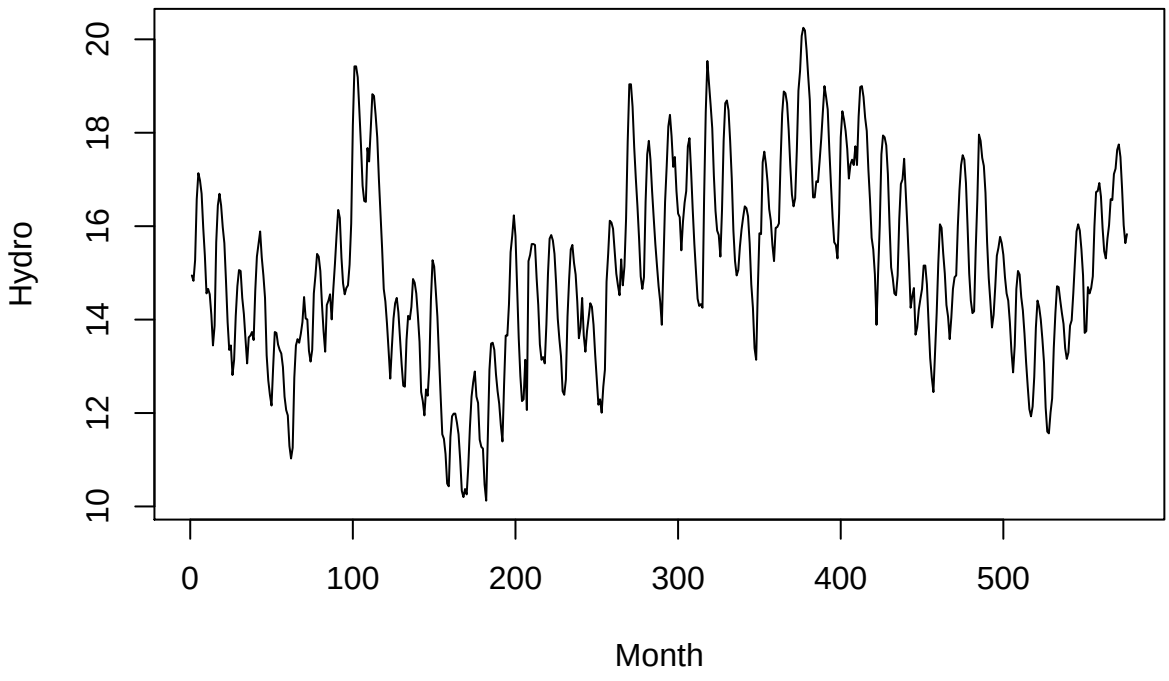
Hydrological

First, we import the data.

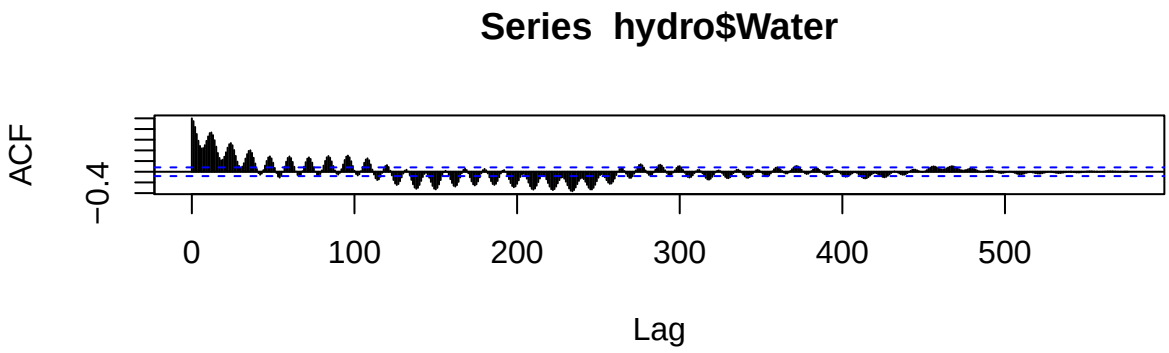
```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Hydrological")
hydro <- read.delim2("hyd_post.txt", header = TRUE, sep = ",")
colnames(hydro) <- c("Month", "Water")
hydro$Water <- as.numeric(unlist(hydro$Water))
```

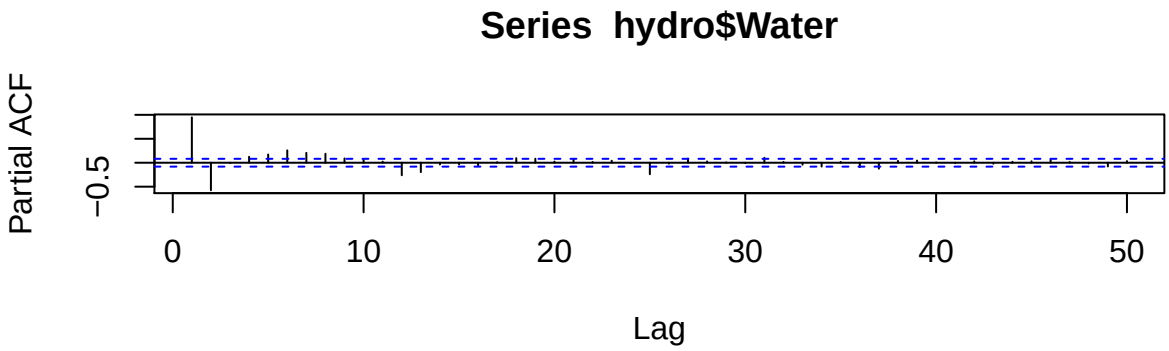
Let's plot the graph, ACF and PACF of the times series.

```
plot(x = hydro$Month, y = hydro$Water, type = "l", xlab = "Month", ylab = "Hydro")
```




```
par(mfrow=c(2,1))
acf(hydro$Water, lag.max = 600)
pacf(hydro$Water, lag.max = 50)
```





The ACF graph of the time series obviously shows seasonal behavior. We can see that there are approximately 9 cycles in 100 lags. Therefore, it is reasonable to infer that the frequency of the data is 12. Therefore, we need to set the frequency of the time series to 12:

```
hydrots <- ts(hydro$Water, frequency = 12)
```

Since the ACF graph geometrically decays and there are some straight lines are beyond the two dashed-blue lines in the PACF graph, it suggests us we should fit SARIMA model to the time series.

```
library("forecast")
```

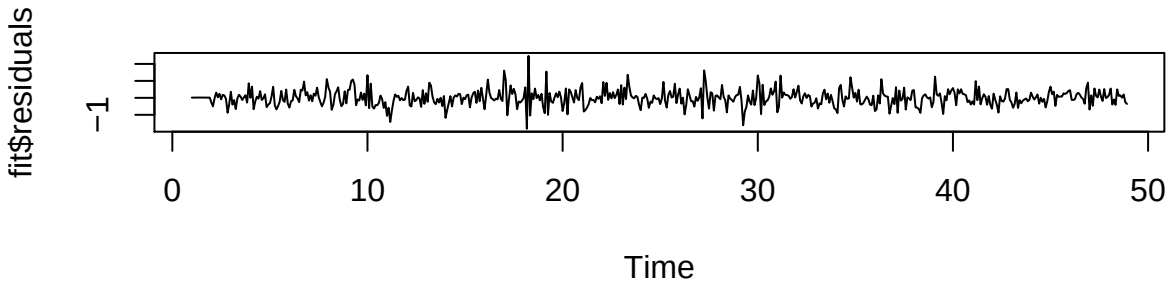
```
## Warning: package 'forecast' was built under R version 4.0.2
```

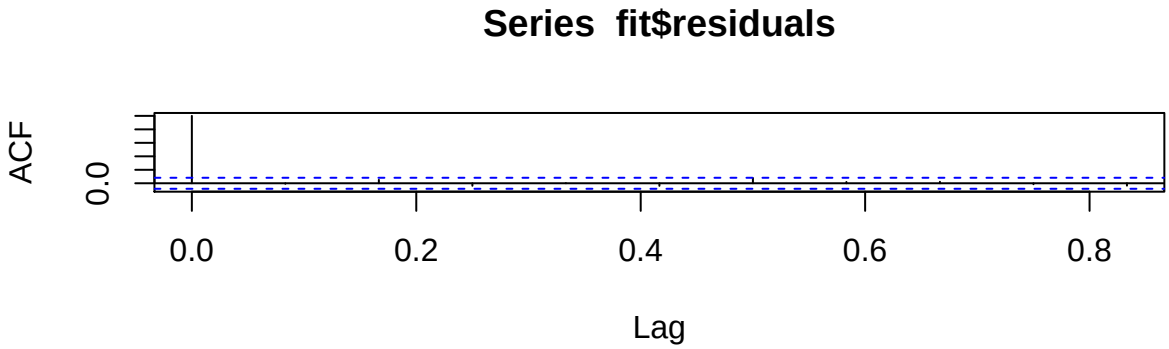
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
fit <- auto.arima(hydrots, max.p = 10, max.q = 10,
                  max.P = 10, max.Q = 10, max.d = 5, max.D = 12,
                  seasonal = TRUE)
```

Then, we use Ljung-Box Test to test the residuals of the model to see if it's strong white noise. If the residuals are white noise, then we can conclude that the model fits very well.

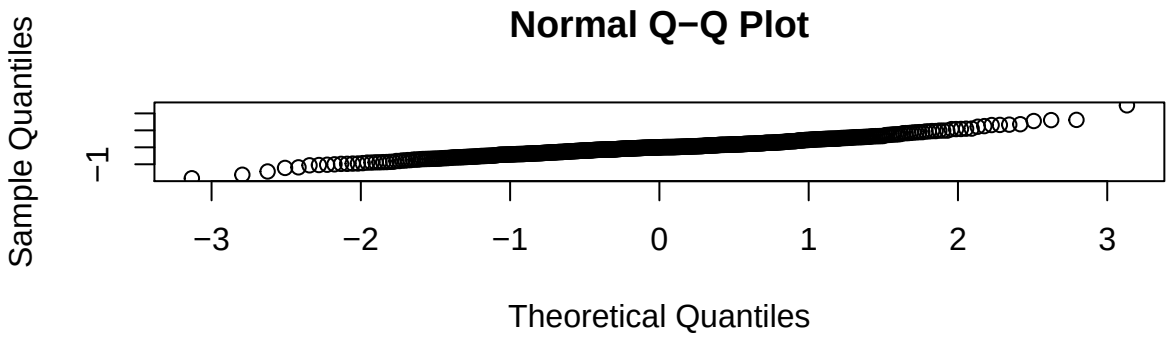
```r
par(mfrow=c(2,1))
plot(fit$residuals)
acf(fit$residuals, lag = 10)
```






```r
qqnorm(fit$residuals)
Box.test(fit$residuals, lag = 10, type = c("Ljung-Box"))
```

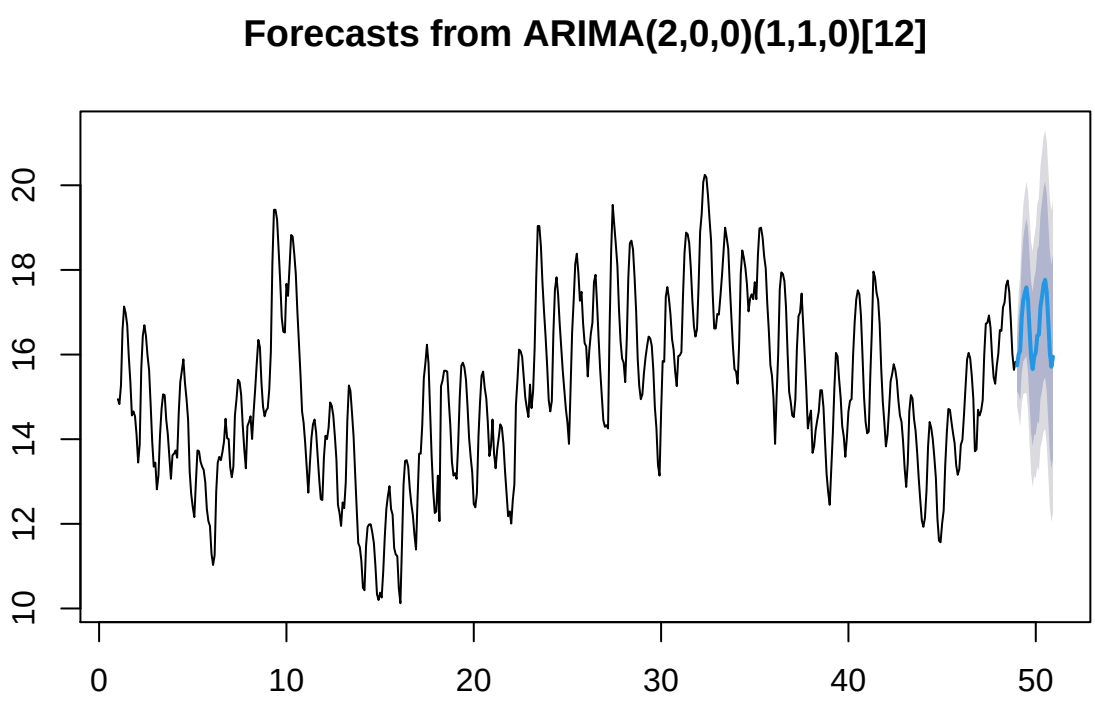
```
##
##  Box-Ljung test
##
## data:  fit$residuals
## X-squared = 8.1127, df = 10, p-value = 0.6178
```



We can see that the ACF plot of the residuals shows that almost all the straight lines are between the dashed-blue lines. The QQ-plot of the residuals shows that the residuals are normally distributed in general. The Box-Ljung test p-value $= 0.6178 > 0.05$, so they both suggest that there is no evidence against the null hypothesis that the residuals are white noise.

At last, we produce the 24 steps ahead forecast and plot it with 95% prediction interval.

```
plot(forecast(fit,h=24))
```




```
last.name <- "Chen"
student.id <- 20756546
write(forecast(fit,h=24)$mean, file = paste("Scenario1_",last.name,student.id,".txt", sep = ""),
      ncolumns = 1 )
```

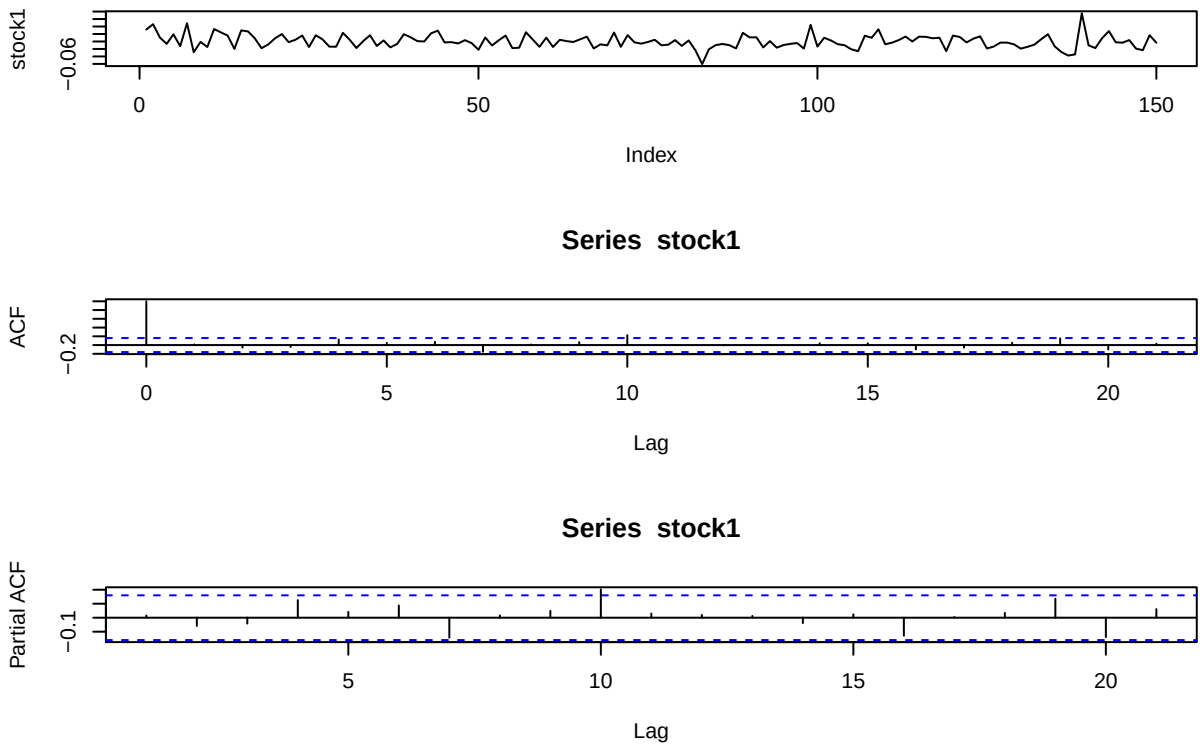
Financial Risk

First, let's look at the first stock and try to forecast 10 steps ahead 15% VaR for it. Let's import the data for the first stock. Notice that the data is the log returns of the stock.

```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Financial Risk")
stock1 <- read.delim2("stock1.txt", header = TRUE, sep = ",")
colnames(stock1) <- c("DayNum", "logDiffPrice")
stock1 <- as.numeric(unlist(stock1$logDiffPrice))
```

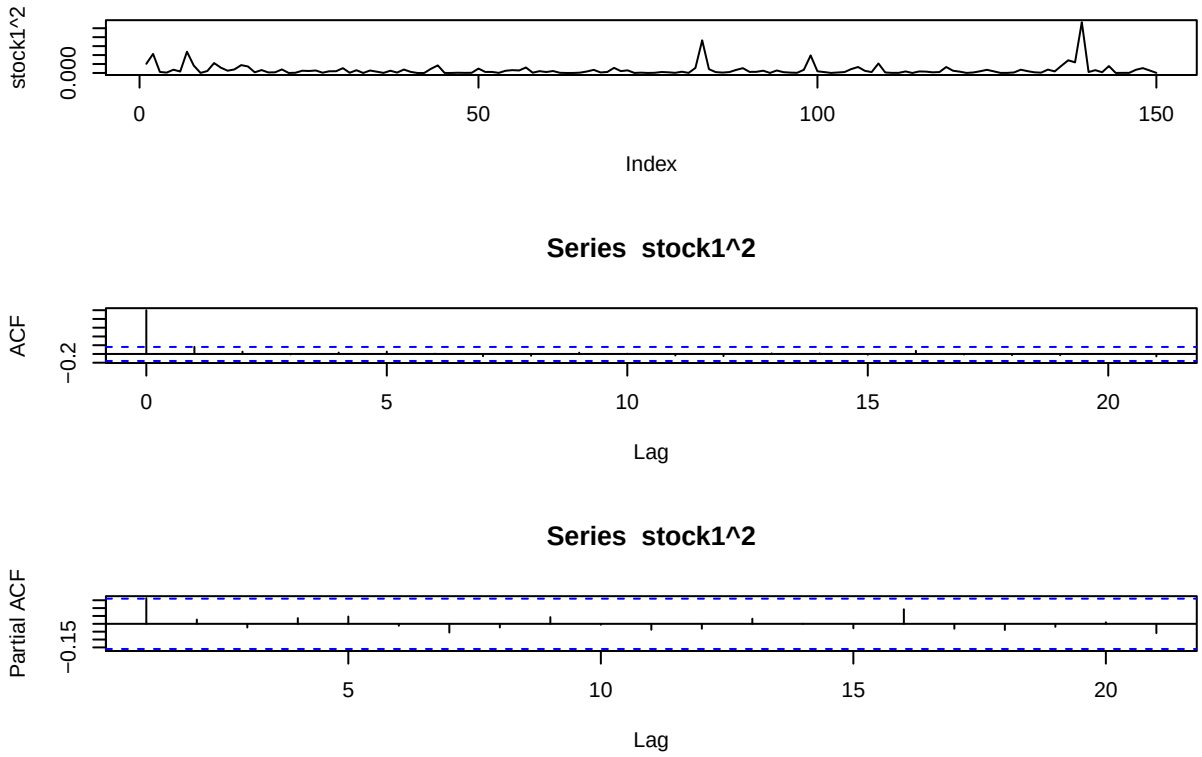
Let's plot the log returns, its ACF and PACF.

```
par(mfrow=c(3,1))
plot(stock1, type = "l")
acf(stock1) # looks very much like white noise
pacf(stock1)
```



We can see that there isn't any obvious serial correlation in the dataset. Almost all the straight lines are between the blue-dashed lines. It seems like the dataset is like a white noise. However, let's plot the squared series, its ACF, and PACF and see if there is any serial correlation.

```r
par(mfrow=c(3,1))
plot(stock1^2, type = "l")
acf(stock1^2) # looks very much like white noise
pacf(stock1^2)
```



Unfortunately, there is no obvious serial correlation in the squared series. Still, almost all the straight lines are between two dashed-blue lines. It looks like the dataset is a white noise. However, we can still try to use ARMA and GARCH model to capture it's minor serial correlation, if there is any.

Let's fit ARMA and GARCH model to the stock's log return. After trying several parameters for ARMA and GARCH model, I found out that ARMA(0, 0) plus GARCH(1, 0) has the lowest AIC. We also can see that the p-values for the Ljung-Box Test is larger than 0.05, so there is no evidence against the null hypothesis that the residuals are white noise. We conclude that GARCH(1,0) is the best model for us.

```r
library(fGarch)
summary(mod1 <- garchFit(~arma(0,0)+garch(1,0), stock1, trace=FALSE))
```

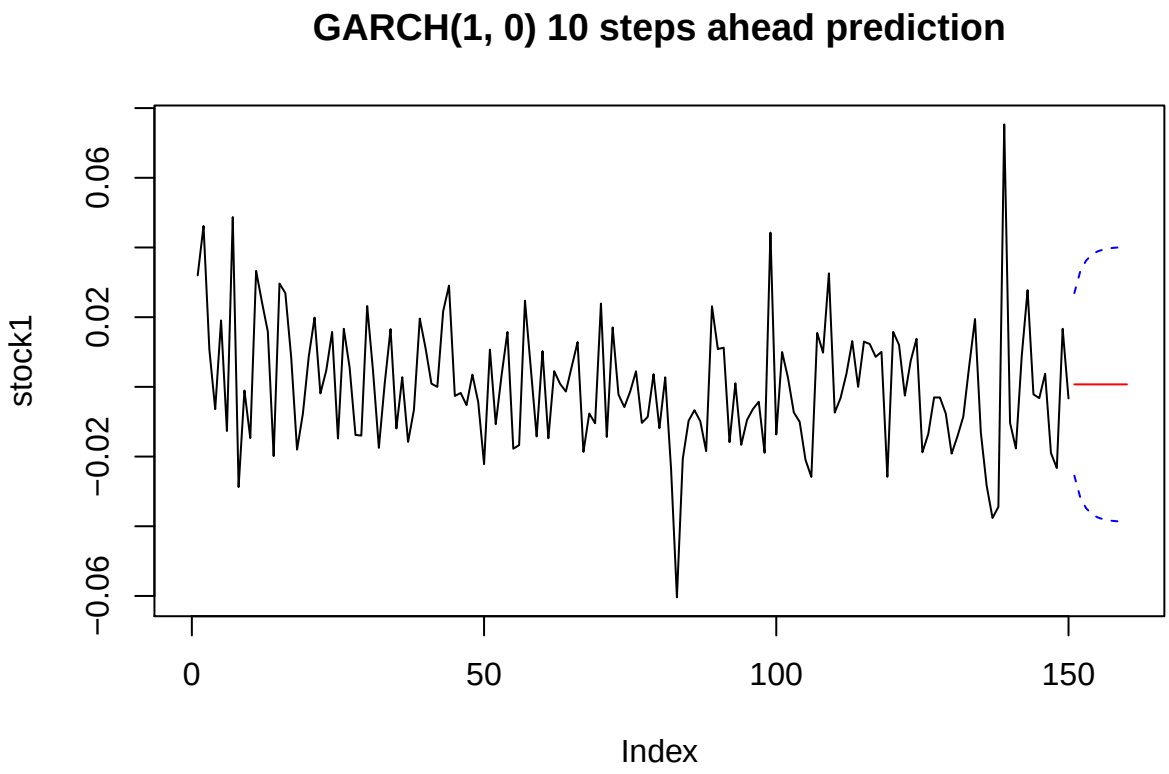
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(1, 0), data = stock1,
##     trace = FALSE)
##
## Mean and Variance Equation:
```

```
##  data ~ arma(0, 0) + garch(1, 0)
## <environment: 0x7fa98d4b1960>
##  [data = stock1]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1
## 0.00071137  0.00016202  0.58461261
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     7.114e-04   1.177e-03    0.604  0.54556
## omega  1.620e-04   3.339e-05    4.853 1.22e-06 ***
## alpha1 5.846e-01   1.910e-01    3.060  0.00221 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  399.1406    normalized:  2.660937
##
## Description:
##  Thu Apr 22 20:40:41 2021 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  2.411331  0.2994927
##  Shapiro-Wilk Test  R    W      0.9902269 0.3853824
##  Ljung-Box Test     R    Q(10)  18.74248  0.04365821
##  Ljung-Box Test     R    Q(15)  19.5139   0.1913834
##  Ljung-Box Test     R    Q(20)  30.65802  0.05987002
##  Ljung-Box Test     R^2  Q(10)  6.723976  0.751221
##  Ljung-Box Test     R^2  Q(15)  9.774538  0.8336773
##  Ljung-Box Test     R^2  Q(20)  13.31375  0.8635203
##  LM Arch Test       R    TR^2   7.834535  0.7979235
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -5.281874 -5.221662 -5.282654 -5.257412
```

Then, we use GARCH(1,0) model to do the 10 steps ahead prediction for stock 1. We will use those predicted values to find the 85% VaR for the stock.

```
len <- length(stock1)
plot(stock1, xlim = c(0, len + 10), type = "l", main = "GARCH(1, 0) 10 steps ahead prediction")
garchPred <- predict(mod1, 10)$meanForecast
garchSTD <- predict(mod1, 10)$standardDeviation
lines(x = (len+1) : (len+10), garchPred + 2 * garchSTD, col = "blue", lty = 2)
lines(x = (len+1) : (len+10), garchPred - 2 * garchSTD, col = "blue", lty = 2)
```

```
lines(x = (len+1) : (len+10), garchPred, col = "red")
```



In lecture, the professor introduced four different methods to forecast the VaR of a stock returns: the historical method, the Risk Metrics method, the GARCH Normal Innovations, and the GARCH Bootstrap Innovations method. Now we are going to backtest the four methods for stock1. In the lectures, for every stock return at time t, the professor calculated the past 250 days VaR and see if the stock return at time t violates the VaR. In our case, we only have 150 data points for each stock, so we will choose 30 days instead of 250 days for backtesting.

```
num = length(31:length(stock1))

histc = 1:num
historic=0
for(j in (30:(length(stock1)-1))){
  cut <- quantile(stock1[(j-30):j], 0.15)
  histc[j-30+1]=cut
  if(stock1[j+1] < cut){historic=historic+1}
}
historic/num
```

```
## [1] 0.175
```

```
#############################################################
Riskc= 1:num
```

```
RiskMetrics=0
lam <- 0.94
sig1 <- mean(stock1[1:29]^2)
for(j in (30:(length(stock1)-1))){
  signew= lam*sig1 + (1-lam)*stock1[j]^2
  sig1=signew

  Riskc[j-30+1]=sqrt(signew)*qnorm(.15)

  if(stock1[j+1] < sqrt(signew)*qnorm(0.15)){RiskMetrics=RiskMetrics+1}
}
RiskMetrics/num
```

```
## [1] 0.1416667
```

```
#################################################################
num=length((31:length(stock1)))
gNorm=0
gNormQ= 1:length((31:length(stock1)))


gNonParam=0
gNonParamQ=1:length((31:length(stock1)))


for(j in (30:(length(stock1)-1))){
  stock1.g.1=garchFit(~arma(0,0)+garch(1,1), stock1[1:(j)], trace=FALSE)

  error.dis=stock1[1:j]/stock1.g.1@sigma.t
  sigthat=predict(stock1.g.1,1)$standardDeviation[1]

  #error.dis=residuals(stock1.g.1)
  #error.dis=(error.dis-mean(error.dis))/sd(error.dis)
  gNormQ[j-30+1]=sigthat*qnorm(0.15)

  gNonParamQ[j-30+1]=sigthat*quantile(error.dis, 0.15)

  if(stock1[j+1] < sigthat*qnorm(0.15)){gNorm=gNorm+1}



  if(stock1[j+1] < sigthat*quantile(error.dis, 0.15)){gNonParam=gNonParam+1}

}
gNorm/num
```

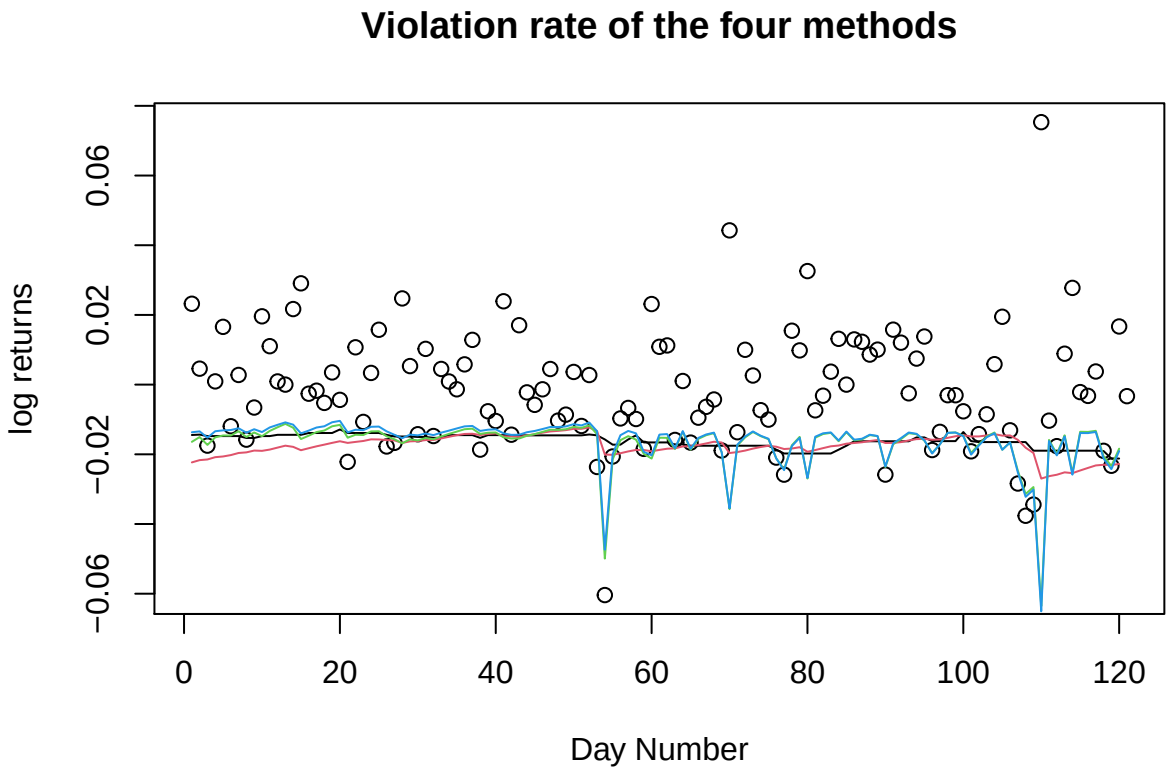
```
## [1] 0.1916667
```

```
gNonParam/num
```

```
## [1] 0.2166667
```

We can see that the risk metrics method has the lowest violation rate, so it is the most accurate for VaR prediction for stock 1. We plot the violation rate along with the log returns in a graph as a summary. The red lines represent the risk metrics method violation rate.

```
plot(stock1[30:150], main = "Violation rate of the four methods", ylab = "log returns",
     xlab = "Day Number")
lines(1:120,histc[1:120])
lines(1:120,Riskc[1:120],col=2)
lines(1:120,gNormQ[1:120],col=3)
lines(1:120,gNonParamQ[1:120],col=4)
```



Since the risk metrics method has the lowest violation rate, we are going to use this method to forecast the 10-steps ahead VaR. Since we are not backtesting now, we can just calculate the empirical VaR based on the 10 steps ahead prediction (we merged the real log returns and the predicted log returns together and treat it as the real log returns).

```
lam <- 0.94
VaR85 <- c()
stock1withPrediction <- c(stock1, garchPred)
for(i in 1:10) {
  sig1 <- mean(stock1withPrediction[1:(149+i)]^2)
  signew <- lam*sig1 + (1-lam)*garchPred[i]^2
  VaR85[i] <- sqrt(signew)*qnorm(.15)
}
VaR85
```

```
##  [1] -0.01818806 -0.01812784 -0.01806821 -0.01800916 -0.01795069 -0.01789279
```

```
##  [7] -0.01783545 -0.01777865 -0.01772240 -0.01766668
```

Now, we have finished forecasting stock 1. We want to forecast all 40 stocks VaR. First, we merged the 40 .txt files into one .csv file and import it. The following shows the first 6 rows of the first 6 stocks.

```
library(tidyverse)
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Financial Risk")
allstocks <- c()
for(i in 1:40) {
  thisStock <- as.numeric(unlist(read.delim2(paste0("stock", i, ".txt"),
                                             header = TRUE, sep = ",")$x))
  allstocks <- cbind(allstocks, thisStock)
  colnames(allstocks)[i] <- paste0("stock", i)
}

head(allstocks[,1:6])
```

```
##           stock1        stock2       stock3       stock4       stock5
## [1,]  0.03200273  0.0147205799 -0.012639087 -0.021429391 -0.012918559
## [2,]  0.04616204  0.0092121864 -0.010138945 -0.011410912  0.056862862
## [3,]  0.01068386  0.0009647854 -0.012034912  0.003644888 -0.009537239
## [4,] -0.00639661  0.0308593203  0.003134097 -0.012552466 -0.017629117
## [5,]  0.01906837  0.0111577169  0.005349990 -0.032088315 -0.029105663
## [6,] -0.01267176 -0.0074246281  0.008853533 -0.052438114  0.009406301
##             stock6
## [1,]  0.0131566175
## [2,]  0.0138600833
## [3,] -0.0251105500
## [4,]  0.0130211622
## [5,]  0.0003316566
## [6,] -0.0075446935
```

Now, we loop through all 40 stocks and record the violation rate for each of them.

```
violation <- data.frame("historical", "Risk Metrics", "Garch Normal Innovations",
                        "GARCH Bootstrap Innovations")
for(i in 1:40) {
  stock <- allstocks[, i]
  num = length(31:length(stock))

  histc = 1:num
  historic=0
  for(j in (30:(length(stock)-1))){
    cut <- quantile(stock[(j-30):j], 0.15)
    histc[j-30+1]=cut
    if(stock[j+1] < cut){historic=historic+1}
  }
  historic/num


  ###############################################################
  Riskc= 1:num
```

```
  RiskMetrics=0
  lam <- 0.94
  sig1 <- mean(stock[1:29]^2)
  for(j in (30:(length(stock)-1))){
    signew= lam*sig1 + (1-lam)*stock[j]^2
    sig1=signew

    Riskc[j-30+1]=sqrt(signew)*qnorm(.15)

    if(stock[j+1] < sqrt(signew)*qnorm(0.15)){RiskMetrics=RiskMetrics+1}
  }
  RiskMetrics/num

  ############################################################
  num=length((31:length(stock)))
  gNorm=0
  gNormQ= 1:length((31:length(stock)))


  gNonParam=0
  gNonParamQ=1:length((31:length(stock)))


  for(j in (30:(length(stock)-1))){
    stock.g.1=garchFit(~arma(0,0)+garch(1,1), stock[1:(j)], trace=FALSE)

    error.dis=stock[1:j]/stock.g.1@sigma.t
    sigthat=predict(stock.g.1,1)$standardDeviation[1]

    #error.dis=residuals(stock.g.1)
    #error.dis=(error.dis-mean(error.dis))/sd(error.dis)
    gNormQ[j-30+1]=sigthat*qnorm(0.15)

    gNonParamQ[j-30+1]=sigthat*quantile(error.dis, 0.15)

    if(stock[j+1] < sigthat*qnorm(0.15)){gNorm=gNorm+1}



    if(stock[j+1] < sigthat*quantile(error.dis, 0.15)){gNonParam=gNonParam+1}

  }
  gNorm/num
  gNonParam/num

  violation[nrow(violation)+1, ] <- c(historic/num, RiskMetrics/num, gNorm/num, gNonParam/num)
}
```

Now we get 40 violation rates for each of the method. We want the method with the lowest violation rate in general, so we sum up the violation rates.

```
violation_rate <- violation
violation_rate <- violation_rate[-c(1), ]
violation_rate <- lapply(violation_rate, as.numeric)
```

```
historical <- 0
riskmetrics <- 0
garchnormal <- 0
garchbootstrap <- 0
for(i in 1:40) {
  historical <- historical + violation_rate$X.historical.[i]
  riskmetrics <- riskmetrics + violation_rate$X.Risk.Metrics.[i]
  garchnormal <- garchnormal + violation_rate$X.Garch.Normal.Innovations.[i]
  garchbootstrap <- garchbootstrap + violation_rate$X.GARCH.Bootstrap.Innovations.[i]
}
```

```
historical
```

```
## [1] 7.008333
```

```
riskmetrics
```

```
## [1] 5.533333
```

```
garchnormal
```

```
## [1] 5.683333
```

```
garchbootstrap
```

```
## [1] 6.716667
```

We can see that the risk metrics method's violation rates sum is significantly lower than others. Therefore, we will use the risk metrics method for all of the 40 stocks. Since GARCH(1,1) model produces a very good model in general, we are going to use GARCH(1,1) to make predictions for every stock.

```
allVaR <- c()
lam <- 0.94
for(j in 1:40) {
  curStock <- ts(allstocks[, j])
  mod <- garchFit(~arma(0,0)+garch(1,1), curStock, trace=FALSE)
  curGarchPred <- predict(mod, 10)$meanForecast

  VaR85 <- c()
  stockwithPrediction <- c(curStock, curGarchPred)

  for(i in 1:10) {
    sig1 <- mean(stockwithPrediction[1:(149+i)]^2)
    signew <- lam*sig1 + (1-lam)*stockwithPrediction[150+i]^2
    VaR85[i] <- sqrt(signew)*qnorm(0.15)
  }
  print(VaR85)
  allVaR <- cbind(allVaR, VaR85)
  colnames(allVaR)[j] <- paste0("stock", j)
}
```

At last, we store the 10 steps ahead VaR forecast in a file.

```
last.name <- "Chen"
student.id <- 20756546
write.table(allVaR, file = paste("Scenario2_", last.name, student.id,".txt", sep = "")
            , sep ="," , col.names = F, row.names = F )
```

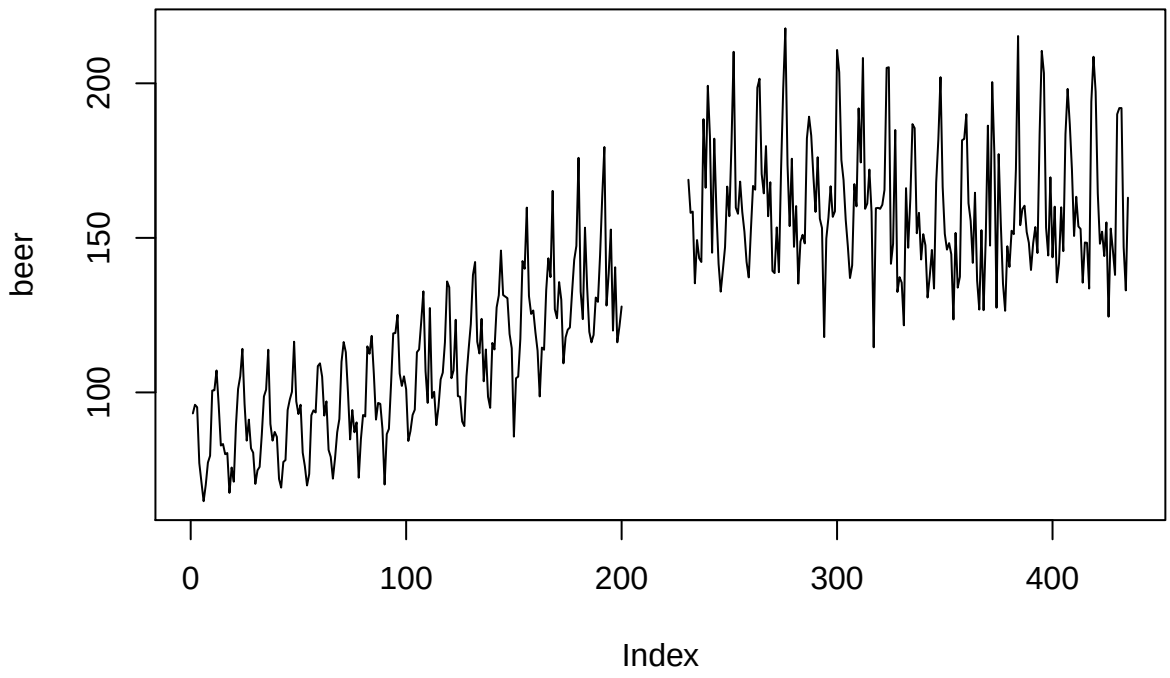
Imputation & Multivariate

Firstly, we are going to import all the data:

```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Imputation")
beer <- as.numeric(unlist(read.delim2("prod_target.txt", header = TRUE, sep = ",")$V2))
car <- as.numeric(unlist(read.delim2("prod_1.txt", header = TRUE, sep = ",")$V2))
steel <- as.numeric(unlist(read.delim2("prod_2.txt", header = TRUE, sep = ",")$V2))
gas <- as.numeric(unlist(read.delim2("eng_1.txt", header = TRUE, sep = ",")$V2))
electricity <- as.numeric(unlist(read.delim2("eng_2.txt", header = TRUE, sep = ",")$V2))
temp <- as.numeric(unlist(read.delim2("temp.txt", header = TRUE, sep = ",")$X20.4))
```

Our first goal is to impute the time series of monthly beer production in Australia. I am going to plot the time series first:

```
plot(beer, type = "l")
```



We can see from the plot that there is clearly a seasonal trend. Therefore, we use the function findfrequency() to find the frequency of the beer data:

```
library("forecast")
beer <- ts(beer, frequency = findfrequency(beer))
findfrequency(beer)
```

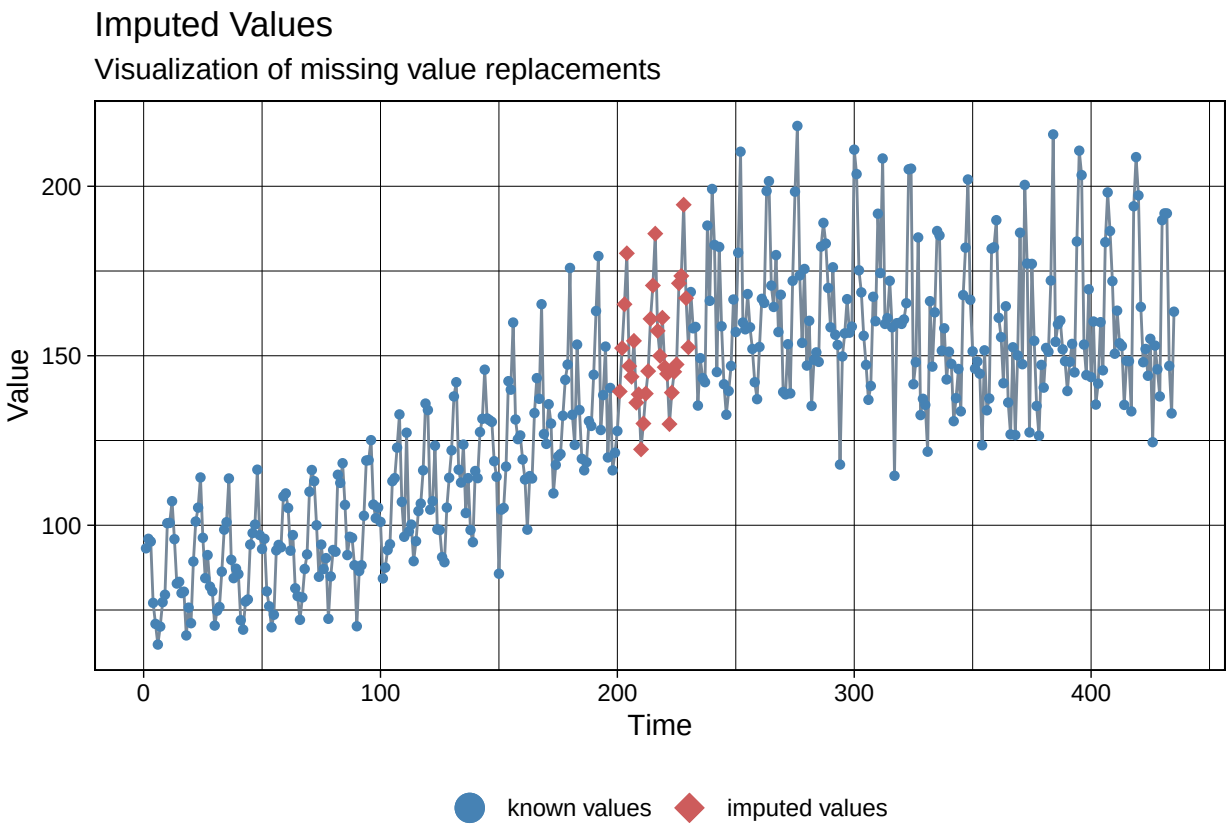
```
## [1] 12
```

Next, I am going to find the best model using auto.arima:

```
library(imputeTS)
x <- auto.arima(beer, max.p = 20, max.q = 20, max.P = 20, max.Q = 20, max.d =20,
                max.D = 24, seasonal = TRUE)
```

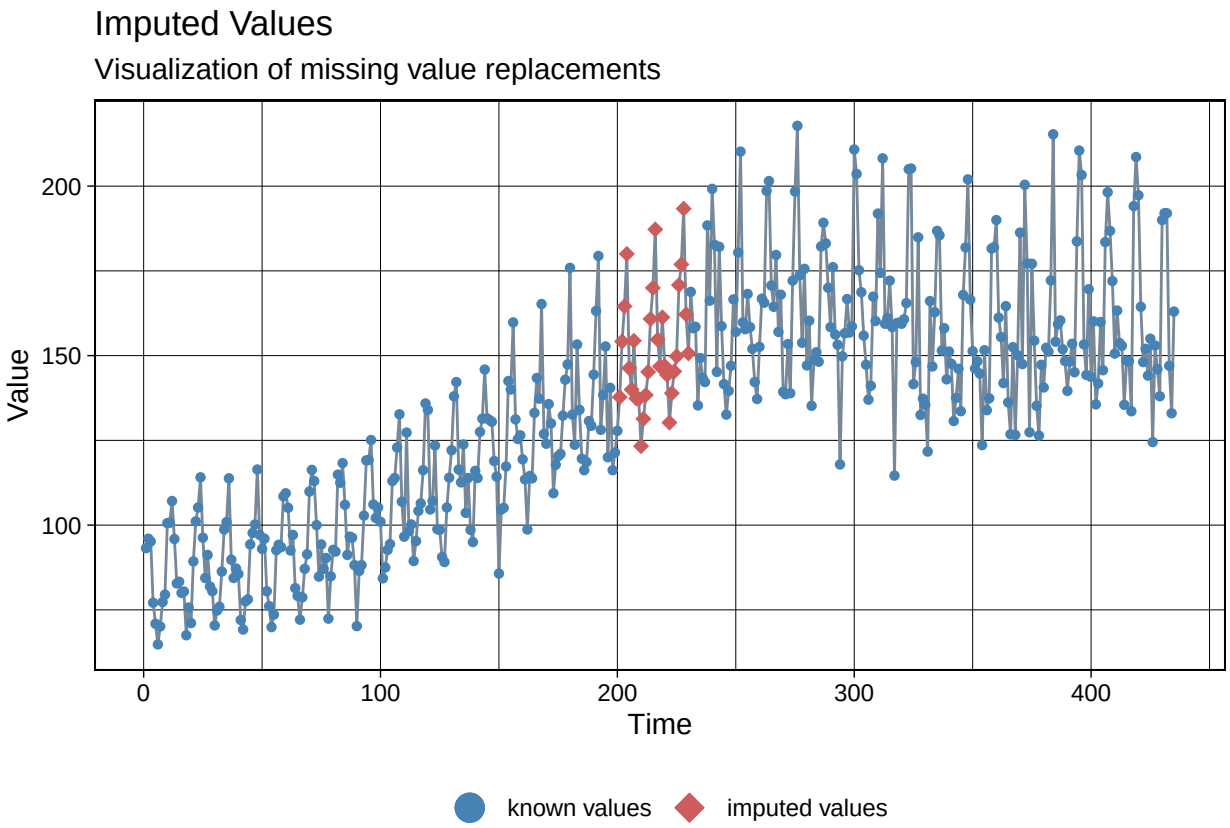
The auto.arima() gives that ARIMA(3,1,1)(0,1,4)[12] is the best model. Therefore, I am going to use the model to do imputation.

```
usermodel = x$model
imp.beer <- na_kalman(beer, model = usermodel)
ggplot_na_imputations(beer, imp.beer)
```



From the above imputed graph, our imputation looks very reasonable (with a reasonable cycle and values).Next, I am going to use the auto.arima model to impute the data again.

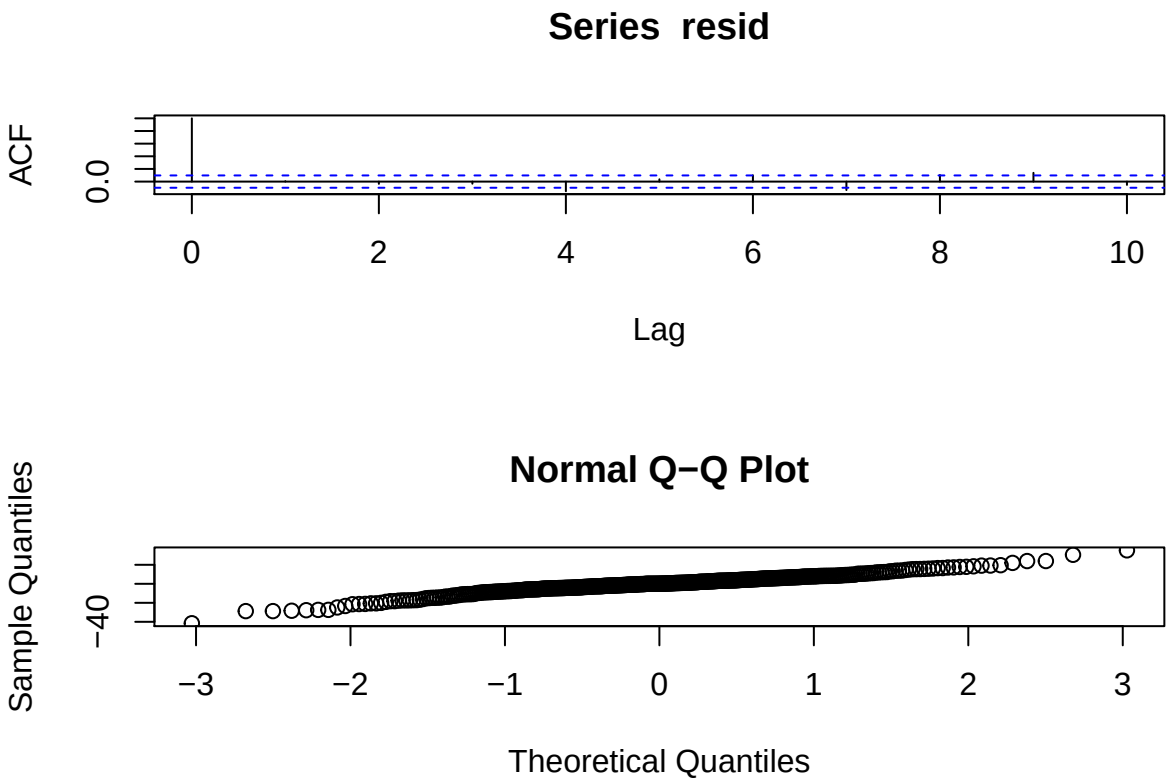
```
imp.beer2 <- na_kalman(beer, model = "auto.arima")
ggplot_na_imputations(beer, imp.beer2)
```



The auto.arima model also did a very good job on imputation. However, for the sake of simplicity, I am just going to use the imputed values from ARIMA(3,1,1)(0,1,4)[12] model.

Next, let's plot the residuals of the ARIMA(3,1,1)(0,1,4)[12] model, its Q-Q plot, its ACF. I am also going to test it with Ljung-Box test.

```
resid <- na_remove(x$residuals)
par(mfrow=c(2,1))
acf(resid, lag = 10)
qqnorm(resid)
```

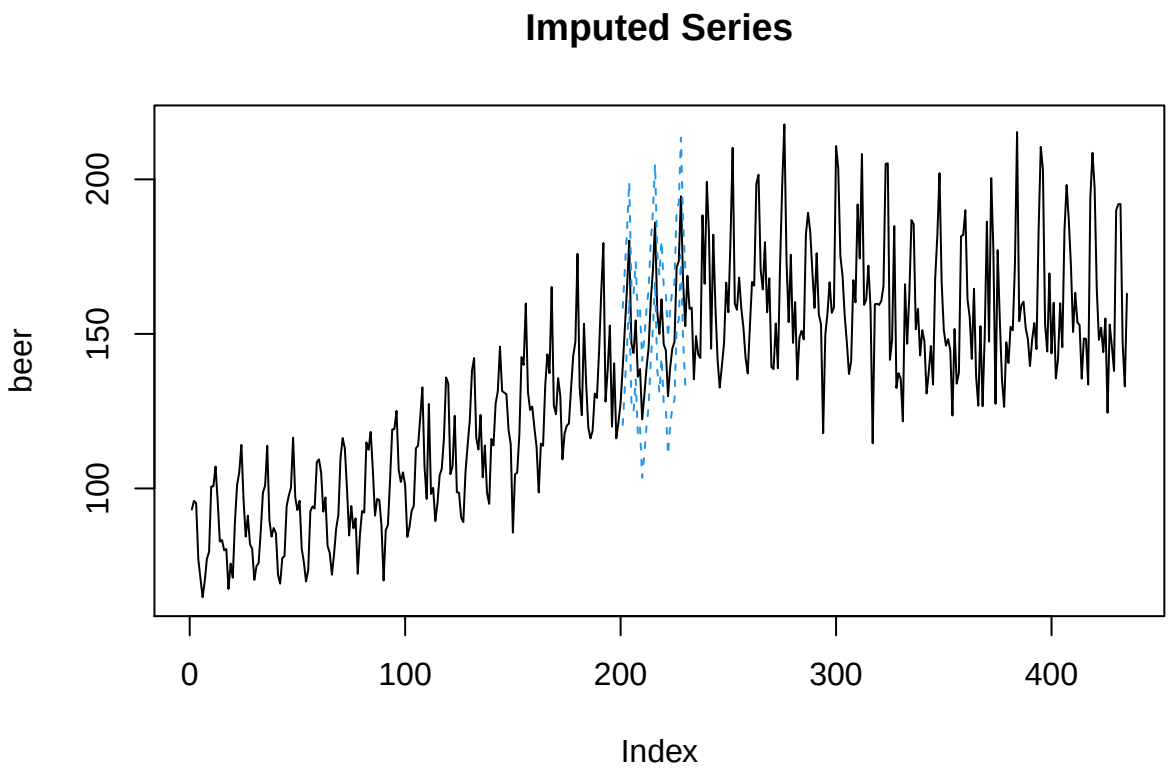



```
Box.test(resid, lag = 5)
```

We can see from the ACF that almost all the straight lines are between the two dashed-blue lines. The normal Q-Q plot shows that the residuals are normally distributed in general. The Ljung-Box test p-value $= 0.05204 > 0.05$, which means that there is no evidence against the null hypothesis that the residuals are strong white noise. This confirmed that our model fits very well to the data.

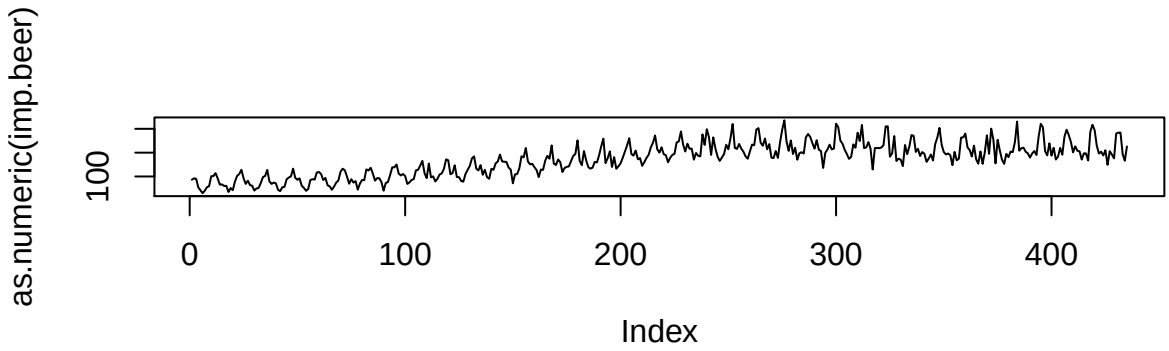
Next, I am going to plot the imputed data with 95% confidence interval.

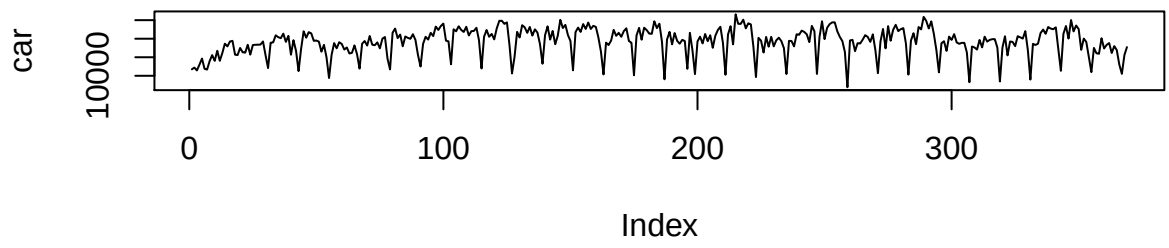
```
plot(as.numeric(imp.beer), type = 'l', main = "Imputed Series", ylab = "beer")
lines(201:230, imp.beer[201:230] + 2*sd(resid), col=4, lty=2)
lines(201:230, imp.beer[201:230] - 2*sd(resid), col=4, lty=2)
```



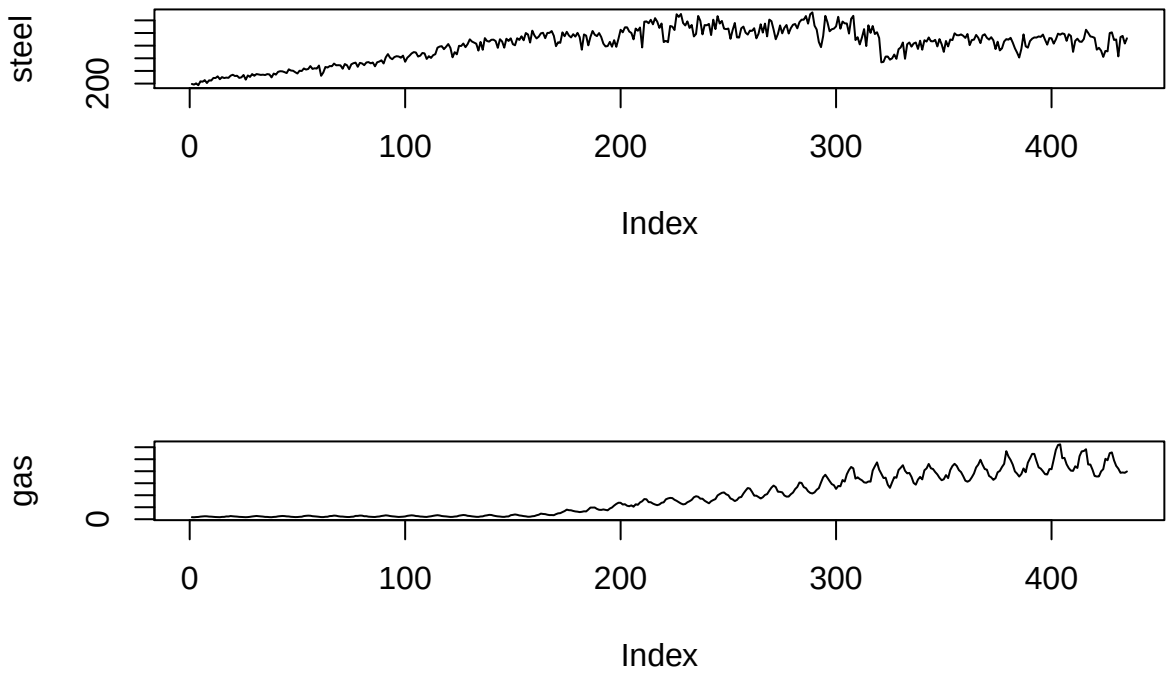

```
ts.beer <- ts(as.numeric(imp.beer), frequency = 12)
ts.car <- ts(car, frequency = 12)
ts.steel <- ts(steel, frequency = 12)
ts.gas <- ts(gas, frequency = 12)
ts.electricity <- ts(electricity, frequency = 12)
ts.temp <- ts(temp[147:581], frequency = 12)
```

```
par(mfrow=c(2,1))
plot(as.numeric(imp.beer), type = "l")
plot(car, type = "l")
```

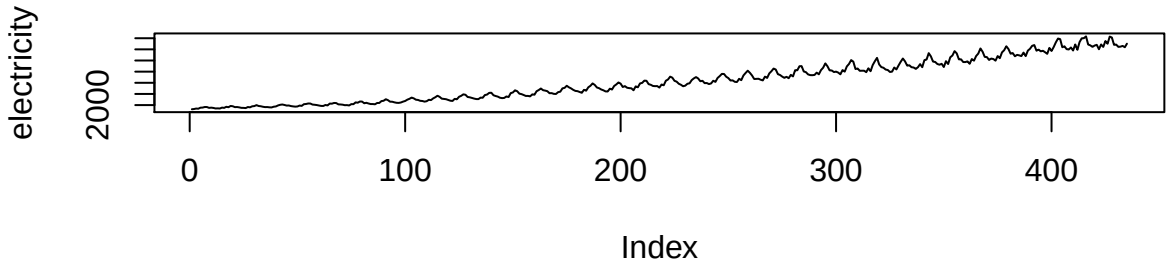


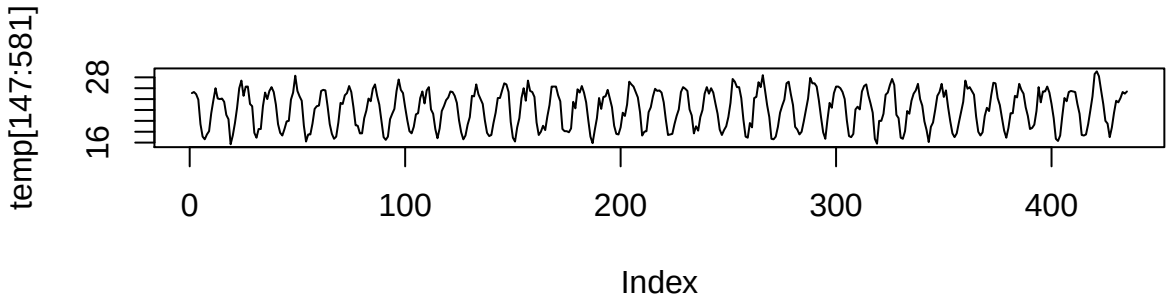



```
plot(steel, type = "l")
plot(gas, type = "l")
```

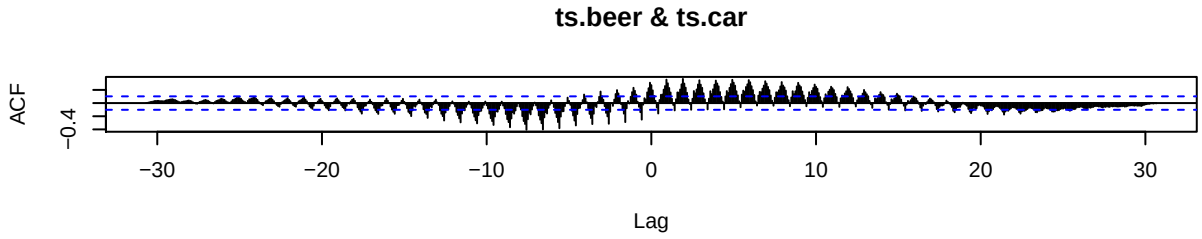


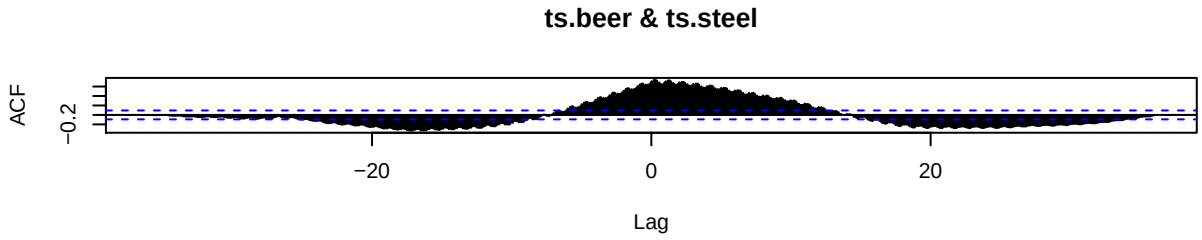

```
plot(electricity, type = "l")
plot(temp[147:581], type = "l")
```

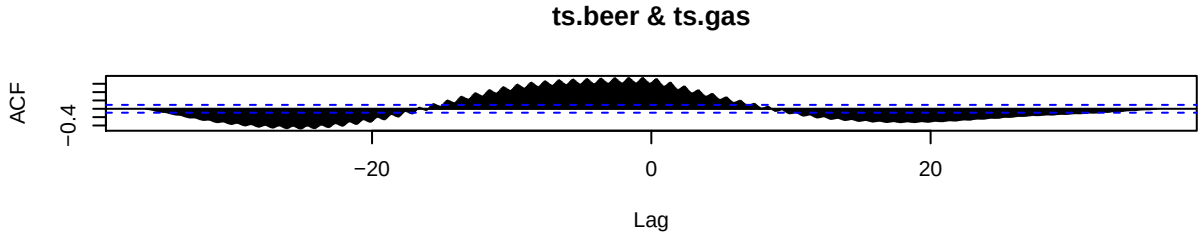




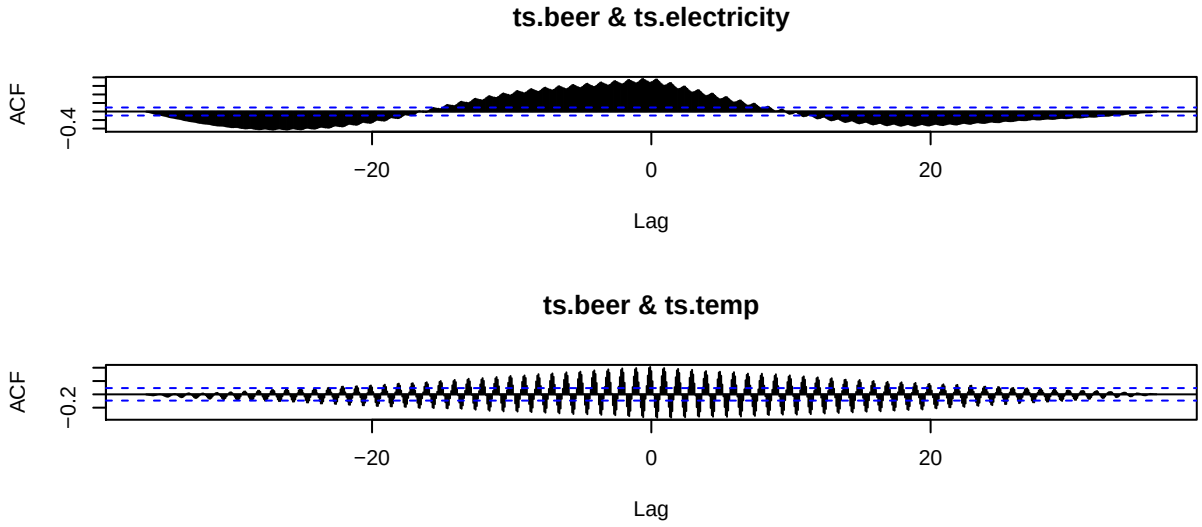

```
par(mfrow=c(3,1))
ccf(ts.beer, ts.car, lag = 500)
ccf(ts.beer, ts.steel, lag = 500)
ccf(ts.beer, ts.gas, lag = 500)
```








```
ccf(ts.beer, ts.electricity, lag = 500)
ccf(ts.beer, ts.temp, lag = 500)
```



From the cross-correlation plots of beer production and the other five dataset, we can see that there is cyclical behavior in every CCF plot. It's reasonable that all of the five datasets are correlated to beer production. The car/steel production, gas/electricity consumption are all periodic data with periods of 12 months. However, those data should be independent of beer production. For example, during summer time, the sales of ice cream and sunglasses will rise. However, we can't say that the sales of ice cream is depend on the sales of sunglasses, or vice versa.

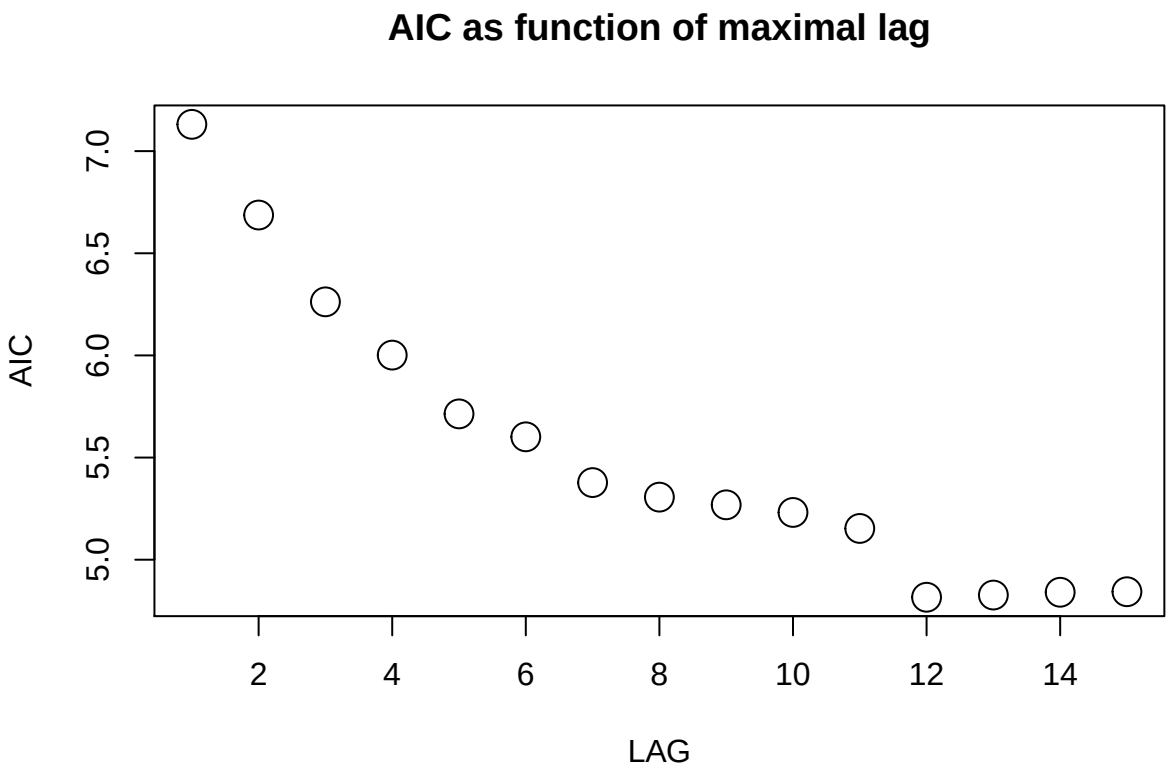
The beer/car/steel production, gas/electricity consumption probably all depend on the season of the year, but they don't depend on each other. However, beer production probably depends on the temperature, since higher temperature usually cause people to drink more beer. Therefore, I am going to use only the temperature series in vector autoregression followed.

First, we combine the dataset beer and temperature.

```
library(vars)
dat.mat.full <- cbind(ts.beer, ts.temp)
```

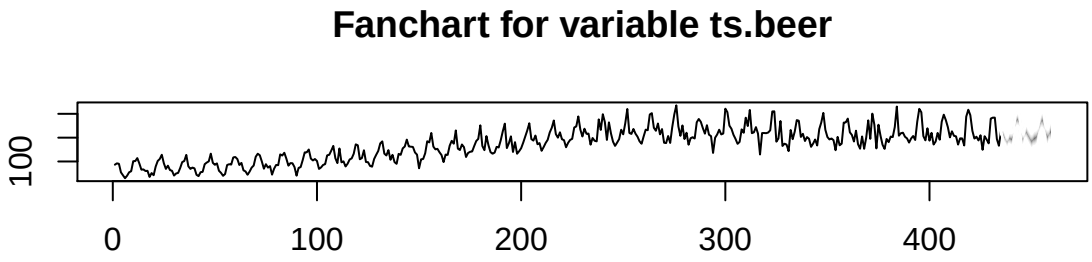
Then, we fit the VAR model and iterate through lag 1 to lag 15 and plot the AIC values of the models.

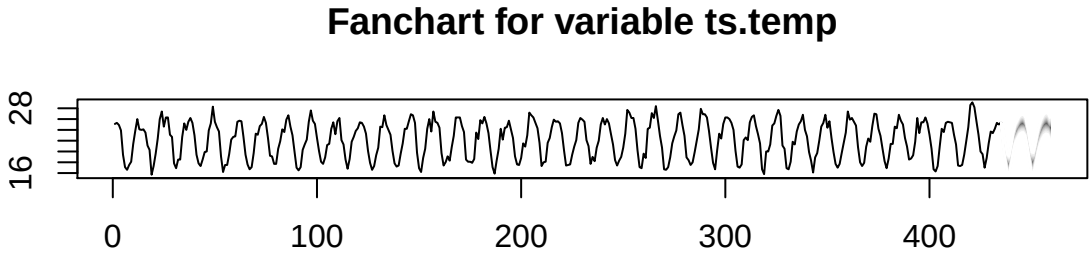
```
beer.mod <- VARselect(dat.mat.full, lag.max = 15)$criteria[1,]
plot(beer.mod, main="AIC as function of maximal lag", xlab= "LAG", ylab="AIC",cex=2)
```



We can see that AIC values are the lowest at lag 12. THerefore, we are going to choose this model. Then, we are going to fit the model with lag 12 and plot the predictions:

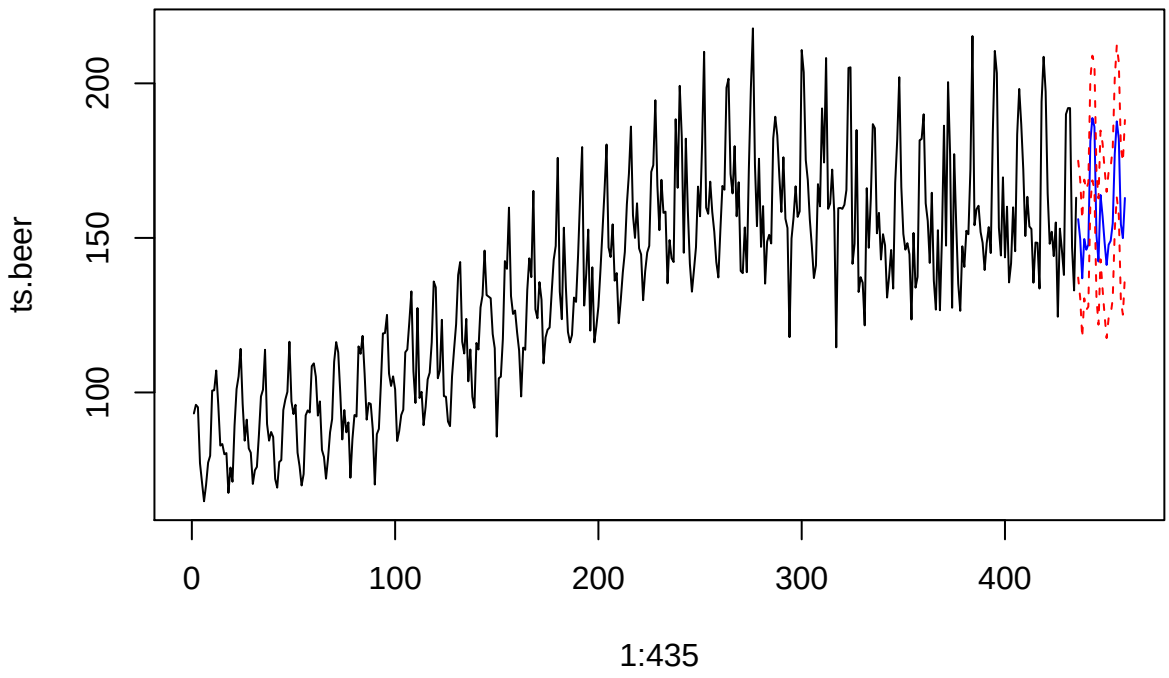
```
var_beer <- VAR(dat.mat.full, p = 12, type = "const")
var_beer_prd <- predict(var_beer, n.ahead = 24, ci = 0.95)
fanchart(var_beer_prd)
```





We can see that the predictions look very plausible. It demostrate a similar cyclical behaviour as the previous data. Next, I am going to plot the 24 steps ahead predictions and 95% confidence interval:

```
plot(x = 1:435, y = ts.beer, xlim = c(0, 460), type = 'l')
lines(x = 436:459, y = var_beer_prd$fcst[1]$ts.beer[,1], col = 'blue')
lines(x = 436:459, y = var_beer_prd$fcst[1]$ts.beer[,2], col = 'red', lty = 2)
lines(x = 436:459, y = var_beer_prd$fcst[1]$ts.beer[,3], col = 'red', lty = 2)
```



At last, let's use Portmanteau test to test the model

```
serial.test(var_beer, lags.pt = 150)
```

The p-value = 0.2022 > 0.05, which means that the model fits very well with the data.

```
last.name <- "Chen"
student.id <- 20756546
imputation3 = imp.beer[201:230]   #imputation of length 30

write(imputation3, file = paste("Scenario3_",last.name,student.id,".txt", sep = ""), ncolumns = 1 )

#Scenario 4

forecast4 = var_beer_prd$fcst[1]$ts.beer[,1] #forecast of length 24

write(forecast4, file = paste("Scenario4_",last.name,student.id,".txt", sep = ""), ncolumns = 1 )
```

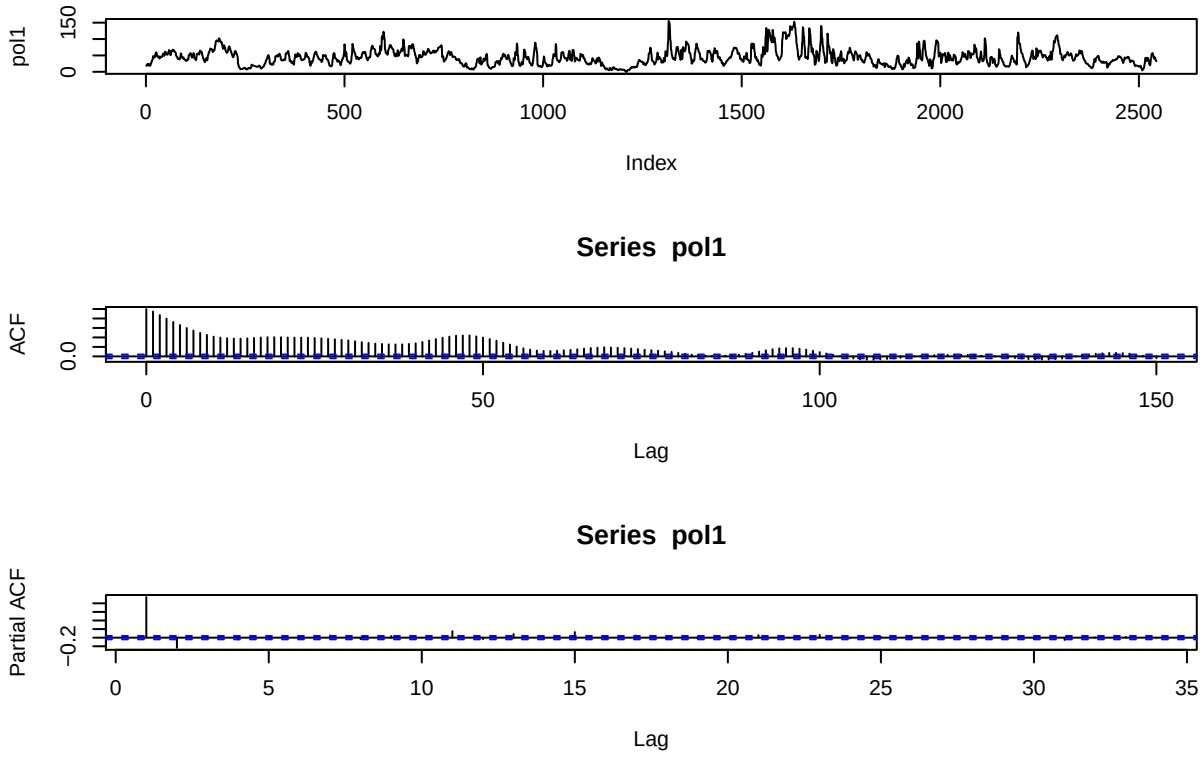
Pollution

First, let's import the dataset for city1.

```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Pollution/")
pol1 <- read.delim2("pollutionCity1.txt", header = TRUE, sep = ",")
pol1 <- as.numeric(unlist(pol1$x))
```

Let's plot the data for city1, its ACF and PACF.

```
library(forecast)
par(mfrow=c(3,1))
plot(pol1, type = 'l')
acf(pol1, lag.max = 150)
pacf(pol1)
```



We can see that there is strong serial correlation indicating by the ACF plot. However, it's hard to tell what is the frequency of the dataset exactly. Therefore, we use the findfrequency() function and get that the frequency of the time series is 28.

```
library("forecast")
pol1 <- ts(pol1, frequency = findfrequency(pol1))
findfrequency(pol1)
```

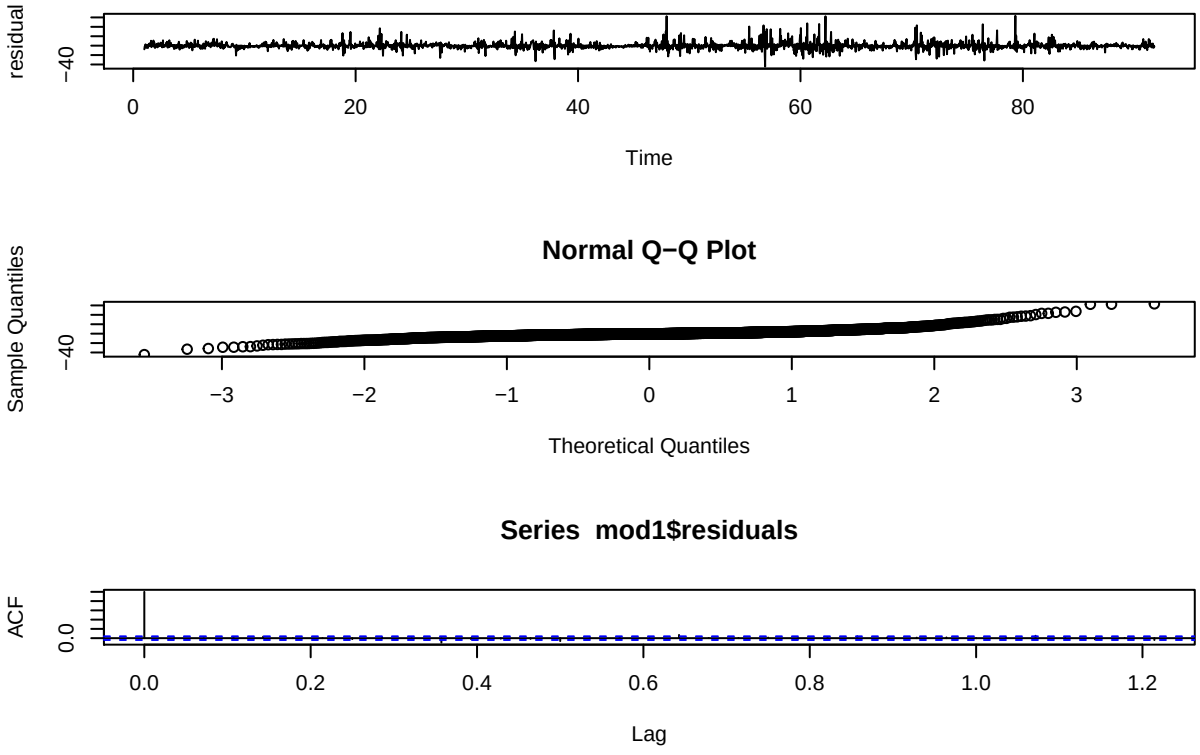
```
## [1] 28
```

I have tried to fit GARCH model to it, but the Ljung-Box test p-value is very small, which means GARCH model doesn't fit our dataset. I have also tried to fit ETS model to it, but the prediction interval of the ETS model blows up. The most suitable model I have found it ARIMA model. Since the dataset records the half-hourly data, we have 48 data points in a day. Therefore, we set the maximal seasonal component as 48.

```
mod1 <- auto.arima(pol1, max.p = 10, max.q = 10,
                   max.P = 10, max.Q = 10, max.d = 5, max.D = 48,
                   seasonal = TRUE)
```

Let's plot the residual of the model and test it with Ljung-Box test.

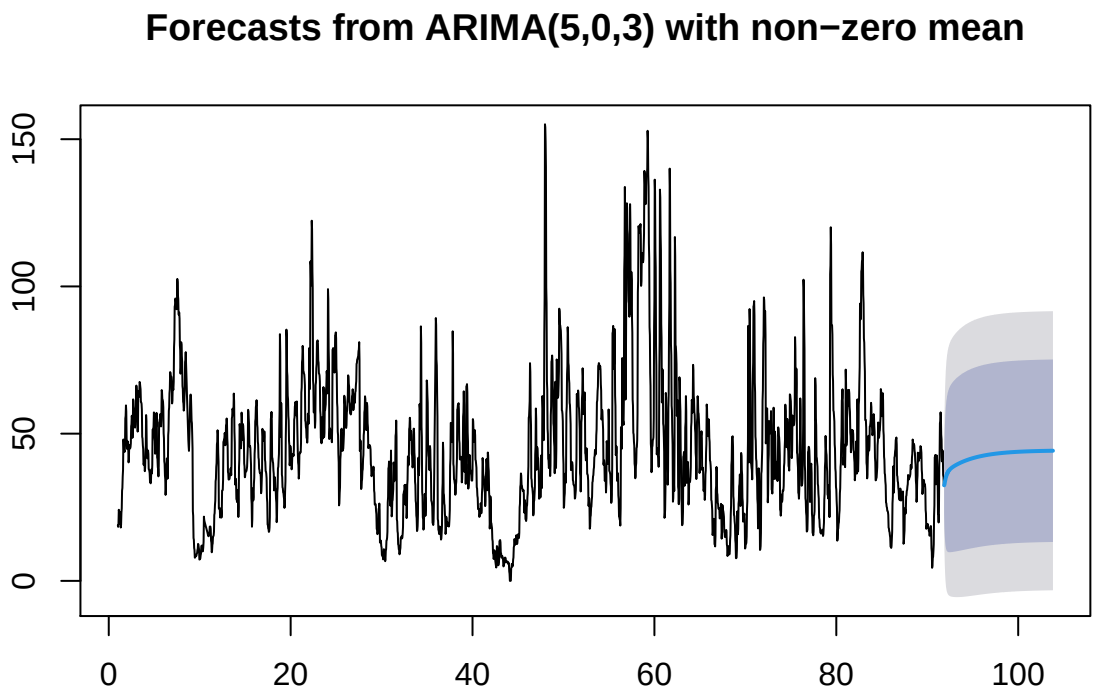
```
par(mfrow=c(3,1))
plot(mod1$residuals, ylab = "residual")
qqnorm(mod1$residuals)
acf(mod1$residuals)
```



```
Box.test(mod1$residuals)
```

From the plot of the residual, the ACF plot and the Q-Q plot, we can see that it's very likely that the residuals are white noise. Almost all the straight lines in the ACF plot are between the two dashed-blue lines. The Q-Q plot of the residuals shows that it's normally distributed. Also, according to the Ljung-box test p-value $= 0.9173 > 0.05$, so there is no evidence against the null hypothesis that the residuals are white noise. Therefore, our model fits very well with the data. Let's plot the forecast and its 95% confidence interval.

```
plot(forecast(mod1, h = 336))
```



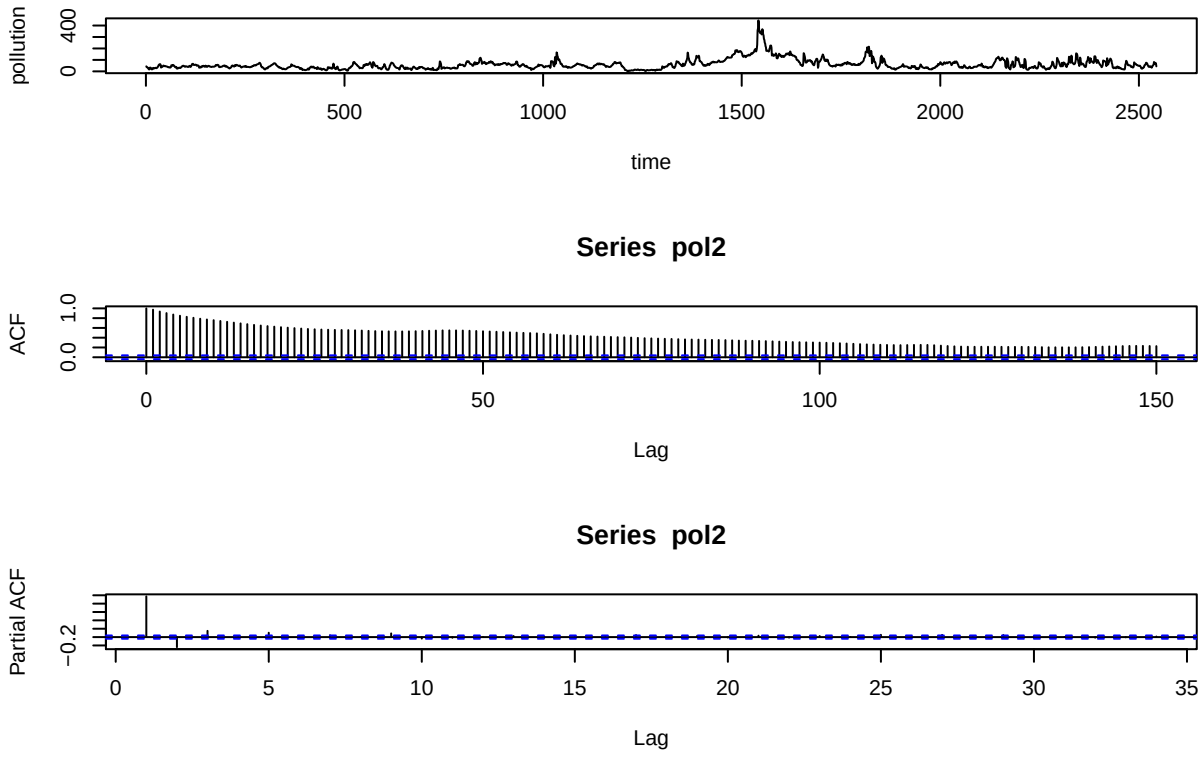
The confidence interval is in reasonable range and gets wider as time increases.

Secondly, let's import the dataset for city2.

```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Pollution/")
pol2 <- read.delim2("pollutionCity2.txt", header = TRUE, sep = ",")
pol2 <- as.numeric(unlist(pol2$x))
```

Now, let's take a look at the plot of the second pollution dataset, its ACF and PACF.

```
par(mfrow=c(3,1))
plot(pol2, type = 'l', ylab = "pollution", xlab = "time")
acf(pol2, lag.max = 150)
pacf(pol2)
```

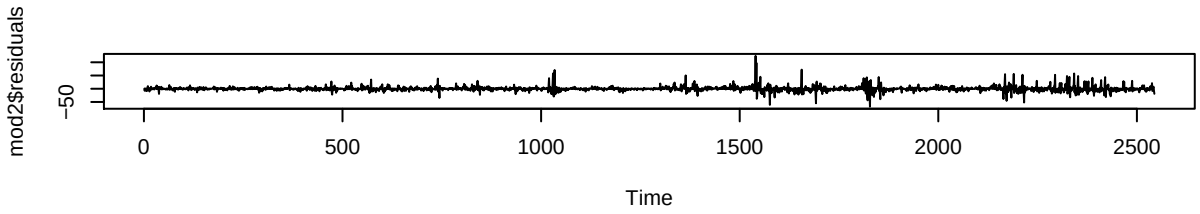


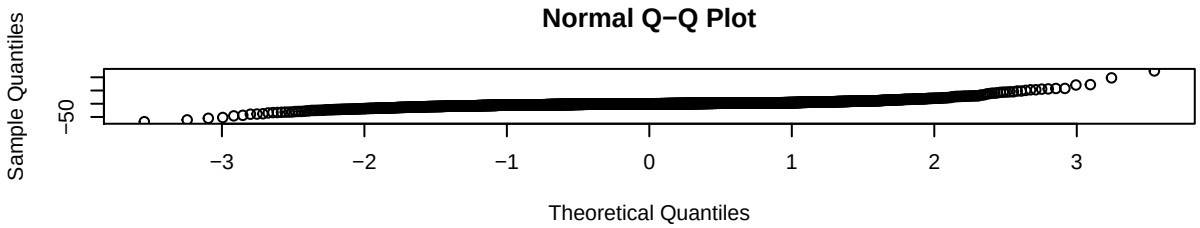
According to the ACF plot, there is strong serial correlation in the time series, but there isn't any seasonal trend. According to the PACF plot, there are several straight lines go beyond the blue-dashed lines, which suggests that ARMA model might fit the data. Therefore, let's fit ARIMA model to it:

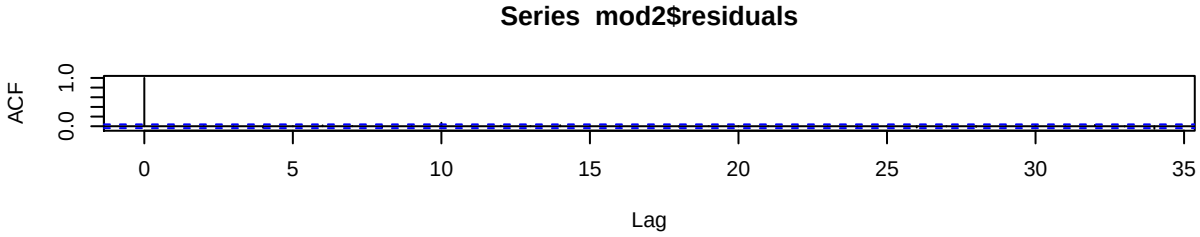
```
mod2 <- auto.arima(pol2, max.p = 10, max.q = 10,
                   max.P = 10, max.Q = 10, max.d = 5, max.D = 48,
                   seasonal = FALSE)
```

Let's plot the residual of the model and test it with Ljung-Box test.

```
par(mfrow=c(3,1))
plot(mod2$residuals)
qqnorm(mod2$residuals)
acf(mod2$residuals)
```



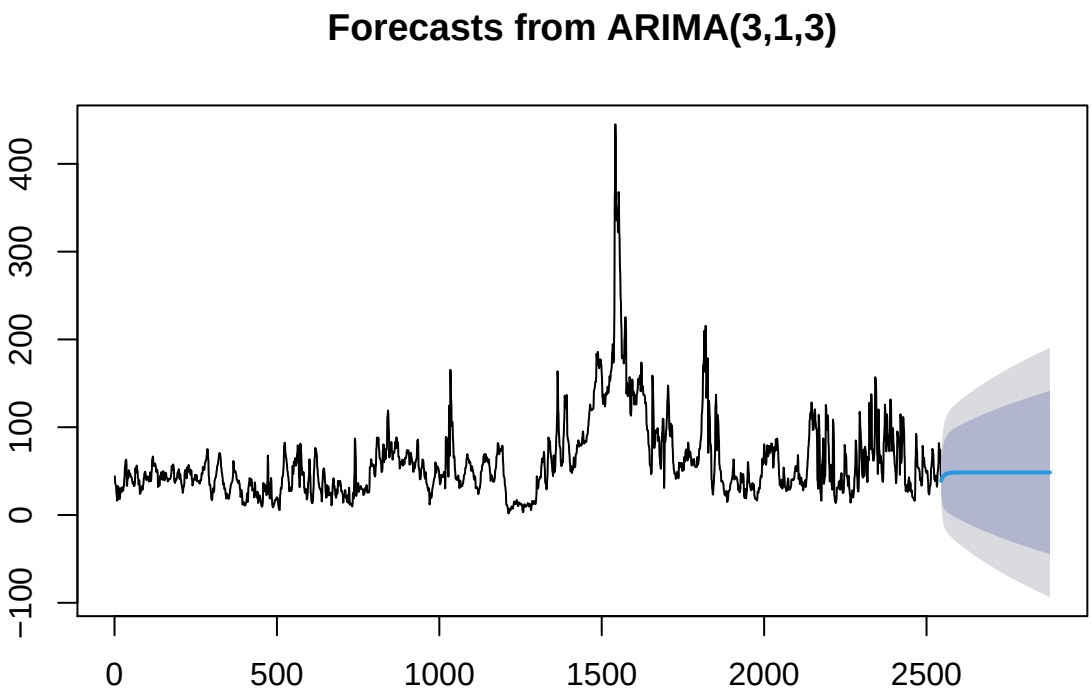





```
Box.test(mod2$residuals)
```

From the plot of the residual the Q-Q plot and the ACF plot, we can see that it's very likely that the residuals are white noise. Almost all the straight lines in the ACF plot are between the two dashed-blue lines. The Q-Q plot of the residuals shows that it's normally distributed. Also, according to the Ljung-box test p-value $= 0.9631 > 0.05$, so there is no evidence against the null hypothesis that the residuals are white noise. Therefore, our model fits very well with the data. Let's plot the forecast and its 95% confidence interval.

```
plot(forecast(mod2, h = 336))
```



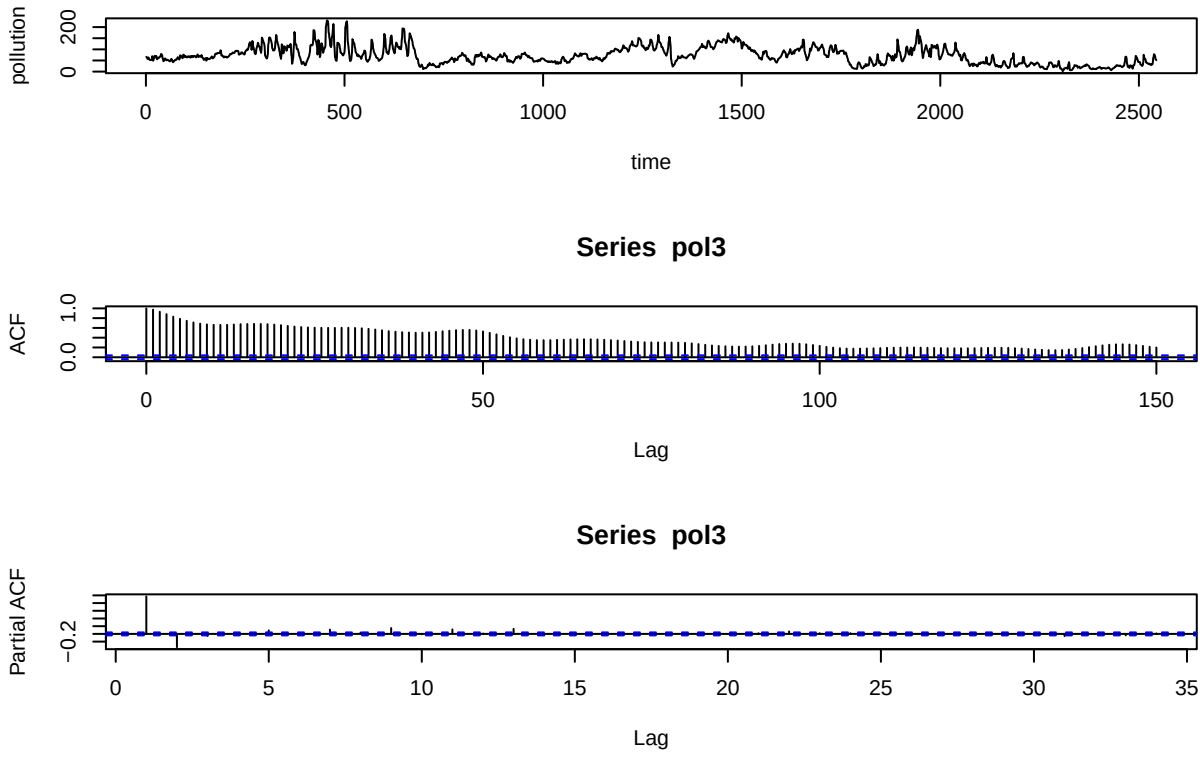
The confidence interval is in reasonable range and gets wider as time increases. I have also tried ETS model and GARCH model, but they are pretty bad, just like in the first dataset. Therefore, the best model I found is ARIMA(3,1,3).

Thirdly, let's import the dataset for city3.

```
setwd("~/Desktop/Winter 2020 3B/STAT 443/Forecasting Competition/Pollution/")
pol3 <- read.delim2("pollutionCity3.txt", header = TRUE, sep = ",")
pol3 <- as.numeric(unlist(pol3$x))
```

Now, let's take a look at the plot of the third pollution dataset, its ACF and PACF.

```
par(mfrow=c(3,1))
plot(pol3, type = 'l', ylab = "pollution", xlab = "time")
acf(pol3, lag.max = 150)
pacf(pol3)
```

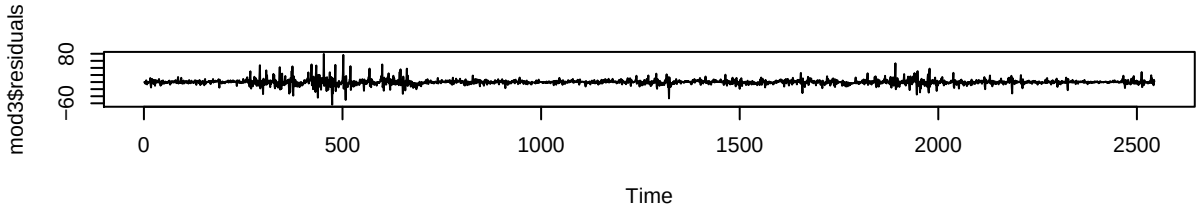


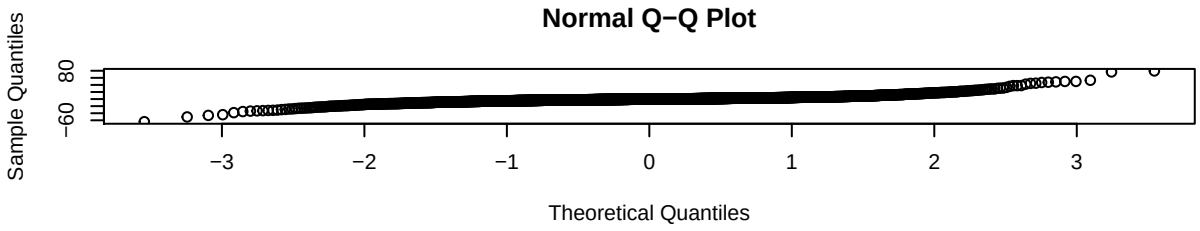
According to the ACF plot, there is strong serial correlation in the time series, but there isn't any seasonal trend. According to the PACF plot, there are several straight lines go beyond the blue-dashed lines, which suggests that ARMA model might fit the data. Therefore, let's fit ARIMA model to it:

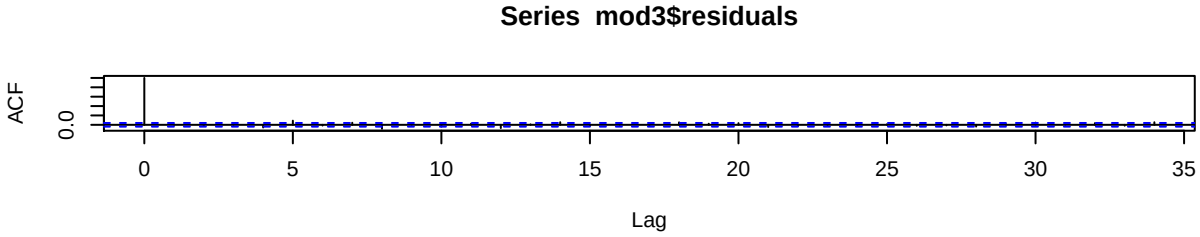
```
mod3 <- auto.arima(pol3, max.p = 10, max.q = 10,
                   max.P = 10, max.Q = 10, max.d = 5, max.D = 5,
                   seasonal = FALSE)
```

Let's plot the residual of the model and test it with Ljung-Box test.

```
par(mfrow=c(3,1))
plot(mod3$residuals)
qqnorm(mod3$residuals)
acf(mod3$residuals)
```



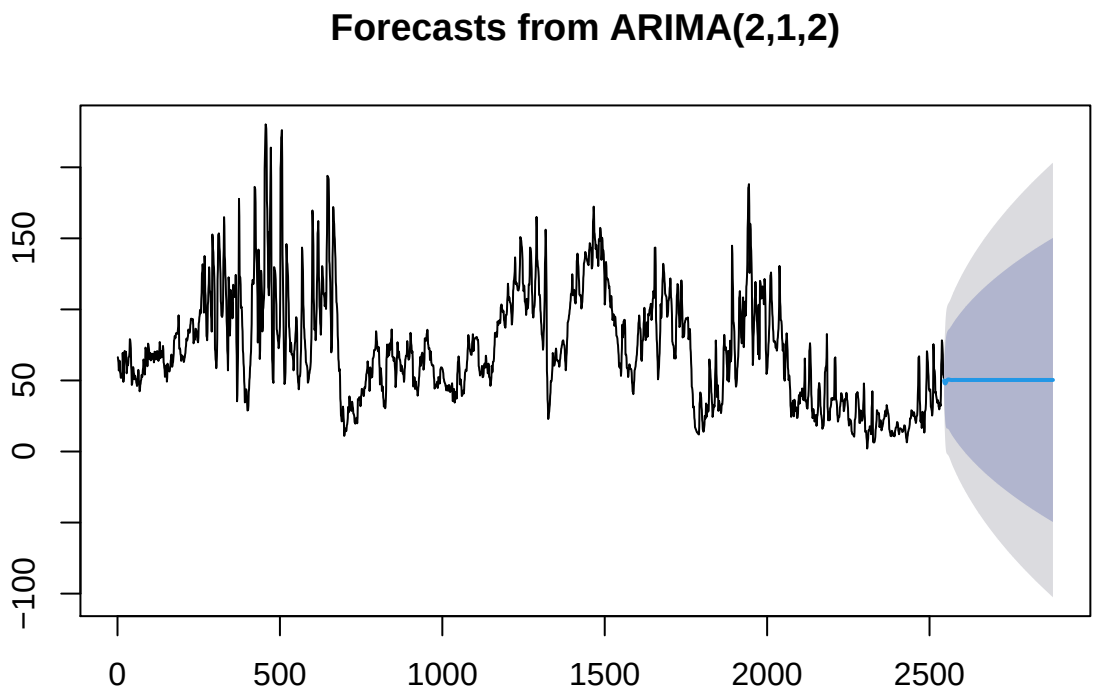





```
Box.test(mod3$residuals)
```

From the plot of the residual the Q-Q plot and the ACF plot, we can see that it's very likely that the residuals are white noise. Almost all the straight lines in the ACF plot are between the two dashed-blue lines. The Q-Q plot of the residuals shows that it's normally distributed. Also, according to the Ljung-box test p-value $= 0.6521 > 0.05$, so there is no evidence against the null hypothesis that the residuals are white noise. Therefore, our model fits very well with the data. Let's plot the forecast and its 95% confidence interval.

```
plot(forecast(mod3, h = 336))
```



The confidence interval is in reasonable range and gets wider as time increases. I have also tried ETS model and GARCH model, but they are pretty bad, just like in the first dataset. Therefore, the best model I found is ARIMA(2,1,2).

```
Q5forecast <- c(forecast(mod1, h = 336)$mean,
                forecast(mod2, h = 336)$mean,
                forecast(mod3, h = 336)$mean)
forecast5 = matrix(Q5forecast,336,3)
last.name <- "Chen"
student.id <- 20756546
write.table(forecast5, file = paste("Scenario5_",last.name,student.id,".txt", sep = ""), sep ="," , col
```