Project 2

Cecilia Smith, Sofia Zajec, TJ Gwilliam, Reese Quillian

Date

# Load data and impute missing values

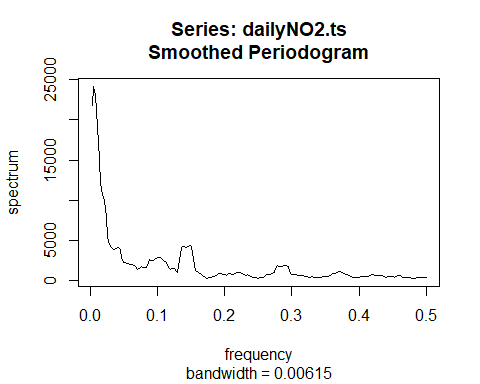
setwd(datadir)  
  
airquality = read.csv('AirQualityUCI.csv')  
  
# replace -200 with NA  
airquality[airquality == -200] <- NA  
  
# convert integer type to numeric  
intcols = c(4,5,7,8,9,10,11,12)  
for(i in 1:length(intcols)){  
 airquality[,intcols[i]] <- as.numeric(airquality[,intcols[i]])  
}  
  
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

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

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

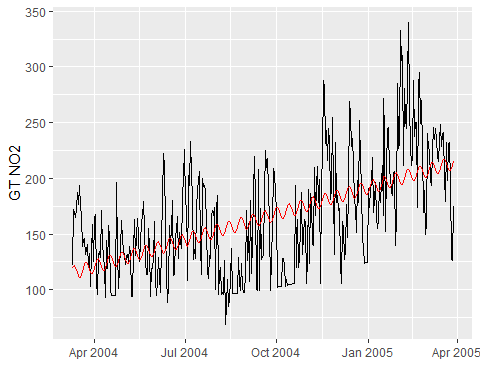
NO SEASONALITY HERE!

# Model seasonality  
NO2.trend.seasonal <- lm(dailyNO2.ts ~ time.NO2 + sin(2\*pi\*time.NO2/12) + cos(2\*pi\*time.NO2/12))  
summary(NO2.trend.seasonal)

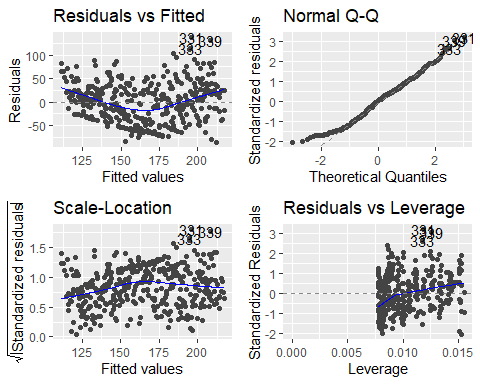
##   
## Call:  
## lm(formula = dailyNO2.ts ~ time.NO2 + sin(2 \* pi \* time.NO2/12) +   
## cos(2 \* pi \* time.NO2/12))  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -85.836 -34.073 2.454 27.183 136.933   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 115.00564 4.39227 26.184 <2e-16 \*\*\*  
## time.NO2 0.25729 0.01977 13.011 <2e-16 \*\*\*  
## sin(2 \* pi \* time.NO2/12) 5.81165 3.09991 1.875 0.0616 .   
## cos(2 \* pi \* time.NO2/12) 1.37990 3.09910 0.445 0.6564   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 42.94 on 380 degrees of freedom  
## Multiple R-squared: 0.3116, Adjusted R-squared: 0.3062   
## F-statistic: 57.34 on 3 and 380 DF, p-value: < 2.2e-16

^ only trend significant

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")

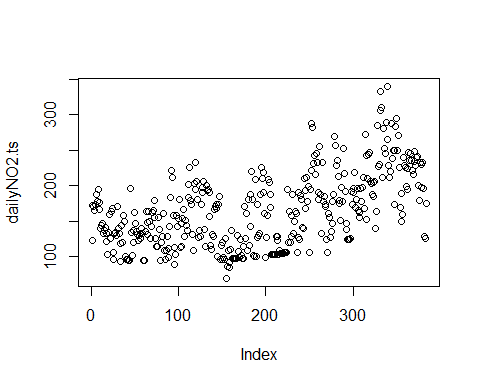


# Model diagnostics for NO2.trend.seasonal  
autoplot(NO2.trend.seasonal, labels.id = NULL)



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

plot(dailyNO2.ts)



Based on the plot above, there appears to be a general increasing trend.

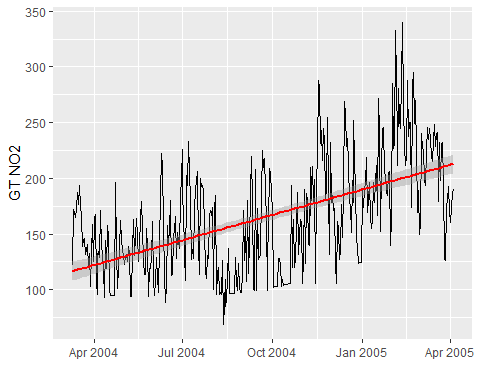
NO2.trend<-lm(dailyNO2.ts ~ time.NO2)  
summary(NO2.trend)

##   
## 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

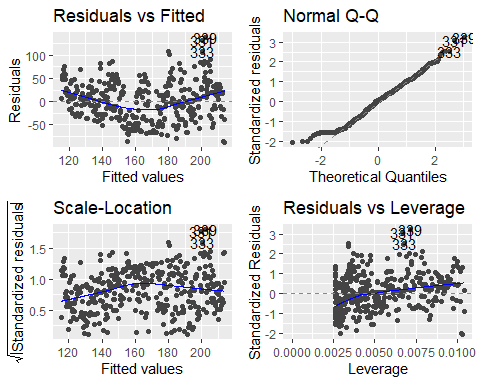
Trend is significant

# 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'



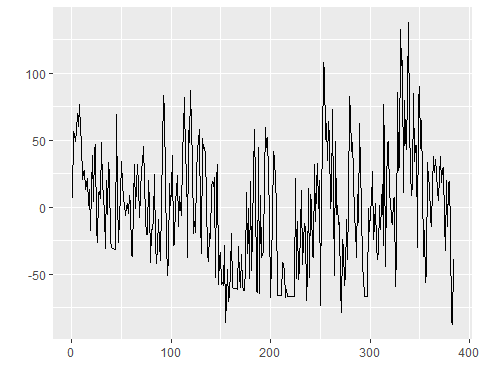
# Model diagnostics for temp.trend  
autoplot(NO2.trend, labels.id = NULL)



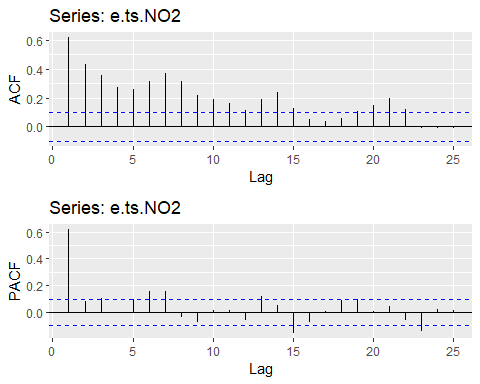
problems with diagnostics: seem to have some influential points, can check using Cooks distance and remove if needed Other issues = non-constant mean

1. How you determined autoregressive and moving average components, if applicable.

# 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)



AR(1) - pacf cuts off after 1 lag - acf shows exponential decay

# 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

# 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

# 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

# AIC:  
AIC(NO2.ar1.1)

## [1] 3796.266

AIC(NO2.auto1)

## [1] 3783.35

# BIC  
BIC(NO2.ar1.1)

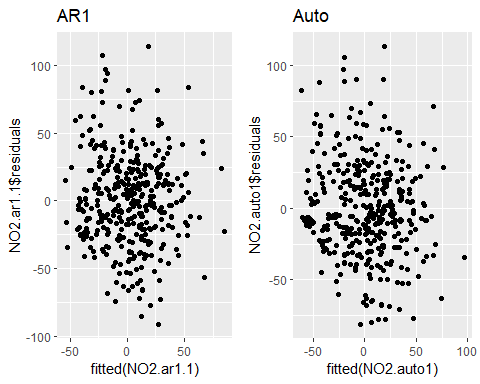
## [1] 3804.167

BIC(NO2.auto1)

## [1] 3795.194

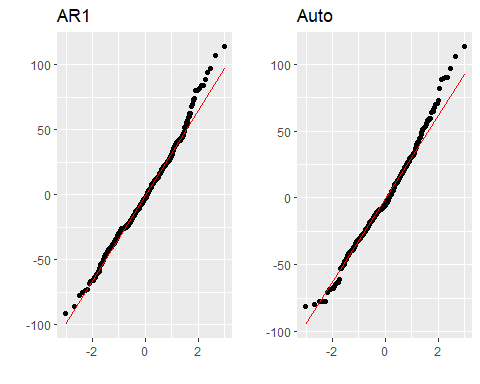
# diagnostics: residuals v. fit  
model1 = ggplot() + geom\_point(aes(x=fitted(NO2.ar1.1), y=NO2.ar1.1$residuals)) + ggtitle("AR1")  
model2 = ggplot() + geom\_point(aes(x=fitted(NO2.auto1), y=NO2.auto1$residuals)) + ggtitle("Auto")  
  
ggarrange(model1, model2, ncol=2, nrow=1)

## 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.

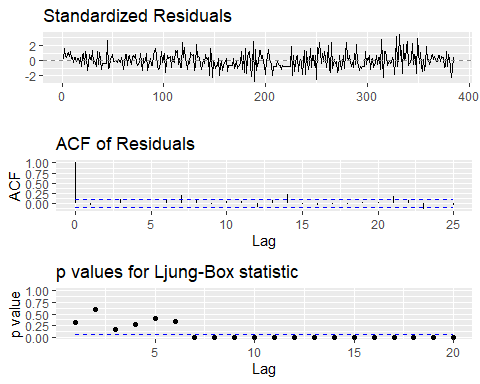


# assess normality of residuals  
model1 = qplot(sample=NO2.ar1.1$residuals) + stat\_qq\_line(color="red") + ggtitle("AR1")  
model2 = qplot(sample=NO2.auto1$residuals) + stat\_qq\_line(color="red") + ggtitle("Auto")  
  
ggarrange(model1, model2, ncol=2, nrow=1)

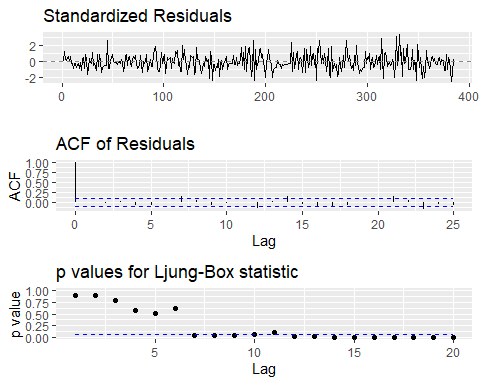
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.  
## Don't know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



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



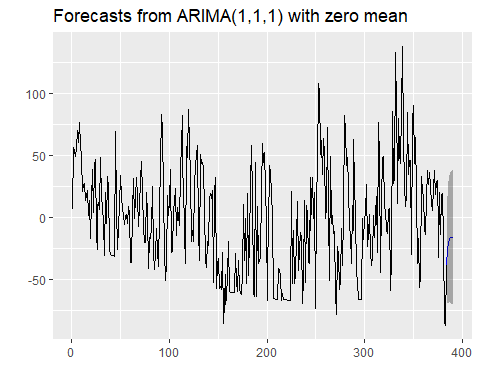
ggtsdiag(NO2.auto1,gof.lag=20)

 both adequate up to lag 7

auto = model selected

1. Forecast the next 7 days of NO2 concentrations using your selected model. Plot the forecasts vs. true values. What is the MSE of the 7-day forecast? (5 points)

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



# 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, 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] 437.0741

# Part 2: Simulating Univariate Time Series Models