139.1622 149.3069 147.7
675 
##       73       74       75       76       77       78       79       
 80 
## 135.8900 122.8054 118.5534 126.5228 140.8994 151.0441 149.5047 137.6
272 
##       81       82       83       84       85       86       87       
 88 
## 124.5426 120.2906 128.2600 142.6366 152.7813 151.2419 139.3644 126.2
798 
##       89       90       91       92       93       94       95       
 96 
## 122.0278 129.9972 144.3738 154.5185 152.9791 141.1016 128.0170 123.7
650 
##       97       98       99      100      101      102      103      
104 
```

```
## 131.7344 146.1110 156.2557 154.7163 142.8388 129.7542 125.5022 133.4
717
##      105      106      107      108      109      110      111
112
## 147.8482 157.9929 156.4535 144.5760 131.4914 127.2394 135.2089 149.5
854
##      113      114      115      116      117      118      119
120
## 159.7301 158.1907 146.3132 133.2286 128.9766 136.9461 151.3226 161.4
673
##      121      122      123      124      125      126      127
128
## 159.9279 148.0504 134.9658 130.7138 138.6833 153.0598 163.2045 161.6
651
##      129      130      131      132      133      134      135
136
## 149.7876 136.7030 132.4510 140.4205 154.7970 164.9417 163.4023 151.5
248
##      137      138      139      140      141      142      143
144
## 138.4402 134.1882 142.1577 156.5342 166.6789 165.1395 153.2620 140.1
774
##      145      146      147      148      149      150      151
152
## 135.9254 143.8949 158.2714 168.4161 166.8767 154.9992 141.9146 137.6
626
##      153      154      155      156      157      158      159
160
## 145.6321 160.0086 170.1533 168.6139 156.7364 143.6518 139.3998 147.3
693
##      161      162      163      164      165      166      167
168
## 161.7458 171.8905 170.3511 158.4736 145.3890 141.1370 149.1065 163.4
830
##      169      170      171      172      173      174      175
176
## 173.6277 172.0883 160.2108 147.1262 142.8742 150.8437 165.2202 175.3
650
##      177      178      179      180      181      182      183
184
## 173.8255 161.9480 148.8634 144.6114 152.5809 166.9574 177.1022 175.5
627
##      185      186      187      188      189      190      191
192
## 163.6852 150.6006 146.3486 154.3181 168.6946 178.8394 177.2999 165.4
224
##      193      194      195      196      197      198      199
200
## 152.3378 148.0858 156.0553 170.4318 180.5766 179.0371 167.1596 154.0
750
```

```
##      201      202      203      204      205      206      207
208
## 149.8230 157.7925 172.1690 182.3138 180.7743 168.8968 155.8122 151.5
602
##      209      210      211      212      213      214      215
216
## 159.5297 173.9062 184.0510 182.5115 170.6340 157.5494 153.2974 161.2
669
##      217      218      219      220      221      222      223
224
## 175.6434 185.7882 184.2487 172.3712 159.2866 155.0346 163.0041 177.3
806
##      225      226      227      228      229      230      231
232
## 187.5254 185.9859 174.1084 161.0238 156.7718 164.7413 179.1178 189.2
626
##      233      234      235      236      237      238      239
240
## 187.7231 175.8457 162.7610 158.5090 166.4785 180.8550 190.9998 189.4
603
##      241      242      243      244      245      246      247
248
## 177.5829 164.4982 160.2462 168.2157 182.5922 192.7370 191.1975 179.3
201
##      249      250      251      252      253      254      255
256
## 166.2354 161.9834 169.9529 184.3294 194.4742 192.9347 181.0573 167.9
726
##      257      258      259      260      261      262      263
264
## 163.7206 171.6901 186.0666 196.2114 194.6719 182.7945 169.7098 165.4
578
##      265      266      267      268      269      270      271
272
## 173.4273 187.8039 197.9486 196.4091 184.5317 171.4470 167.1950 175.1
645
##      273      274      275      276      277      278      279
280
## 189.5411 199.6858 198.1463 186.2689 173.1842 168.9322 176.9017 191.2
783
##      281      282      283      284      285      286      287
288
## 201.4230 199.8835 188.0061 174.9214 170.6694 178.6389 193.0155 203.1
602
##      289      290      291      292      293      294      295
296
## 201.6207 189.7433 176.6586 172.4066 180.3761 194.7527 204.8974 203.3
579
##      297      298      299      300      301      302      303
304
```

```
## 191.4805 178.3958 174.1438 182.1133 196.4899 206.6346 205.0951 193.2
177
##      305      306      307      308      309      310      311
312
## 180.1330 175.8810 183.8505 198.2271 208.3718 206.8323 194.9549 181.8
702
##      313      314      315      316      317      318      319
320
## 177.6182 185.5877 199.9643 210.1090 208.5695 196.6921 183.6074 179.3
554
##      321      322      323      324      325      326      327
328
## 187.3249 201.7015 211.8462 210.3067 198.4293 185.3446 181.0927 189.0
621
##      329      330      331      332      333      334      335
336
## 203.4387 213.5834 212.0440 200.1665 187.0818 182.8299 190.7993 205.1
759
##      337      338      339      340      341      342      343
344
## 215.3206 213.7812 201.9037 188.8190 184.5671 192.5365 206.9131 217.0
578
##      345      346      347      348      349      350      351
352
## 215.5184 203.6409 190.5562 186.3043 194.2737 208.6503 218.7950 217.2
556
##      353      354      355      356      357      358      359
360
## 205.3781 192.2934 188.0415 196.0109 210.3875 220.5322 218.9928 207.1
153
##      361      362      363      364      365
## 194.0306 189.7787 197.7481 212.1247 222.2694
```

2. simulate 1 year data for multivariate models

```
sim_muti = VARMAsim(365,phi=varma.model$Phi,theta=varma.model$Theta,sig
ma=varma.model$Sigma)

time.next.yr <- c(1:365)
next.yr.df <- data.frame(time.index = time.next.yr)

mean.co <- predict(co.trend.seasonal.predict, newdata=next.yr.df)
mean.co

##        1        2        3        4        5        6        7
  8
## 4.745610 4.588922 3.877465 3.148916 2.953823 3.441031 4.245597 4.763
602
##        9       10       11       12       13       14       15
 16
## 4.606914 3.895457 3.166908 2.971815 3.459023 4.263589 4.781594 4.624
```

```
906
##        17        18        19        20        21        22        23
 24
## 3.913449 3.184900 2.989808 3.477015 4.281581 4.799587 4.642898 3.931
441
##        25        26        27        28        29        30        31
 32
## 3.202892 3.007800 3.495007 4.299573 4.817579 4.660890 3.949433 3.220
885
##        33        34        35        36        37        38        39
 40
## 3.025792 3.512999 4.317565 4.835571 4.678882 3.967425 3.238877 3.043
784
##        41        42        43        44        45        46        47
 48
## 3.530991 4.335557 4.853563 4.696875 3.985418 3.256869 3.061776 3.548
984
##        49        50        51        52        53        54        55
 56
## 4.353550 4.871555 4.714867 4.003410 3.274861 3.079768 3.566976 4.371
542
##        57        58        59        60        61        62        63
 64
## 4.889547 4.732859 4.021402 3.292853 3.097760 3.584968 4.389534 4.907
539
##        65        66        67        68        69        70        71
 72
## 4.750851 4.039394 3.310845 3.115752 3.602960 4.407526 4.925531 4.768
843
##        73        74        75        76        77        78        79
 80
## 4.057386 3.328837 3.133745 3.620952 4.425518 4.943523 4.786835 4.075
378
##        81        82        83        84        85        86        87
 88
## 3.346829 3.151737 3.638944 4.443510 4.961516 4.804827 4.093370 3.364
822
##        89        90        91        92        93        94        95
 96
## 3.169729 3.656936 4.461502 4.979508 4.822819 4.111362 3.382814 3.187
721
##        97        98        99       100       101       102       103
104
## 3.674928 4.479494 4.997500 4.840812 4.129354 3.400806 3.205713 3.692
920
##       105       106       107       108       109       110       111
112
## 4.497487 5.015492 4.858804 4.147347 3.418798 3.223705 3.710913 4.515
479
##       113       114       115       116       117       118       119
```

```
120
## 5.033484 4.876796 4.165339 3.436790 3.241697 3.728905 4.533471 5.051
476
##      121      122      123      124      125      126      127      
128
## 4.894788 4.183331 3.454782 3.259689 3.746897 4.551463 5.069468 4.912
780
##      129      130      131      132      133      134      135      
136
## 4.201323 3.472774 3.277682 3.764889 4.569455 5.087460 4.930772 4.219
315
##      137      138      139      140      141      142      143      
144
## 3.490766 3.295674 3.782881 4.587447 5.105453 4.948764 4.237307 3.508
759
##      145      146      147      148      149      150      151      
152
## 3.313666 3.800873 4.605439 5.123445 4.966756 4.255299 3.526751 3.331
658
##      153      154      155      156      157      158      159      
160
## 3.818865 4.623431 5.141437 4.984748 4.273291 3.544743 3.349650 3.836
857
##      161      162      163      164      165      166      167      
168
## 4.641423 5.159429 5.002741 4.291284 3.562735 3.367642 3.854850 4.659
416
##      169      170      171      172      173      174      175      
176
## 5.177421 5.020733 4.309276 3.580727 3.385634 3.872842 4.677408 5.195
413
##      177      178      179      180      181      182      183      
184
## 5.038725 4.327268 3.598719 3.403626 3.890834 4.695400 5.213405 5.056
717
##      185      186      187      188      189      190      191      
192
## 4.345260 3.616711 3.421618 3.908826 4.713392 5.231397 5.074709 4.363
252
##      193      194      195      196      197      198      199      
200
## 3.634703 3.439611 3.926818 4.731384 5.249390 5.092701 4.381244 3.652
695
##      201      202      203      204      205      206      207      
208
## 3.457603 3.944810 4.749376 5.267382 5.110693 4.399236 3.670688 3.475
595
##      209      210      211      212      213      214      215      
216
## 3.962802 4.767368 5.285374 5.128685 4.417228 3.688680 3.493587 3.980
```

```
794
##      217      218      219      220      221      222      223      
224
## 4.785360 5.303366 5.146678 4.435220 3.706672 3.511579 3.998787 4.803
353
##      225      226      227      228      229      230      231      
232
## 5.321358 5.164670 4.453213 3.724664 3.529571 4.016779 4.821345 5.339
350
##      233      234      235      236      237      238      239      
240
## 5.182662 4.471205 3.742656 3.547563 4.034771 4.839337 5.357342 5.200
654
##      241      242      243      244      245      246      247      
248
## 4.489197 3.760648 3.565555 4.052763 4.857329 5.375334 5.218646 4.507
189
##      249      250      251      252      253      254      255      
256
## 3.778640 3.583548 4.070755 4.875321 5.393326 5.236638 4.525181 3.796
632
##      257      258      259      260      261      262      263      
264
## 3.601540 4.088747 4.893313 5.411319 5.254630 4.543173 3.814625 3.619
532
##      265      266      267      268      269      270      271      
272
## 4.106739 4.911305 5.429311 5.272622 4.561165 3.832617 3.637524 4.124
731
##      273      274      275      276      277      278      279      
280
## 4.929297 5.447303 5.290615 4.579157 3.850609 3.655516 4.142723 4.947
290
##      281      282      283      284      285      286      287      
288
## 5.465295 5.308607 4.597150 3.868601 3.673508 4.160716 4.965282 5.483
287
##      289      290      291      292      293      294      295      
296
## 5.326599 4.615142 3.886593 3.691500 4.178708 4.983274 5.501279 5.344
591
##      297      298      299      300      301      302      303      
304
## 4.633134 3.904585 3.709492 4.196700 5.001266 5.519271 5.362583 4.651
126
##      305      306      307      308      309      310      311      
312
## 3.922577 3.727485 4.214692 5.019258 5.537263 5.380575 4.669118 3.940
569
##      313      314      315      316      317      318      319      
```

```
320
## 3.745477 4.232684 5.037250 5.555256 5.398567 4.687110 3.958562 3.763
469
##      321      322      323      324      325      326      327
328
## 4.250676 5.055242 5.573248 5.416559 4.705102 3.976554 3.781461 4.268
668
##      329      330      331      332      333      334      335
336
## 5.073234 5.591240 5.434551 4.723094 3.994546 3.799453 4.286660 5.091
226
##      337      338      339      340      341      342      343
344
## 5.609232 5.452544 4.741087 4.012538 3.817445 4.304653 5.109219 5.627
224
##      345      346      347      348      349      350      351
352
## 5.470536 4.759079 4.030530 3.835437 4.322645 5.127211 5.645216 5.488
528
##      353      354      355      356      357      358      359
360
## 4.777071 4.048522 3.853429 4.340637 5.145203 5.663208 5.506520 4.795
063
##      361      362      363      364      365
## 4.066514 3.871421 4.358629 5.163195 5.681200
```

```
mean.no2 <- predict(no2.trend.seasonal.predict, newdata=next.yr.df)
mean.no2
```

```
##        1        2        3        4        5        6        7
  8
## 131.9349 130.3955 118.5180 105.4333 101.1814 109.1508 123.5274 133.6
721
##        9       10       11       12       13       14       15
 16
## 132.1327 120.2552 107.1705 102.9186 110.8880 125.2646 135.4093 133.8
699
##       17       18       19       20       21       22       23
 24
## 121.9924 108.9077 104.6558 112.6252 127.0018 137.1465 135.6071 123.7
296
##       25       26       27       28       29       30       31
 32
## 110.6449 106.3930 114.3624 128.7390 138.8837 137.3443 125.4668 112.3
821
##       33       34       35       36       37       38       39
 40
## 108.1302 116.0996 130.4762 140.6209 139.0815 127.2040 114.1193 109.8
674
##       41       42       43       44       45       46       47
```

```
 48
## 117.8368 132.2134 142.3581 140.8187 128.9412 115.8565 111.6046 119.5
740
##       49       50       51       52       53       54       55
 56
## 133.9506 144.0953 142.5559 130.6784 117.5937 113.3418 121.3112 135.6
878
##       57       58       59       60       61       62       63
 64
## 145.8325 144.2931 132.4156 119.3309 115.0790 123.0484 137.4250 147.5
697
##       65       66       67       68       69       70       71
 72
## 146.0303 134.1528 121.0682 116.8162 124.7856 139.1622 149.3069 147.7
675
##       73       74       75       76       77       78       79
 80
## 135.8900 122.8054 118.5534 126.5228 140.8994 151.0441 149.5047 137.6
272
##       81       82       83       84       85       86       87
 88
## 124.5426 120.2906 128.2600 142.6366 152.7813 151.2419 139.3644 126.2
798
##       89       90       91       92       93       94       95
 96
## 122.0278 129.9972 144.3738 154.5185 152.9791 141.1016 128.0170 123.7
650
##       97       98       99      100      101      102      103
104
## 131.7344 146.1110 156.2557 154.7163 142.8388 129.7542 125.5022 133.4
717
##      105      106      107      108      109      110      111
112
## 147.8482 157.9929 156.4535 144.5760 131.4914 127.2394 135.2089 149.5
854
##      113      114      115      116      117      118      119
120
## 159.7301 158.1907 146.3132 133.2286 128.9766 136.9461 151.3226 161.4
673
##      121      122      123      124      125      126      127
128
## 159.9279 148.0504 134.9658 130.7138 138.6833 153.0598 163.2045 161.6
651
##      129      130      131      132      133      134      135
136
## 149.7876 136.7030 132.4510 140.4205 154.7970 164.9417 163.4023 151.5
248
##      137      138      139      140      141      142      143
144
## 138.4402 134.1882 142.1577 156.5342 166.6789 165.1395 153.2620 140.1
```

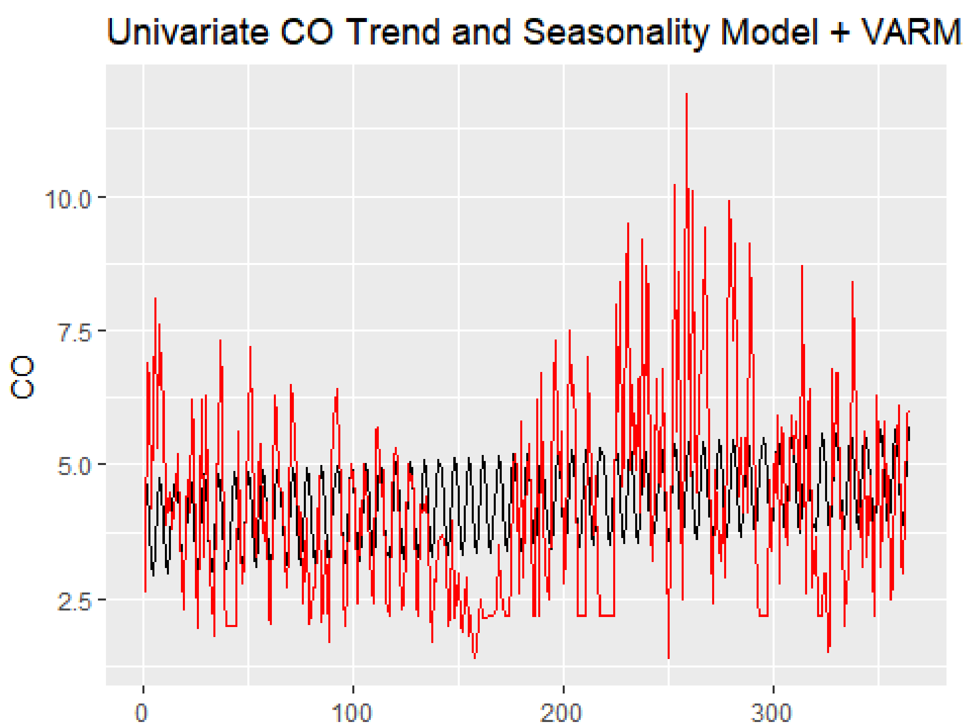
```
774
##      145      146      147      148      149      150      151      
152 
## 135.9254 143.8949 158.2714 168.4161 166.8767 154.9992 141.9146 137.6
626 
##      153      154      155      156      157      158      159      
160 
## 145.6321 160.0086 170.1533 168.6139 156.7364 143.6518 139.3998 147.3
693 
##      161      162      163      164      165      166      167      
168 
## 161.7458 171.8905 170.3511 158.4736 145.3890 141.1370 149.1065 163.4
830 
##      169      170      171      172      173      174      175      
176 
## 173.6277 172.0883 160.2108 147.1262 142.8742 150.8437 165.2202 175.3
650 
##      177      178      179      180      181      182      183      
184 
## 173.8255 161.9480 148.8634 144.6114 152.5809 166.9574 177.1022 175.5
627 
##      185      186      187      188      189      190      191      
192 
## 163.6852 150.6006 146.3486 154.3181 168.6946 178.8394 177.2999 165.4
224 
##      193      194      195      196      197      198      199      
200 
## 152.3378 148.0858 156.0553 170.4318 180.5766 179.0371 167.1596 154.0
750 
##      201      202      203      204      205      206      207      
208 
## 149.8230 157.7925 172.1690 182.3138 180.7743 168.8968 155.8122 151.5
602 
##      209      210      211      212      213      214      215      
216 
## 159.5297 173.9062 184.0510 182.5115 170.6340 157.5494 153.2974 161.2
669 
##      217      218      219      220      221      222      223      
224 
## 175.6434 185.7882 184.2487 172.3712 159.2866 155.0346 163.0041 177.3
806 
##      225      226      227      228      229      230      231      
232 
## 187.5254 185.9859 174.1084 161.0238 156.7718 164.7413 179.1178 189.2
626 
##      233      234      235      236      237      238      239      
240 
## 187.7231 175.8457 162.7610 158.5090 166.4785 180.8550 190.9998 189.4
603 
##      241      242      243      244      245      246      247      
```

```
248
## 177.5829 164.4982 160.2462 168.2157 182.5922 192.7370 191.1975 179.3
201
##      249      250      251      252      253      254      255
256
## 166.2354 161.9834 169.9529 184.3294 194.4742 192.9347 181.0573 167.9
726
##      257      258      259      260      261      262      263
264
## 163.7206 171.6901 186.0666 196.2114 194.6719 182.7945 169.7098 165.4
578
##      265      266      267      268      269      270      271
272
## 173.4273 187.8039 197.9486 196.4091 184.5317 171.4470 167.1950 175.1
645
##      273      274      275      276      277      278      279
280
## 189.5411 199.6858 198.1463 186.2689 173.1842 168.9322 176.9017 191.2
783
##      281      282      283      284      285      286      287
288
## 201.4230 199.8835 188.0061 174.9214 170.6694 178.6389 193.0155 203.1
602
##      289      290      291      292      293      294      295
296
## 201.6207 189.7433 176.6586 172.4066 180.3761 194.7527 204.8974 203.3
579
##      297      298      299      300      301      302      303
304
## 191.4805 178.3958 174.1438 182.1133 196.4899 206.6346 205.0951 193.2
177
##      305      306      307      308      309      310      311
312
## 180.1330 175.8810 183.8505 198.2271 208.3718 206.8323 194.9549 181.8
702
##      313      314      315      316      317      318      319
320
## 177.6182 185.5877 199.9643 210.1090 208.5695 196.6921 183.6074 179.3
554
##      321      322      323      324      325      326      327
328
## 187.3249 201.7015 211.8462 210.3067 198.4293 185.3446 181.0927 189.0
621
##      329      330      331      332      333      334      335
336
## 203.4387 213.5834 212.0440 200.1665 187.0818 182.8299 190.7993 205.1
759
##      337      338      339      340      341      342      343
344
## 215.3206 213.7812 201.9037 188.8190 184.5671 192.5365 206.9131 217.0
```

```
578
##       345       346       347       348       349       350       351
352
## 215.5184 203.6409 190.5562 186.3043 194.2737 208.6503 218.7950 217.2
556
##       353       354       355       356       357       358       359
360
## 205.3781 192.2934 188.0415 196.0109 210.3875 220.5322 218.9928 207.1
153
##       361       362       363       364       365
## 194.0306 189.7787 197.7481 212.1247 222.2694
```

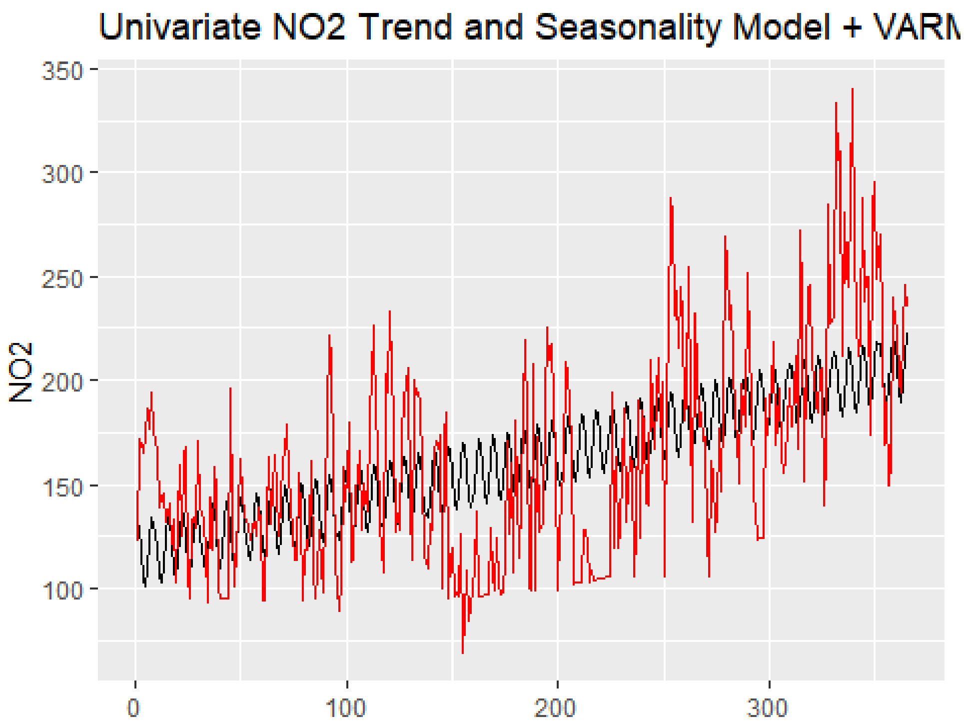
#3a)

```r
# Univariate CO
ggplot() +
  geom_line(aes(x=1:365,y=next.yr.co.predictions),color="black") +
  geom_line(aes(x=1:365,y=co.ts[1:365]),color="red") +
  xlab("") + ylab("CO") +
  ggtitle("Univariate CO Trend and Seasonality Model + VARMA of Residua
ls")
```

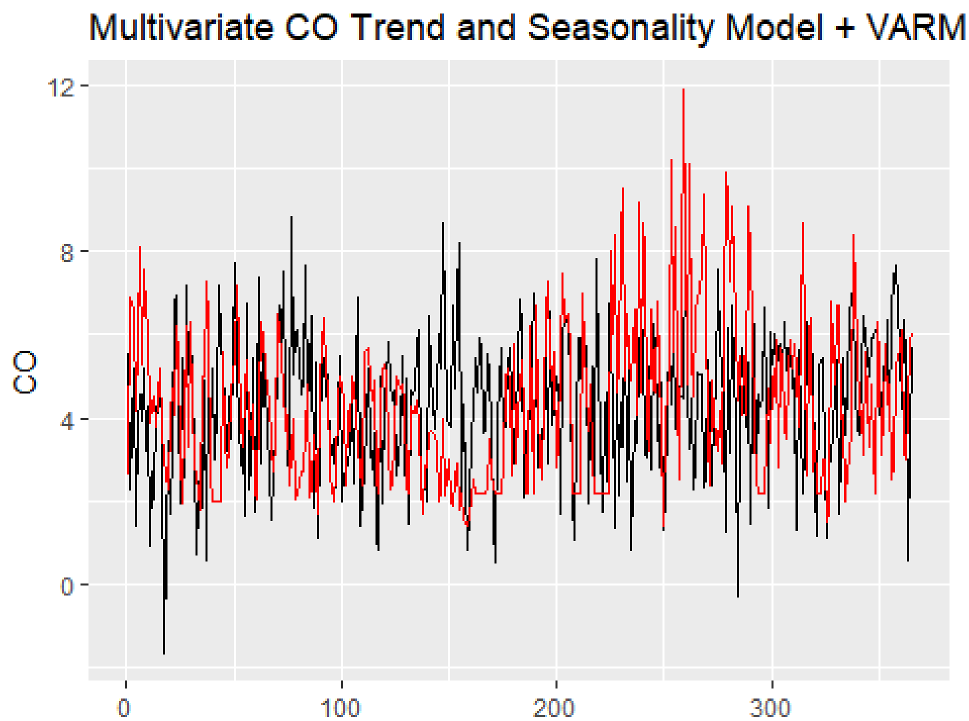



```r
# Univariate NO2
ggplot() +
  geom_line(aes(x=1:365,y=next.yr.no2.predictions),color="black") +
  geom_line(aes(x=1:365,y=no2.ts[1:365]),color="red") +
  xlab("") + ylab("NO2") +
```

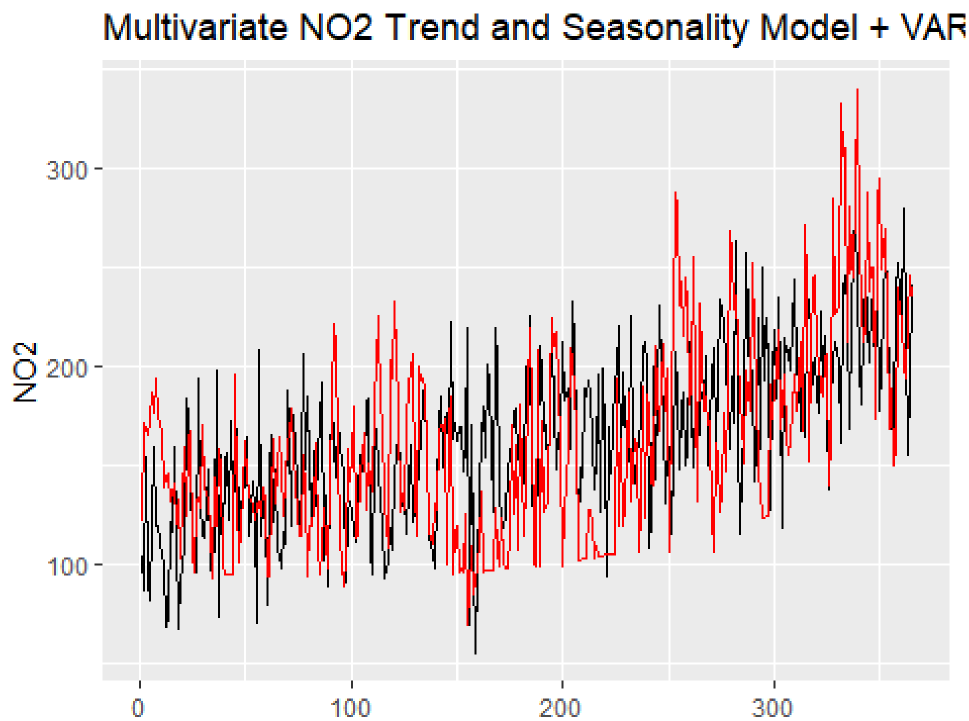
```
  ggtitle("Univariate NO2 Trend and Seasonality Model + VARMA of Residu
als")
```




```
# Multivariate CO
ggplot() +
  geom_line(aes(x=1:365,y=sim_muti$series[,1]+mean.co),color="black") +

  geom_line(aes(x=1:365,y=co.ts[1:365]),color="red") +
  xlab("") + ylab("CO") +
  ggtitle("Multivariate CO Trend and Seasonality Model + VARMA of Resid
uals")
```



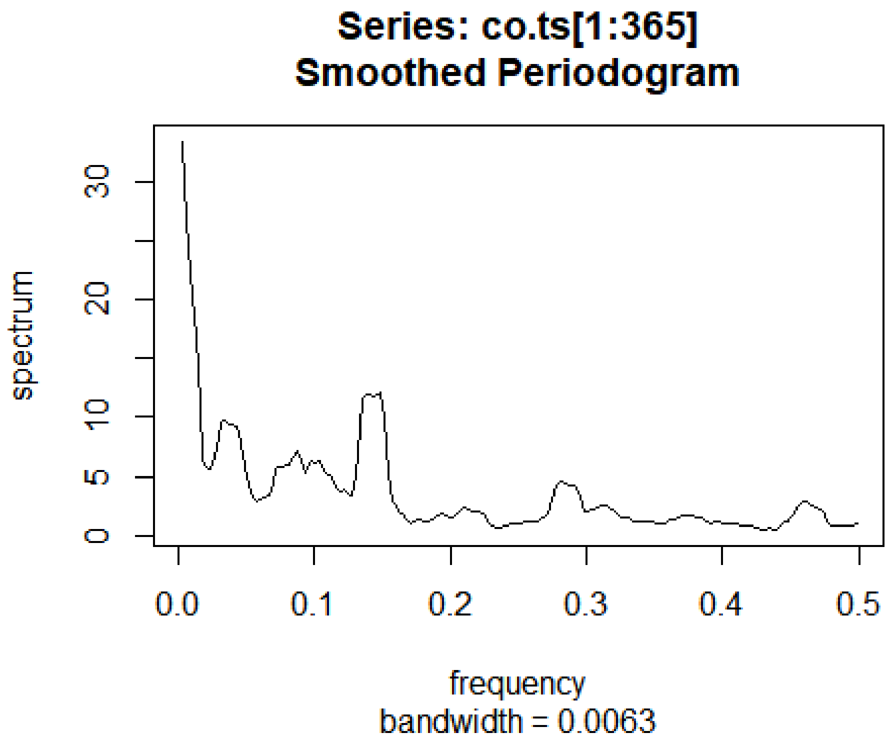

```r
# Multivariate NO2
ggplot() +
  geom_line(aes(x=1:365,y=sim_muti$series[,2]+mean.no2),color="black")
+
  geom_line(aes(x=1:365,y=no2.ts[1:365]),color="red") +
  xlab("") + ylab("NO2") +
  ggtitle("Multivariate NO2 Trend and Seasonality Model + VARMA of Resi
duals")
```



While the univariate models show the overall trend of the series, they lack the resolution to model the seasonality with the accuriacy of the multivariate models.

#3b)

```
# Original Data
pg.co <- spec.pgram(co.ts[1:365],spans=9,demean=T,log='no')
```




```
pg.co

## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
```

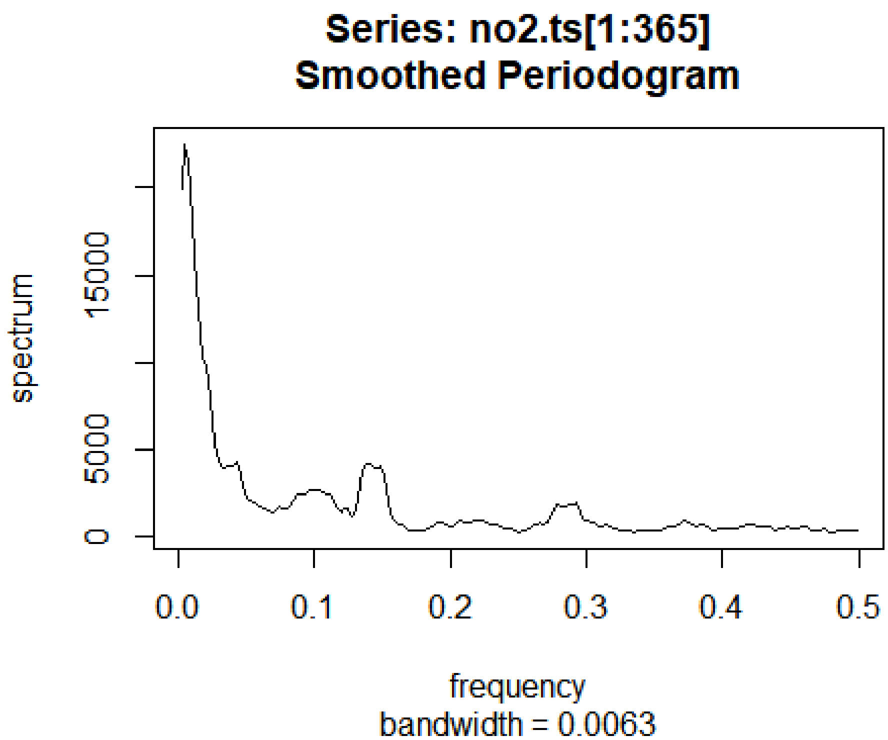
```
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
##
## $spec
##   [1] 33.4068928 29.7923025 24.2594087 20.3541235 16.8476099 11.1661
331
##   [7]  6.4906741  5.8206726  5.6507687  6.6996715  8.4348383  9.6374
367
##  [13]  9.7303679  9.4634122  9.4043079  9.2838359  8.8550135  6.8010
016
##  [19]  4.8521438  3.7924240  3.0767493  2.9238570  3.0279981  3.2242
576
##  [25]  3.3563208  4.3898205  5.5470322  5.7973802  5.8375563  6.0236
```

```
528
##  [31]  6.0162083  6.6225338  7.1870765  6.3393258  5.3371935  5.6646
483
##  [37]  6.3565682  6.2123924  6.2712804  5.6830319  5.2577176  5.1811
489
##  [43]  4.6222630  3.9317396  3.6456021  3.8665882  3.5838492  3.3585
479
##  [49]  5.0598277  9.0948844 11.6137819 11.9773613 11.9914738 11.7074
366
##  [55] 11.9973054 12.2063704  9.8312195  5.2587787  2.8461259  2.4844
719
##  [61]  2.0460697  1.8977327  1.4067664  0.9985561  1.1422096  1.3561
512
##  [67]  1.2865196  1.1083075  1.1824506  1.3310059  1.4878340  1.7284
698
##  [73]  1.8178385  1.6421472  1.4769213  1.4809235  1.8769393  2.2244
029
##  [79]  2.2916562  2.1777995  1.9925543  1.9948278  1.9378814  1.8236
668
##  [85]  1.3730325  0.8990701  0.7521747  0.6799232  0.6747660  0.7442
144
##  [91]  0.8390481  0.9127938  0.9284668  0.9061261  1.0058706  1.1641
189
##  [97]  1.1829857  1.0902904  1.2262218  1.3764281  1.5550109  1.9015
840
## [103]  3.0722748  4.1939073  4.4790806  4.6464907  4.4511777  4.2959
352
## [109]  4.1594329  4.0878306  3.1406174  2.0543490  2.0730928  2.1918
297
## [115]  2.2656342  2.4114962  2.6053621  2.5132788  2.2383065  2.1112
089
## [121]  1.8690284  1.5641922  1.5307766  1.5185077  1.3525303  1.2093
037
## [127]  1.2186609  1.1528720  1.1218806  1.2386514  1.2245317  1.0079
247
## [133]  0.9213242  1.0334144  1.1132609  1.2691796  1.3526824  1.5262
473
## [139]  1.7268640  1.7259787  1.7169653  1.6386816  1.5486119  1.4598
666
## [145]  1.3113832  1.1289271  1.0056550  1.0868979  1.0974177  0.9703
477
## [151]  1.0074372  1.0423840  1.0533460  0.9651490  0.8480325  0.8867
830
## [157]  0.8867912  0.8708807  0.7223604  0.5609828  0.5387092  0.5506
609
## [163]  0.5596977  0.4868711  0.5450492  0.8677397  1.1225085  1.2229
856
## [169]  1.4566900  1.8521784  2.4977409  2.9318335  2.9130943  2.7065
474
## [175]  2.4595701  2.3789910  2.1906136  1.7872251  1.1351416  0.7449
```

```
723
## [181]  0.7706477  0.7394935  0.7564253  0.8348201  0.8060831  0.8350
103
## [187]  0.9828717
##
## $coh
## NULL
##
## $phase
## NULL
##
## $kernel
## mDaniell(4)
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "co.ts[1:365]"
##
## $snames
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
```

```
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

```
pg.no2 <- spec.pgram(no2.ts[1:365],spans=9,demean=T,log='no')
```




```
pg.no2

## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
```

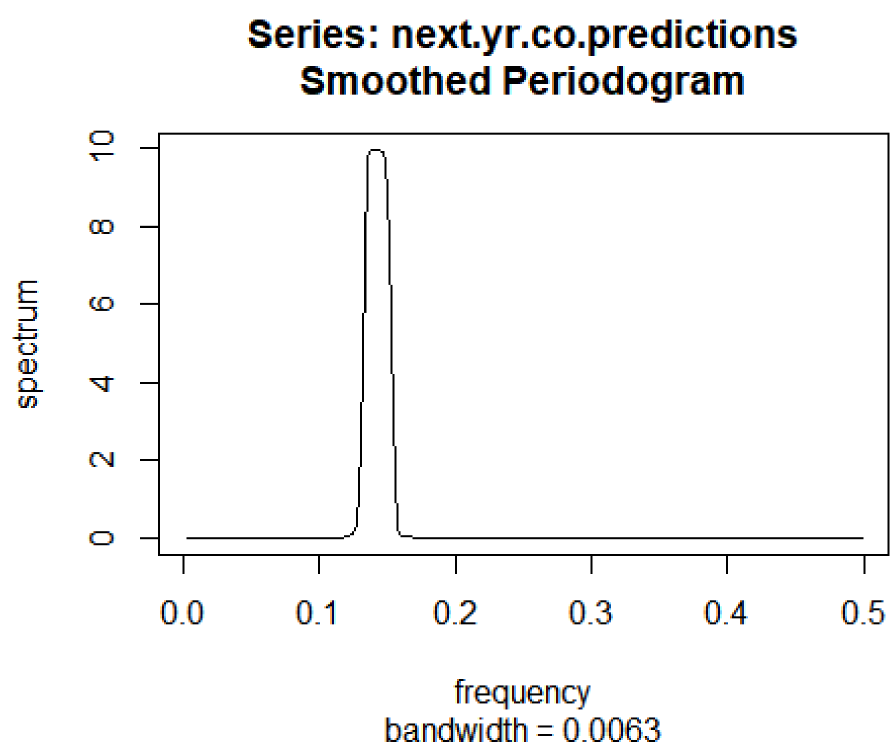
```
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
```

```
##
## $spec
##   [1] 19903.1576 22479.3287 21508.8672 18114.2072 14373.0088 11662.8
224
##   [7] 10375.1663  9821.4436  7889.3427  5311.4449  4476.4084  4088.3
938
##  [13]  3978.1249  4060.2416  4105.6146  4295.3032  4038.1761  2983.2
473
##  [19]  2260.3058  2083.3963  2019.0559  1905.2124  1796.4206  1678.3
685
##  [25]  1516.1425  1401.4690  1536.8090  1725.8089  1621.7385  1636.9
148
##  [31]  1769.1546  2150.7363  2504.2930  2476.1290  2421.7859  2559.1
665
##  [37]  2755.1376  2728.7686  2748.7469  2588.1984  2451.5000  2451.4
297
##  [43]  2111.1585  1623.6166  1431.4683  1608.6107  1604.1687  1224.9
890
##  [49]  1569.9378  2966.0461  3909.9620  4157.2148  4172.9011  4001.8
053
##  [55]  3993.2210  4118.7053  3487.1100  1983.8421  1079.2155   892.1
594
##  [61]   751.5273   685.9692   558.0284   423.3135   342.8464   374.6
862
##  [67]   429.5188   420.4474   497.7056   623.2079   726.4400   810.2
326
##  [73]   814.4986   748.0641   662.9261   623.7611   799.0745   920.2
897
##  [79]   825.8371   790.2810   879.0923   983.8556  1000.6271   993.4
081
##  [85]   875.7054   734.9866   705.9363   719.6938   665.7479   536.4
099
##  [91]   469.3262   491.2547   403.9674   298.0545   374.2355   366.8
016
##  [97]   474.8047   688.7364   777.1448   792.8953   771.4145   804.4
852
## [103]  1346.2238  1905.6850  1851.2339  1730.1742  1724.5037  1832.6
393
## [109]  1939.8136  1963.4788  1424.6377   912.5718   910.6516   866.5
346
## [115]   793.0356   671.4480   666.2691   689.8119   619.1965   532.9
949
## [121]   446.9339   408.0726   399.2104   426.6466   387.1646   322.2
616
## [127]   353.5228   371.8612   378.1545   398.2132   417.6417   383.8
877
## [133]   388.4768   522.5853   632.6330   643.6489   636.5224   782.8
535
## [139]   930.5967   911.0808   857.7769   719.1080   643.0735   682.9
968
```

```
## [145]   697.8642   563.4078   409.9163   435.0813   449.9438   453.4
096
## [151]   490.2808   492.9446   501.5675   535.5106   560.5622   648.0
060
## [157]   740.3719   759.2247   680.5511   590.1319   580.5424   612.5
729
## [163]   646.4193   530.9854   431.1169   452.9863   520.9940   581.3
465
## [169]   537.8220   479.9736   523.8813   610.6501   603.6429   534.8
652
## [175]   436.0115   371.2106   433.0011   464.2171   373.5799   294.1
896
## [181]   281.0479   345.5674   407.9972   425.8582   433.6705   420.2
213
## [187]   419.9945
##
## $coh
## NULL
##
## $phase
## NULL
##
## $kernel
## mDaniell(4)
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "no2.ts[1:365]"
##
## $snames
```

```
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

```
# Univariate Simulations
pg.co.uni <- spec.pgram(next.yr.co.predictions,spans=9,demean=T,log='no
')
```




```
pg.co.uni
```

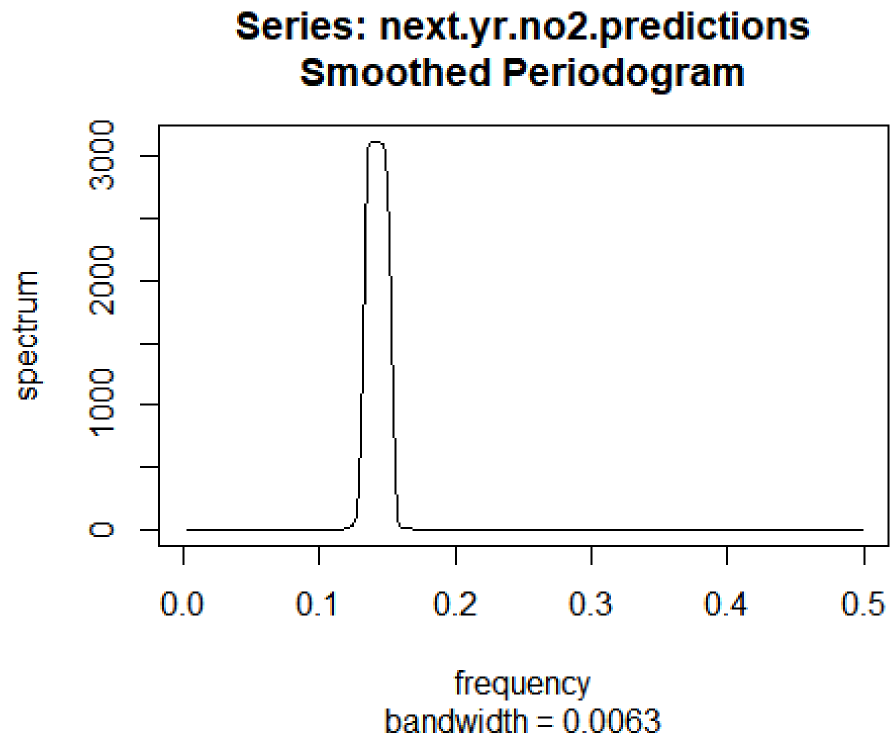
```
## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
```

```
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
##
## $spec
##   [1] 2.818130e-04 2.726003e-04 2.248711e-04 1.441796e-04 6.333352e-
05
##   [6] 1.523651e-05 5.586881e-06 3.052548e-06 2.332548e-06 1.921381e-
06
##  [11] 1.487518e-06 1.105021e-06 8.633852e-07 7.318281e-07 6.537416e-
07
##  [16] 6.817100e-07 9.252496e-07 1.347729e-06 1.720981e-06 1.875190e-
06
##  [21] 1.909304e-06 1.995665e-06 2.050352e-06 1.902792e-06 1.841466e-
06
##  [26] 2.506045e-06 3.856438e-06 4.846469e-06 5.206236e-06 7.284491e-
06
##  [31] 1.402058e-05 2.393710e-05 3.078950e-05 3.163410e-05 3.212360e-
05
##  [36] 3.593910e-05 3.910666e-05 9.127860e-05 4.530248e-04 1.727071e-
03
##  [41] 4.683924e-03 9.589251e-03 1.534145e-02 1.948691e-02 2.081367e-
02
##  [46] 2.696444e-02 7.446148e-02 3.140622e-01 2.324614e+00 6.901041e+
00
##  [51] 9.806166e+00 9.960396e+00 9.973614e+00 9.970106e+00 9.930409e+
00
##  [56] 9.700598e+00 7.696537e+00 3.121032e+00 2.127774e-01 5.519244e-
02
##  [61] 4.076969e-02 3.808517e-02 2.994306e-02 1.889079e-02 9.442323e-
03
##  [66] 3.614185e-03 9.948332e-04 2.039157e-04 8.023183e-05 6.836654e-
05
##  [71] 4.431091e-05 2.631338e-05 2.363603e-05 2.390100e-05 1.972008e-
05
##  [76] 1.468143e-05 1.264484e-05 1.211084e-05 1.016636e-05 6.794775e-
06
##  [81] 4.016918e-06 2.924303e-06 2.913649e-06 3.053128e-06 3.072944e-
06
```

```
##  [86] 3.053369e-06 2.888137e-06 2.391466e-06 1.657100e-06 1.016723e-
06
##  [91] 6.858971e-07 6.048733e-07 6.074702e-07 6.111846e-07 6.092439e-
07
##  [96] 5.785652e-07 4.799899e-07 3.252991e-07 1.820559e-07 1.029515e-
07
## [101] 8.076959e-08 7.906852e-08 7.976747e-08 8.516611e-08 9.190539e-
08
## [106] 8.657540e-08 6.542567e-08 4.181311e-08 2.929038e-08 2.673201e-
08
## [111] 2.519359e-08 2.231668e-08 2.286949e-08 2.844253e-08 3.405267e-
08
## [116] 3.540897e-08 3.406868e-08 3.327179e-08 3.245479e-08 2.901153e-
08
## [121] 2.348227e-08 1.961952e-08 1.956813e-08 2.159455e-08 2.304553e-
08
## [126] 2.339960e-08 2.330768e-08 2.229148e-08 1.922782e-08 1.460227e-
08
## [131] 1.062747e-08 8.860486e-09 8.770443e-09 9.000833e-09 9.068865e-
09
## [136] 9.151643e-09 8.929284e-09 7.686763e-09 5.484444e-09 3.388869e-
09
## [141] 2.295342e-09 2.058655e-09 2.056772e-09 2.123000e-09 2.461707e-
09
## [146] 2.939653e-09 3.049064e-09 2.602223e-09 2.011276e-09 1.715440e-
09
## [151] 1.661562e-09 1.578944e-09 1.508084e-09 1.724705e-09 2.238940e-
09
## [156] 2.705258e-09 2.871001e-09 2.846042e-09 2.820122e-09 2.743714e-
09
## [161] 2.484438e-09 2.157157e-09 2.052960e-09 2.258782e-09 2.552761e-
09
## [166] 2.712441e-09 2.744803e-09 2.728656e-09 2.599138e-09 2.256253e-
09
## [171] 1.805751e-09 1.505793e-09 1.467315e-09 1.552470e-09 1.600861e-
09
## [176] 1.613664e-09 1.636602e-09 1.589035e-09 1.354852e-09 9.928800e-
10
## [181] 7.089161e-10 6.103249e-10 6.126513e-10 6.205968e-10 6.727936e-
10
## [186] 8.224735e-10 9.747731e-10
##
## $coh
## NULL
##
## $phase
## NULL
##
## $kernel
## mDaniell(4)
```

```
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "next.yr.co.predictions"
##
## $snames
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

```
pg.no2.uni <- spec.pgram(next.yr.no2.predictions,spans=9,demean=T,log='
no')
```




```
pg.no2.uni

## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
```

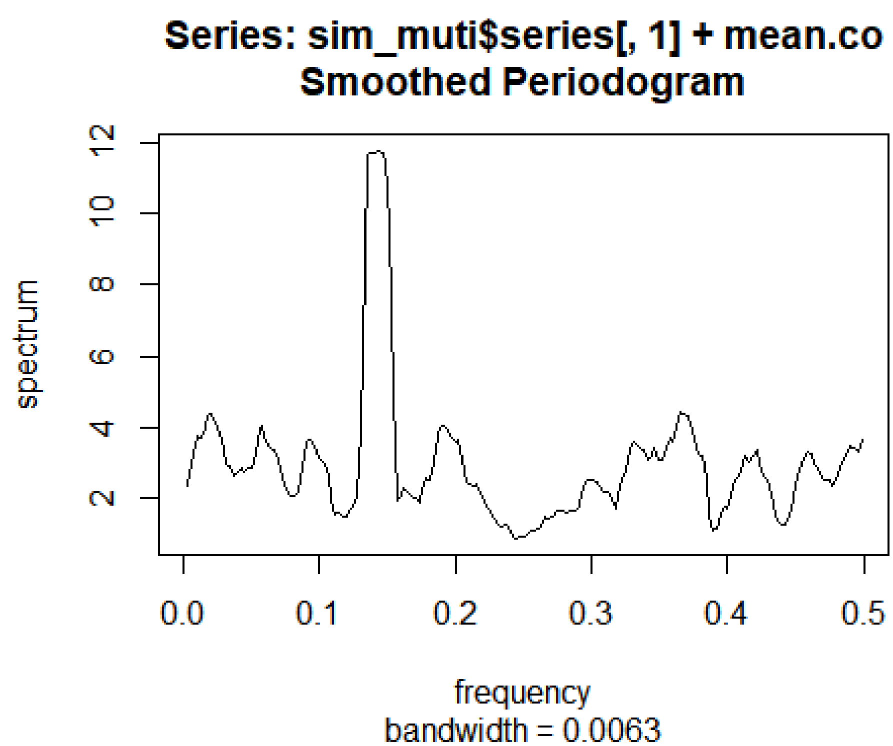
```
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
##
## $spec
##   [1] 1.405408e-01 1.363125e-01 1.124281e-01 7.159576e-02 3.068557e-
02
##   [6] 6.623622e-03 2.217749e-03 1.232648e-03 9.982823e-04 8.270427e-
04
##  [11] 6.106518e-04 4.135117e-04 2.899017e-04 2.303556e-04 2.038549e-
04
##  [16] 2.151178e-04 2.937807e-04 4.279697e-04 5.462591e-04 5.951899e-
04
##  [21] 6.059004e-04 6.327271e-04 6.493343e-04 6.022345e-04 5.824240e-
```

```
04
##  [26] 7.910522e-04 1.215306e-03 1.526396e-03 1.639301e-03 2.290153e-
03
##  [31] 4.399505e-03 7.503376e-03 9.645516e-03 9.906029e-03 1.005783e-
02
##  [36] 1.125340e-02 1.224592e-02 2.858626e-02 1.418819e-01 5.408863e-
01
##  [41] 1.466881e+00 3.003042e+00 4.804385e+00 6.102558e+00 6.518036e+
00
##  [46] 8.444206e+00 2.331835e+01 9.835188e+01 7.279806e+02 2.161147e+
03
##  [51] 3.070924e+03 3.119223e+03 3.123363e+03 3.122264e+03 3.109833e+
03
##  [56] 3.037865e+03 2.410269e+03 9.773914e+02 6.663368e+01 1.728387e+
01
##  [61] 1.276720e+01 1.192651e+01 9.376747e+00 5.915664e+00 2.956852e+
00
##  [66] 1.131771e+00 3.115275e-01 6.385524e-02 2.512431e-02 2.140850e-
02
##  [71] 1.387449e-02 8.236917e-03 7.397561e-03 7.481239e-03 6.173652e-
03
##  [76] 4.597079e-03 3.959756e-03 3.792590e-03 3.183757e-03 2.127910e-
03
##  [81] 1.257672e-03 9.150375e-04 9.114182e-04 9.550697e-04 9.613200e-
04
##  [86] 9.552096e-04 9.035457e-04 7.482022e-04 5.184185e-04 3.179243e-
04
##  [91] 2.142377e-04 1.887688e-04 1.895439e-04 1.907029e-04 1.901086e-
04
##  [96] 1.805695e-04 1.498423e-04 1.015601e-04 5.680231e-05 3.205411e-
05
## [101] 2.510000e-05 2.456274e-05 2.477715e-05 2.646039e-05 2.858363e-
05
## [106] 2.696652e-05 2.041542e-05 1.307437e-05 9.172773e-06 8.373957e-
06
## [111] 7.891075e-06 6.980806e-06 7.131670e-06 8.855095e-06 1.060808e-
05
## [116] 1.104553e-05 1.063857e-05 1.039315e-05 1.013894e-05 9.064613e-
06
## [121] 7.332137e-06 6.108799e-06 6.071749e-06 6.691165e-06 7.140858e-
06
## [126] 7.251770e-06 7.223862e-06 6.911038e-06 5.963736e-06 4.527117e-
06
## [131] 3.285154e-06 2.725686e-06 2.690497e-06 2.759823e-06 2.780772e-
06
## [136] 2.807390e-06 2.742961e-06 2.366201e-06 1.691516e-06 1.045160e-
06
## [141] 7.052550e-07 6.304908e-07 6.294781e-07 6.489952e-07 7.521474e-
07
## [146] 9.005158e-07 9.388700e-07 8.069295e-07 6.284153e-07 5.382177e-
```

```
07
## [151] 5.216245e-07 4.956113e-07 4.712436e-07 5.332886e-07 6.874074e-
07
## [156] 8.297042e-07 8.819095e-07 8.754966e-07 8.678785e-07 8.446431e-
07
## [161] 7.651827e-07 6.632169e-07 6.272391e-07 6.855484e-07 7.725287e-
07
## [166] 8.205020e-07 8.303674e-07 8.256325e-07 7.871316e-07 6.844388e-
07
## [171] 5.484224e-07 4.564975e-07 4.429865e-07 4.674773e-07 4.817560e-
07
## [176] 4.855575e-07 4.925087e-07 4.788995e-07 4.100379e-07 3.029347e-
07
## [181] 2.185948e-07 1.891889e-07 1.898483e-07 1.921630e-07 2.074234e-
07
## [186] 2.514823e-07 2.964108e-07
##
## $coh
## NULL
##
## $phase
## NULL
##
## $kernel
## mDaniell(4)
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "next.yr.no2.predictions"
##
```

```
## $snames
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

```
# Multivariate Simulations
pg.co.uni <- spec.pgram(sim_muti$series[,1]+mean.co,spans=9,demean=T,lo
g='no')
```




```
pg.co.uni
```

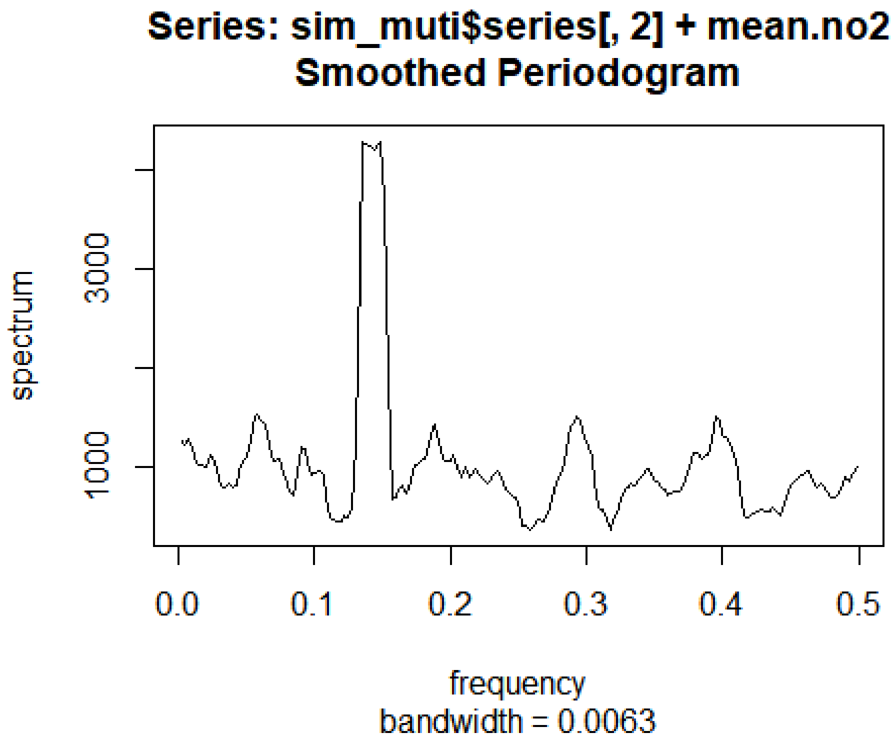
```
## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
```

```
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
##
## $spec
##   [1]  2.3497142  2.6686524  3.2532144  3.7377650  3.6838294  4.0054
460
##   [7]  4.2980919  4.3579136  4.1512522  3.8276852  3.3920844  2.9563
298
##  [13]  2.9004899  2.6365111  2.7480454  2.8383318  2.7158684  2.8357
518
##  [19]  2.8675027  3.2624106  3.9210800  4.0335631  3.5969604  3.4391
926
##  [25]  3.3804713  3.0821177  2.7769126  2.3955547  2.1118963  2.0229
191
##  [31]  2.0729369  2.1515549  2.8423114  3.5656726  3.6616470  3.6095
851
##  [37]  3.2891116  3.0810304  2.9476620  2.6342939  1.9751113  1.5536
762
##  [43]  1.6042532  1.4554564  1.4613878  1.6317150  1.7727543  2.0572
319
##  [49]  3.8623497  8.5759242 11.6641695 11.7178073 11.6920630 11.7798
722
##  [55] 11.7325668 11.4727192  9.6997563  4.9511218  1.9125010  2.0714
511
##  [61]  2.2571295  2.1509352  2.0689659  2.0146812  1.8705265  2.2207
113
##  [67]  2.5486717  2.4740964  3.0410593  3.7718320  4.0012388  4.0200
363
##  [73]  3.9095407  3.7060979  3.6000731  3.6573797  3.0623587  2.4336
362
##  [79]  2.3777895  2.3345589  2.3839419  2.1688632  1.8878628  1.6978
068
##  [85]  1.5585331  1.4466112  1.2318770  1.1944517  1.2289599  1.0671
343
##  [91]  0.9079714  0.8574722  0.8867484  0.8854825  0.9425912  1.0836
912
##  [97]  1.0688457  1.1290639  1.3023574  1.5043440  1.4089171  1.4595
027
```

```
## [103]  1.6582524  1.6410028  1.6200229  1.5925108  1.6218121  1.6638
115
## [109]  1.7829714  2.1167424  2.4319315  2.5007033  2.4900263  2.4435
564
## [115]  2.3228423  2.1497918  2.1839374  2.0128929  1.6870086  2.0849
980
## [121]  2.5289828  2.8005361  3.3636702  3.5883050  3.5458508  3.4004
219
## [127]  3.3718614  3.0733378  3.1865603  3.4386555  3.0619551  3.0635
887
## [133]  3.3878400  3.6933381  3.5613856  3.9859799  4.4150634  4.3866
861
## [139]  4.3099751  4.0884021  3.7082690  3.2647200  3.1986806  2.5424
172
## [145]  1.5583294  1.0693568  1.1481298  1.5475610  1.7356464  1.7217
233
## [151]  2.0631719  2.4787268  2.6008060  2.9740749  3.1985827  3.0392
612
## [157]  3.1565945  3.3787616  2.9704270  2.5895337  2.5073075  2.0791
443
## [163]  1.5491707  1.3373787  1.2285617  1.2252731  1.4552118  1.9610
040
## [169]  2.3883989  2.7608002  3.1503592  3.3221342  3.2628937  2.9543
102
## [175]  2.8506998  2.5849840  2.4886149  2.4897946  2.3186771  2.5231
341
## [181]  2.8258950  3.0571368  3.2810185  3.4723326  3.4115141  3.2700
263
## [187]  3.6592810
##
## $coh
## NULL
##
## $phase
## NULL
##
## $kernel
## mDaniell(4)
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
```

```
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "sim_muti$series[, 1] + mean.co"
##
## $snames
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

```
pg.no2.uni <- spec.pgram(sim_muti$series[,2]+mean.no2,spans=9,demean=T,
log='no')
```




```
pg.no2.uni

## $freq
##   [1] 0.002666667 0.005333333 0.008000000 0.010666667 0.013333333 0.
016000000
##   [7] 0.018666667 0.021333333 0.024000000 0.026666667 0.029333333 0.
032000000
##  [13] 0.034666667 0.037333333 0.040000000 0.042666667 0.045333333 0.
048000000
##  [19] 0.050666667 0.053333333 0.056000000 0.058666667 0.061333333 0.
064000000
##  [25] 0.066666667 0.069333333 0.072000000 0.074666667 0.077333333 0.
080000000
##  [31] 0.082666667 0.085333333 0.088000000 0.090666667 0.093333333 0.
096000000
##  [37] 0.098666667 0.101333333 0.104000000 0.106666667 0.109333333 0.
112000000
##  [43] 0.114666667 0.117333333 0.120000000 0.122666667 0.125333333 0.
128000000
##  [49] 0.130666667 0.133333333 0.136000000 0.138666667 0.141333333 0.
144000000
##  [55] 0.146666667 0.149333333 0.152000000 0.154666667 0.157333333 0.
160000000
##  [61] 0.162666667 0.165333333 0.168000000 0.170666667 0.173333333 0.
176000000
##  [67] 0.178666667 0.181333333 0.184000000 0.186666667 0.189333333 0.
192000000
```

```
##  [73] 0.194666667 0.197333333 0.200000000 0.202666667 0.205333333 0.
208000000
##  [79] 0.210666667 0.213333333 0.216000000 0.218666667 0.221333333 0.
224000000
##  [85] 0.226666667 0.229333333 0.232000000 0.234666667 0.237333333 0.
240000000
##  [91] 0.242666667 0.245333333 0.248000000 0.250666667 0.253333333 0.
256000000
##  [97] 0.258666667 0.261333333 0.264000000 0.266666667 0.269333333 0.
272000000
## [103] 0.274666667 0.277333333 0.280000000 0.282666667 0.285333333 0.
288000000
## [109] 0.290666667 0.293333333 0.296000000 0.298666667 0.301333333 0.
304000000
## [115] 0.306666667 0.309333333 0.312000000 0.314666667 0.317333333 0.
320000000
## [121] 0.322666667 0.325333333 0.328000000 0.330666667 0.333333333 0.
336000000
## [127] 0.338666667 0.341333333 0.344000000 0.346666667 0.349333333 0.
352000000
## [133] 0.354666667 0.357333333 0.360000000 0.362666667 0.365333333 0.
368000000
## [139] 0.370666667 0.373333333 0.376000000 0.378666667 0.381333333 0.
384000000
## [145] 0.386666667 0.389333333 0.392000000 0.394666667 0.397333333 0.
400000000
## [151] 0.402666667 0.405333333 0.408000000 0.410666667 0.413333333 0.
416000000
## [157] 0.418666667 0.421333333 0.424000000 0.426666667 0.429333333 0.
432000000
## [163] 0.434666667 0.437333333 0.440000000 0.442666667 0.445333333 0.
448000000
## [169] 0.450666667 0.453333333 0.456000000 0.458666667 0.461333333 0.
464000000
## [175] 0.466666667 0.469333333 0.472000000 0.474666667 0.477333333 0.
480000000
## [181] 0.482666667 0.485333333 0.488000000 0.490666667 0.493333333 0.
496000000
## [187] 0.498666667
##
## $spec
##   [1] 1252.4414 1207.4102 1278.9467 1187.0762 1022.7562 1016.9597 10
18.6029
##   [8]  993.5793 1108.4952 1043.1165  876.1073  796.4737  792.3247  8
29.3095
##  [15]  794.9914  813.5872  941.9328 1049.9491 1095.5833 1253.8665 15
08.3768
##  [22] 1514.8629 1466.2623 1414.5954 1220.7050 1051.8545 1059.9177 10
75.7207
##  [29]  938.0577  817.9112  752.3161  712.0262  927.7580 1196.6357 11
```

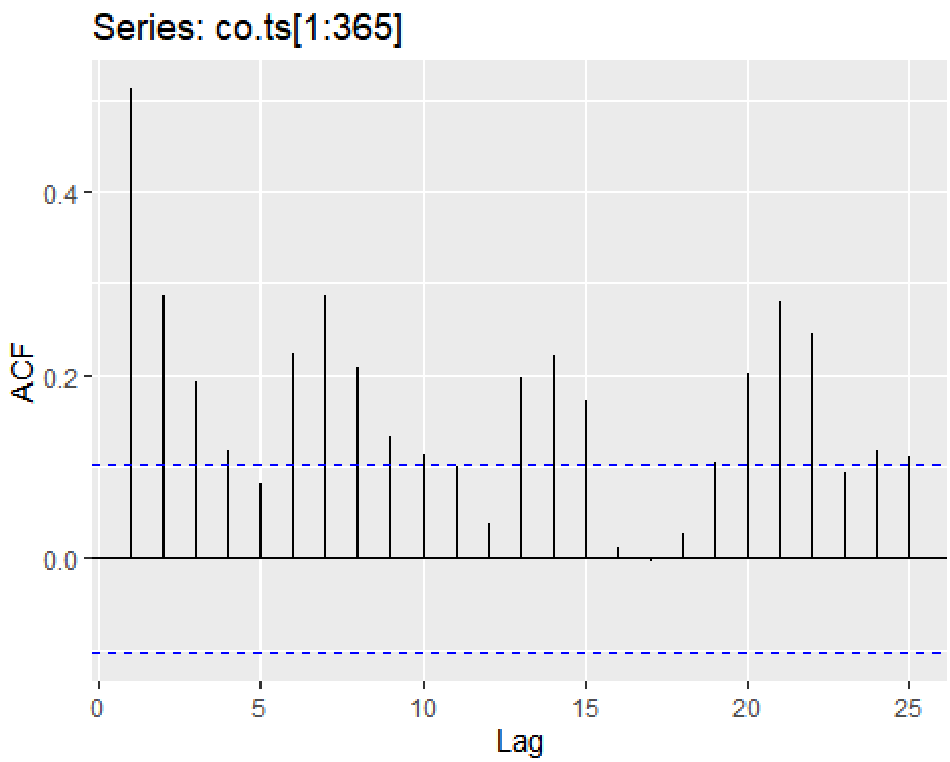
```
79.9504
##  [36] 1028.8673  913.1140  921.4283  948.7140  907.5519  681.8434  4
70.3617
##  [43]  458.4264  430.8307  447.6990  497.0123  478.6394  551.7247 12
36.6366
##  [50] 3072.6133 4283.9814 4271.9387 4249.7561 4200.0236 4255.4728 42
85.7512
##  [57] 3684.3372 1888.0147  659.8148  685.8307  759.9325  806.7417  7
29.4306
##  [64]  846.1670 1003.9378 1001.0572 1053.5711 1072.4277 1215.3386 13
87.9196
##  [71] 1409.8622 1263.9293 1067.3500 1053.5780 1054.9304 1118.5652  9
97.4305
##  [78]  878.6337  983.5039  887.2599  904.8078  964.3534  905.3777  8
67.5502
##  [85]  828.6684  852.0674  903.2838  945.1979  878.3697  771.9798  7
40.5995
##  [92]  701.8685  688.9976  560.8338  403.5983  388.5667  361.3656  3
89.0861
##  [99]  467.2402  452.0705  436.5406  512.5776  666.0353  818.1830  8
95.6164
## [106]  958.7523 1157.3160 1399.1895 1441.9608 1496.9051 1459.4512 12
90.0128
## [113] 1201.8902 1093.2690  847.0716  586.0619  554.1662  457.8962  3
57.9431
## [120]  449.0960  520.3314  655.9346  760.9166  781.6401  817.1004  8
15.0729
## [127]  875.3930  903.2561  976.4144  968.0877  868.9839  854.9768  7
93.8912
## [134]  762.6898  701.8483  724.8129  753.6306  734.5333  810.0170  8
65.5339
## [141]  995.0664 1138.1196 1131.7528 1080.9429 1087.4376 1107.7795 12
56.2536
## [148] 1496.5105 1453.8769 1319.3355 1300.2957 1244.5339 1102.6580  9
74.8315
## [155]  762.1579  508.8771  486.4098  526.4892  512.6502  539.8754  5
54.5627
## [162]  547.7829  541.6518  578.6516  540.9459  503.5870  615.9758  7
38.5291
## [169]  797.4536  848.7852  887.9587  910.4614  950.9762  939.2861  8
51.0859
## [176]  794.2733  829.7005  778.2815  750.0531  683.9954  673.8838  7
30.3031
## [183]  805.5230  887.8444  839.8326  925.3854  979.9456
##
## $coh
## NULL
##
## $phase
## NULL
```

```
##
## $kernel
## mDaniell(4)
## coef[-4] = 0.0625
## coef[-3] = 0.1250
## coef[-2] = 0.1250
## coef[-1] = 0.1250
## coef[ 0] = 0.1250
## coef[ 1] = 0.1250
## coef[ 2] = 0.1250
## coef[ 3] = 0.1250
## coef[ 4] = 0.0625
##
## $df
## [1] 14.88055
##
## $bandwidth
## [1] 0.006301088
##
## $n.used
## [1] 375
##
## $orig.n
## [1] 365
##
## $series
## [1] "sim_muti$series[, 2] + mean.no2"
##
## $snames
## NULL
##
## $method
## [1] "Smoothed Periodogram"
##
## $taper
## [1] 0.1
##
## $pad
## [1] 0
##
## $detrend
## [1] TRUE
##
## $demean
## [1] TRUE
##
## attr(,"class")
## [1] "spec"
```

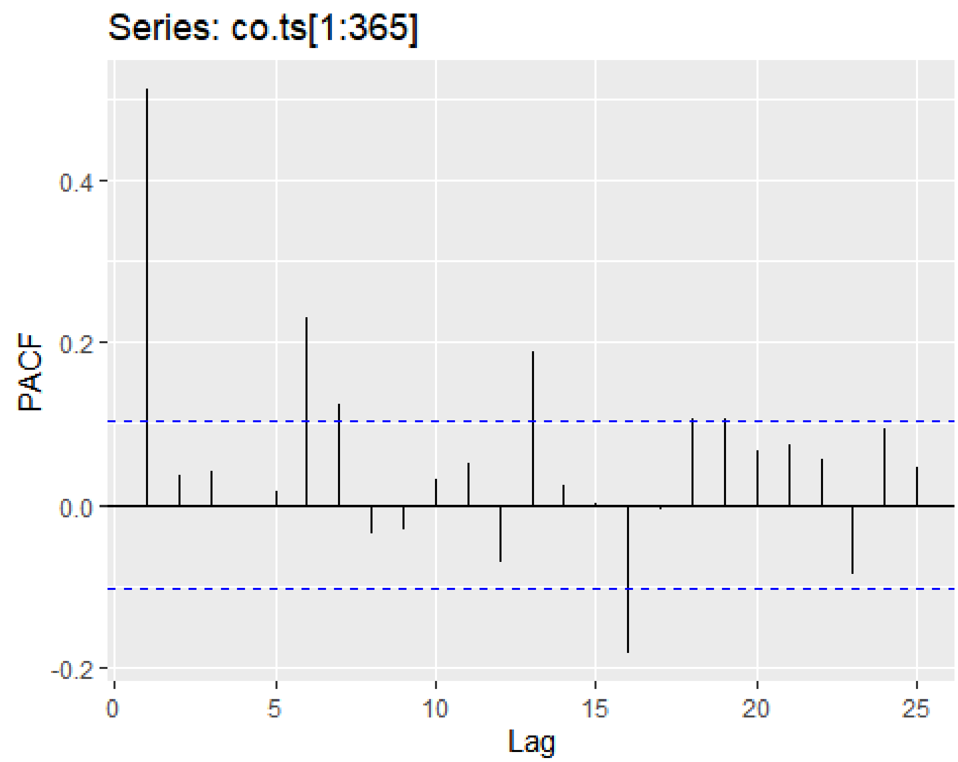
Here again, the univariate models capture the main seasonality (highest peak) of the original data, but the multivariate models also capture some of the smaller peaks that the original data contains. However, the multivariate data seems to contain several additional peaks as well, showing that they may be overestimating some of the seasonality in the data.

#3c)

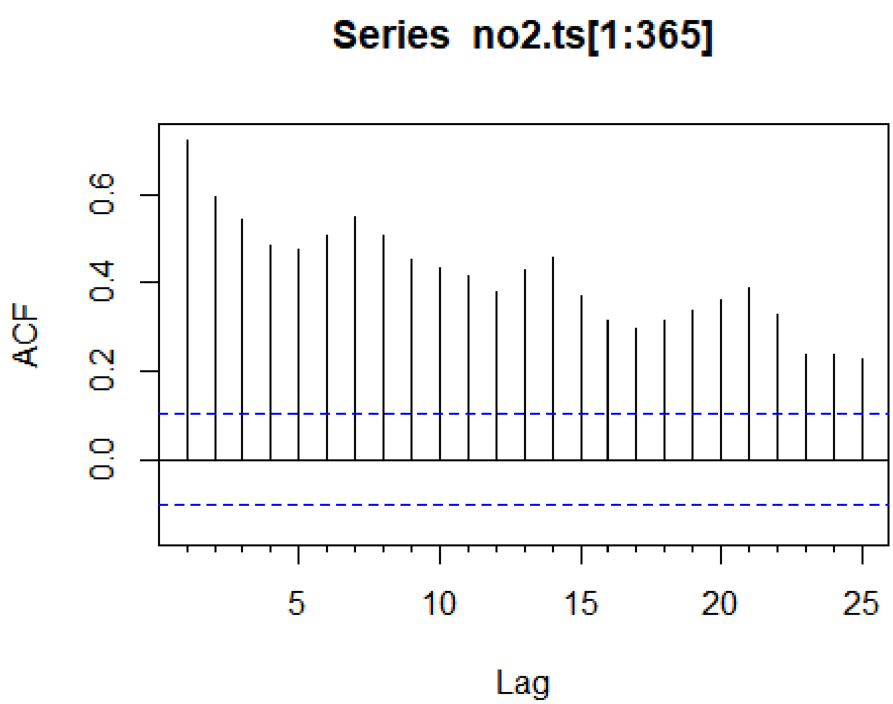
```
# Original CO
co.365.acf <- ggAcf(co.ts[1:365])
co.365.acf
```



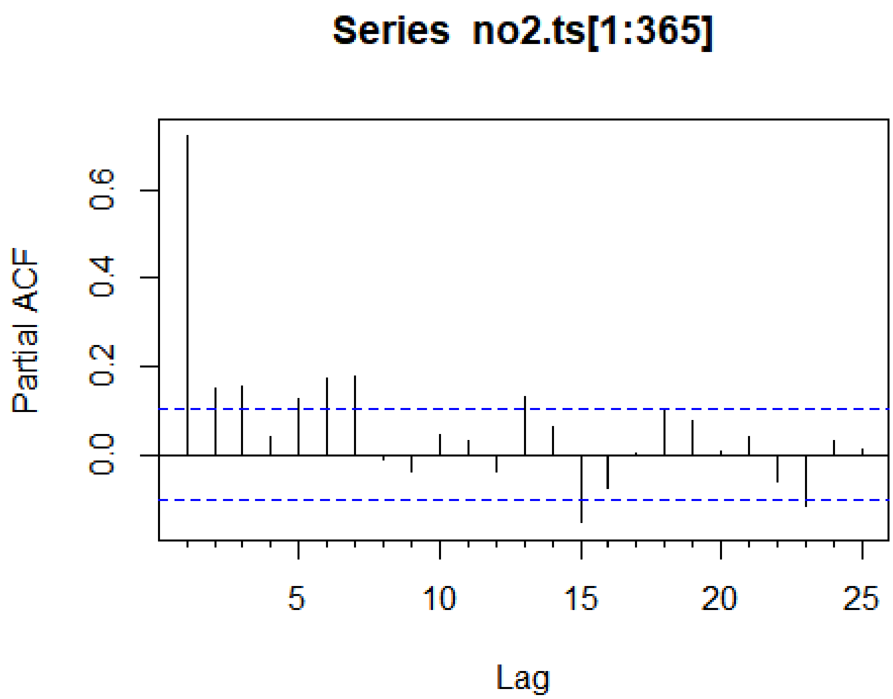

```
co.365.pacf <- ggPacf(co.ts[1:365])
co.365.pacf
```



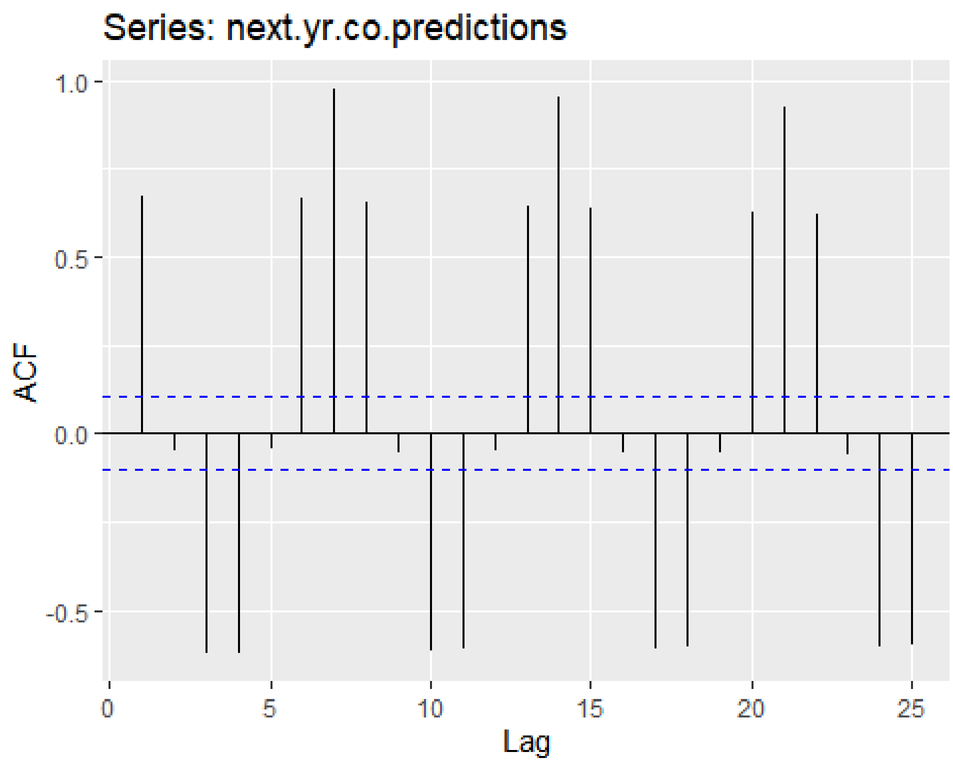

```r
# Original NO2
no2.365.acf <- Acf(no2.ts[1:365])
```



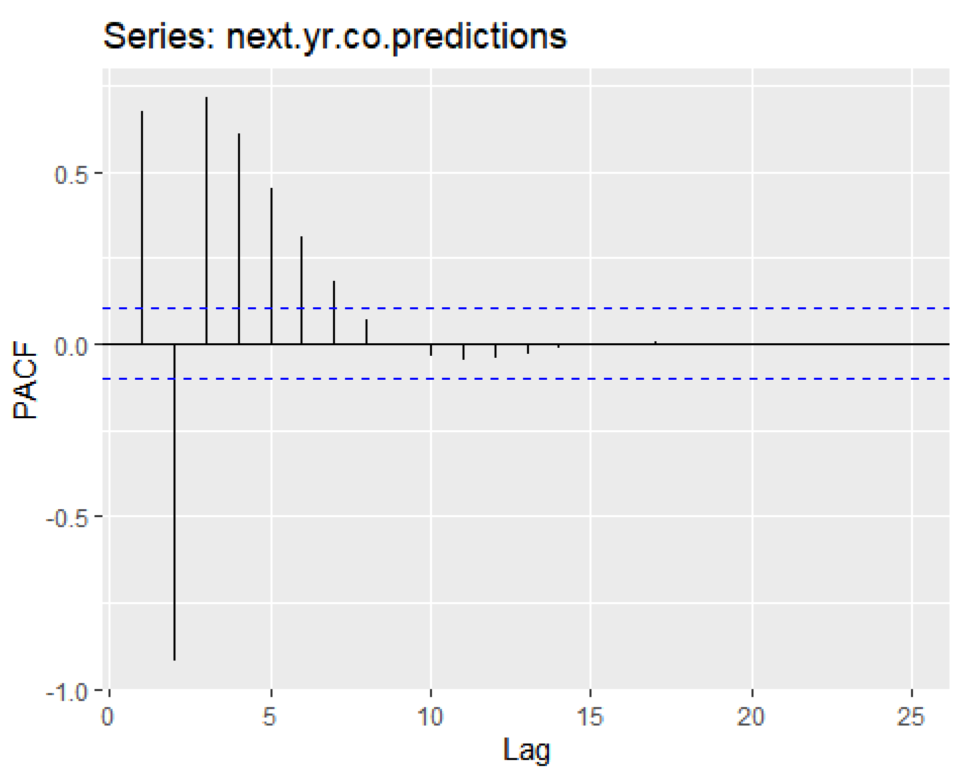
```r
no2.365.pacf <- Pacf(no2.ts[1:365])
```



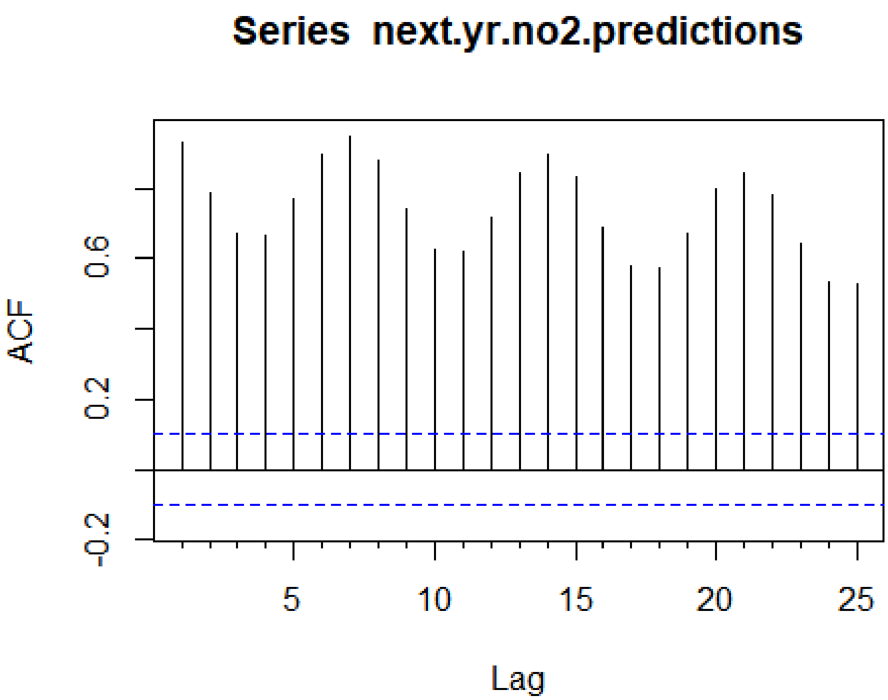

```r
# Univariate CO
co.uni.acf <- ggAcf(next.yr.co.predictions)
co.uni.acf
```



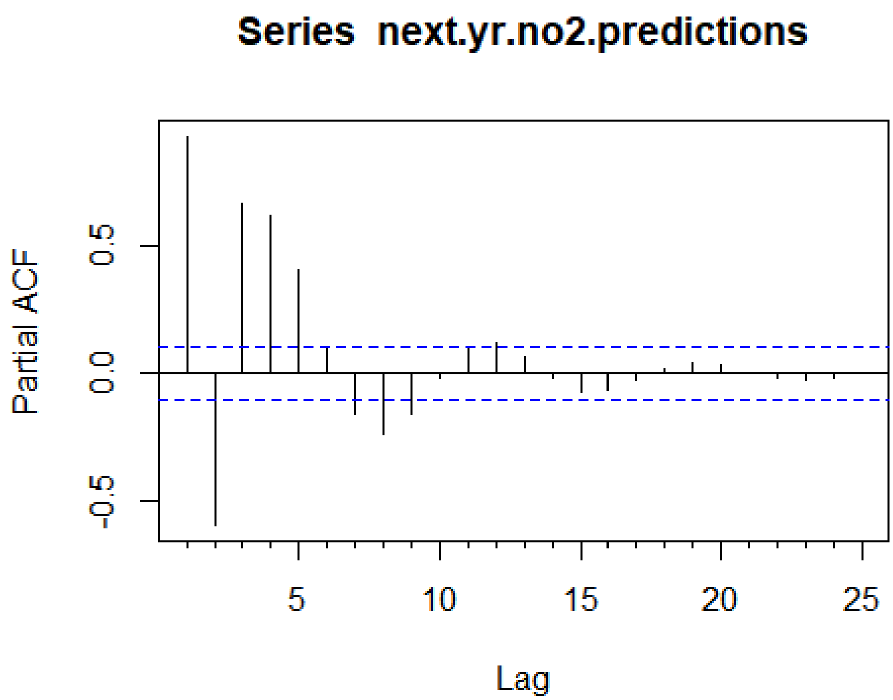

```r
co.uni.pacf <- ggPacf(next.yr.co.predictions)
co.uni.pacf
```



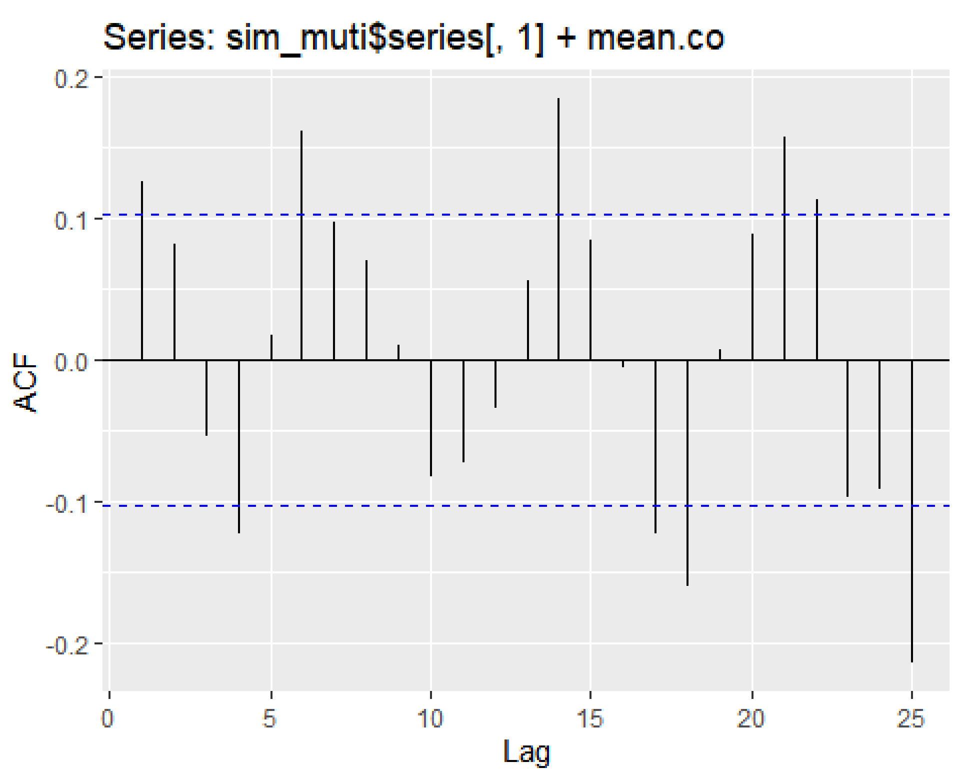
```r
# Univariate NO2
no2.uni.acf <- Acf(next.yr.no2.predictions)
```



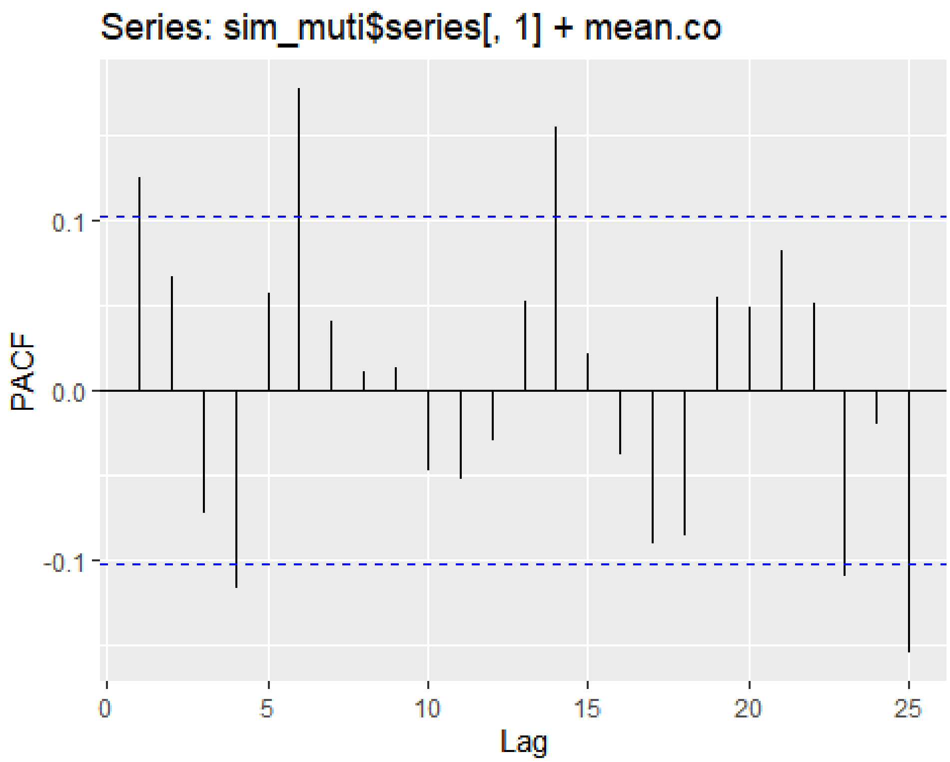

```r
no2.uni.pacf <- Pacf(next.yr.no2.predictions)
```




```r
# Multivariate CO
co.multi.acf <- ggAcf(sim_muti$series[,1]+mean.co)
co.multi.acf
```

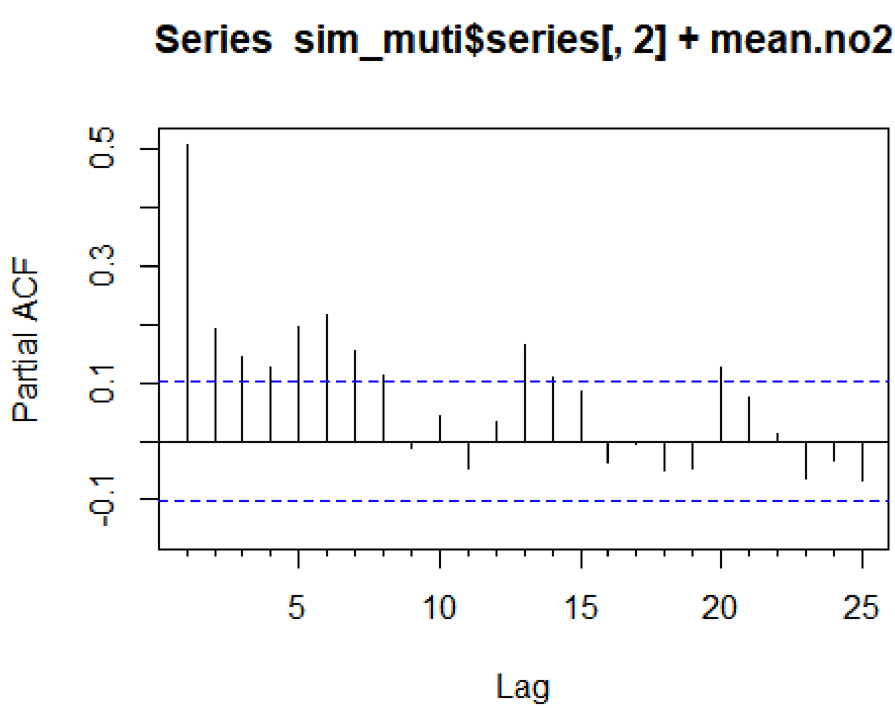


```
co.multi.pacf <- ggPacf(sim_muti$series[,1]+mean.co)
co.multi.pacf
```




```
# Multivariate NO2
no2.multi.acf <- Acf(sim_muti$series[,2]+mean.no2)
```


```
no2.multi.pacf <- Pacf(sim_muti$series[,2]+mean.no2)
```



Neither the univariate nor multivariate CO model captures the ACF or PACF plots of the original data. Both NO2 models seem to do a decent job getting the overall shape of the plots, but the multivariate model better captures the magnitude of the correlations.

#3d)

```r
cor(co.ts[1:365],no2.ts[1:365]) # 0.605

## [1] 0.6053903

cor(next.yr.co.predictions, next.yr.no2.predictions) # 0.721

## [1] 0.7212431

cor(sim_muti$series[,1]+mean.co, sim_muti$series[,2]+mean.no2) # 0.616

## [1] 0.6159244
```

The multivariate model much more closely resembles the cross correlation value of the original data when compared to the univariate data.