Project 2

Cecilia Smith, Sofia Zajec, TJ Gwilliam, Reese Quillian

December 8, 2022

# Load data and impute missing values

setwd(datadir)  
  
airquality = read.csv('AirQualityUCI.csv')  
  
# replace -200 with NA  
airquality[airquality == -200] <- NA  
  
# convert integer type to numeric  
intcols = c(4,5,7,8,9,10,11,12)  
for(i in 1:length(intcols)){  
 airquality[,intcols[i]] <- as.numeric(airquality[,intcols[i]])  
}  
  
setwd(sourcedir)  
  
# create new data frame with just CO and NO2  
AQdata = airquality[,c(3,10)]  
  
# impute missing air quality data  
f <- ~ CO.GT. + NO2.GT.  
t <- c(seq(1,dim(AQdata)[1],1))  
i <- mnimput(f, AQdata, eps=1e-3, ts=TRUE, method='gam', ga.control=list(formula=paste(names(AQdata)[c(1:2)],'~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)

# Part 1: Building Univariate Time Series Models

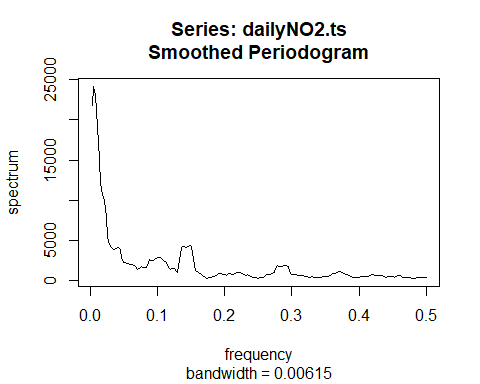
Build a time series model of daily maximum nitrogen dioxide (NO2) concentrations using all but the last 7 days of observations

# use dataframe dailyAQ  
dailyNO2 <- dailyAQ$NO2.GT.  
  
dailyNO2.ts <- ts(dailyNO2)  
  
# remove last 7 days  
dailyNO2.ts <- (dailyNO2.ts[1:(length(dailyNO2.ts)-7)])  
  
# create time element  
time.NO2<-c(1:(length(dailyNO2.ts)))

#### 1.1 Seasonal components

To begin, we check for seasonal components using a periodogram:

pg.NO2 <- spec.pgram(dailyNO2.ts,spans=9,demean=T,log='no')



# Find the peak, max.omega.NO2  
max.omega.NO2<-pg.NO2$freq[which(pg.NO2$spec==max(pg.NO2$spec))]  
  
# Where is the peak?  
max.omega.NO2

## [1] 0.005208333

# What is the period?  
1/max.omega.NO2

## [1] 192

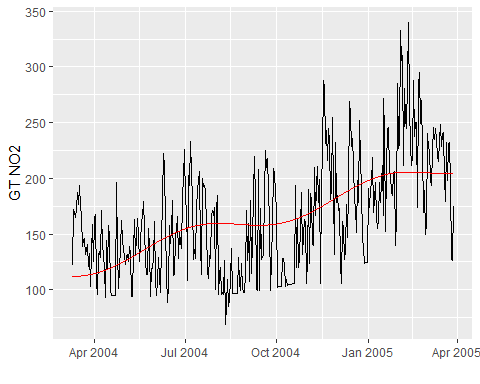
In order to determine if there is any seasonality, we created a periodogram of daily NO2 values. This yielded a maximum omega value of 0.0052, corresponding to a period of 192 days. This period can further be represented as approximately 6 and a half months (~192/30). We hypothesize that this could be due to the fact that during the colder months of the year, a greater amount of emissions are released in order to heat homes, thereby releasing a greater amount of NO2 into the atmosphere.

# Model seasonality  
# from before: period = 192  
NO2.trend.seasonal <- lm(dailyNO2.ts ~ time.NO2 + sin(2\*pi\*time.NO2/192) + cos(2\*pi\*time.NO2/192))  
summary(NO2.trend.seasonal)

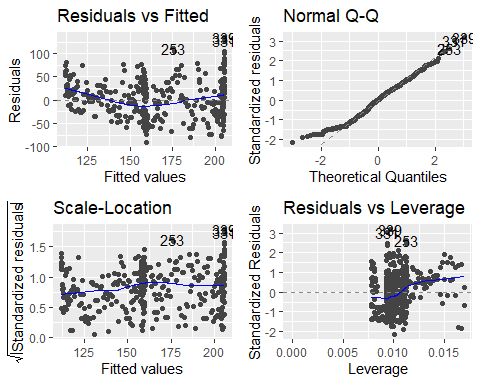
##   
## Call:  
## lm(formula = dailyNO2.ts ~ time.NO2 + sin(2 \* pi \* time.NO2/192) +   
## cos(2 \* pi \* time.NO2/192))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -90.458 -33.603 0.671 29.988 134.304   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 118.28961 4.64945 25.442 <2e-16 \*\*\*  
## time.NO2 0.24023 0.02134 11.258 <2e-16 \*\*\*  
## sin(2 \* pi \* time.NO2/192) -6.63400 3.34509 -1.983 0.0481 \*   
## cos(2 \* pi \* time.NO2/192) -6.46088 3.08054 -2.097 0.0366 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 42.68 on 380 degrees of freedom  
## Multiple R-squared: 0.3198, Adjusted R-squared: 0.3145   
## F-statistic: 59.57 on 3 and 380 DF, p-value: < 2.2e-16

We then created a model to include the trend and seasonality which determined that both the trend and ~6 month seasonality are significant. We can then plot the trend/seasonal model onto the data in order to get a better visualization.

dailyAQ1 <- head(dailyAQ, -7)   
  
# Plot seasonal model  
ggplot(dailyAQ1, aes(x=Group.1,y=NO2.GT.)) + geom\_line() +   
 geom\_line(aes(x=Group.1,y=NO2.trend.seasonal$fitted.values),color="red") +  
 xlab("") + ylab("GT NO2")

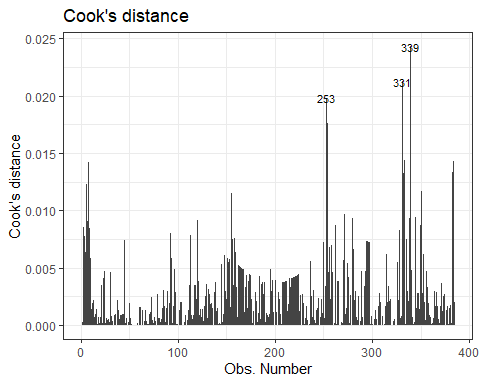


# Model diagnostics for NO2.trend.seasonal  
autoplot(NO2.trend.seasonal, labels.id = NULL)



We then checked the diagnostic plots to make sure that the underlying assumptions were being met. As seen above, the residual versus fitted and residuals versus leverage could definitely be improved (the variance does not appear constant and there is slight indication of nonzero mean), however the normality and scale-location seem to be acceptable.

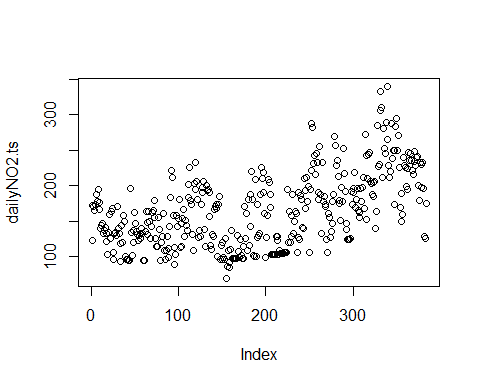
# check to make sure no points have cooks distance higher than 0.5  
autoplot(NO2.trend.seasonal,which=4,ncol=1,label.size=3) + theme\_bw()



Lastly, we considered Cooks distance to ensure that no influential points were present and skewing or influencing our data. As none of the data points had a cooks distance greater than 0.5, there proved no reason to remove any of the data points.

#### 1.2 Trend

plot(dailyNO2.ts)



To determine any trend within the data, we plotted the data as a whole to begin with a simple visualization. Based on the plot above, there appears to be a general increasing trend.

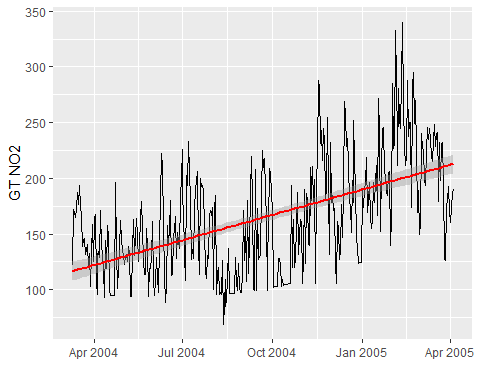
NO2.trend<-lm(dailyNO2.ts ~ time.NO2)  
summary(NO2.trend)

##   
## Call:  
## lm(formula = dailyNO2.ts ~ time.NO2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -87.389 -34.365 2.159 27.847 137.895   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 115.16473 4.40111 26.17 <2e-16 \*\*\*  
## time.NO2 0.25646 0.01981 12.94 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 43.04 on 382 degrees of freedom  
## Multiple R-squared: 0.3049, Adjusted R-squared: 0.3031   
## F-statistic: 167.6 on 1 and 382 DF, p-value: < 2.2e-16

After the initial visualization plot, we created the trend model to determine if the trend was significant. As seen in the output above, the trend is significant with a very low P-value of <2.2e-16.

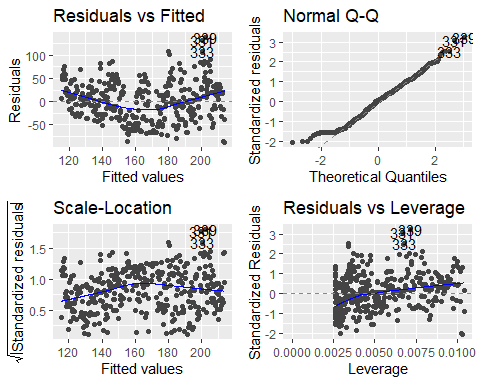
# Plot NO2.trend model  
ggplot(dailyAQ, aes(x=Group.1,y=NO2.GT.)) + geom\_line() +  
 stat\_smooth(method="lm",col="red") + xlab("") + ylab("GT NO2")

## `geom\_smooth()` using formula 'y ~ x'



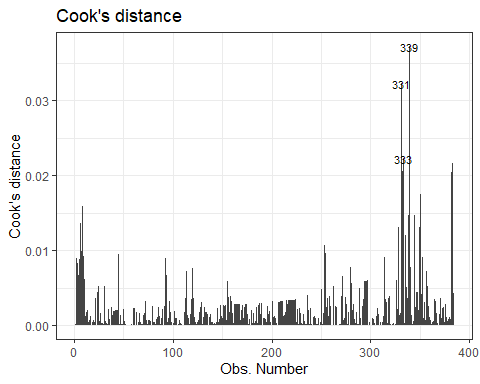
Furthermore, we can then plot the trend model on the existing data to visualize the relationships.

# Model diagnostics for temp.trend  
autoplot(NO2.trend, labels.id = NULL)



These diagnostic plots reveal that there are three potential outliers. Therefore, we will check the Cook’s distances below. The diagnostics of this model are very similar to our trend.seasonal model: decent meeting of assumptions in the QQ and Residuals v. Leverage plot, but Residuals v. Fitted and Scale-Location show some issues with non-constant mean and variance.

# check to make sure no points have cooks distance higher than 0.05  
autoplot(NO2.trend,which=4,ncol=1,label.size=3) + theme\_bw()

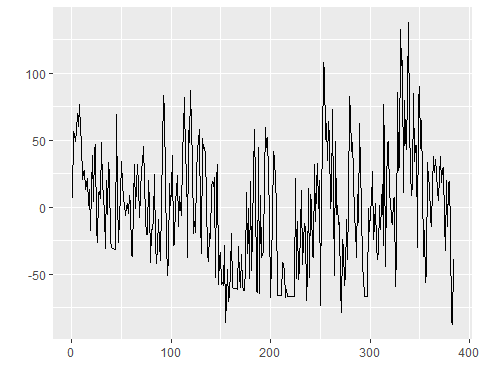


Again, similar to the seasonality model, none of the points have a Cook’s distance greater than 0.05 and therefore do not require removal.

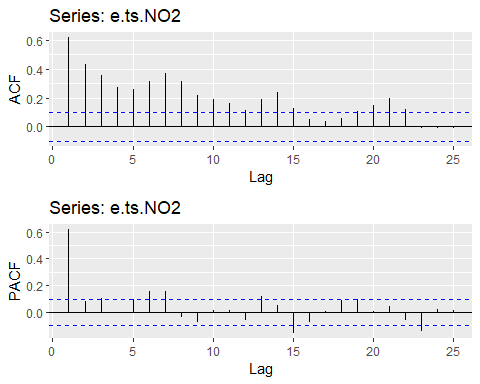
#### 1.3 Autoregressive and moving average components

Now we will check for autoregressive and moving average components.

# using acf and pcf  
# Get the residuals from the NO2.trend model above and store in e.ts.NO2:  
e.ts.NO2 <- ts(NO2.trend$residuals)  
   
# Plot the residuals for the temp.trend model  
autoplot(e.ts.NO2)



# ACF  
NO2.acf <- ggAcf(e.ts.NO2)  
  
# PACF  
NO2.pacf <- ggPacf(e.ts.NO2)  
  
# Plot acf and pacf side by side for easier examination  
ggarrange(NO2.acf,NO2.pacf,nrow=2,ncol=1)



Based on the shapes of both plots, we assume that there are definitely autoregressive terms present. Further, we believe that we can model the NO2 emissions as an AR(1) model. This is because:

* The acf shows exponential decay
* The pacf cuts off after 1 lag

# build ar1 model  
NO2.ar1 <- arma(e.ts.NO2, order=c(1,0))  
summary(NO2.ar1)

##   
## Call:  
## arma(x = e.ts.NO2, order = c(1, 0))  
##   
## Model:  
## ARMA(1,0)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -91.604 -23.325 -2.764 20.703 114.324   
##   
## Coefficient(s):  
## Estimate Std. Error t value Pr(>|t|)   
## ar1 0.61929 0.04014 15.430 <2e-16 \*\*\*  
## intercept -0.08139 1.72328 -0.047 0.962   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Fit:  
## sigma^2 estimated as 1143, Conditional Sum-of-Squares = 436758.5, AIC = 3797.76

# without intercept  
NO2.ar1 <- arma(e.ts.NO2, order=c(1,0), include.intercept = FALSE)

## Warning in optim(coef, err, gr = NULL, hessian = TRUE, ...): one-dimensional optimization by Nelder-Mead is unreliable:  
## use "Brent" or optimize() directly

summary(NO2.ar1)

##   
## Call:  
## arma(x = e.ts.NO2, order = c(1, 0), include.intercept = FALSE)  
##   
## Model:  
## ARMA(1,0)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -91.685 -23.407 -2.845 20.621 114.243   
##   
## Coefficient(s):  
## Estimate Std. Error t value Pr(>|t|)   
## ar1 0.61928 0.04014 15.43 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Fit:  
## sigma^2 estimated as 1143, Conditional Sum-of-Squares = 436760.9, AIC = 3795.76

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

##   
## Call:  
## arima(x = e.ts.NO2, order = c(1, 0, 0), include.mean = FALSE)  
##   
## Coefficients:  
## ar1  
## 0.6177  
## s.e. 0.0400  
##   
## sigma^2 estimated as 1137: log likelihood = -1896.13, aic = 3796.27  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.06582209 33.72644 26.44602 67.34444 173.0708 0.9496177  
## ACF1  
## Training set -0.05098442

We will move forward with the arima model, because the trend from earlier indicates that there is a need to take a first order difference.

Now we will use automatic model selection to build a second model, and later compare the two to use for the rest of our analysis.

# automatic model selection  
NO2.auto <- auto.arima(e.ts.NO2)  
summary(NO2.auto)

## Series: e.ts.NO2   
## ARIMA(2,1,1)   
##   
## Coefficients:  
## ar1 ar2 ma1  
## 0.4362 -0.0223 -0.8828  
## s.e. 0.0699 0.0617 0.0485  
##   
## sigma^2 = 1130: log likelihood = -1888.61  
## AIC=3785.22 AICc=3785.33 BIC=3801.01  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.6115355 33.44208 26.09984 48.32784 193.0488 0.9371873  
## ACF1  
## Training set 0.0004664631

NO2.auto1 <- auto.arima(e.ts.NO2,approximation=FALSE)  
summary(NO2.auto1) # smaller AIC - use this

## Series: e.ts.NO2   
## ARIMA(1,1,1)   
##   
## Coefficients:  
## ar1 ma1  
## 0.4393 -0.8926  
## s.e. 0.0696 0.0398  
##   
## sigma^2 = 1128: log likelihood = -1888.67  
## AIC=3783.35 AICc=3783.41 BIC=3795.19  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.6204464 33.44706 26.09811 48.60256 192.0112 0.9371249  
## ACF1  
## Training set 0.006682074

#### 1.4 Model assessment

We start by comparing the AIC of the two models:

# AIC:  
AIC(NO2.ar1.1)

## [1] 3796.266

AIC(NO2.auto1)

## [1] 3783.35

And now BIC:

# BIC  
BIC(NO2.ar1.1)

## [1] 3804.167

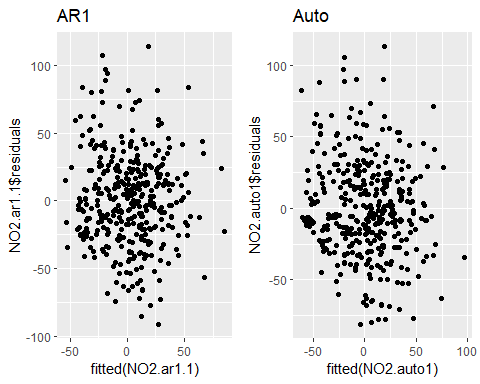
BIC(NO2.auto1)

## [1] 3795.194

The automatic selection model has both a lower AIC and BIC, making it a better choice. We will now check other diagnostics before moving forward with the auto model.

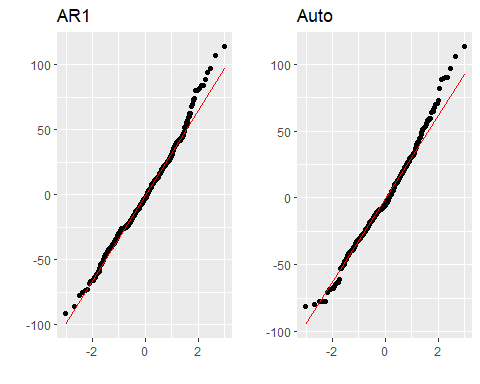
# diagnostics: residuals v. fit  
model1 = ggplot() + geom\_point(aes(x=fitted(NO2.ar1.1), y=NO2.ar1.1$residuals)) + ggtitle("AR1")  
model2 = ggplot() + geom\_point(aes(x=fitted(NO2.auto1), y=NO2.auto1$residuals)) + ggtitle("Auto")  
  
ggarrange(model1, model2, ncol=2, nrow=1)

## 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.

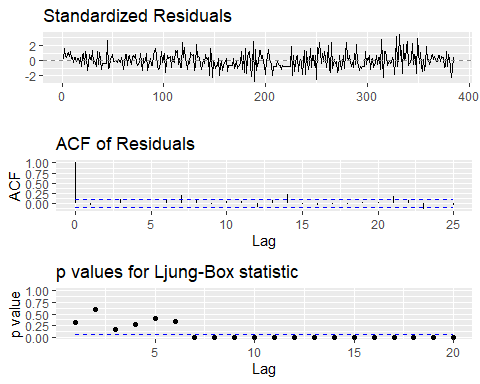


# assess normality of residuals  
model1 = qplot(sample=NO2.ar1.1$residuals) + stat\_qq\_line(color="red") + ggtitle("AR1")  
model2 = qplot(sample=NO2.auto1$residuals) + stat\_qq\_line(color="red") + ggtitle("Auto")  
  
ggarrange(model1, model2, ncol=2, nrow=1)

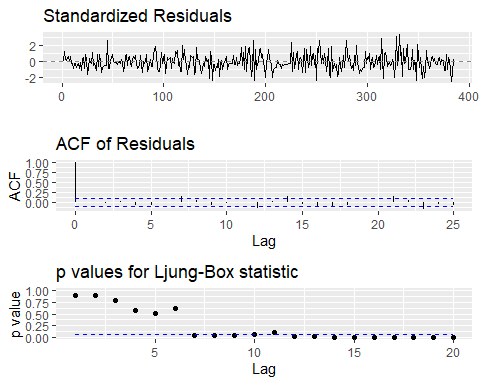
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know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



ggtsdiag(NO2.ar1.1,gof.lag=20)



ggtsdiag(NO2.auto1,gof.lag=20)



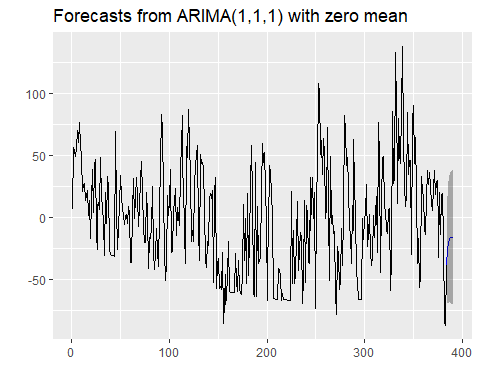
The Ljung-Box plot shows that both models are adequate up to lag 7. In terms of the residual v. fit and QQ plots, both models are very similar and meet assumptions.

Because both models are very similar in diagnostics, we choose the automatically selected model due to the lower AIC and BIC. This is what will be used for the remainder of our analysis.

#### 1.5 7 day forecast

Now we forecast the next 7 days of NO2 concentrations using the NO2.auto model.

NO2.auto.forecast <- forecast(NO2.auto1, h=7)  
  
autoplot(NO2.auto.forecast,main="Forecasts from ARIMA(1,1,1) with zero mean")



anova(NO2.trend,NO2.trend.seasonal)

## Analysis of Variance Table  
##   
## Model 1: dailyNO2.ts ~ time.NO2  
## Model 2: dailyNO2.ts ~ time.NO2 + sin(2 \* pi \* time.NO2/192) + cos(2 \*   
## pi \* time.NO2/192)  
## Res.Df RSS Df Sum of Sq F Pr(>F)   
## 1 382 707556   
## 2 380 692335 2 15221 4.1772 0.01605 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

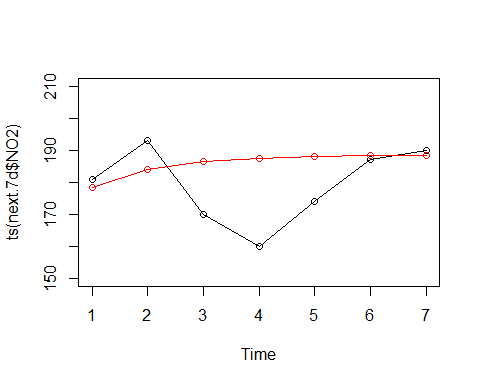
# Prediction performance  
# Create test set from temp data set with last 7 days  
  
# The test period in days  
next.7d.time <- c((length(dailyNO2)-6):(length(dailyNO2)))  
  
# The test data frame  
next.7d <- data.frame(time.NO2 = next.7d.time, NO2 = dailyNO2[next.7d.time])  
  
# The actual time series for the test period  
next.7d.ts <- ts(next.7d$NO2)  
  
# Prediction for the next 6 months by temp.auto:  
E\_Y.pred <- predict(NO2.trend.seasonal, newdata=next.7d)  
e\_t.pred <- forecast(NO2.auto1, h=7)  
next.7d.prediction <- E\_Y.pred + e\_t.pred$mean

# MSE:  
mean((next.7d.prediction-next.7d$NO2)^2)

## [1] 188.9485

Our forecast yields a MSE of 189. We can now plot the predicted vs. actual values to see how they compare for the last 7 days of the data:

# Plot actual values and predicted values  
plot(ts(next.7d$NO2),type='o',ylim=c(150,210))  
lines(ts(next.7d.prediction),col='red',type='o')  
lines(1:7, E\_Y.pred + e\_t.pred$lower[,2], col = "red", lty = "dashed")  
lines(1:7, E\_Y.pred + e\_t.pred$upper[,2], col = "red", lty = "dashed")  
legend(1,60, legend = c("Actual", "Predicted"), lwd = 2, col = c("black", "red"))



Based on the above graph, our forecast was reasonably close for 4/7 predictions (the first two and the last two). It can be observed that our simulation predicted more of a smooth curve compared to the real data which is much more variable. This makes sense because the real data is messy, and our model may not be equipped to replicate this without overfitting.

# Part 2: Simulating Univariate Time Series Models

We now simulate a year of synthetic observations of daily maximum nitrogen dioxide (NO2) concentrations from your selected model. The seed is set to 1 so that the same results can be obtained each time. You will need to consider the sum of the linear models of the trend + seasonality, and the residual models. Assess and compare the model’s performance with respect to:

#### 2.1 Ability to reproduce the appearance of time series. Plot observations and simulations and visually compare their characteristics.

We are selecting the arima(1,1,1) model derived from automatic selection for this section because it has a lower AIC and BIC than the ar(1) model, with similar diagnostics.

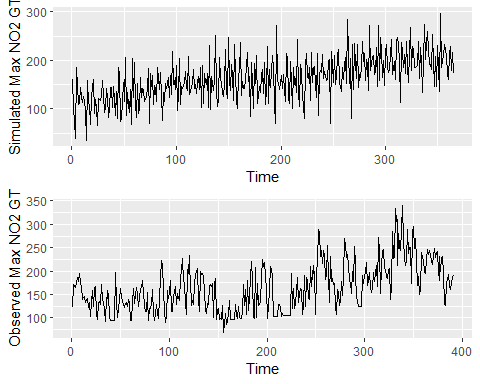
# Simulate 50 years of monthly minimum temperatures with the best model  
# auto.arima builds ARMA(1,1) model  
set.seed(1)  
auto.sim <- arima.sim(n=365, list(ar=c(NO2.auto1$coef[1]),  
 ma=c(NO2.auto1$coef[2])),  
 sd=sqrt(NO2.auto1$sigma2))

# Add mean predictions and plot simulation of Tmin  
next.1yr.time <- c(1:365)  
next.1yr <- data.frame(time.NO2 = next.1yr.time)  
  
next.1.yr.predictions <- predict(NO2.trend.seasonal, newdata=next.1yr)  
  
# plot simulated temperatures  
predicted\_1yrplot <- autoplot(ts(next.1.yr.predictions + auto.sim),xlab="Time",ylab="Simulated Max NO2 GT")

Here, we generate the predictions and simulation for our ARIMA model. In order to determine the viability of the model, we then turn toward visualizing the simulation with the observed data.

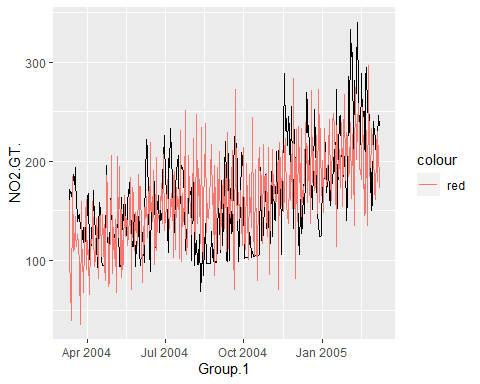
observed\_plot <- autoplot(ts(dailyNO2),xlab="Time",ylab="Observed Max NO2 GT")

ggarrange(predicted\_1yrplot,observed\_plot,nrow=2,ncol=1)



After seeing the two plots we generated, we overlay the plots to get a stronger visualization.

dailyAQ\_1yr <- head(dailyAQ,365)  
ggplot(dailyAQ\_1yr, aes(x=Group.1,y=NO2.GT.)) + geom\_line() +   
 geom\_line(aes(Group.1, next.1.yr.predictions + auto.sim, color='red'))



The visualization above shows the simulated data in red, and the observed data in black. We can see that the simulated data seems to match the observed data within a reasonable deviation.

#### 2.2 Ability to reproduce observed trends. You can assess this by building a linear model of the trend + seasonality of the simulations and comparing the coefficient estimates with the linear model of the trend + seasonality of the observations. What is the percent difference in the coefficient on time?

summary(NO2.trend.seasonal)

##   
## Call:  
## lm(formula = dailyNO2.ts ~ time.NO2 + sin(2 \* pi \* time.NO2/192) +   
## cos(2 \* pi \* time.NO2/192))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -90.458 -33.603 0.671 29.988 134.304   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 118.28961 4.64945 25.442 <2e-16 \*\*\*  
## time.NO2 0.24023 0.02134 11.258 <2e-16 \*\*\*  
## sin(2 \* pi \* time.NO2/192) -6.63400 3.34509 -1.983 0.0481 \*   
## cos(2 \* pi \* time.NO2/192) -6.46088 3.08054 -2.097 0.0366 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 42.68 on 380 degrees of freedom  
## Multiple R-squared: 0.3198, Adjusted R-squared: 0.3145   
## F-statistic: 59.57 on 3 and 380 DF, p-value: < 2.2e-16

# Model seasonality  
dailyNO2.ts.sim <- ts(next.1.yr.predictions + auto.sim)  
  
NO2.trend.seasonal.sim <- lm(dailyNO2.ts.sim ~ next.1yr.time + sin(2\*pi\*next.1yr.time/192) + cos(2\*pi\*next.1.yr.predictions/192))  
  
summary(NO2.trend.seasonal.sim)

##   
## Call:  
## lm(formula = dailyNO2.ts.sim ~ next.1yr.time + sin(2 \* pi \* next.1yr.time/192) +   
## cos(2 \* pi \* next.1.yr.predictions/192))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -103.958 -23.636 -1.672 24.817 106.684   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 123.15774 6.45009 19.094 < 2e-16  
## next.1yr.time 0.19727 0.04888 4.036 6.64e-05  
## sin(2 \* pi \* next.1yr.time/192) -5.27361 3.05198 -1.728 0.0849  
## cos(2 \* pi \* next.1.yr.predictions/192) 9.33246 8.96897 1.041 0.2988  
##   
## (Intercept) \*\*\*  
## next.1yr.time \*\*\*  
## sin(2 \* pi \* next.1yr.time/192) .   
## cos(2 \* pi \* next.1.yr.predictions/192)   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 37.41 on 361 degrees of freedom  
## Multiple R-squared: 0.3573, Adjusted R-squared: 0.352   
## F-statistic: 66.91 on 3 and 361 DF, p-value: < 2.2e-16

# percent difference  
((0.24023 - 0.19727) / ((0.24023+0.19727)/2))\*100

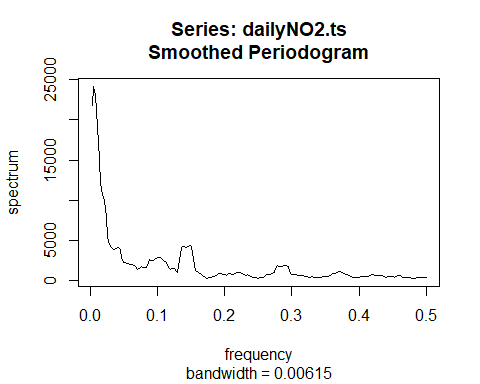
## [1] 19.63886

Percent difference for coefficient on time: -19.6%

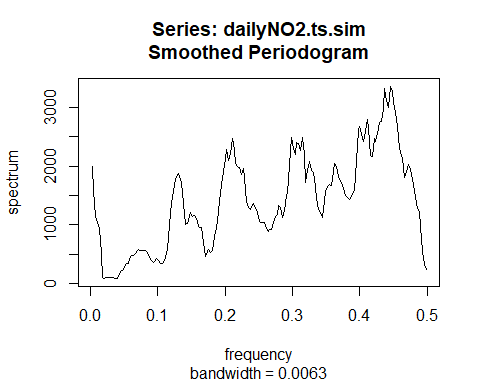
In order to prove that the observed trends can be reproduced, we compared a simulated linear model of trend + seasonality with the observed linear model of trend + seasonality. For both models, the objective is to compare the coefficients on time and minimize the percent difference between them. The coefficient on time for the simulated model is 0.24023, and the coefficient on time for the observed model is 0.19727%. There is a percent difference of 19.64%. With the objective of this exercise to minimize the percent difference, this is an acceptable model and proves that the observed trends can be reproduced with adequate accuracy.

#### 2.3 Ability to reproduce seasonality of the time series. Analysis can be visual, simply comparing the periodogram of the observations and simulations.

# original  
pg.NO2<- spec.pgram(dailyNO2.ts,spans=9,demean=T,log='no')



pg.NO2.sim<- spec.pgram(dailyNO2.ts.sim,spans=9,demean=T,log='no')



The periodograms of the original and simulated data do not match up well. The most concerning difference is the general decrease in the original periodogram and the general increase in the simulated periodogram. Next we will sort the first 20 peaks and match them up 1:1 to see if they align, since the periodograms look very different.

# Find the peak, max.omega.NO2  
max.omega.NO2<-pg.NO2$freq[which(pg.NO2$spec==max(pg.NO2$spec))]  
sim.max.omega.NO2<-pg.NO2.sim$freq[which(pg.NO2.sim$spec==max(pg.NO2.sim$spec))]  
  
# Where is the peak?  
max.omega.NO2

## [1] 0.005208333

sim.max.omega.NO2

## [1] 0.4453333

# What is the period?  
1/max.omega.NO2

## [1] 192

1/sim.max.omega.NO2

## [1] 2.245509

# compare first 20 periods of real and simulated data to see how they match up  
  
# sort spectrum from largest to smallest and find index  
sorted.spec <- sort(pg.NO2.sim$spec, decreasing=T, index.return=T)  
names(sorted.spec)

## [1] "x" "ix"

# corresponding periods (omegas = frequencies, Ts = periods)  
sorted.omegas <- pg.NO2.sim$freq[sorted.spec$ix]  
sorted.Ts <- 1/pg.NO2.sim$freq[sorted.spec$ix]  
  
# look at first 20  
#sorted.omegas[1:20]  
sorted.Ts[1:20]

## [1] 2.245509 2.286585 2.232143 2.272727 2.218935 2.259036 2.300613 2.205882  
## [9] 2.435065 2.314815 2.329193 2.192982 2.500000 2.419355 2.516779 2.450980  
## [17] 2.483444 2.343750 3.177966 3.348214

# the Ts should roughly mimic those of the original data (period)

# original data   
  
sorted.spec <- sort(pg.NO2$spec, decreasing=T, index.return=T)  
sorted.Ts.og <- 1/pg.NO2$freq[sorted.spec$ix]  
  
sorted.Ts.og[1:20]

## [1] 192.000000 128.000000 384.000000 96.000000 76.800000 64.000000  
## [7] 54.857143 48.000000 42.666667 38.400000 34.909091 6.736842  
## [13] 7.245283 6.620690 7.111111 6.857143 6.981818 32.000000  
## [19] 24.000000 7.384615

Comparing the first 20 periods of the original data with the simulated data, we can see that the simulated periods do not mimic those of the original data. While some variation is expected, as there are discrepancies between the simulated and observed data, the contrast between trends in the periodograms is concerning. This could be due to the use of real and messy data, an incorrect or inadequate selection of models, our limited modeling abilities, or a combination of the three. However, in our initial model of the data, we found seasonality of the real data to be insignificant (Section 1.2). Thus, since we are able to reproduce the trend reasonably well, we are less concerned about the lack of matching up of the periodograms.

#### 2.4 Ability to reproduce observed mean and variance of the time series (Hint: Use the functions ‘mean(ts)’ and ‘var(ts)’ where ts is a time series, and find the percent difference between observations and simulations)

mean(dailyNO2.ts)

## [1] 164.5335

mean(dailyNO2.ts.sim)

## [1] 162.5446

# % difference  
(abs((mean(dailyNO2.ts.sim) - mean(dailyNO2.ts)))/ ((mean(dailyNO2.ts.sim)+mean(dailyNO2.ts))/2))\*100

## [1] 1.216182

var(dailyNO2.ts)

## [1] 2657.722

var(dailyNO2.ts.sim)

## [1] 2160.124

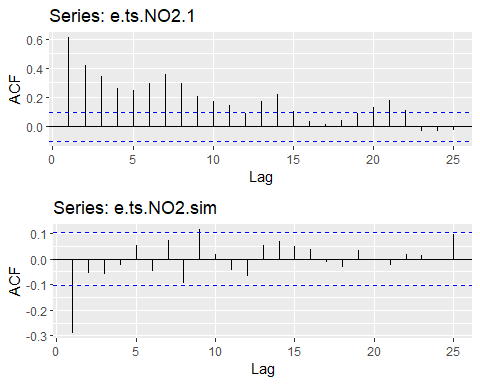
# % difference  
(abs((var(dailyNO2.ts.sim) - var(dailyNO2.ts)))/ ((var(dailyNO2.ts.sim)+var(dailyNO2.ts))/2))\*100

## [1] 20.65646

As seen above, the mean of the simulated time series was only 1.2% different than the mean of the observed time series. The variance has a 20.7% difference. This makes sense because the observed data has a higher degree of variability in comparison to our simulation.

#### 2.5 Ability to reproduce the autocorrelation of the time series. Analysis can be visual, simply comparing the ACF and PACF of the observations and simulations

# compare ACF of observed and simulated time series  
e.ts.NO2.1 <- ts(NO2.trend.seasonal$residuals)  
e.ts.NO2.sim <- ts(NO2.trend.seasonal.sim$residuals)  
  
NO2.acf1 <- ggAcf(e.ts.NO2.1)  
NO2.acf.sim <- ggAcf(e.ts.NO2.sim)  
  
ggarrange(NO2.acf1,NO2.acf.sim,nrow=2,ncol=1)



As seen in the ACF’s above, despite not matching up perfectly, both the original data and simulated data exhibit sinusoidal decay. Therefore, the simulated model is an acceptable representation of the observed model. Based on their respective ACF’s, the autocorrelation of the time series can be reproduced.

NO2.pacf1 <- ggPacf(e.ts.NO2.1)  
NO2.pacf.sim <- ggPacf(e.ts.NO2.sim)  
  
ggarrange(NO2.pacf1,NO2.pacf.sim,nrow=2,ncol=1)

