Time Series Analysis 4

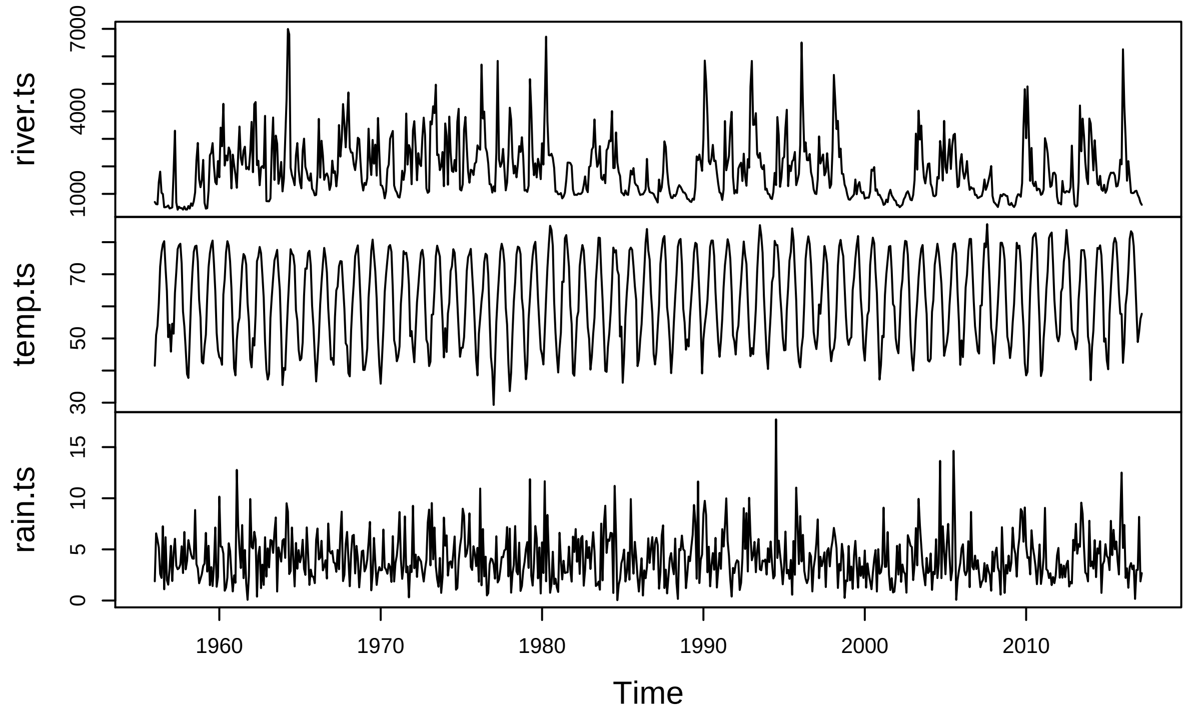
Multivariate time series, Vector autoregression (VAR)

Time Series Analysis
Zhe Zheng

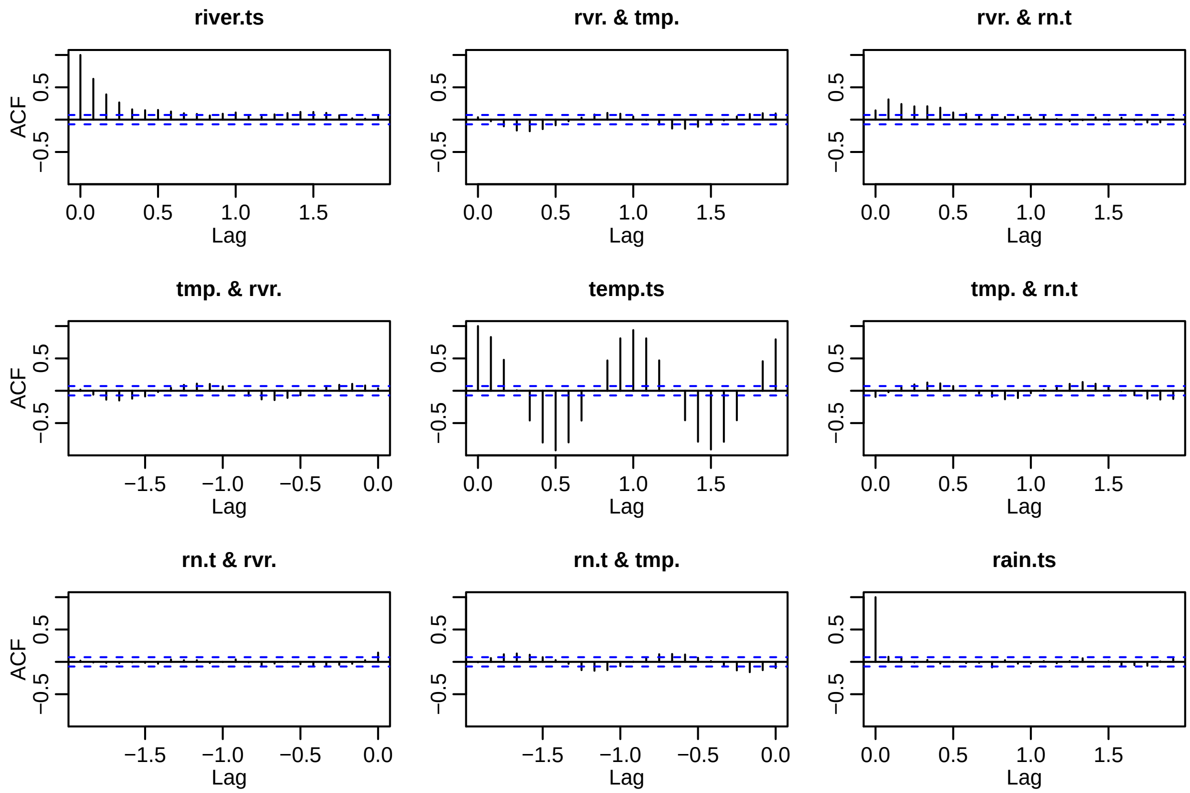
```r
rm(list=ls())                # clearing
library(tseries)
library(forecast)
library(vars)
library(aod)
setwd("E:/1AAAAGatech/Time Series")
temp = read.csv("Temperature.csv",header=T)
rain  = read.csv("Precipitation.csv",header=T)
river = read.csv("RiverFlow.csv",header=F)
```

1. Data exploration and check correlations between different time series

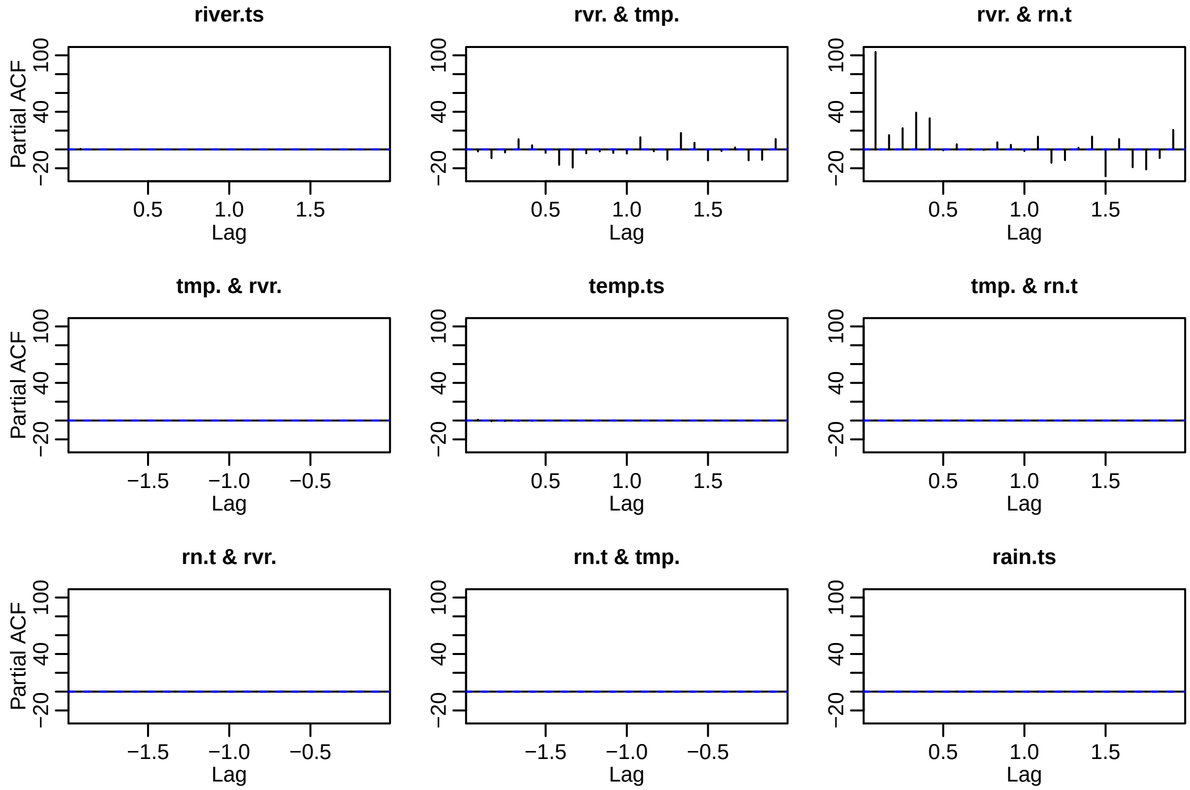
```r
#Fill in missing value in rain for October, 1964
#using the average between September and November
#they are factor data, need to change it to numeric first
# rain[,11]=as.numeric(rain[,11])
# rain[14,11]=0.5*(as.numeric(rain[14,10])+as.numeric(rain[14,12]))

rain[,11]=as.numeric(as.character(rain[,11]))
rain[14,11]=0.5*(as.numeric(as.character(rain[14,10]))+as.numeric(as.character(rain[14,12])))
rain[,11]=as.factor(rain[,11])

## All variables for the same time period
temp = as.vector(t(temp[,-1])) #drop the first column
temp = temp[-c(1:(12*6))]     #drop the first 72 data
temp = temp[-c(736:744)]
rain = as.vector(t(rain[,-1]))
rain = rain[-c(1:(12*6))]
rain = rain[-c(736:744)]
river = as.vector(river[,3])

temp.ts = ts(as.numeric(temp),start=1956, freq=12)
rain.ts = ts(as.numeric(rain),start=1956, freq=12)
river.ts = ts(as.numeric(river),start=1956, freq=12)

data.ts = ts.union(river.ts,temp.ts,rain.ts)
plot(data.ts, type="l",main="")
```




```r
acf(data.ts, mar=c(3.5,3,1.9,0))
```




```r
pacf(data.ts, mar=c(3.5,3,1.9,0))
```




```r
#library(tseries)
adf.test(river.ts, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  river.ts
## Dickey-Fuller = -6.9666, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(rain.ts, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rain.ts
## Dickey-Fuller = -8.7695, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(temp.ts, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
```

```
## data:  temp.ts
## Dickey-Fuller = -8.7844, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

There are some correlation in cross ACF and PACF among different series, and adf.test also show these 3 series are not stationary. So we should do some transformation for the data (differencing, log transformation)

```r
#library(forecast)
#Function to determine differencing order to achieve stationarity
ndiffs(river.ts, alpha = 0.05, test = c("adf"))
```

```
## [1] 0
```

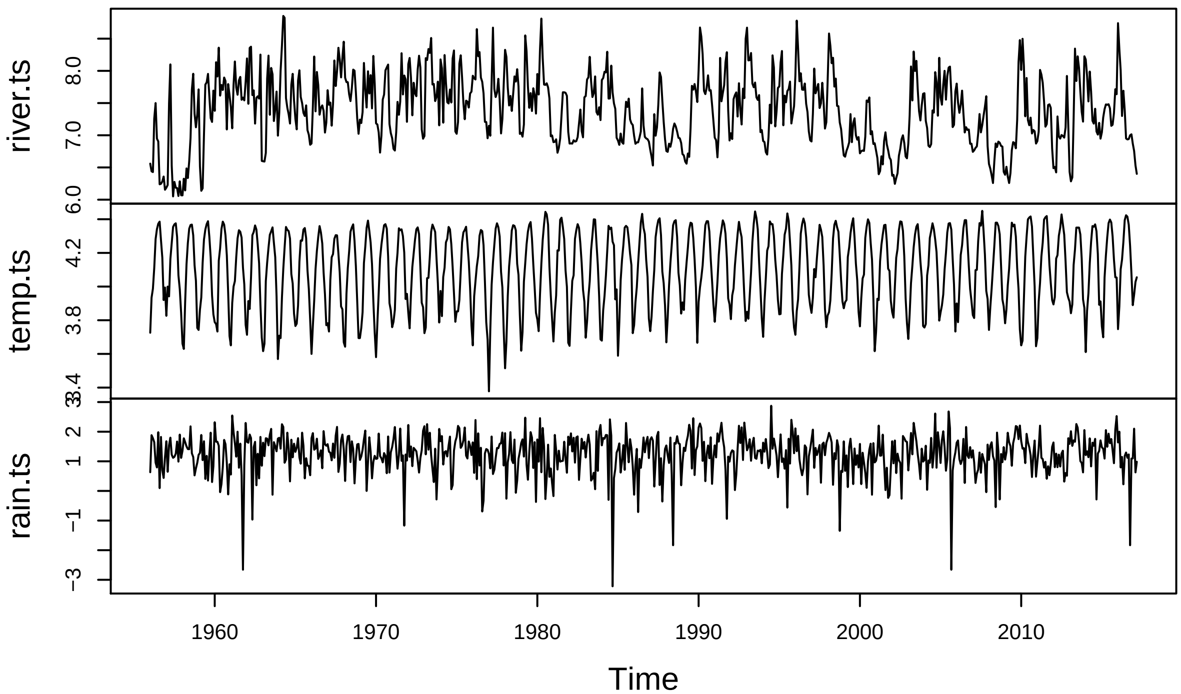
```r
ndiffs(rain.ts, alpha = 0.05, test = c("adf"))
```

```
## [1] 0
```

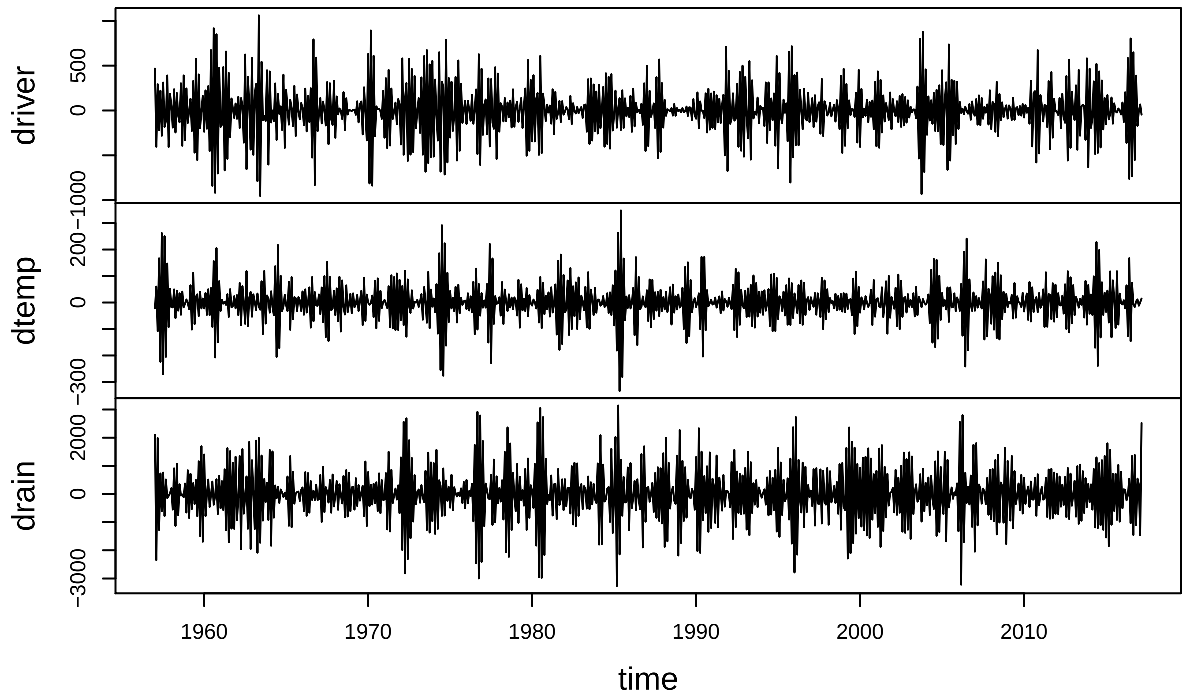
```r
ndiffs(temp.ts, alpha = 0.05, test = c("adf"))
```

```
## [1] 0
```

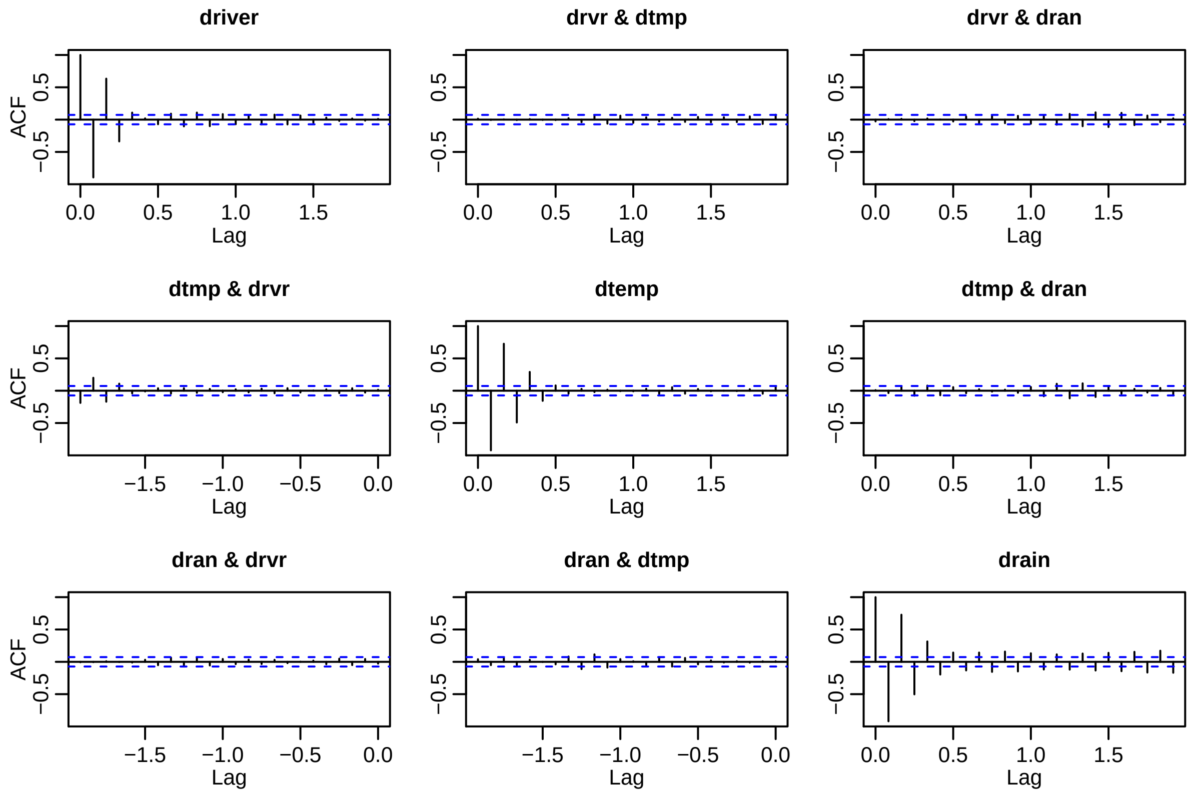
```r
#Log transformation
plot(log(data.ts), type="l",main="")
```



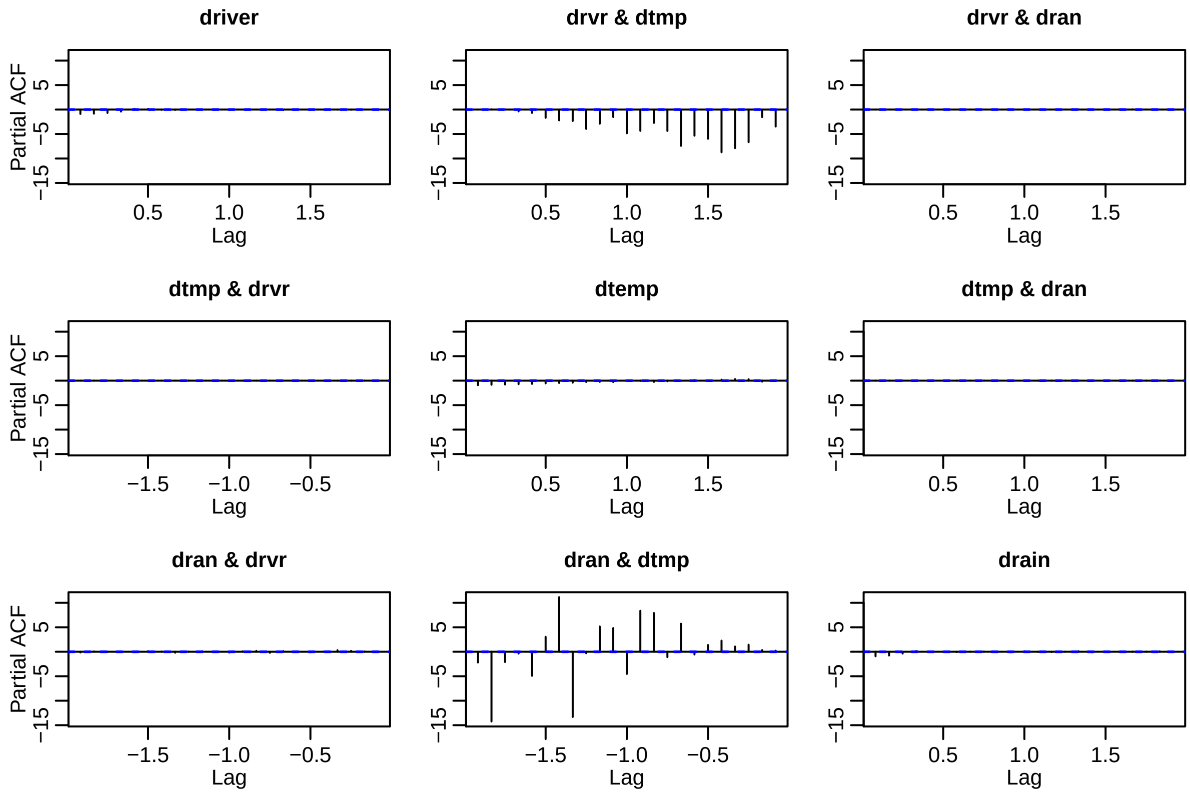
```
#Using seasonal differencing of 12 months and log transformation(make the data more symmetric)
dtemp=diff(log(temp.ts),differences = 12)
drain=diff(log(rain.ts),differences = 12)
driver=diff(log(river.ts),differences = 12)
ddata.ts = ts.union(driver,dtemp,drain)
plot(ddata.ts,xlab="time",main="",type="l")
```




```
acf(ddata.ts, mar=c(3.5,3,1.9,0))
```




```r
pacf(ddata.ts, mar=c(3.5,3,1.9,0))
```



Still there are some correlation in cross ACF and PACF, so we can still borrow some strengths from other series, here comes with VAR

Data preparation for models: Testing Vs Training

```
data=data.ts
n = nrow(data)
## Training data: 1956 to 2015
data.train=data[1:(n-15),]
## Test data: 2016 and 3 months in 2017
data.test=data[(n-14):n,]

ts_river=ts(log(data.train[,"river.ts"]),start=1956, freq=12)
ts_rain=ts(log(data.train[,"rain.ts"]),start=1956, freq=12)
ts_temp=ts(log(data.train[,"temp.ts"]),start=1956, freq=12)


ts_river2=ts(log(data.test[,"river.ts"]),start=2016, freq=12)
ts_rain2=ts(log(data.test[,"rain.ts"]),start=2016, freq=12)
ts_temp2=ts(log(data.test[,"temp.ts"]),start=2016, freq=12)
```

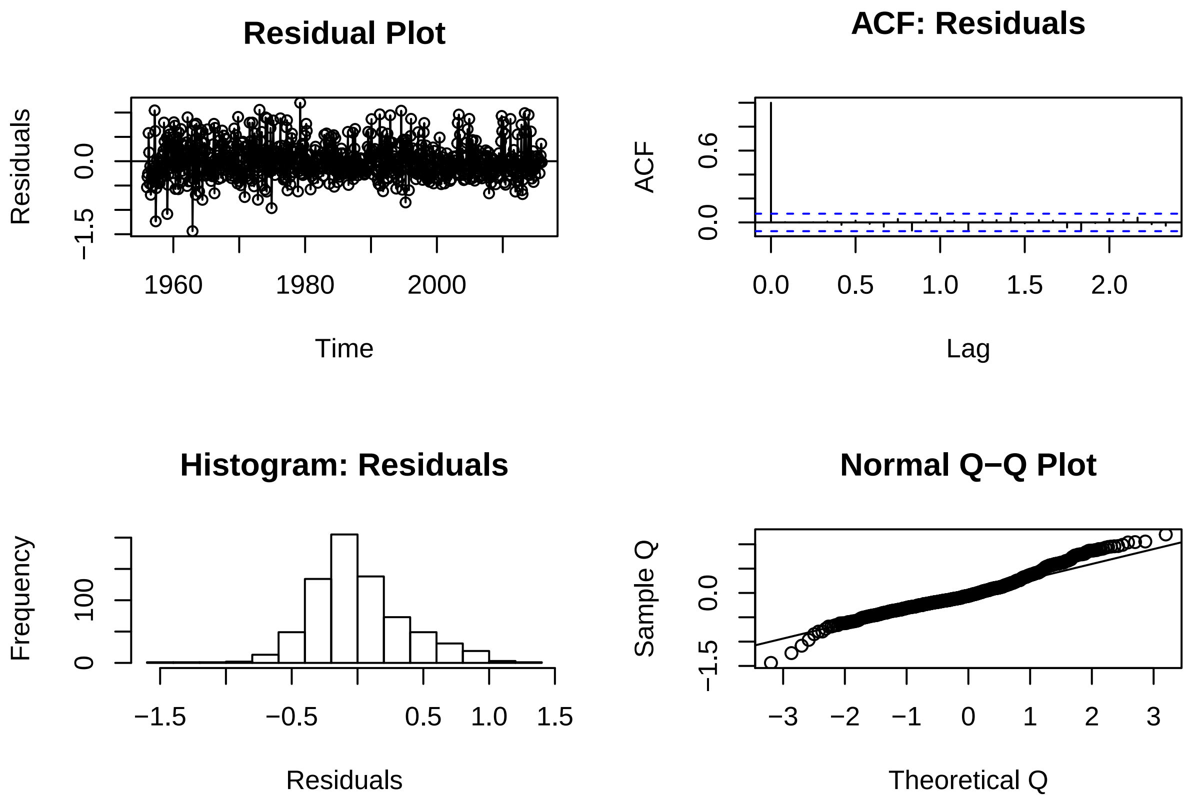
2.Use univariate model, ARIMA and ARIMAX to predict the time series

Use Univariate ARIMA model to fit

```
# final.aic = Inf
# final.order = c(0,0,0,0)
# # trying to select the order (p, d, q) and seasonality (s) for ARIMA model by comparing AIC. For ts_r
# for (p in 1:6) for (d in 0:1) for (q in 1:6) for(s in 0:1){
#    current.aic = AIC(arima(ts_river, order=c(p, d, q), seasonal = list(order=c(0,s,0),
#    period=12), method="ML"))
#    if (current.aic < final.aic) {
#      final.aic = current.aic
#      final.order = c(p, d, q,s)
#
#    }
#  }
# # > final.order 1 0 4 0. I comment this out because it takes time to run in knit, feel free to uncomm
```

Residual analysis for final ARIMA model

```
model.arima = arima(ts_river, order=c(1,0,4),seasonal = list(order=c(0,0,0),
   period=12), method="ML")

par(mfrow=c(2,2))
plot(resid(model.arima), ylab='Residuals',type='o',main="Residual Plot")
abline(h=0)
acf(resid(model.arima),main="ACF: Residuals")
hist(resid(model.arima),xlab='Residuals',main='Histogram: Residuals')
qqnorm(resid(model.arima),ylab="Sample Q",xlab="Theoretical Q")
qqline(resid(model.arima))
```




```
Box.test(model.arima$resid, lag = (1+4+1), type = "Box-Pierce", fitdf = (1+4))
```

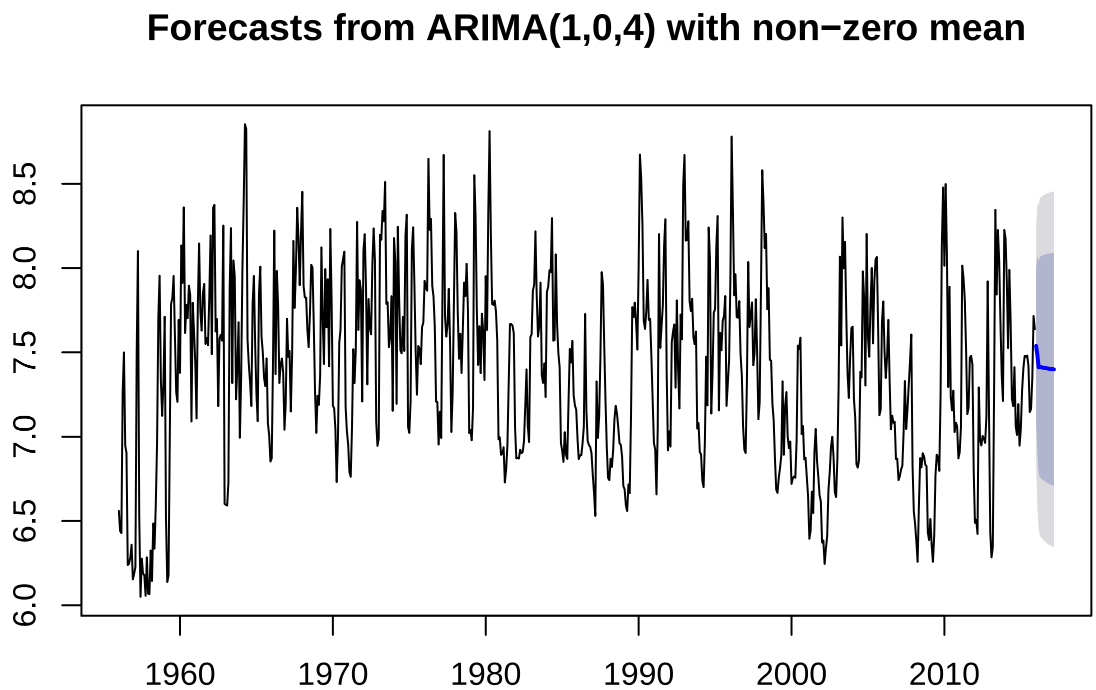
```
##
##  Box-Pierce test
##
## data:  model.arima$resid
## X-squared = 0.51611, df = 1, p-value = 0.4725
```

```
Box.test(model.arima$resid, lag = (1+4+1), type = "Ljung-Box", fitdf = (1+4))
```

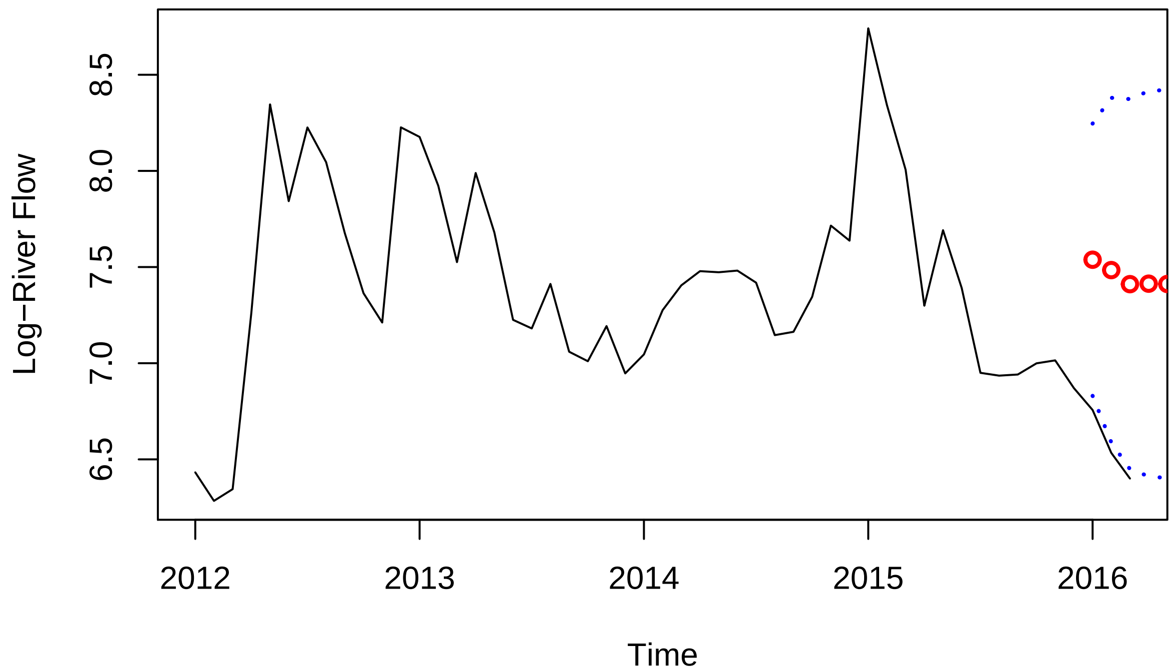
```
##
##  Box-Ljung test
##
## data:  model.arima$resid
## X-squared = 0.52129, df = 1, p-value = 0.4703
```

Predictions versus actual Using ARIMA(1,0,4,0)

```
# Using "forecast" is much better than using "predict", cuz it provides with many other stats informati
model.arima$x <- ts_river
plot(forecast(model.arima,h=15)) #plot with the original data before this period
```



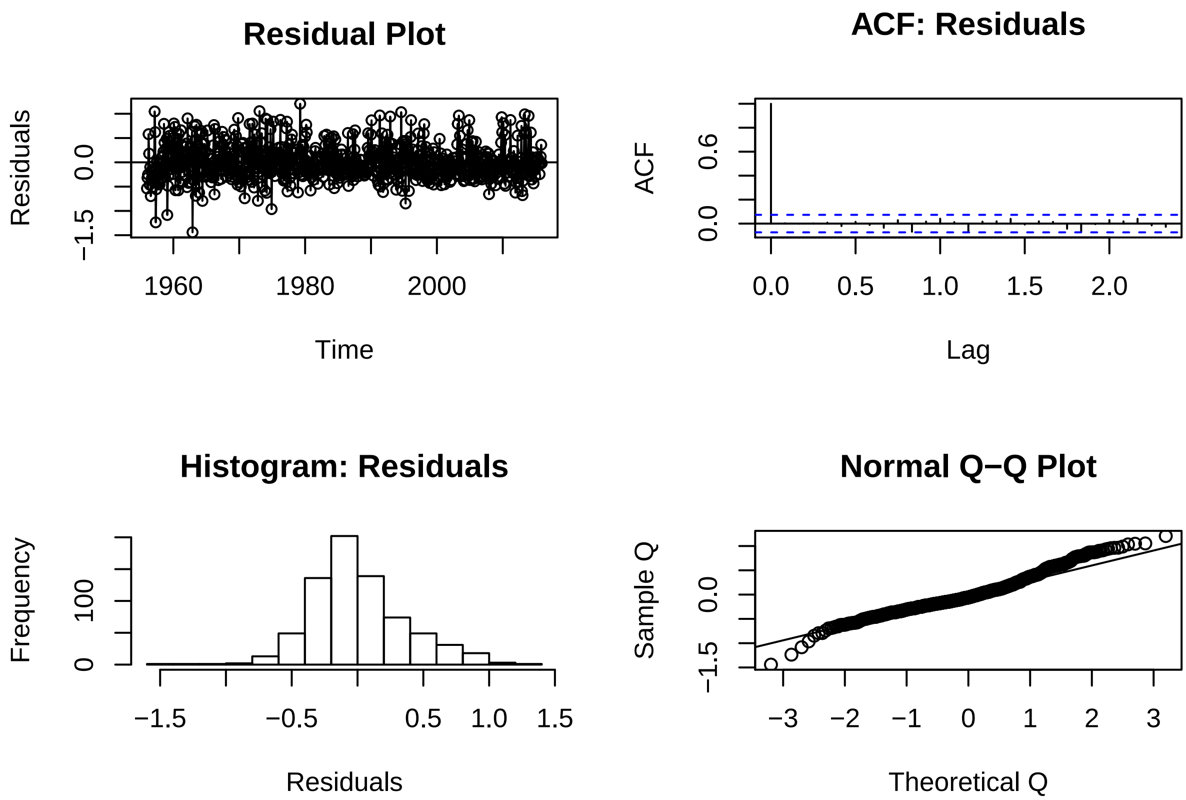

```
fore = forecast(model.arima,h=15) #h ->Number of periods for forecasting
fore=as.data.frame(fore)
point.fore = ts(fore[,1],start=2016, freq=12)
lo.fore = ts(fore[,4],start=2016, freq=12)
up.fore = ts(fore[,5],start=2016, freq=12)
ymin=min(c(log(river[(n-50):n]),lo.fore))
ymax=max(c(log(river)[(n-50):n],up.fore))
plot(ts(log(as.numeric(river[(n-50):n])),start=2012, freq=12), ylim=c(ymin,ymax), ylab="Log-River Flow"
points(point.fore,lwd=2,col="red")
lines(lo.fore,lty=3,lwd= 2, col="blue")
lines(up.fore,lty=3,lwd= 2, col="blue")
```



Predictions versus actual Using ARIMAX(1,0,4,0)

```
##### ARIMAX model
# final.aic = Inf
# final.order = c(0,0,0,0)
# # When an ARIMA model includes other time series as input variables, the model is sometimes referred
# for (p in 1:6) for (d in 0:1) for (q in 1:6) for(s in 0:1){
#    current.aic = AIC(arima(ts_river, order=c(p, d, q), seasonal = list(order=c(0,s,0),
#    period=12), method="ML",xreg=data.frame(ts_rain,ts_temp)))
#    if (current.aic < final.aic) {
#      final.aic = current.aic
#      final.order = c(p, d, q,s)
#
#    }
#  }
# > final.order
# [1] 1 0 4 0

model.arima2 = Arima(ts_river, order = c(1,0,4), method="ML",xreg=data.matrix(ts_rain,ts_temp))
## Residual analysis
par(mfrow=c(2,2))
plot(resid(model.arima2), ylab='Residuals',type='o',main="Residual Plot")
abline(h=0)
acf(resid(model.arima2),main="ACF: Residuals")
```

```
hist(resid(model.arima2),xlab='Residuals',main='Histogram: Residuals')
qqnorm(resid(model.arima2),ylab="Sample Q",xlab="Theoretical Q")
qqline(resid(model.arima2))
```




```
Box.test(model.arima2$resid, lag = (1+4+1), type = "Box-Pierce", fitdf = (1+4))
```

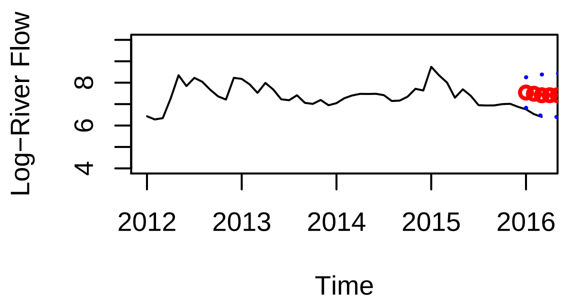
```
##
##  Box-Pierce test
##
## data:  model.arima2$resid
## X-squared = 0.50785, df = 1, p-value = 0.4761
```

```
Box.test(model.arima2$resid, lag = (1+4+1), type = "Ljung-Box", fitdf = (1+4))
```

```
##
##  Box-Ljung test
##
## data:  model.arima2$resid
## X-squared = 0.51294, df = 1, p-value = 0.4739
```

```
#Predictions versus actual
plot(ts(log(as.numeric(river[(n-50):n])),start=2012, freq=12), ylim=c(4,10), ylab="Log-River Flow", typ
fore = forecast(model.arima2,h=15,xreg=as.matrix(ts_rain2,ts_temp2))
```

```
fore=as.data.frame(fore)
point.fore = ts(fore[,1],start=2016, freq=12)
lo.fore = ts(fore[,4],start=2016, freq=12)
up.fore = ts(fore[,5],start=2016, freq=12)
points(point.fore,lwd=2,col="red")
lines(lo.fore,lty=3,lwd= 2, col="blue")
lines(up.fore,lty=3,lwd= 2, col="blue")
```



3.Using Multivariate model-> VAR(restricted and unrestricted VAR) to predict time series, and test Granger Causality

```
data.train=cbind(ts_river,ts_temp,ts_rain)
data.test=cbind(ts_river2,ts_temp2,ts_rain2)

#library(vars)
###VAR Model with Deterministic Components ##
##Model Selection
VARselect(data.train, lag.max = 20,season=12,type="both")$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      1      1      5
```

```r
## Model Fitting: Unrestricted VAR. Do regression separately, 3 sets of coefficients
model.var=VAR(data.train, p=1,type="both",season=12)
summary(model.var)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: ts_river, ts_temp, ts_rain
## Deterministic variables: both
## Sample size: 719
## Log Likelihood: 38.259
## Roots of the characteristic polynomial:
## 0.7405 0.1843 0.07091
## Call:
## VAR(y = data.train, p = 1, type = "both", season = 12L)
##
##
## Estimation results for equation ts_river:
## =========================================
## ts_river = ts_river.l1 + ts_temp.l1 + ts_rain.l1 + const + trend + sd1 + sd2 + sd3 + sd4 + sd5 + sd6
##
##               Estimate Std. Error t value Pr(>|t|)
## ts_river.l1  7.383e-01  2.511e-02  29.396  < 2e-16 ***
## ts_temp.l1  -8.597e-02  2.279e-01  -0.377 0.706155
## ts_rain.l1   6.787e-02  2.053e-02   3.306 0.000995 ***
## const        2.233e+00  9.692e-01   2.304 0.021541 *
## trend       -7.753e-05  6.698e-05  -1.157 0.247481
## sd1          1.074e-01  7.577e-02   1.418 0.156678
## sd2          8.888e-02  8.406e-02   1.057 0.290725
## sd3          1.773e-01  7.371e-02   2.405 0.016426 *
## sd4          2.301e-01  6.609e-02   3.481 0.000530 ***
## sd5          1.703e-01  7.487e-02   2.275 0.023218 *
## sd6          9.742e-02  9.096e-02   1.071 0.284504
## sd7          2.136e-01  1.063e-01   2.009 0.044899 *
## sd8          2.724e-01  1.133e-01   2.404 0.016480 *
## sd9          1.304e-01  1.117e-01   1.167 0.243612
## sd10         9.941e-02  9.832e-02   1.011 0.312307
## sd11         1.287e-01  7.606e-02   1.692 0.091149 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3594 on 703 degrees of freedom
## Multiple R-Squared: 0.5841,  Adjusted R-squared: 0.5752
## F-statistic: 65.82 on 15 and 703 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation ts_temp:
## ========================================
## ts_temp = ts_river.l1 + ts_temp.l1 + ts_rain.l1 + const + trend + sd1 + sd2 + sd3 + sd4 + sd5 + sd6 +
##
##               Estimate Std. Error t value Pr(>|t|)
## ts_river.l1 -8.187e-03  4.097e-03  -1.998  0.04606 *
```

```
## ts_temp.l1   2.074e-01  3.718e-02   5.578 3.47e-08 ***
## ts_rain.l1  -8.019e-03  3.349e-03  -2.395  0.01690 *
## const        3.301e+00  1.581e-01  20.879  < 2e-16 ***
## trend        5.800e-05  1.093e-05   5.308 1.49e-07 ***
## sd1         -3.793e-02  1.236e-02  -3.069  0.00223 **
## sd2          6.070e-02  1.371e-02   4.427 1.11e-05 ***
## sd3          1.976e-01  1.202e-02  16.437  < 2e-16 ***
## sd4          3.154e-01  1.078e-02  29.252  < 2e-16 ***
## sd5          4.020e-01  1.221e-02  32.919  < 2e-16 ***
## sd6          4.699e-01  1.484e-02  31.668  < 2e-16 ***
## sd7          4.846e-01  1.734e-02  27.938  < 2e-16 ***
## sd8          4.719e-01  1.848e-02  25.533  < 2e-16 ***
## sd9          3.985e-01  1.823e-02  21.862  < 2e-16 ***
## sd10         2.595e-01  1.604e-02  16.178  < 2e-16 ***
## sd11         1.214e-01  1.241e-02   9.784  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.05863 on 703 degrees of freedom
## Multiple R-Squared: 0.9354,  Adjusted R-squared: 0.934
## F-statistic: 678.5 on 15 and 703 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation ts_rain:
## ========================================
## ts_rain = ts_river.l1 + ts_temp.l1 + ts_rain.l1 + const + trend + sd1 + sd2 + sd3 + sd4 + sd5 + sd6 +
##
##               Estimate Std. Error t value Pr(>|t|)
## ts_river.l1  1.627e-02  4.636e-02   0.351  0.72580
## ts_temp.l1   4.220e-01  4.208e-01   1.003  0.31624
## ts_rain.l1   5.006e-02  3.790e-02   1.321  0.18692
## const       -6.520e-01  1.789e+00  -0.364  0.71565
## trend       -4.891e-05  1.236e-04  -0.396  0.69251
## sd1          2.044e-01  1.399e-01   1.461  0.14444
## sd2          1.916e-01  1.552e-01   1.235  0.21738
## sd3          3.269e-01  1.361e-01   2.402  0.01654 *
## sd4         -7.866e-02  1.220e-01  -0.645  0.51929
## sd5         -1.508e-01  1.382e-01  -1.091  0.27573
## sd6         -2.804e-01  1.679e-01  -1.670  0.09541 .
## sd7         -3.596e-02  1.963e-01  -0.183  0.85469
## sd8         -3.325e-01  2.091e-01  -1.590  0.11238
## sd9         -4.468e-01  2.063e-01  -2.166  0.03063 *
## sd10        -5.776e-01  1.815e-01  -3.182  0.00152 **
## sd11        -1.445e-01  1.404e-01  -1.030  0.30360
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6635 on 703 degrees of freedom
## Multiple R-Squared: 0.08041, Adjusted R-squared: 0.06079
## F-statistic: 4.098 on 15 and 703 DF,  p-value: 3.11e-07
##
##
```

```
##
## Covariance matrix of residuals:
##            ts_river    ts_temp   ts_rain
## ts_river  0.1292002 -0.0005391  0.018715
## ts_temp  -0.0005391  0.0034378 -0.003069
## ts_rain   0.0187153 -0.0030692  0.440250
##
## Correlation matrix of residuals:
##          ts_river  ts_temp  ts_rain
## ts_river  1.00000 -0.02558  0.07847
## ts_temp  -0.02558  1.00000 -0.07889
## ts_rain   0.07847 -0.07889  1.00000
```

```
## Model Fitting: Restricted VAR
model.var.restrict=restrict(model.var)
summary(model.var.restrict)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: ts_river, ts_temp, ts_rain
## Deterministic variables: both
## Sample size: 719
## Log Likelihood: 22.286
## Roots of the characteristic polynomial:
## 0.7405 0.2026 0.01255
## Call:
## VAR(y = data.train, p = 1, type = "both", season = 12L)
##
##
## Estimation results for equation ts_river:
## =========================================
## ts_river = ts_river.l1 + ts_rain.l1 + const + sd4 + sd8
##
##             Estimate Std. Error t value Pr(>|t|)
## ts_river.l1  0.74053    0.02440  30.353  < 2e-16 ***
## ts_rain.l1   0.07221    0.02000   3.610 0.000328 ***
## const        1.82980    0.18147  10.083  < 2e-16 ***
## sd4          0.11653    0.04941   2.358 0.018625 *
## sd8          0.12510    0.04904   2.551 0.010953 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.3605 on 714 degrees of freedom
## Multiple R-Squared: 0.9977,  Adjusted R-squared: 0.9976
## F-statistic: 6.067e+04 on 5 and 714 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation ts_temp:
## ========================================
## ts_temp = ts_temp.l1 + ts_rain.l1 + const + trend + sd1 + sd2 + sd3 + sd4 + sd5 + sd6 + sd7 + sd8 + s
##
##              Estimate Std. Error t value Pr(>|t|)
```

```
## ts_temp.l1  2.151e-01  3.706e-02   5.805 9.74e-09 ***
## ts_rain.l1 -8.392e-03  3.351e-03  -2.505   0.0125 *
## const       3.209e+00  1.515e-01  21.178  < 2e-16 ***
## trend       6.001e-05  1.090e-05   5.504 5.20e-08 ***
## sd1        -3.561e-02  1.233e-02  -2.888   0.0040 **
## sd2         6.338e-02  1.367e-02   4.634 4.26e-06 ***
## sd3         1.994e-01  1.202e-02  16.594  < 2e-16 ***
## sd4         3.152e-01  1.080e-02  29.173  < 2e-16 ***
## sd5         3.995e-01  1.217e-02  32.817  < 2e-16 ***
## sd6         4.664e-01  1.477e-02  31.581  < 2e-16 ***
## sd7         4.811e-01  1.730e-02  27.818  < 2e-16 ***
## sd8         4.679e-01  1.841e-02  25.411  < 2e-16 ***
## sd9         3.936e-01  1.810e-02  21.743  < 2e-16 ***
## sd10        2.558e-01  1.597e-02  16.020  < 2e-16 ***
## sd11        1.196e-01  1.240e-02   9.647  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.05876 on 704 degrees of freedom
## Multiple R-Squared: 0.9998,  Adjusted R-squared: 0.9998
## F-statistic: 2.344e+05 on 15 and 704 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation ts_rain:
## ========================================
## ts_rain = ts_temp.l1 + sd1 + sd2 + sd3 + sd9 + sd10
##
##             Estimate Std. Error t value Pr(>|t|)
## ts_temp.l1  0.302917   0.006038  50.165  < 2e-16 ***
## sd1         0.306551   0.092419   3.317 0.000956 ***
## sd2         0.293689   0.091742   3.201 0.001429 **
## sd3         0.438382   0.091729   4.779 2.14e-06 ***
## sd9        -0.279361   0.091714  -3.046 0.002404 **
## sd10       -0.426789   0.091709  -4.654 3.89e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.6645 on 713 degrees of freedom
## Multiple R-Squared: 0.7826,  Adjusted R-squared: 0.7808
## F-statistic: 427.8 on 6 and 713 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##            ts_river    ts_temp   ts_rain
## ts_river  0.1320121 -0.0005336  0.018740
## ts_temp  -0.0005336  0.0034573 -0.003108
## ts_rain   0.0187396 -0.0031080  0.447804
##
## Correlation matrix of residuals:
##          ts_river  ts_temp  ts_rain
## ts_river  1.00000 -0.02498  0.07707
```

```
## ts_temp  -0.02498  1.00000 -0.07899
## ts_rain   0.07707 -0.07899  1.00000
```

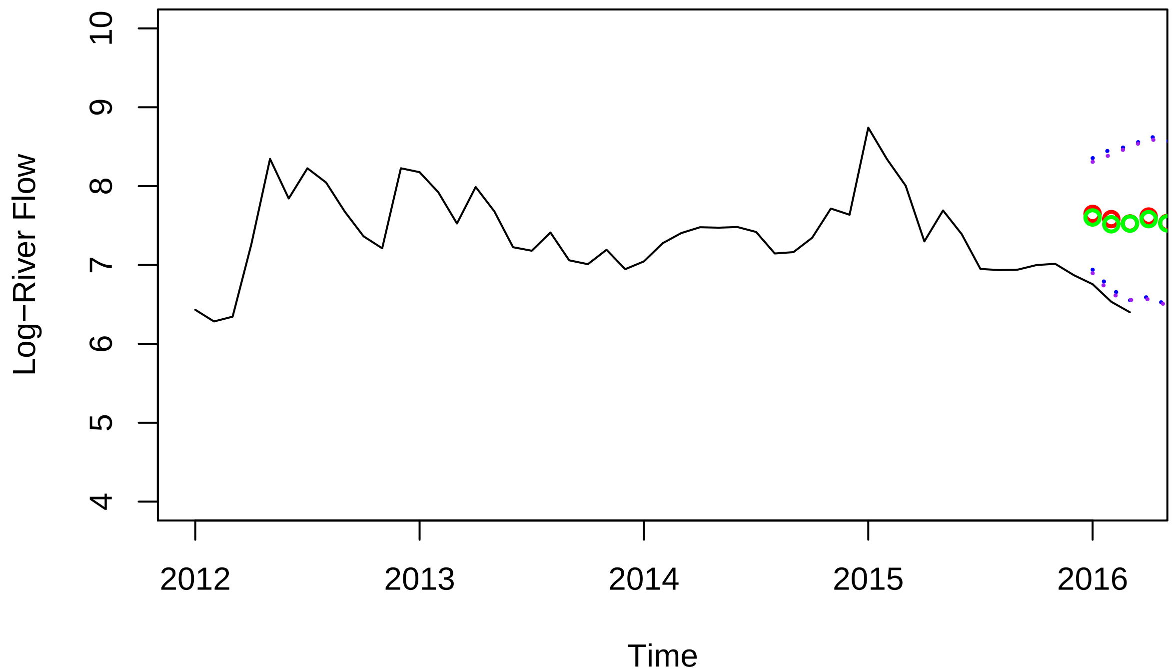
```
## Granger Causality: Wald Test
#library(aod)
coef.riverflow = coefficients(model.var)$ts_river[c(1:3),1]
var.model = vcov(model.var)[1:3,1:3]
## Granger Causality: Rain & Temperature
wald.test(b=coef.riverflow, var.model, Terms=seq(2,3))
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 12.2, df = 2, P(> X2) = 0.0023
```

```
## Granger Causality: Rain
#wald.test(b=coef.riverflow, var.model, Terms=seq(3, 9, 5))
## Granger Causality: Temperature
#wald.test(b=coef.riverflow, var.model, Terms=seq(2, 9, 5))
```

Compare plot of restricted and unrestricted VAR prediction

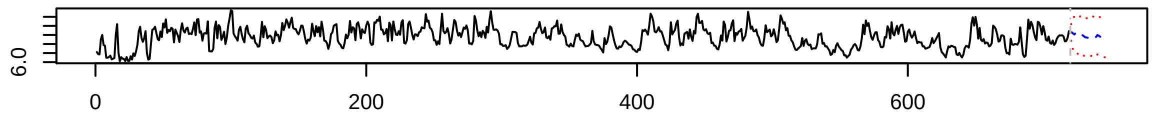
```
pred = predict(model.var.restrict, n.ahead=15, ci=0.95)[[1]]$ts_river #restricted VAR
point.pred = ts(pred[,1],start=2016, freq=12)
lo.pred = ts(pred[,2],start=2016, freq=12)
up.pred = ts(pred[,3],start=2016, freq=12)
pred.f = predict(model.var,n.ahead=15, ci=0.95)[[1]]$ts_river #unrestricted BAR
point.pred.f = ts(pred.f[,1],start=2016, freq=12)
lo.pred.f = ts(pred.f[,2],start=2016, freq=12)
up.pred.f = ts(pred.f[,3],start=2016, freq=12)

ymin=min(c(log(river[(n-50):n]),lo.pred,lo.pred.f))
ymax=max(c(log(river)[(n-50):n],up.pred,up.pred.f))
plot(ts(log(as.numeric(river[(n-50):n])),start=2012, freq=12), ylim=c(4,10), ylab="Log-River Flow", typ
points(point.pred,lwd=2,col="red")
lines(lo.pred,lty=3,lwd= 2, col="blue")
lines(up.pred,lty=3,lwd= 2, col="blue")
points(point.pred.f,lwd=2,col="green")
lines(lo.pred.f,lty=3,lwd= 2, col="purple")
lines(up.pred.f,lty=3,lwd= 2, col="purple")
```



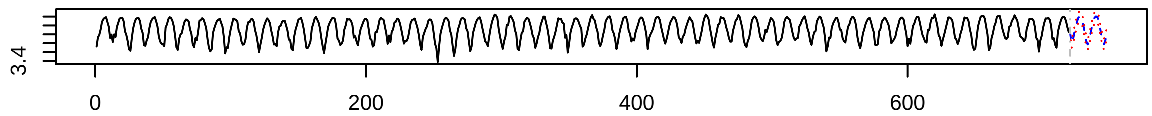
Another approach for prediction & visualizing predictions

```
fcst1 = predict(model.var.restrict, n.ahead=27, ci=0.95) #directly predict for all 3 series
plot(fcst1)
```

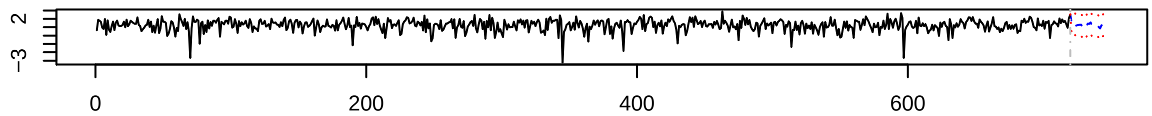
Forecast of series ts_river



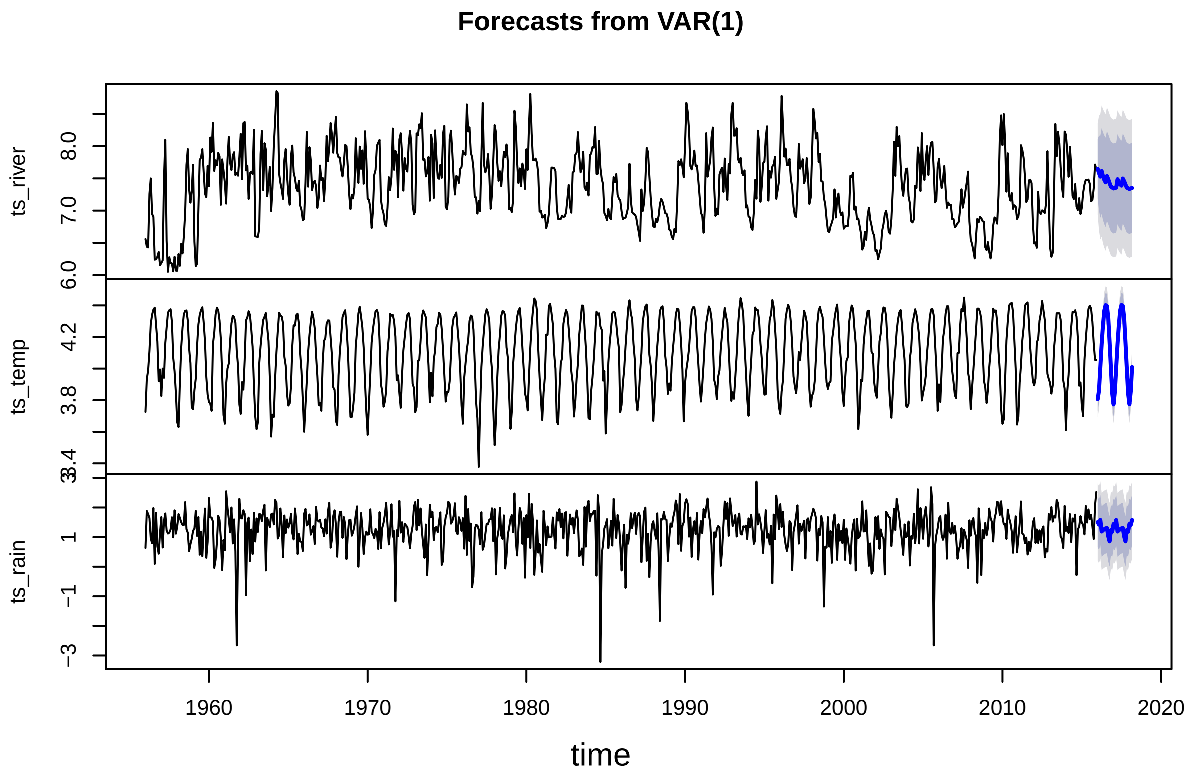
Forecast of series ts_temp



Forecast of series ts_rain




```
fcst2 = forecast(model.var.restrict,h=27)
plot(fcst2)
```


Forecasts from VAR(1)
ts_river
ts_temp
ts_rain
6.0
7.0
8.0
3.4
3.8
4.2
−3
−1
1
3
1960
1970
1980
1990
2000
2010
2020
time