Lake Michigan Water Level Forecasting

Nuka Gvilia, Veda Kilaru, Anisha BharathSingh, Thomas Harmon, Ani Baghdasaryan

5/23/2022

STEP 0: Set-Up

```
defaultW <- getOption("warn")
options(warn = -1)

#library(fpp)
library(tseries)
```

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

```
library(ggplot2)
library(forecast)
library(TSA)
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast

##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```
library(vars)
```

```
## Loading required package: MASS

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
## Loading required package: urca
## Loading required package: lmtest
```

```r
#library(plotly)
```

```r
water_level <- read.csv("water_level_data.csv")
temperature <- read.csv("temperature.csv")
precipitation <- read.csv("precipitation.csv")
```

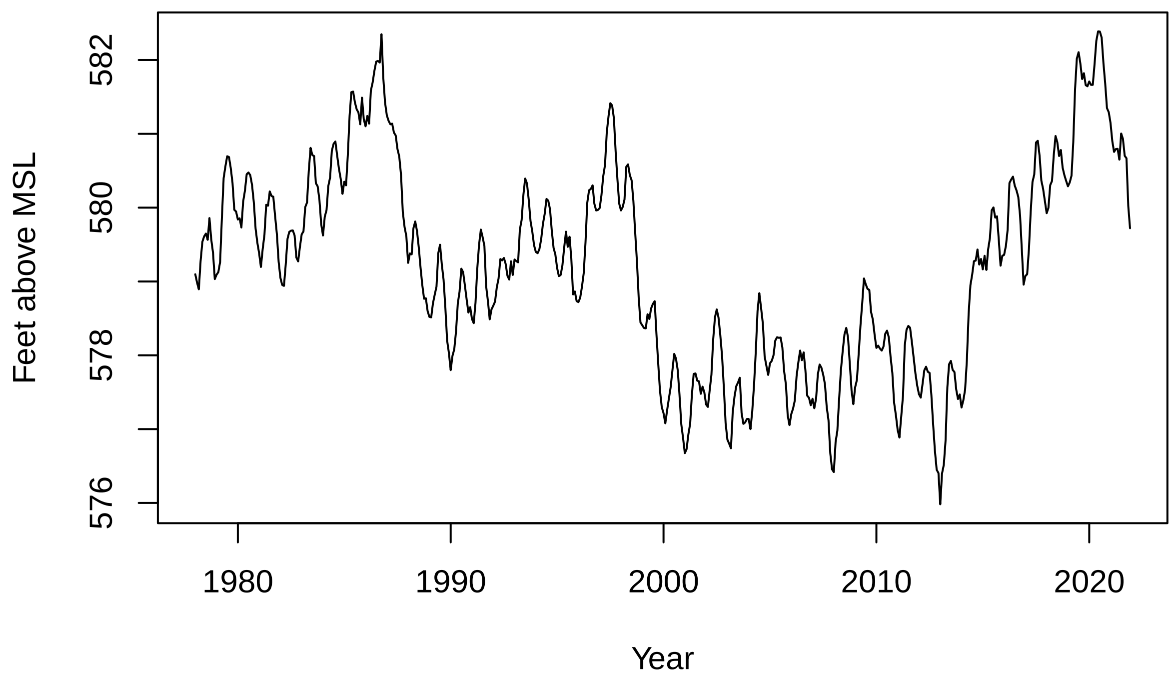
STEP 1: Data pre-processing

```r
# Water level
water_level <- water_level[1:528,]
water_level.ts <- ts(water_level$MSL, start = 1978, frequency = 12) # Convert to time series object

# Temperature
temperature <- temperature[1:528, ]
temp_transpose <- data.frame(t(temperature[, 2:13]))
temp <- data.frame(Temperature=unlist(temp_transpose, use.names = FALSE))
temperature.ts <- ts(temp, start = 1978, end = c(2021, 12), frequency = 12) # Convert to time series ob

# Precipitation
precipitation <- precipitation[1:528, ]
precip_transpose <- data.frame(t(precipitation[, 2:13]))
precip <- data.frame(Precipitation=unlist(precip_transpose, use.names = FALSE))
precipitation.ts <- ts(precip, start = 1978, end = c(2021, 12), frequency = 12) # Convert to time serie

# Final dataframe
df = data.frame(water_level.ts, temperature.ts, precipitation.ts)
```

STEP 2: Initial data exploration

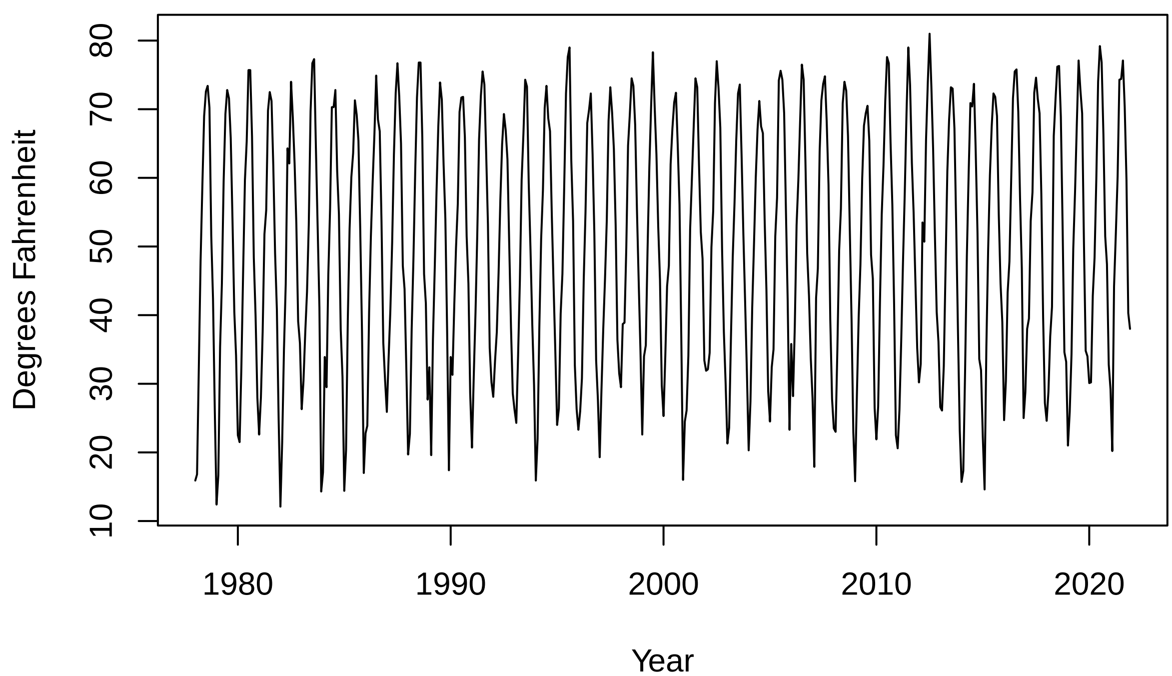
```r
plot(water_level.ts, ylab="Feet above MSL", xlab="Year", main="Average Water Level of Lake Michigan")
```

Average Water Level of Lake Michigan

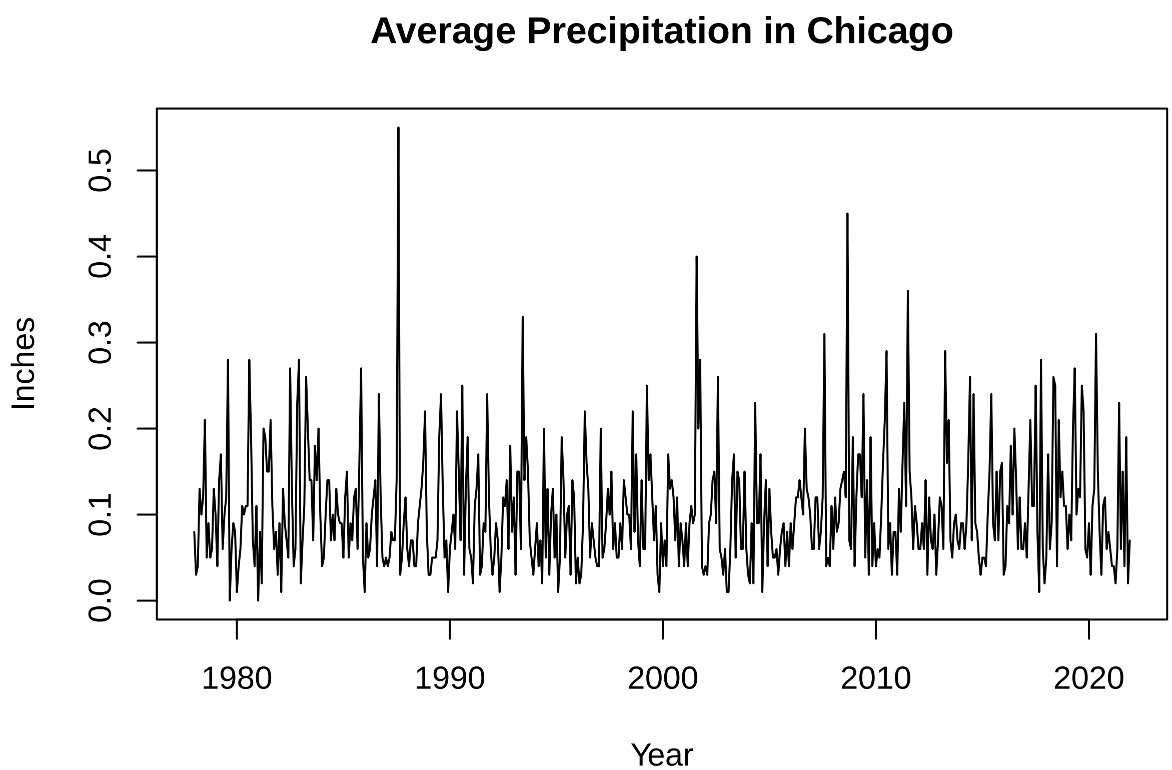



```
plot(temperature.ts, ylab="Degrees Fahrenheit", xlab="Year", main="Average Temperature of Chicago")
```

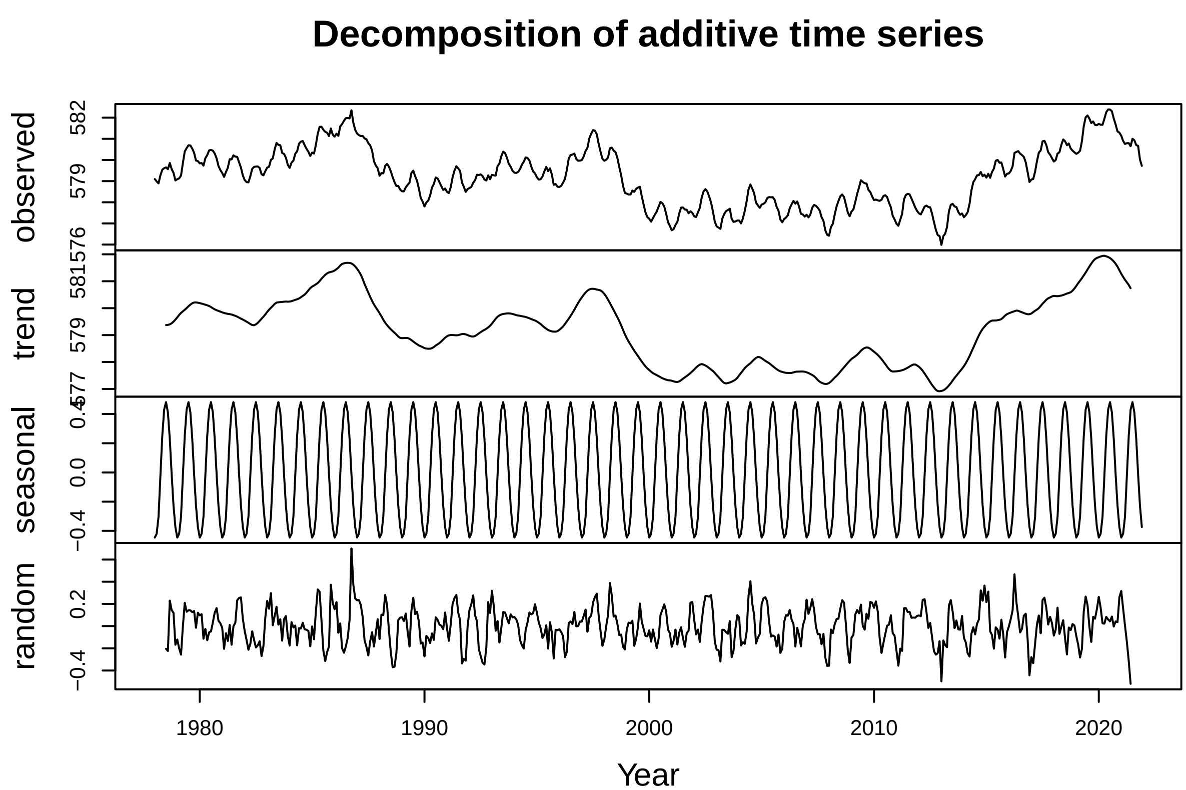
Average Temperature of Chicago




```
plot(precipitation.ts, ylab="Inches", xlab="Year", main="Average Precipitation in Chicago")
```

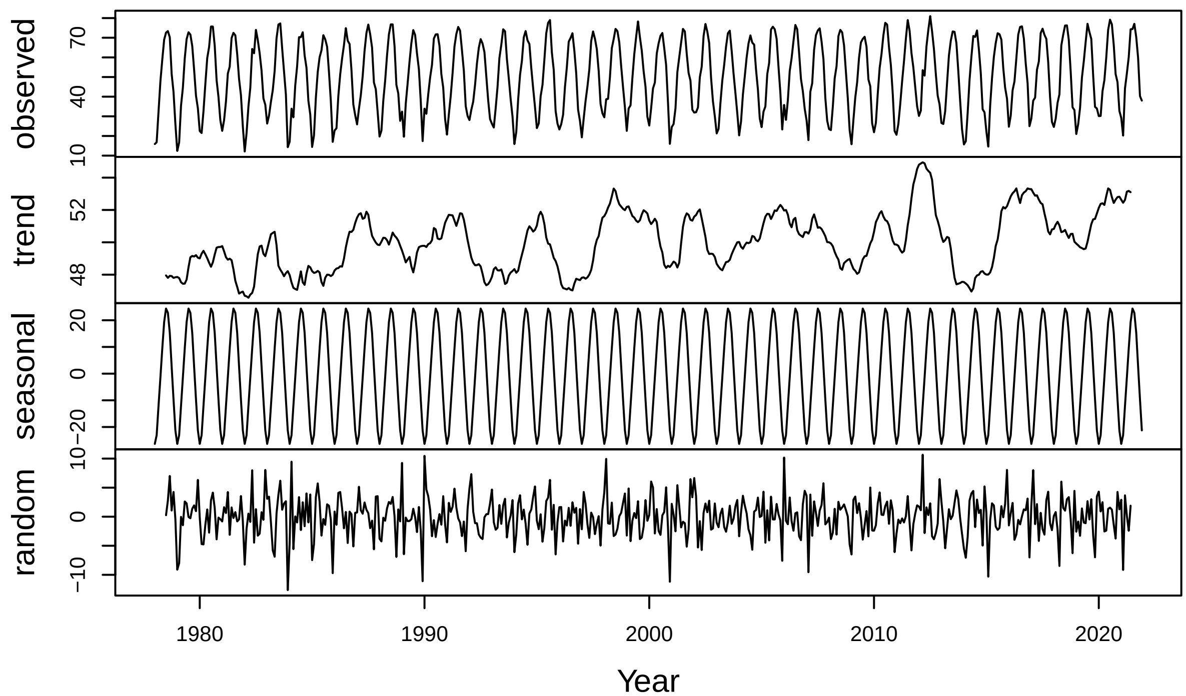



```
plot(decompose(water_level.ts), xlab="Year")
```



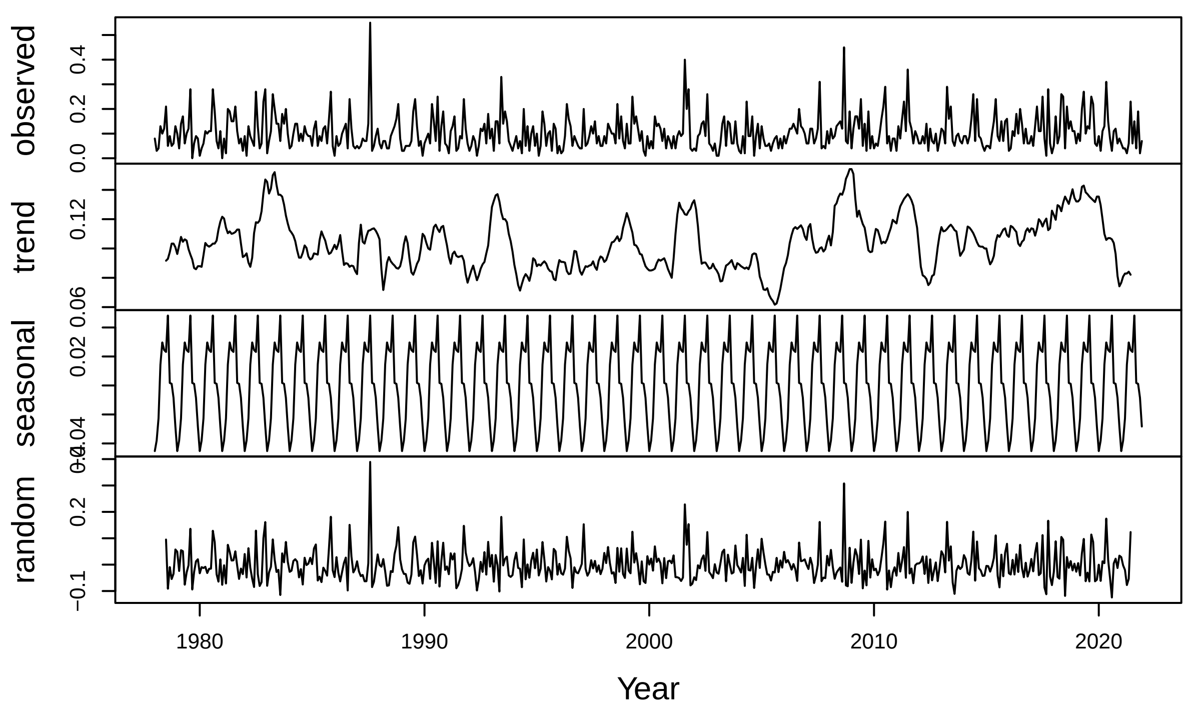

```
plot(decompose(temperature.ts), xlab="Year")
```

Decomposition of additive time series

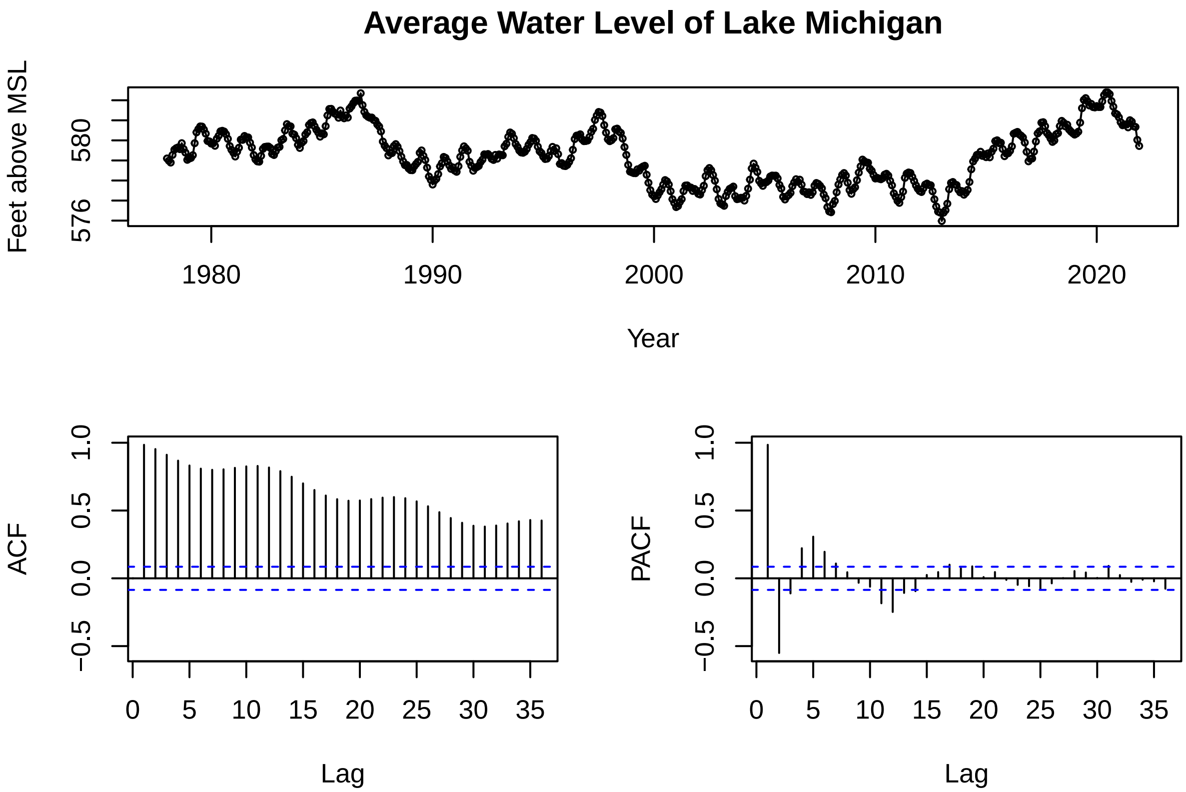



```
plot(decompose(precipitation.ts), xlab="Year")
```

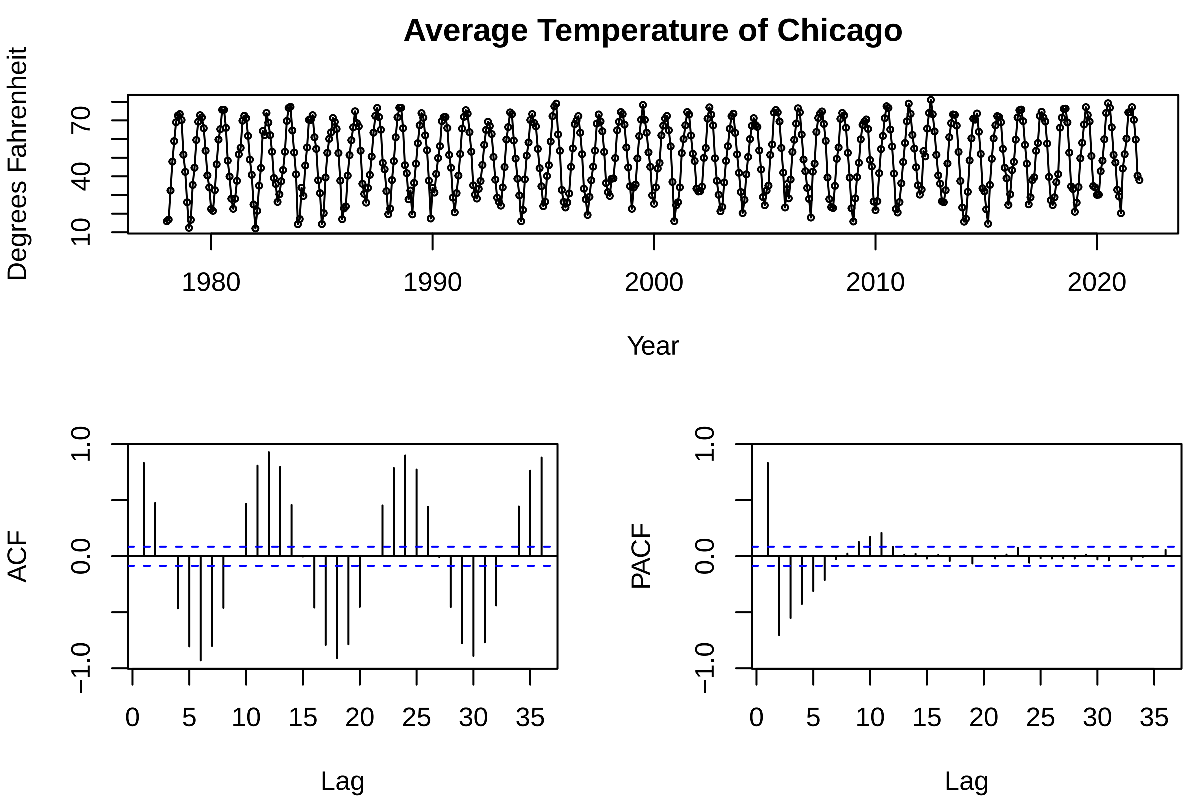
Decomposition of additive time series



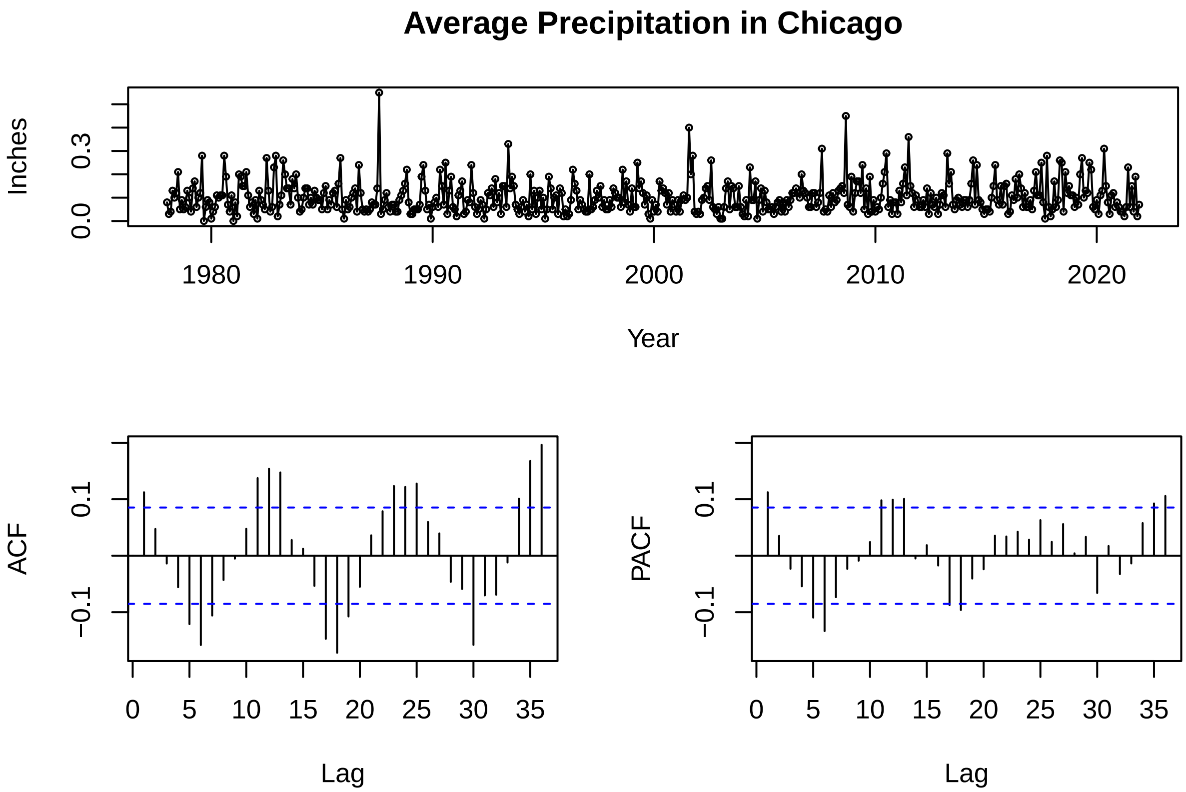

```
tsdisplay(water_level.ts, ylab="Feet above MSL", xlab="Year", main="Average Water Level of Lake Michigan
```



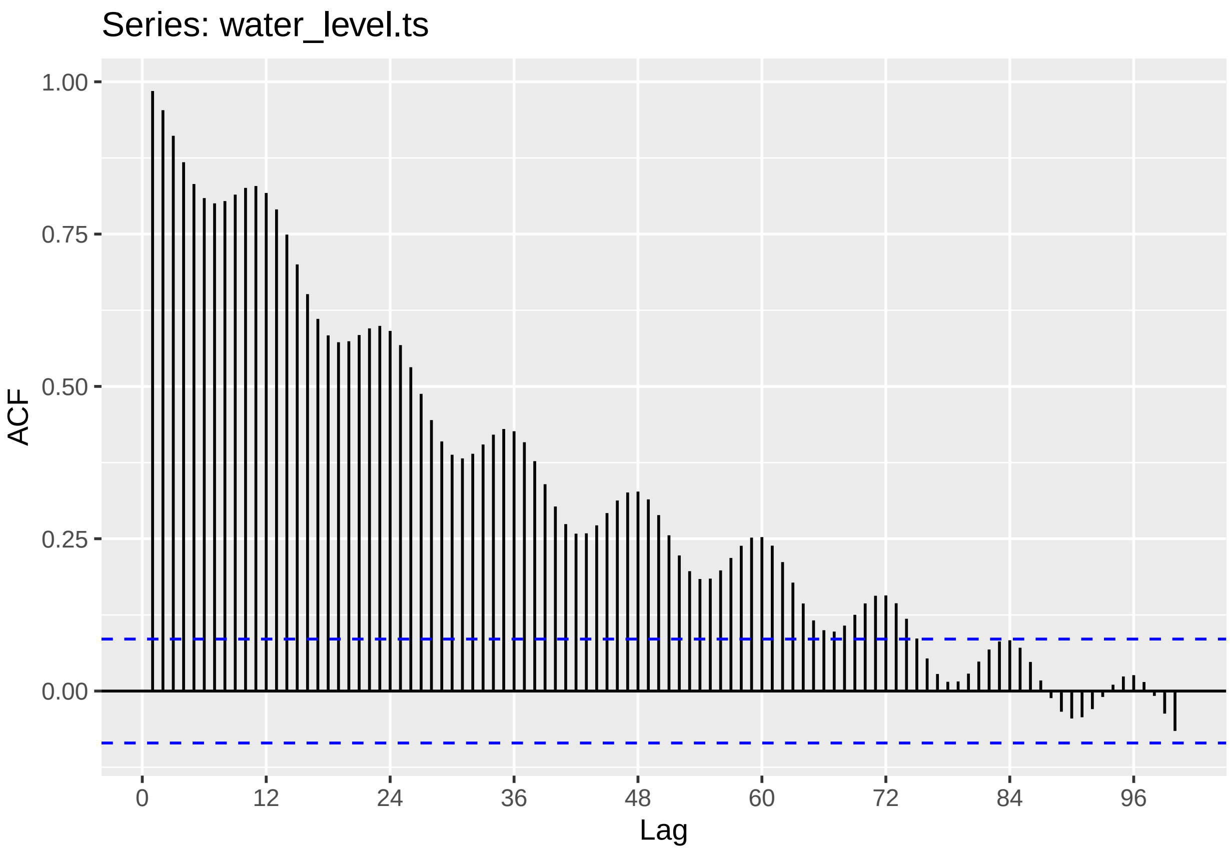

```
tsdisplay(temperature.ts, ylab="Degrees Fahrenheit", xlab="Year", main="Average Temperature of Chicago")
```



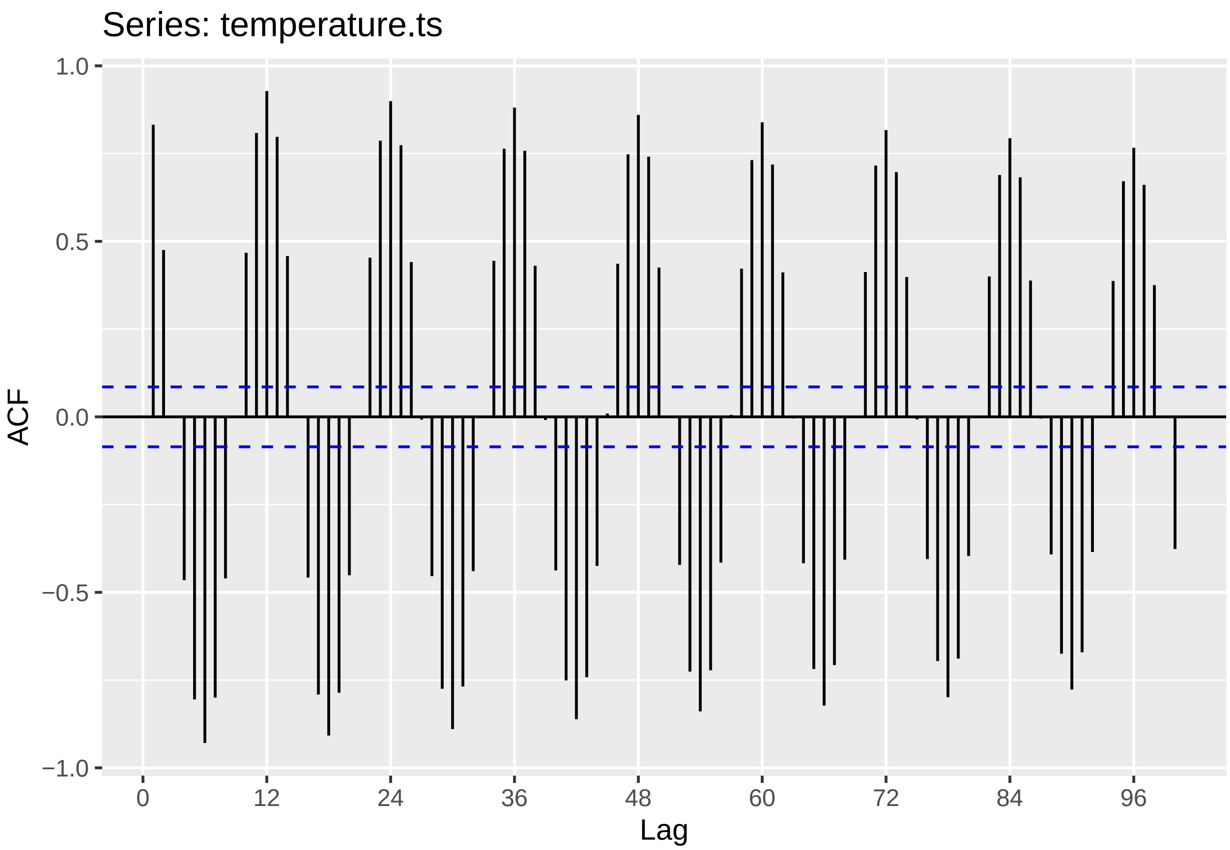
```r
tsdisplay(precipitation.ts, ylab="Inches", xlab="Year", main="Average Precipitation in Chicago")
```



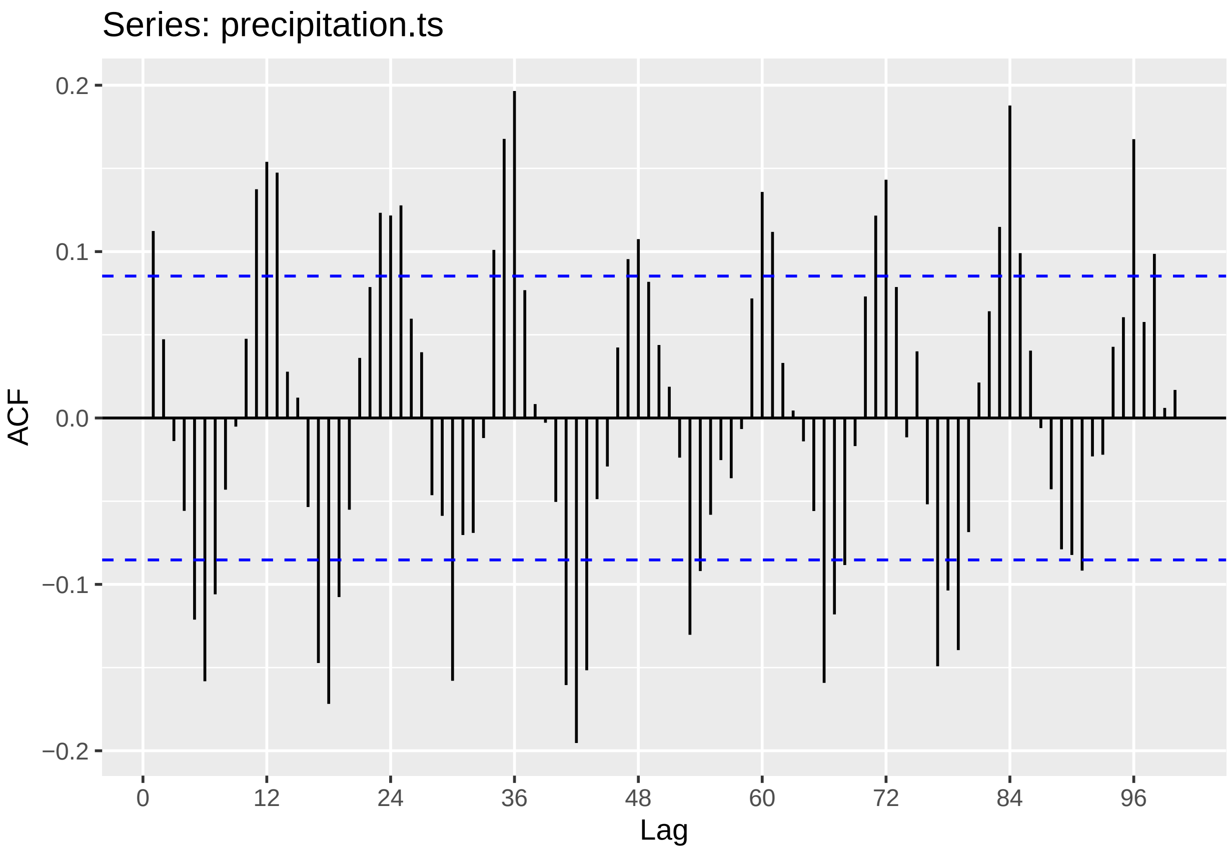

```r
ggAcf(water_level.ts, lag=100)
```



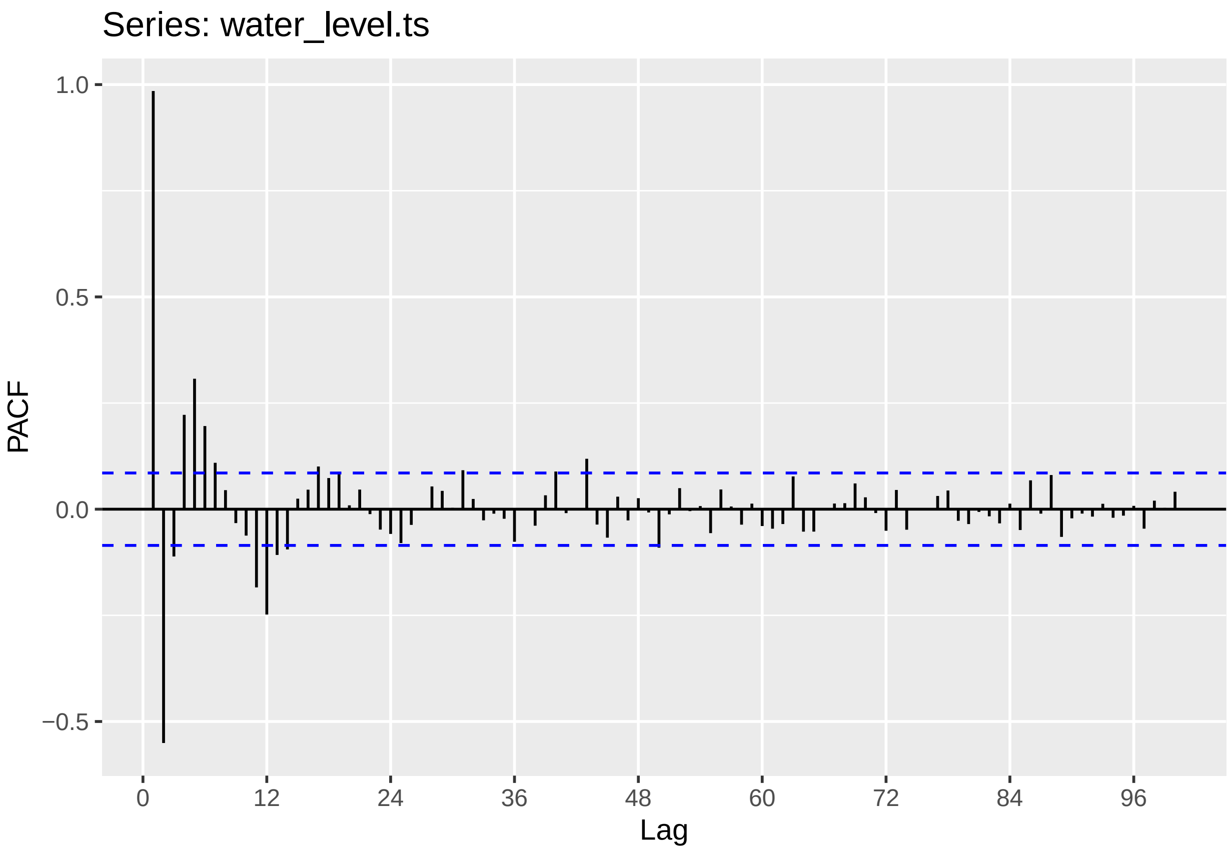

```
ggAcf(temperature.ts, lag=100)
```



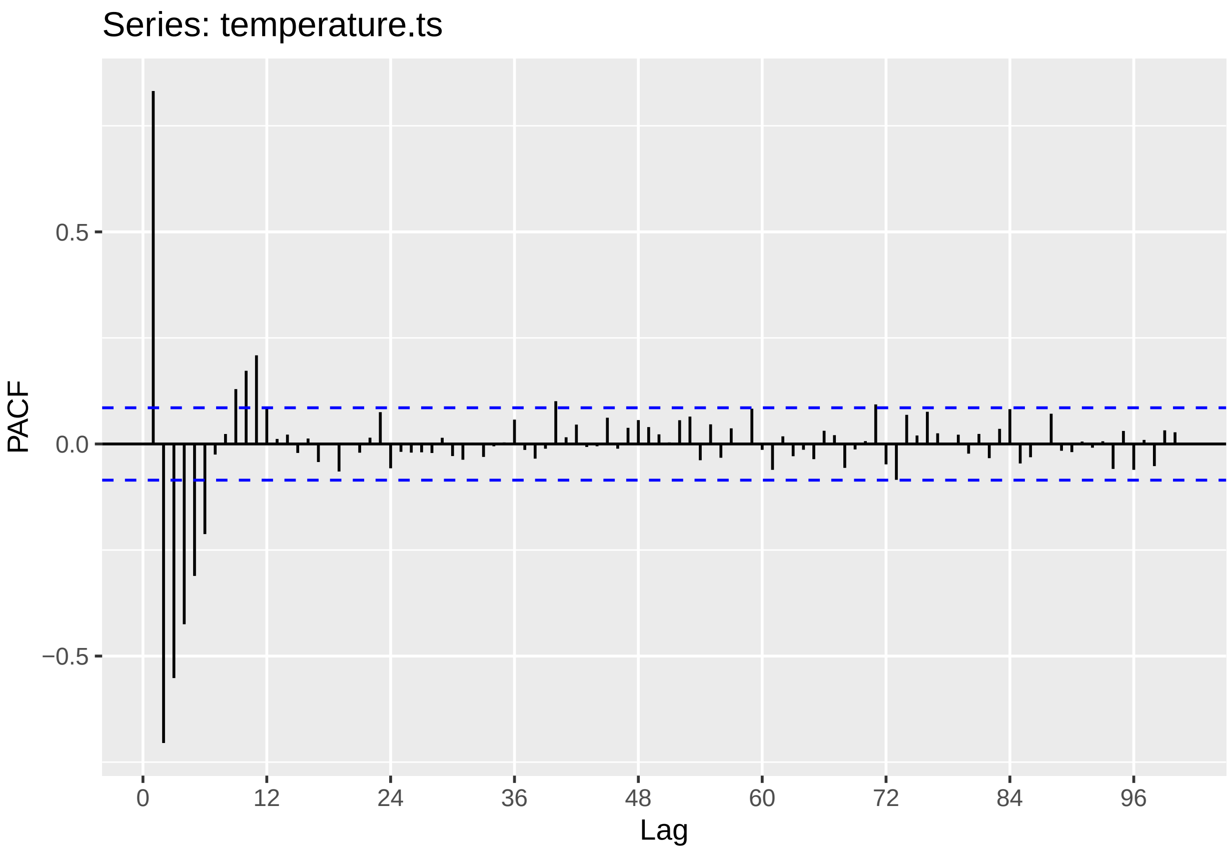

```
ggAcf(precipitation.ts, lag=100)
```



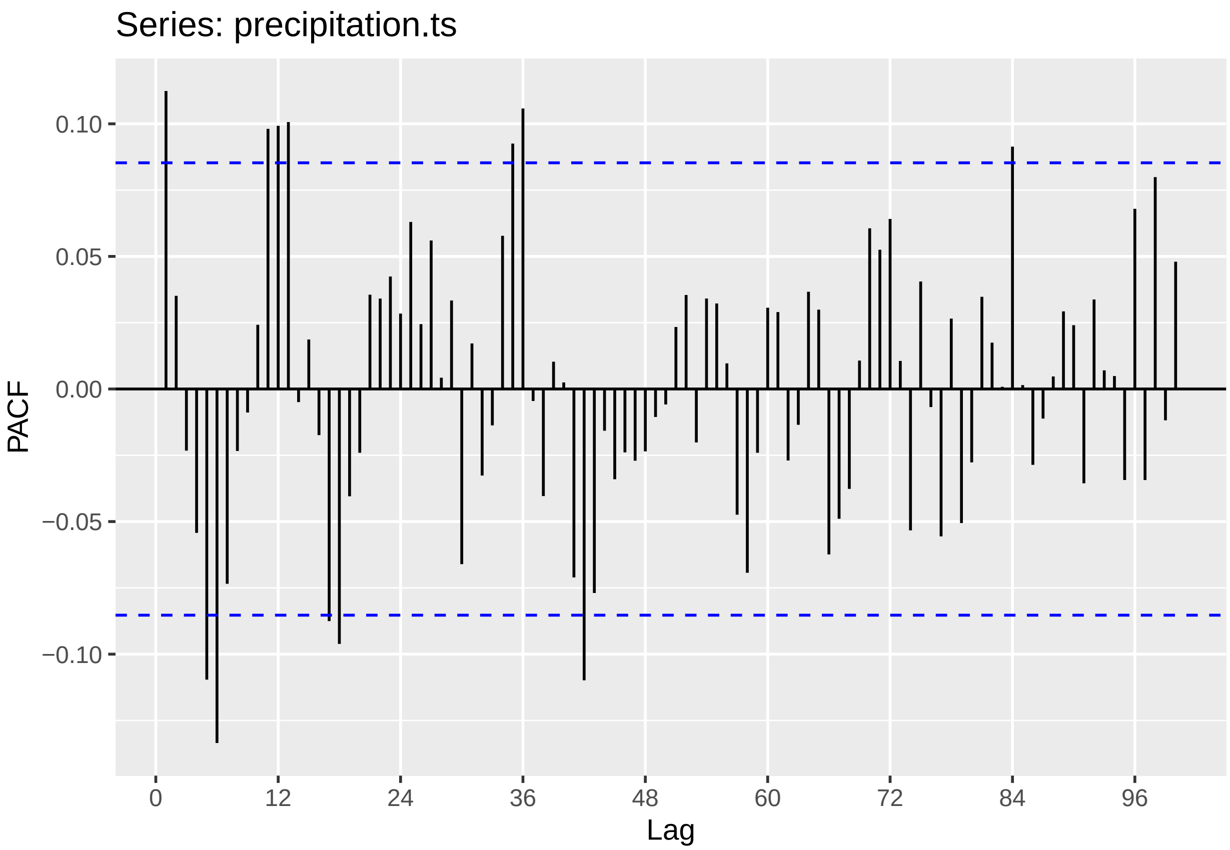

```r
ggPacf(water_level.ts, lag=100)
```




```
ggPacf(temperature.ts, lag=100)
```




```
ggPacf(precipitation.ts, lag=100)
```

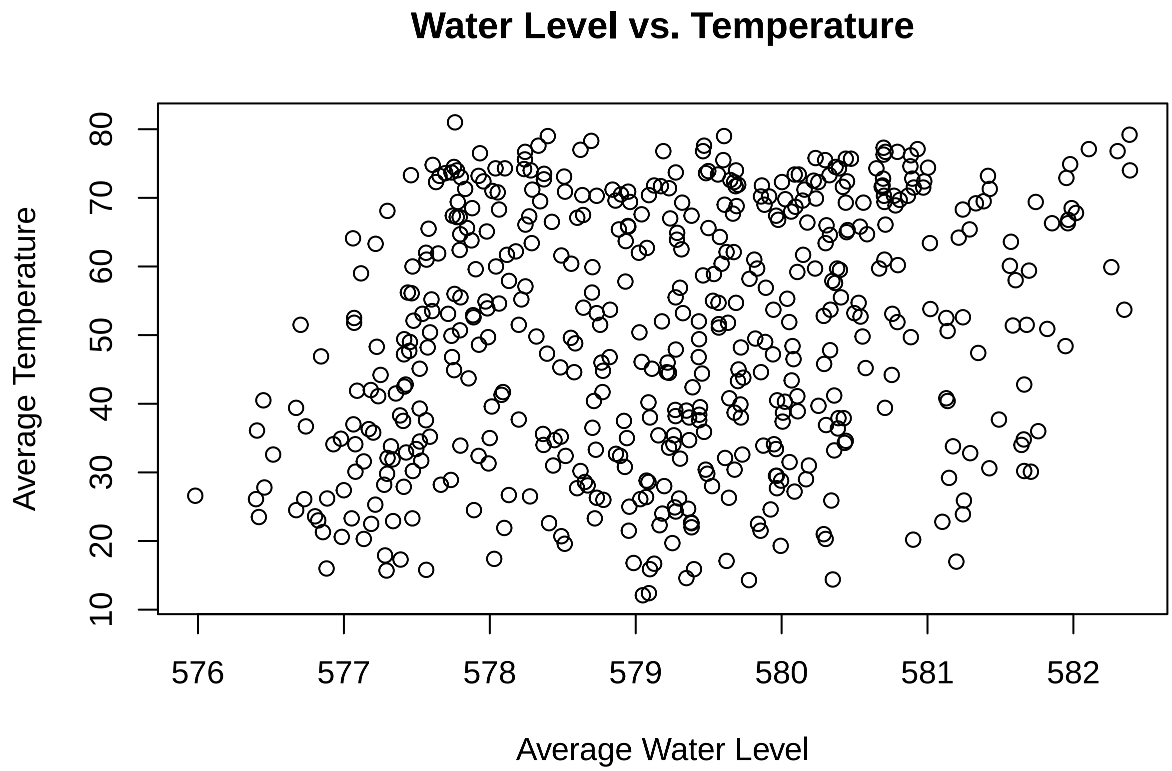


CORRELATION

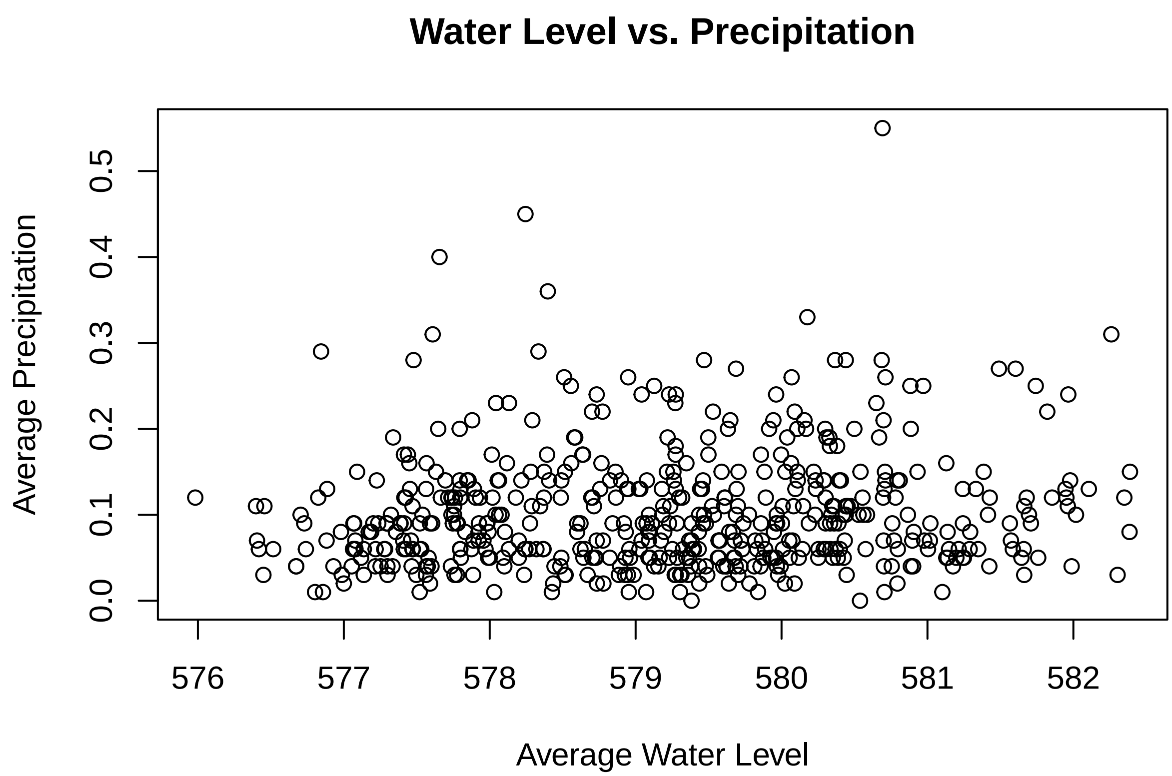
```
cor(df)
```

```
##                    water_level.ts temperature.ts precipitation.ts
## water_level.ts          1.0000000      0.2361988        0.1095364
## temperature.ts          0.2361988      1.0000000        0.3642743
## precipitation.ts        0.1095364      0.3642743        1.0000000
```

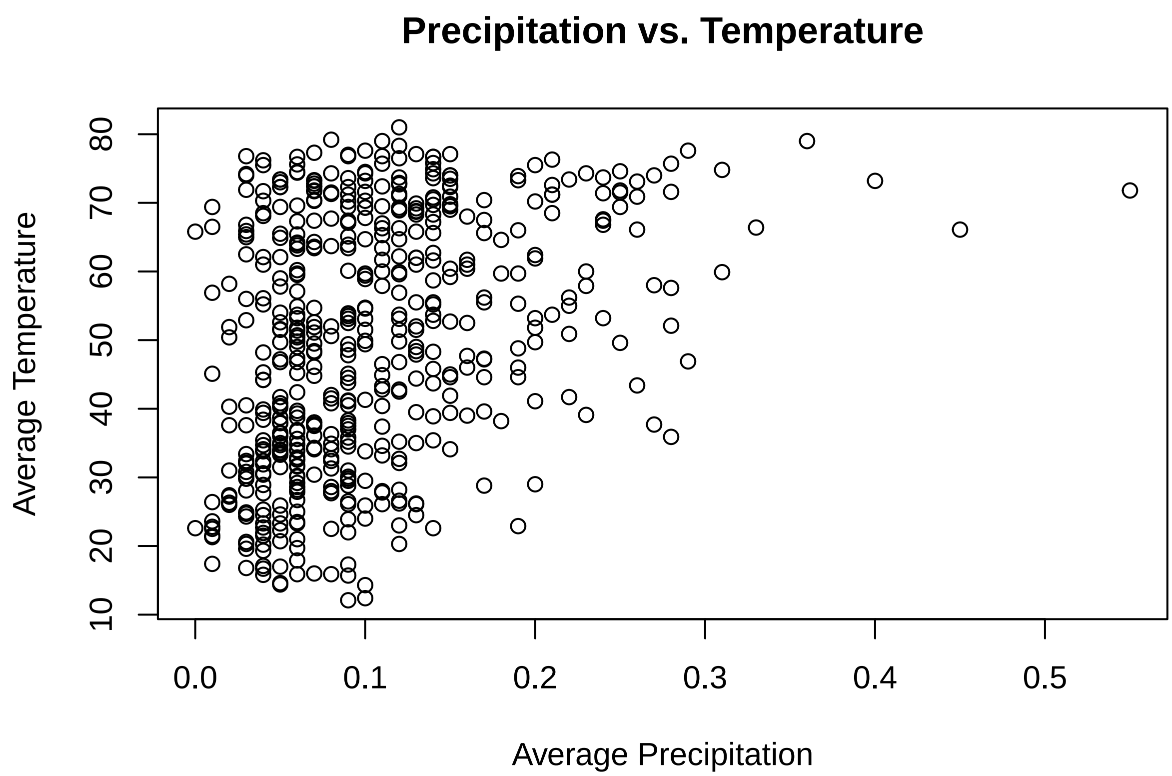
```
#plot_ly(x=temperature.ts, y=water_level.ts, z=precipitation.ts, type="scatter3d", mode="markers", colo

plot(x=water_level.ts, y=temperature.ts, xlab="Average Water Level", ylab="Average Temperature", main="
```




```
plot(x=water_level.ts, y=precipitation.ts, xlab="Average Water Level", ylab="Average Precipitation", ma
```




```
plot(x=precipitation.ts, y=temperature.ts, xlab="Average Precipitation", ylab="Average Temperature", ma
```



STEP 3: Data transformations

```
defaultW <- getOption("warn")
options(warn = -1)

lw <- BoxCox.lambda(water_level.ts)
lw # close to 2
```

```
## [1] 1.999924
```

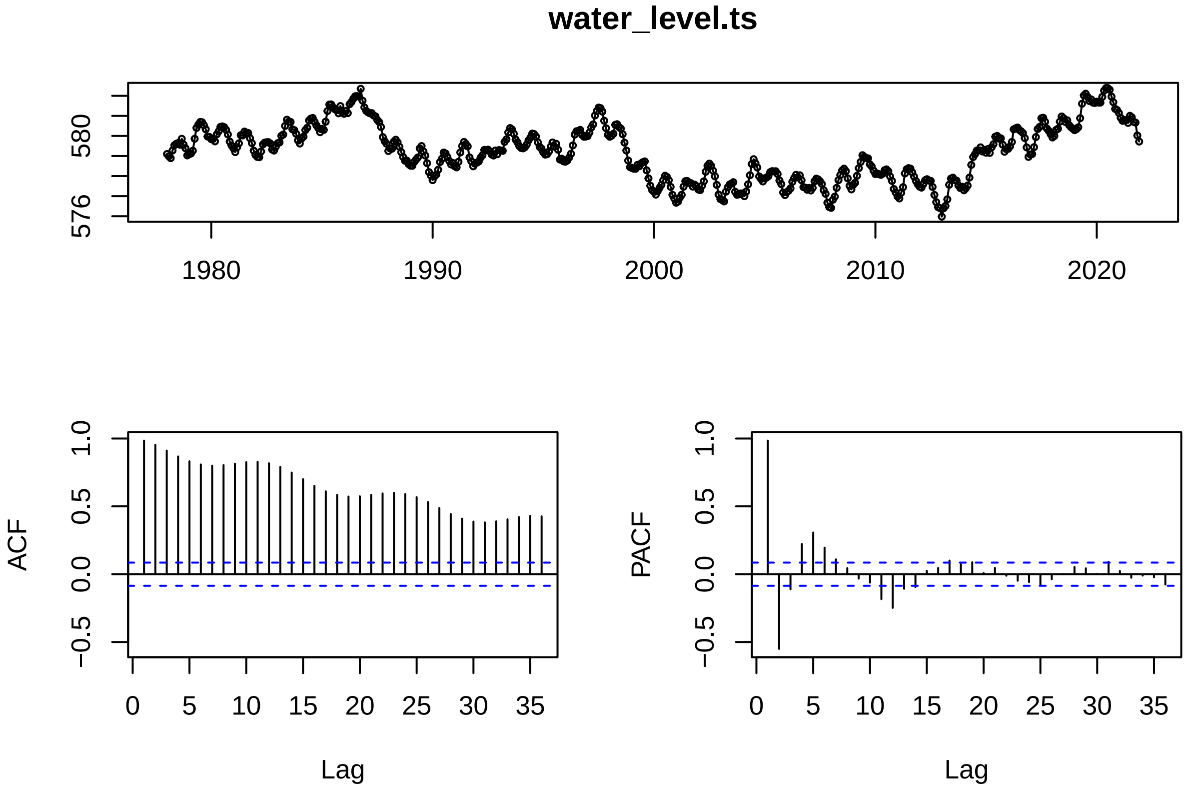
```
lt <- BoxCox.lambda(temperature.ts)
lt # close to 2
```

```
## [1] 1.891525
```

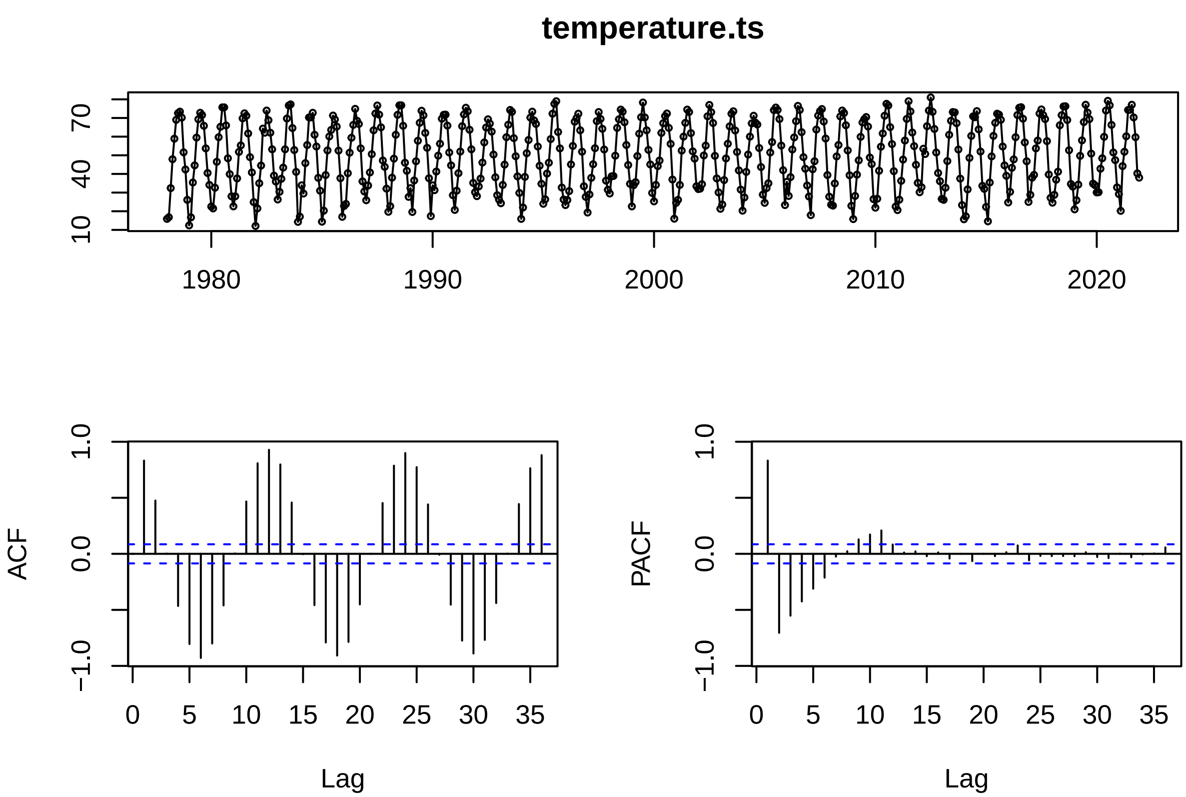
```
lp <- BoxCox.lambda(precipitation.ts)
lp # close to 0 i.e. ln()
```

```
## [1] 4.102259e-05
```

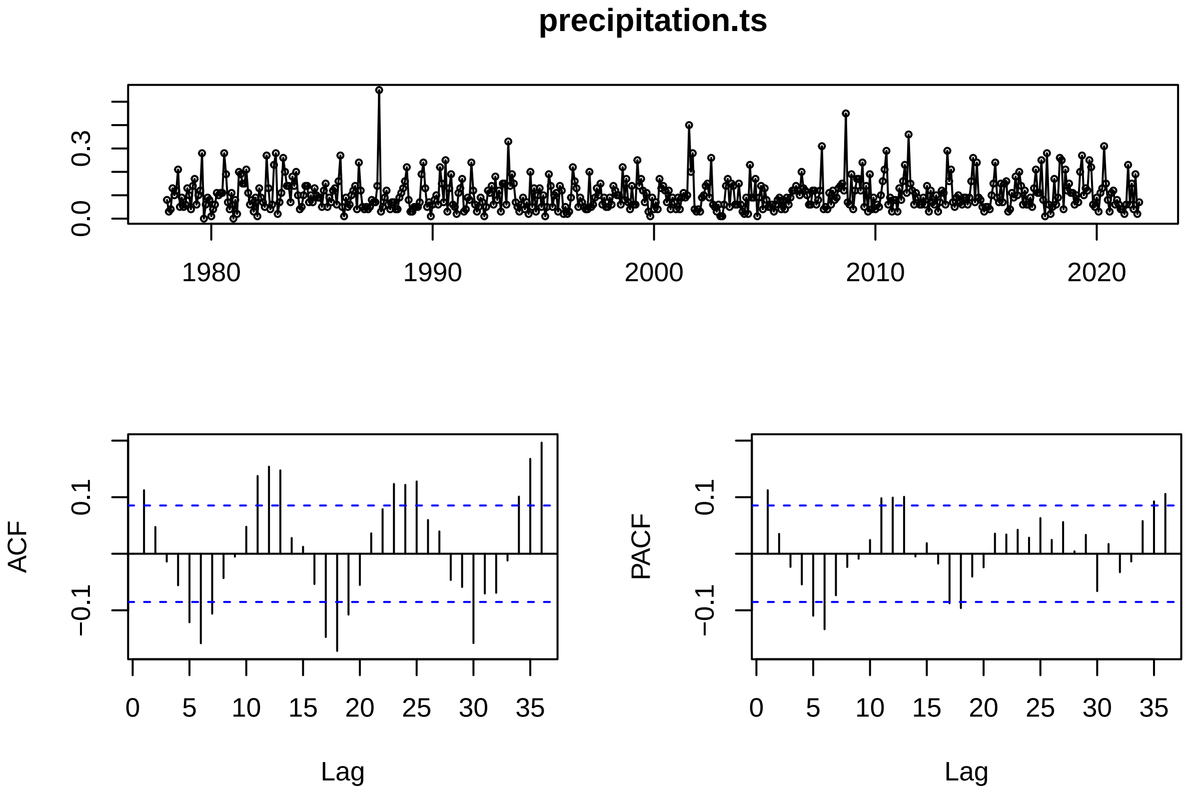
```
tsdisplay(water_level.ts)
```




```
tsdisplay(temperature.ts)
```



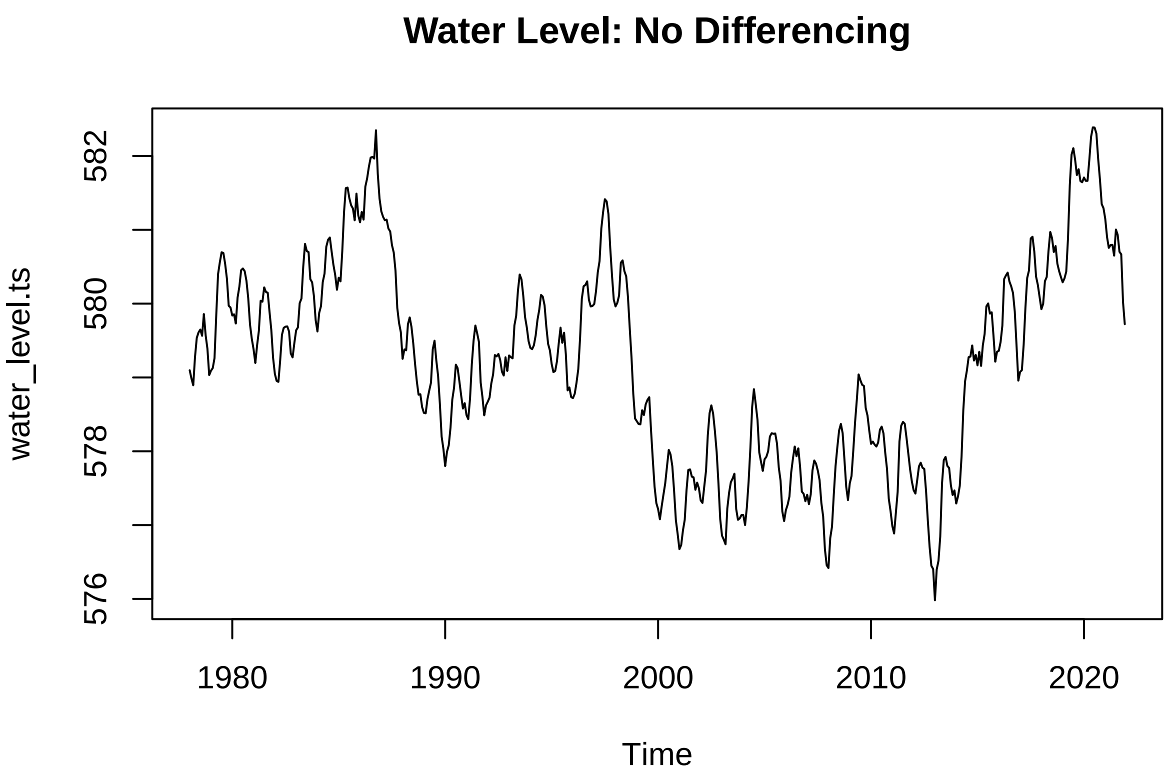
```r
tsdisplay(precipitation.ts)
```



Variation for all variables does not appear to increase/decrease with the level of the series; transformation not needed.

STEP 4: Stationarity

```r
# No differencing
plot(water_level.ts, main="Water Level: No Differencing")
```



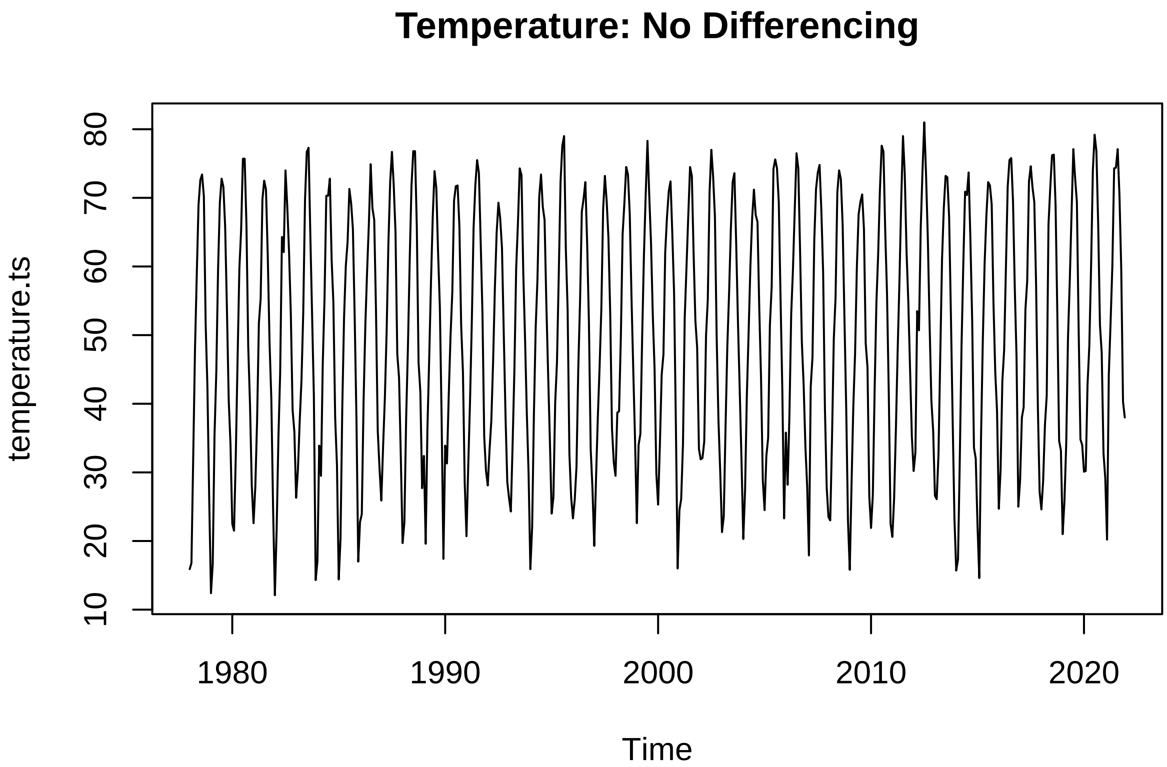

```
kpss.test(water_level.ts) # small p-value, series is *not* level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  water_level.ts
## KPSS Level = 1.3185, Truncation lag parameter = 6, p-value = 0.01
```

```
adf.test(water_level.ts) # large p-value, series is *not* level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  water_level.ts
## Dickey-Fuller = -1.4421, Lag order = 8, p-value = 0.8144
## alternative hypothesis: stationary
```

```
plot(temperature.ts, main="Temperature: No Differencing")
```



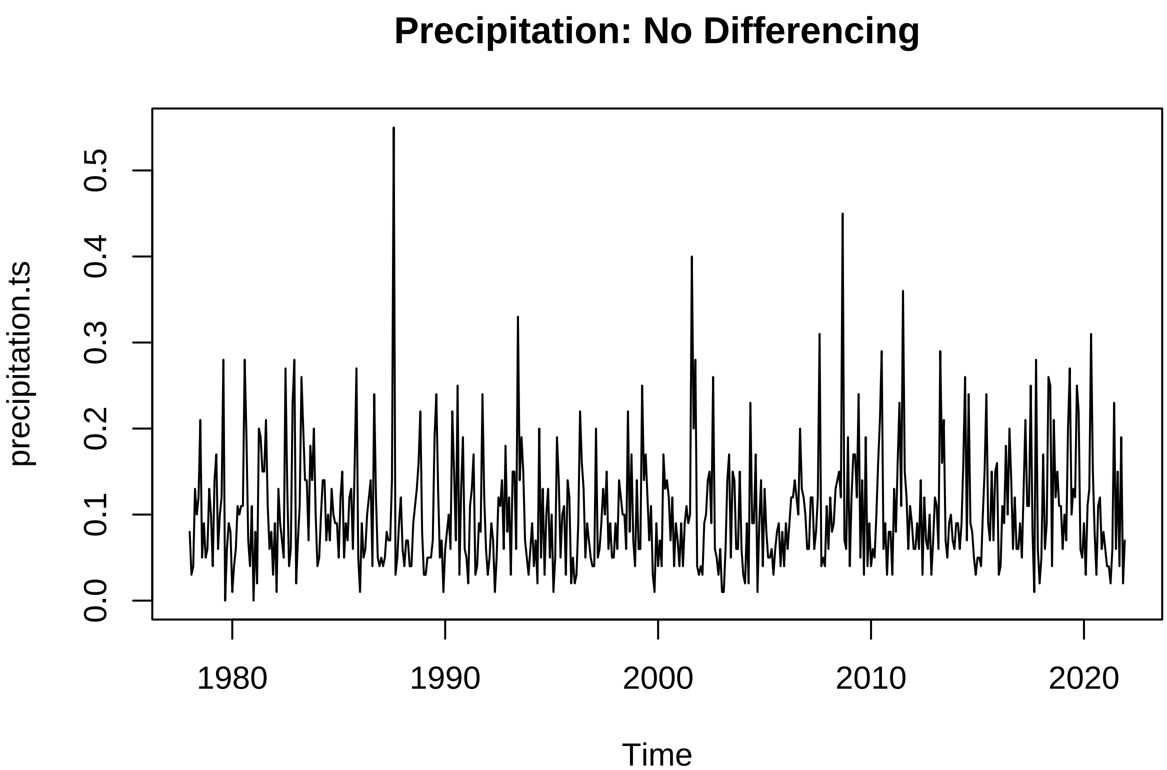

```
kpss.test(temperature.ts) # large p-value, series *is* level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  temperature.ts
## KPSS Level = 0.089951, Truncation lag parameter = 6, p-value = 0.1
```

```
adf.test(temperature.ts) # small p-value, series *is* level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temperature.ts
## Dickey-Fuller = -10.509, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
plot(precipitation.ts, main="Precipitation: No Differencing")
```



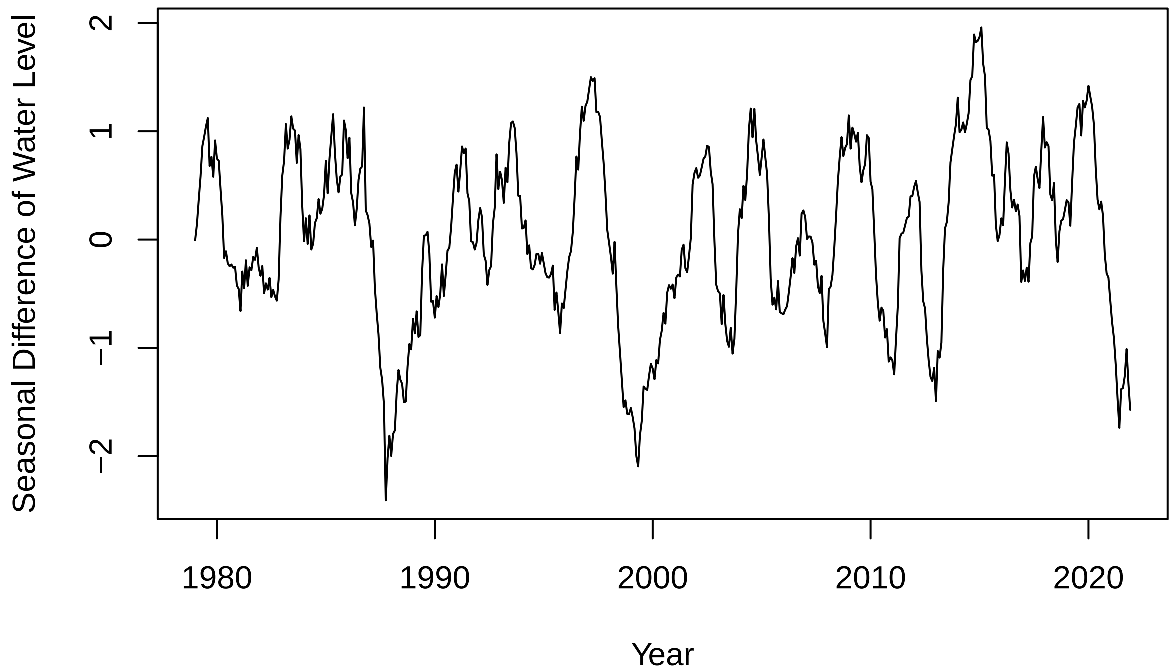

```
kpss.test(precipitation.ts) # large p-value, series *is* level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  precipitation.ts
## KPSS Level = 0.20013, Truncation lag parameter = 6, p-value = 0.1
```

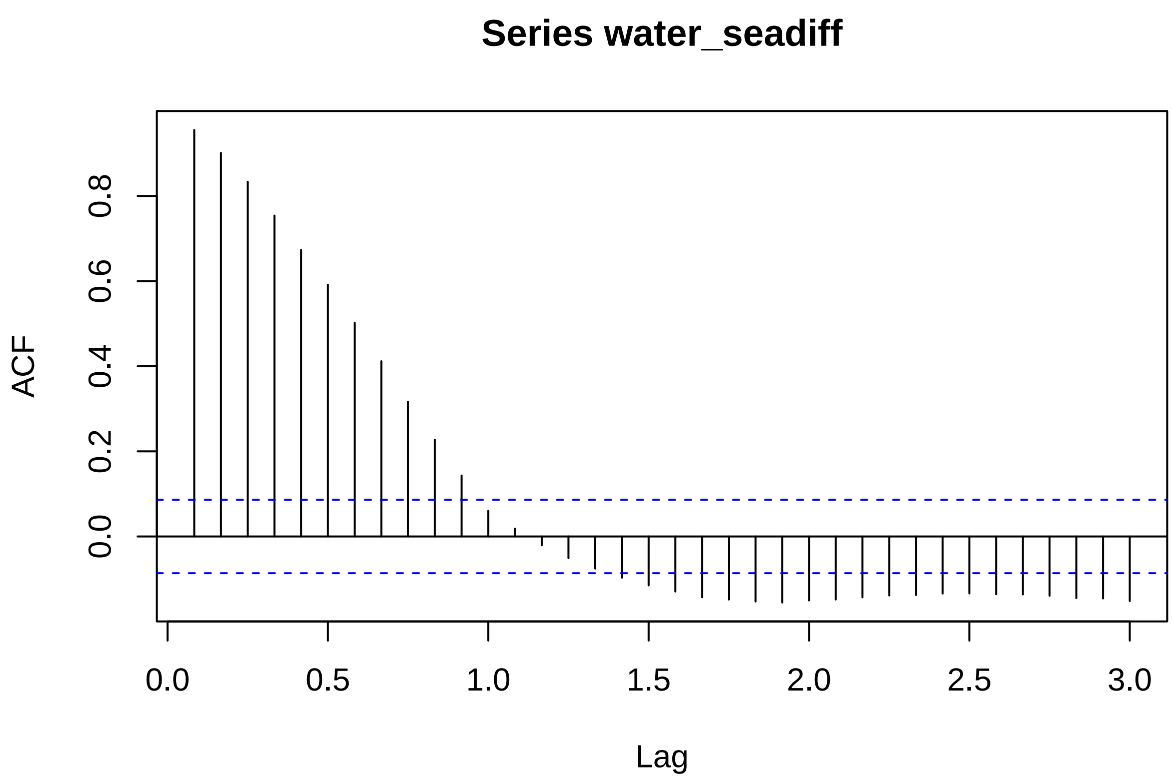
```
adf.test(precipitation.ts) # small p-value, series *is* level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  precipitation.ts
## Dickey-Fuller = -9.5048, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

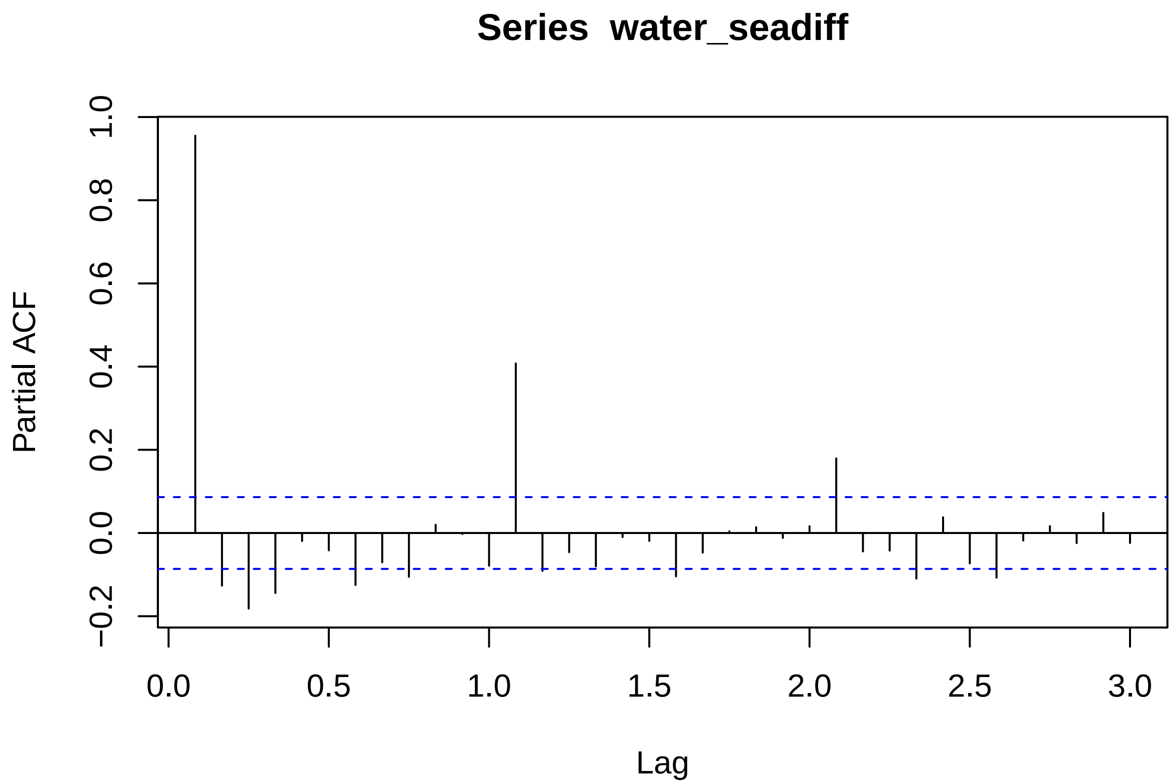
```
# Try with one order of seasonal differencing first for water level
water_seadiff <- diff(water_level.ts, lag=12)
plot(water_seadiff, ylab="Seasonal Difference of Water Level", xlab="Year")
```



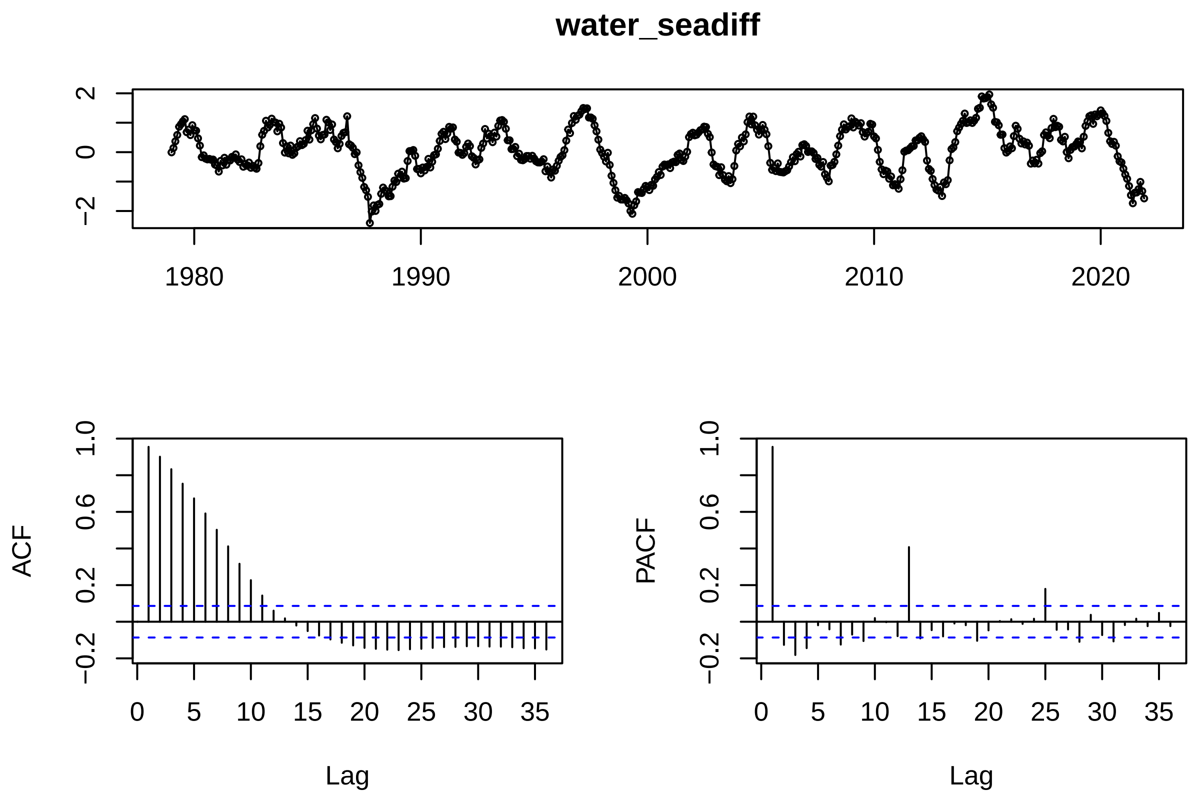

```
acf(water_seadiff, lag=36)
```



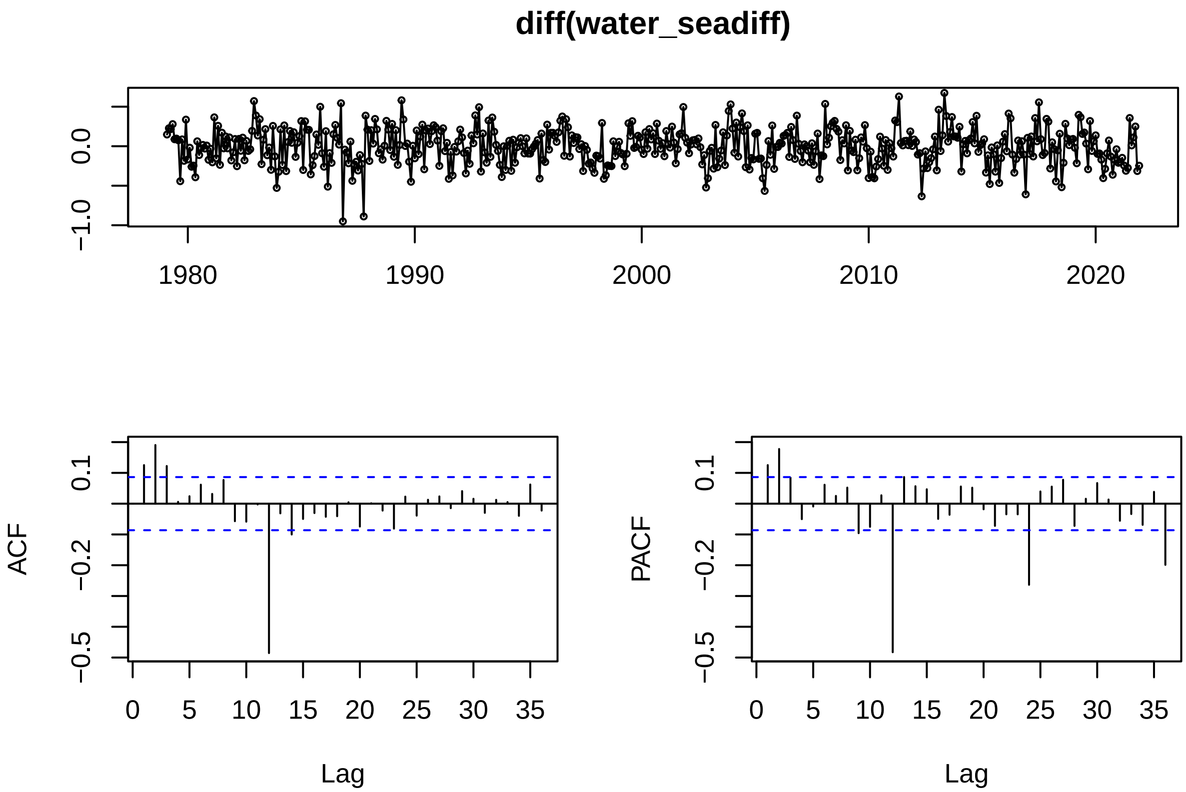

```
pacf(water_seadiff, lag=36)
```



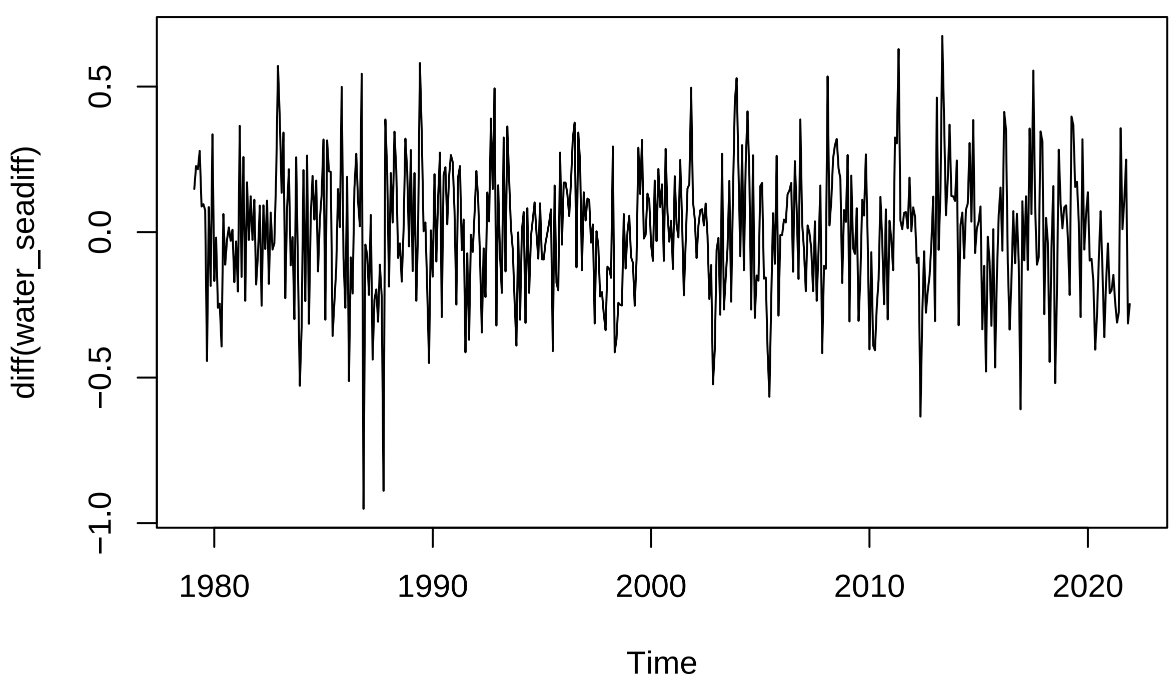

```
tsdisplay(water_seadiff)
```



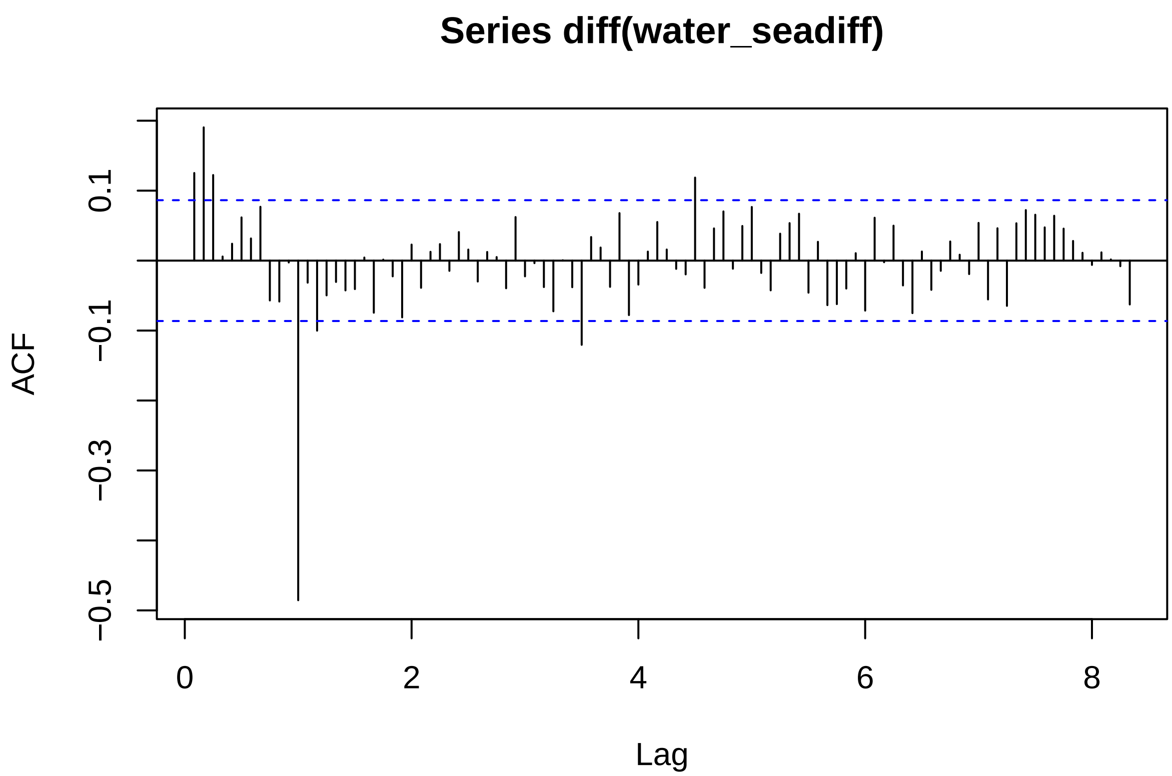
```r
tsdisplay(diff(water_seadiff))
```



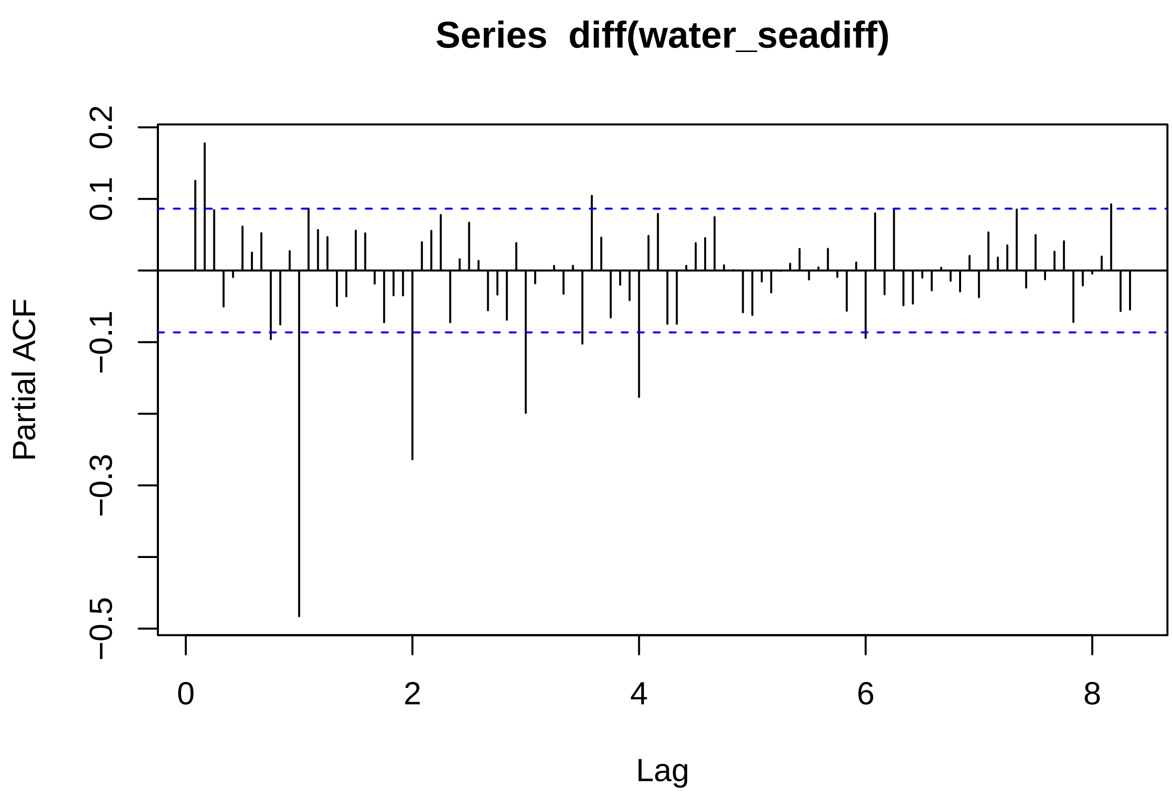

```r
#checkresiduals(water_seadiff)

plot(diff(water_seadiff))
```




```r
acf(diff(water_seadiff), lag=100)
```

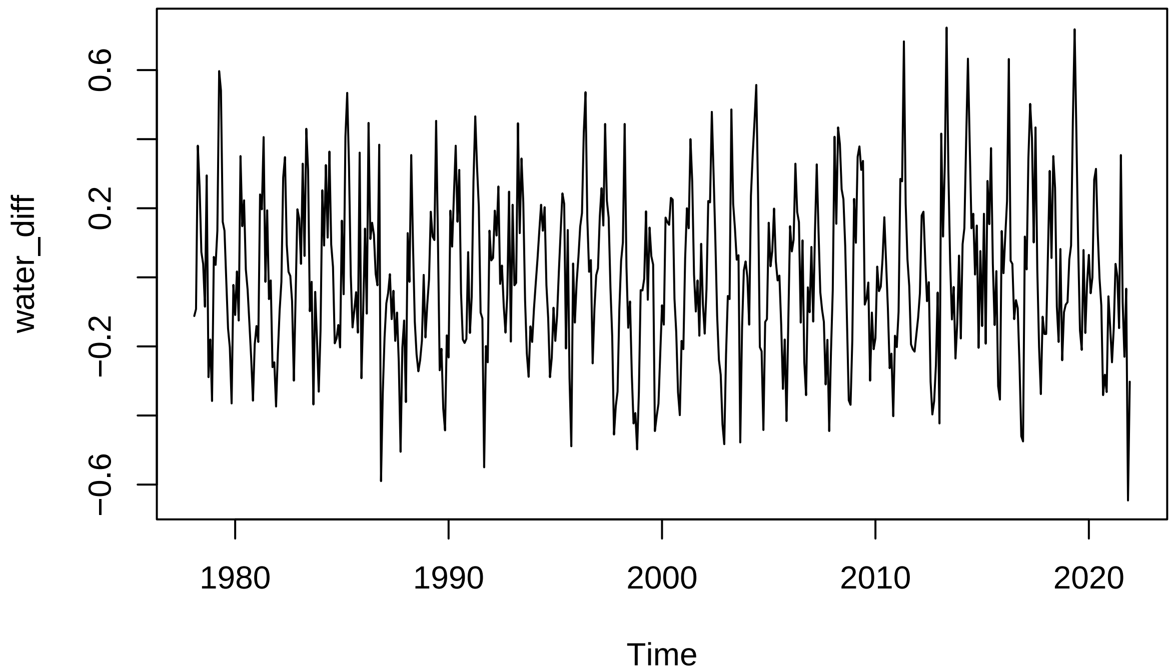



```
pacf(diff(water_seadiff), lag=100)
```

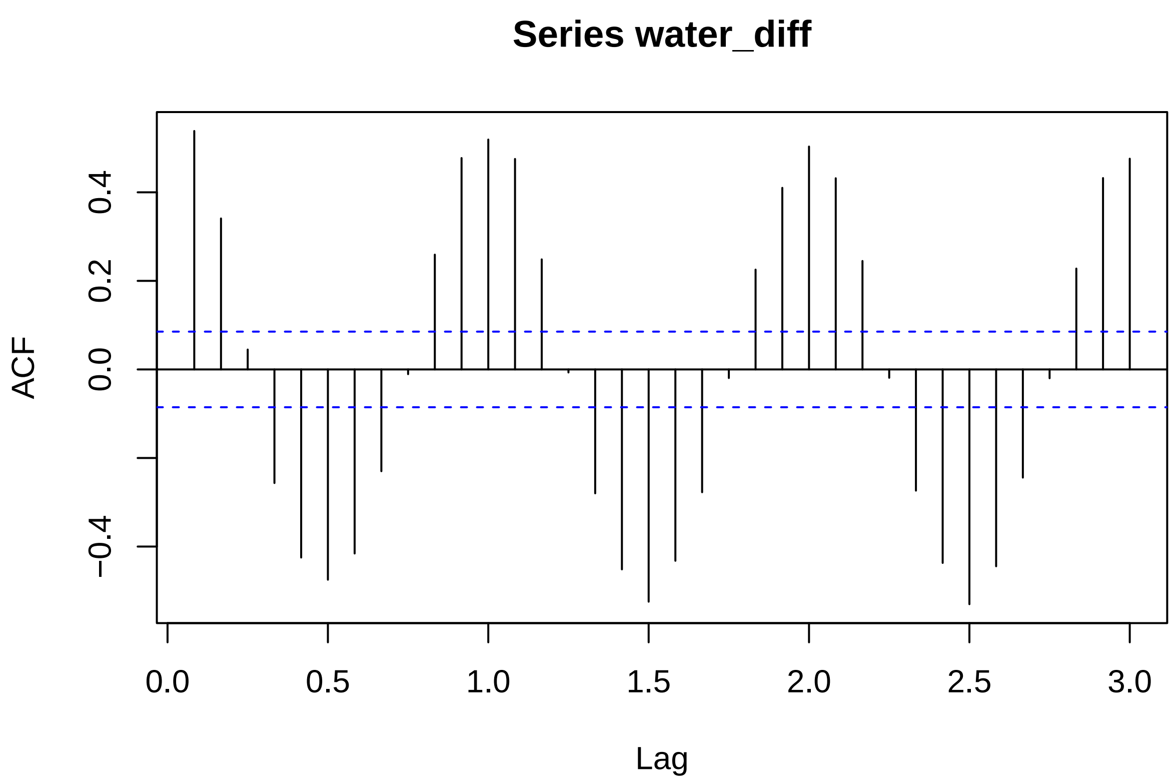



```
#checkresiduals(diff(water_seadiff))

# Try with one order regular differencing first for water level
```

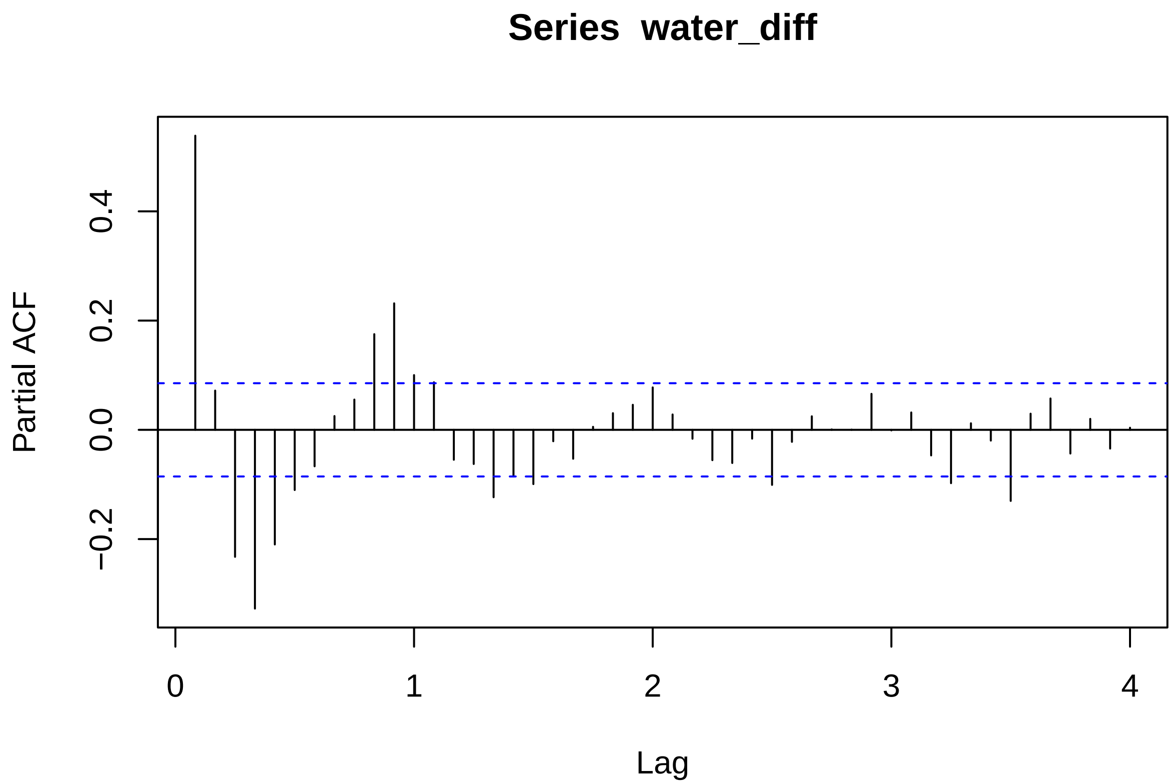
```r
water_diff <- diff(water_level.ts)
plot(water_diff)
```



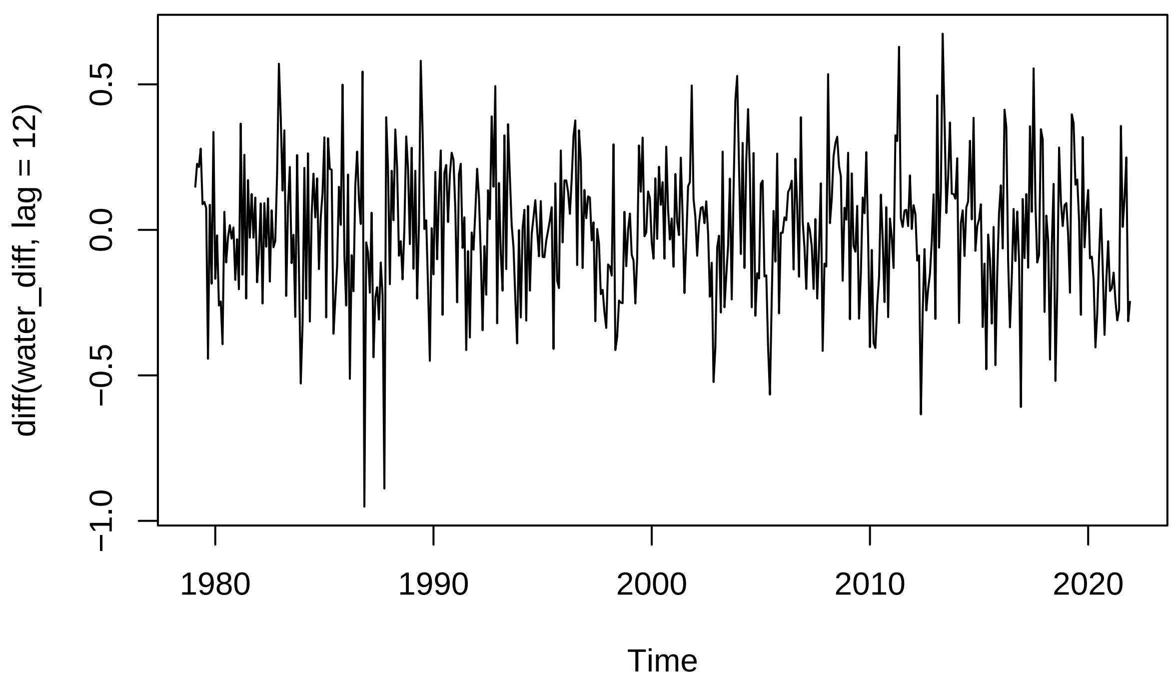

```r
acf(water_diff, lag=36)
```



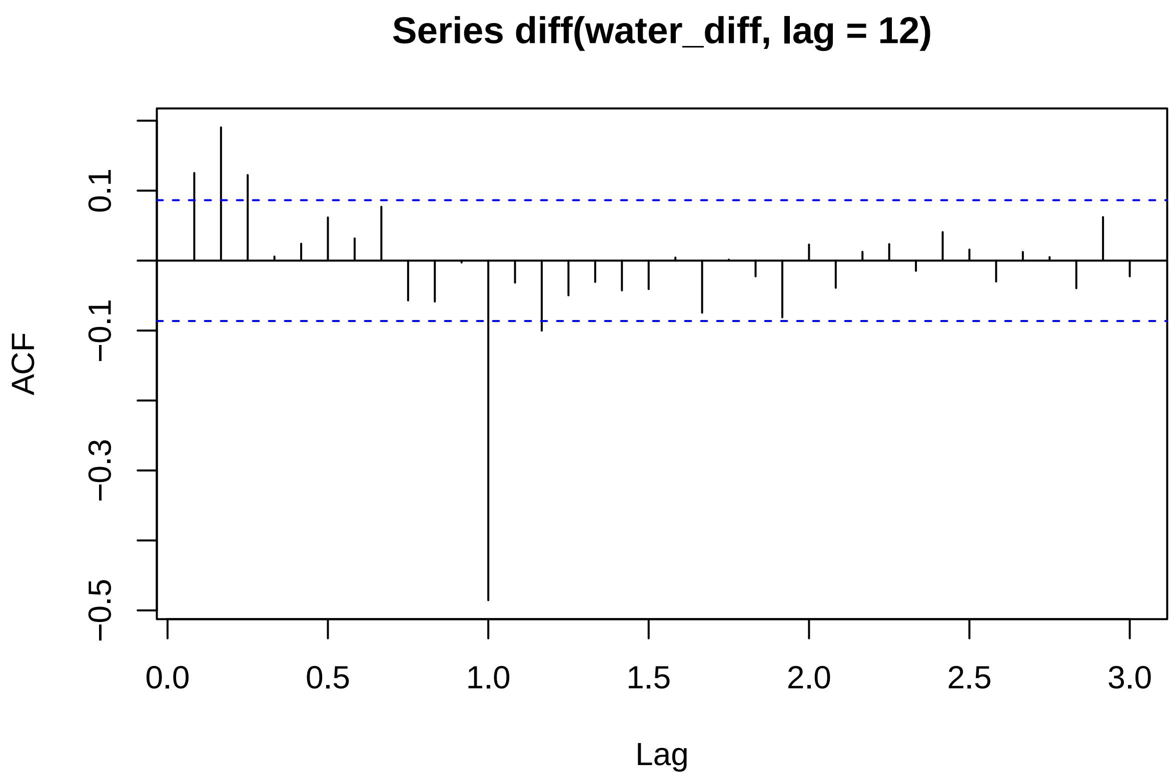

```r
pacf(water_diff, lag=48)
```



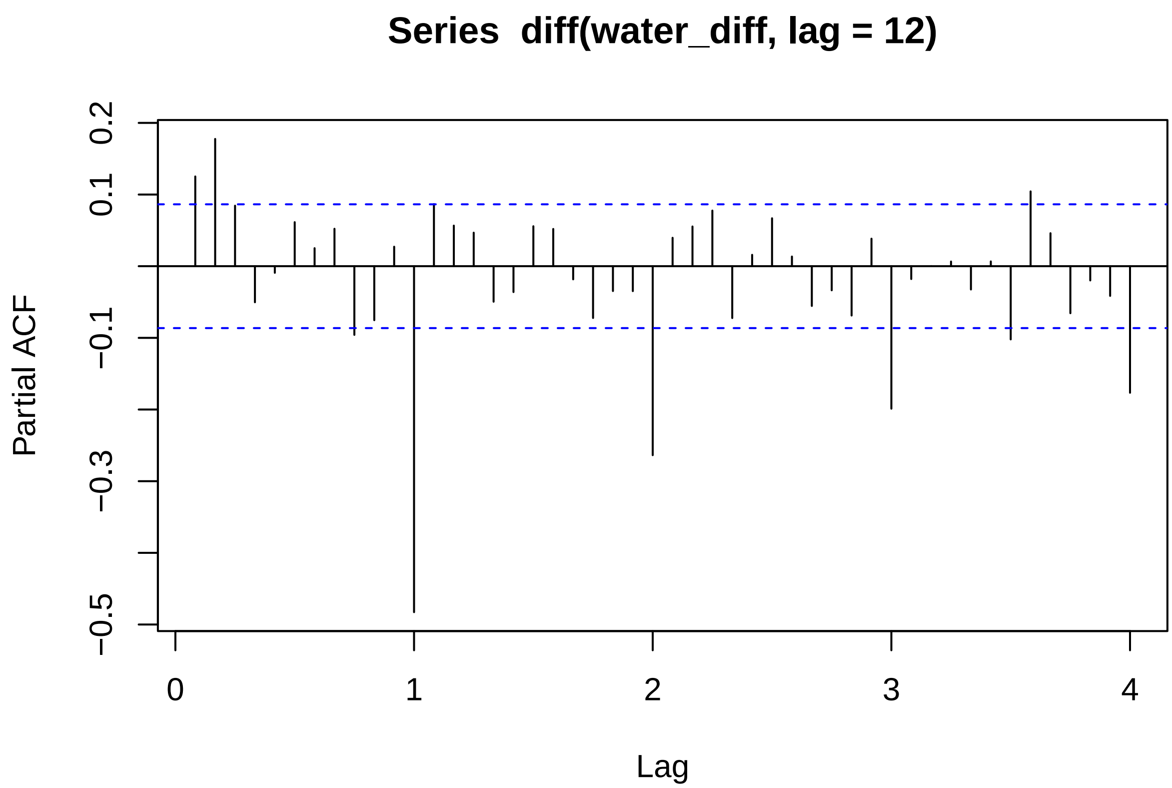

```
plot(diff(water_diff, lag=12))
```



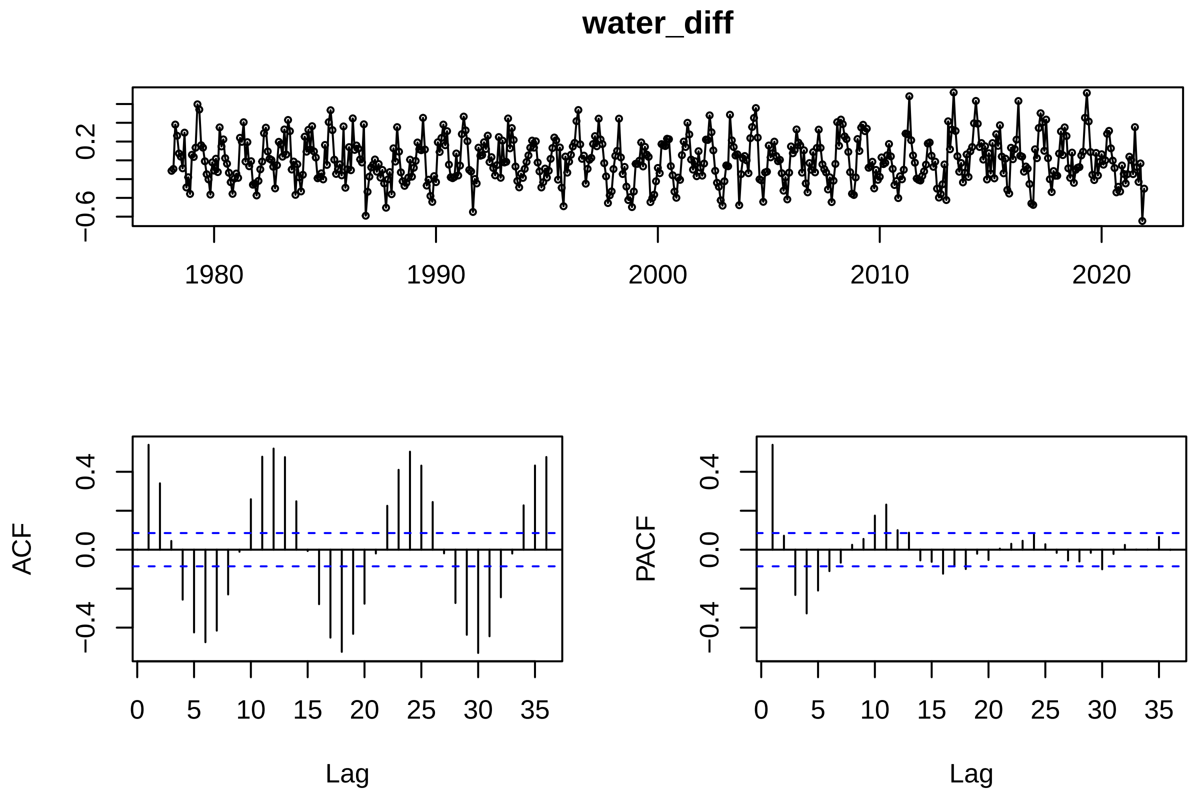

```
acf(diff(water_diff, lag=12), lag=36)
```



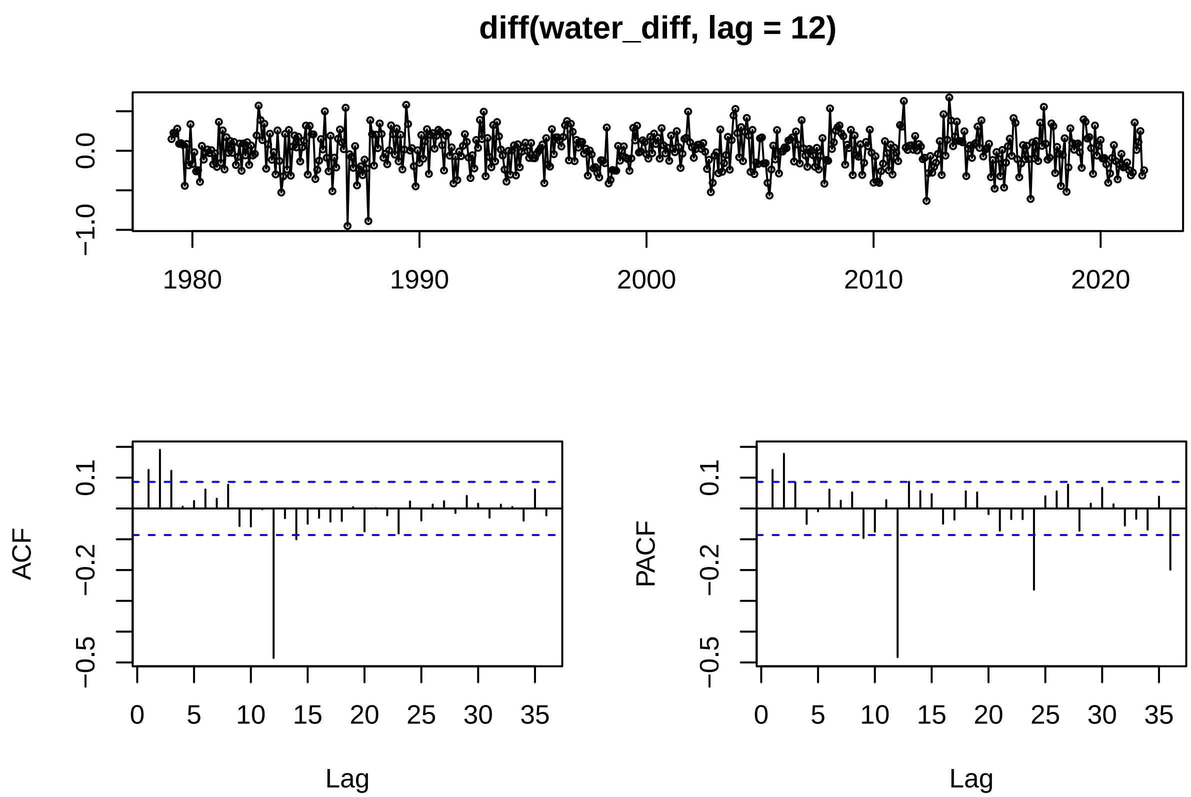

```
pacf(diff(water_diff, lag=12), lag=48)
```




```
tsdisplay(water_diff)
```

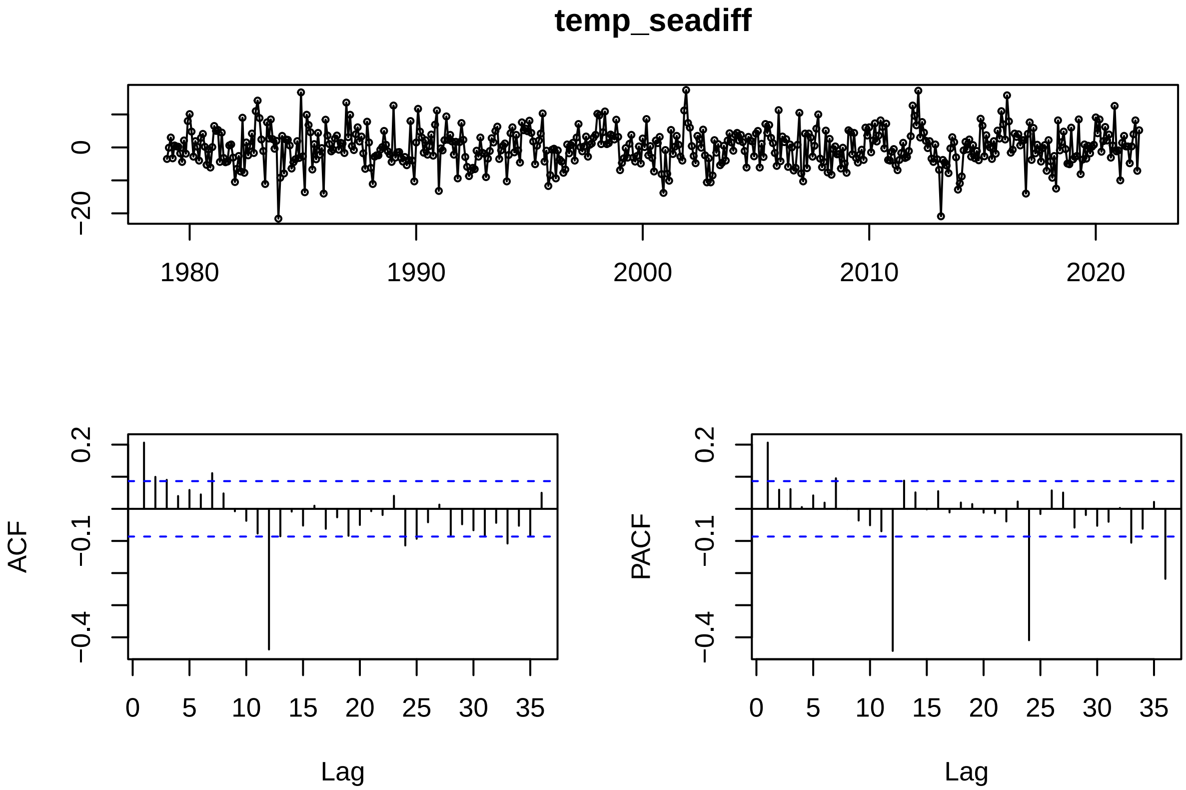



```r
tsdisplay(diff(water_diff, lag=12))
```

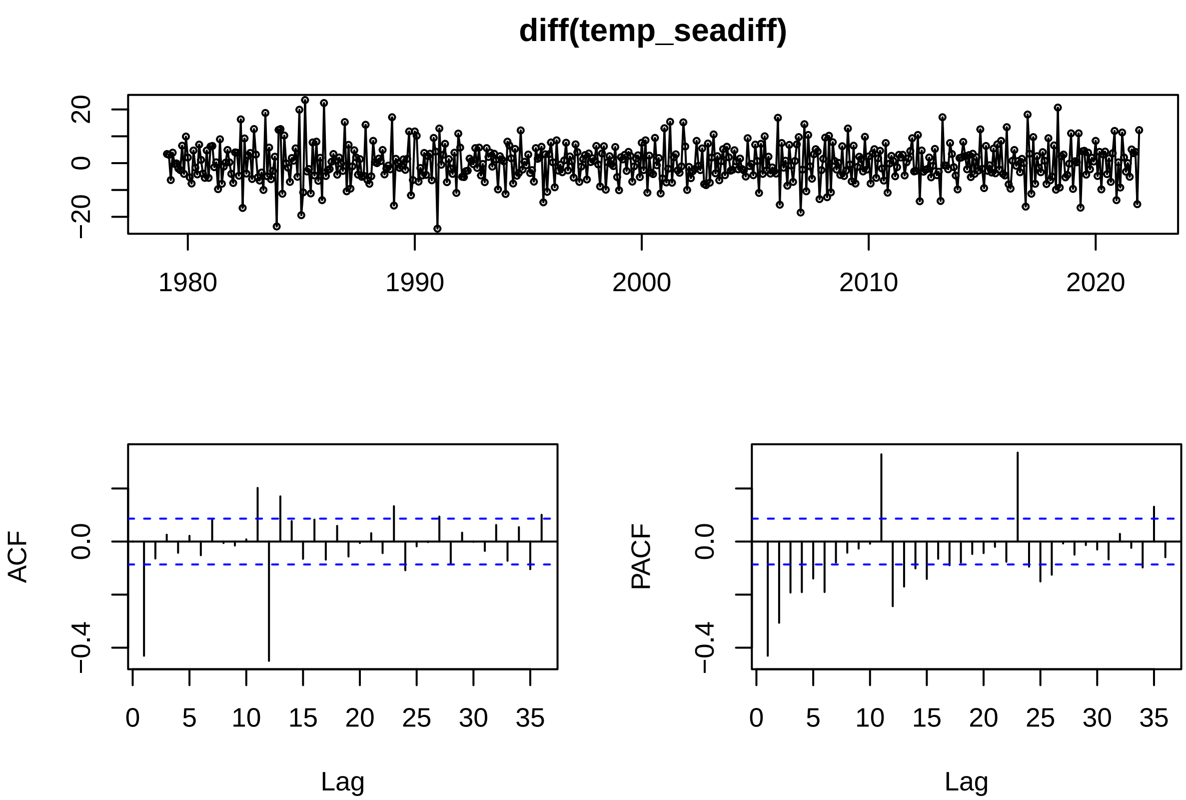



```r
# One order regular differencing with temperature
temp_seadiff <- diff(temperature.ts, lag=12)
```

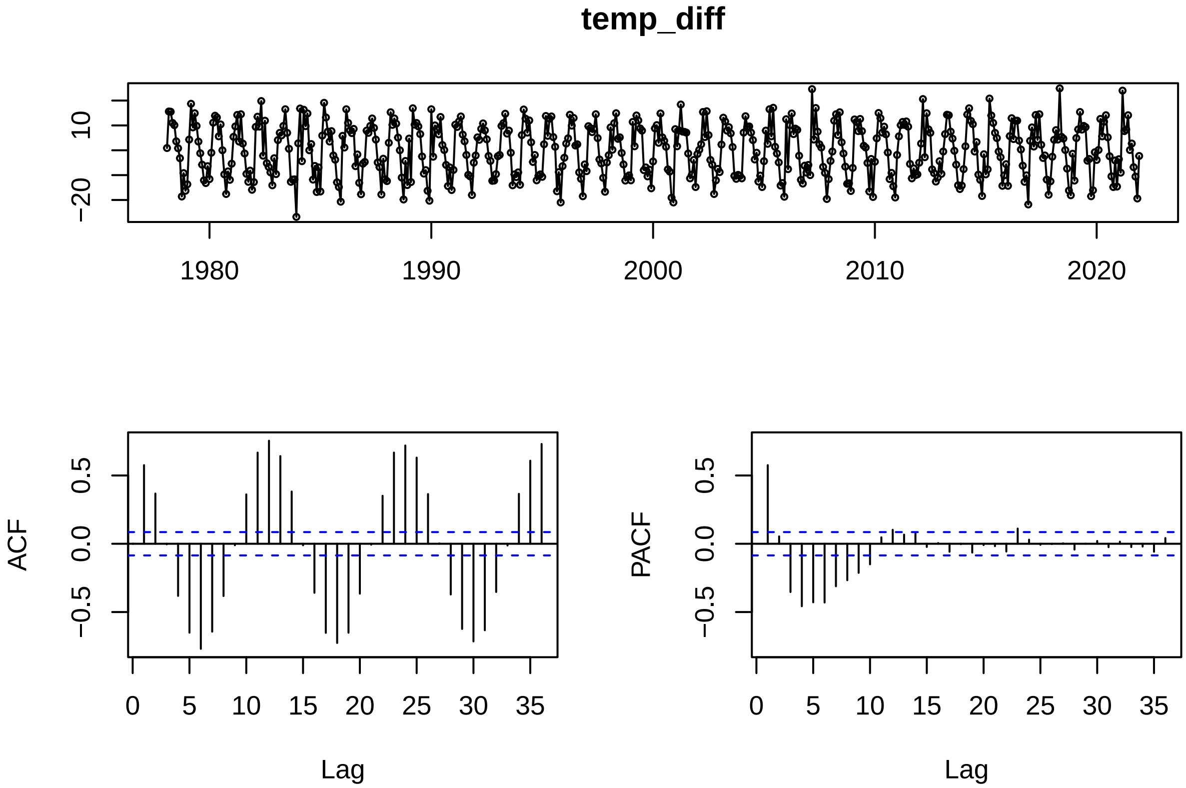
```
tsdisplay(temp_seadiff)
```



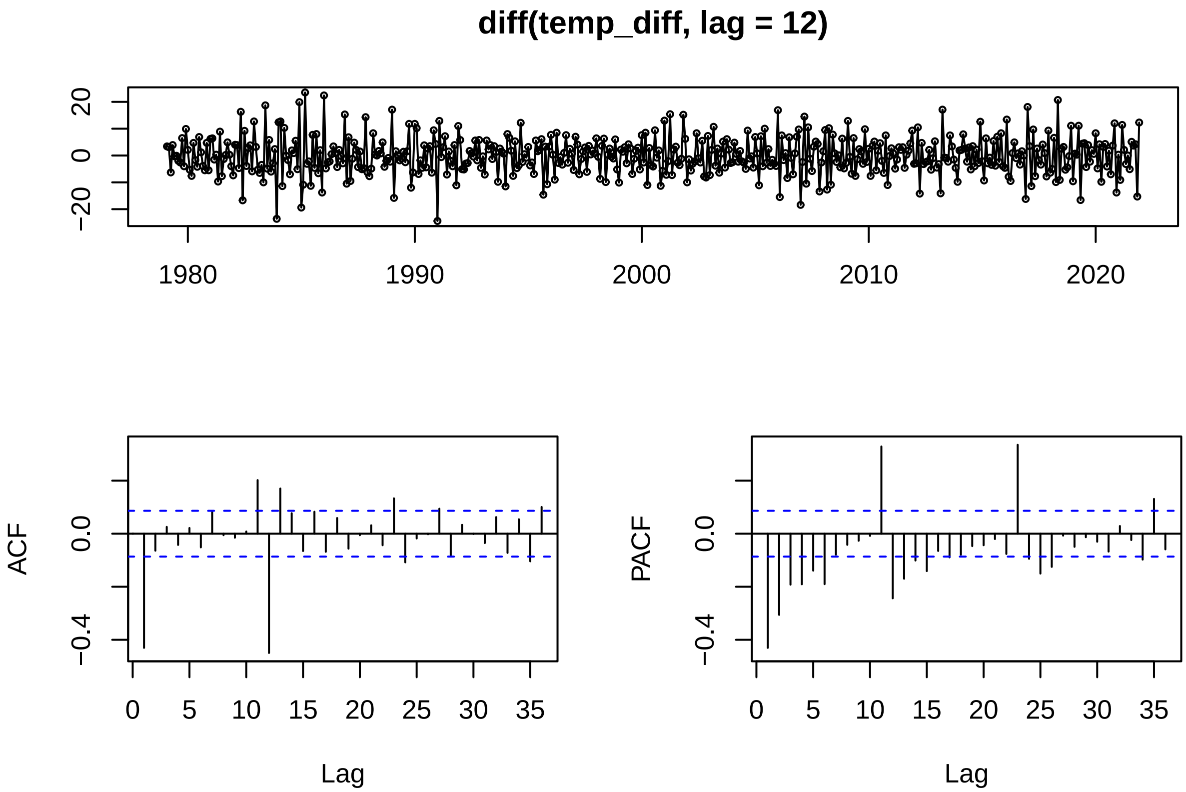

```
tsdisplay(diff(temp_seadiff))
```



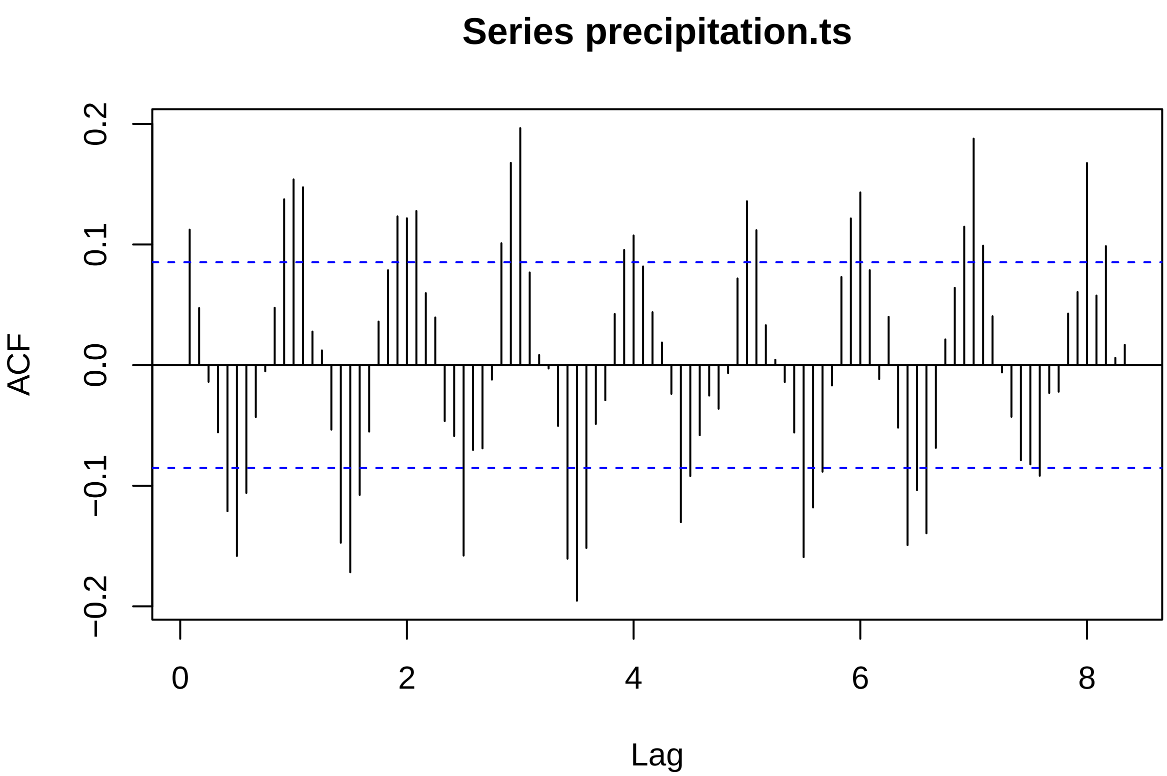
```
temp_diff <- diff(temperature.ts)
tsdisplay(temp_diff)
```



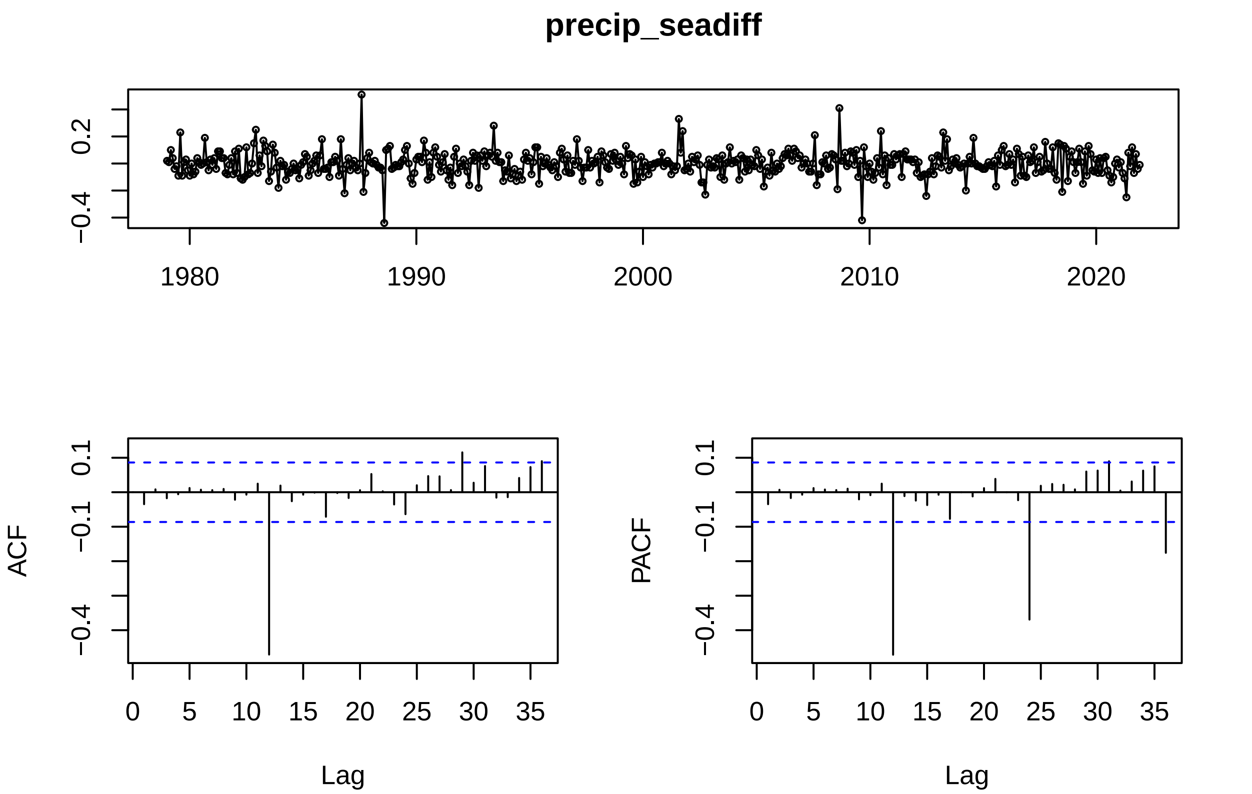

```
tsdisplay(diff(temp_diff, lag=12))
```



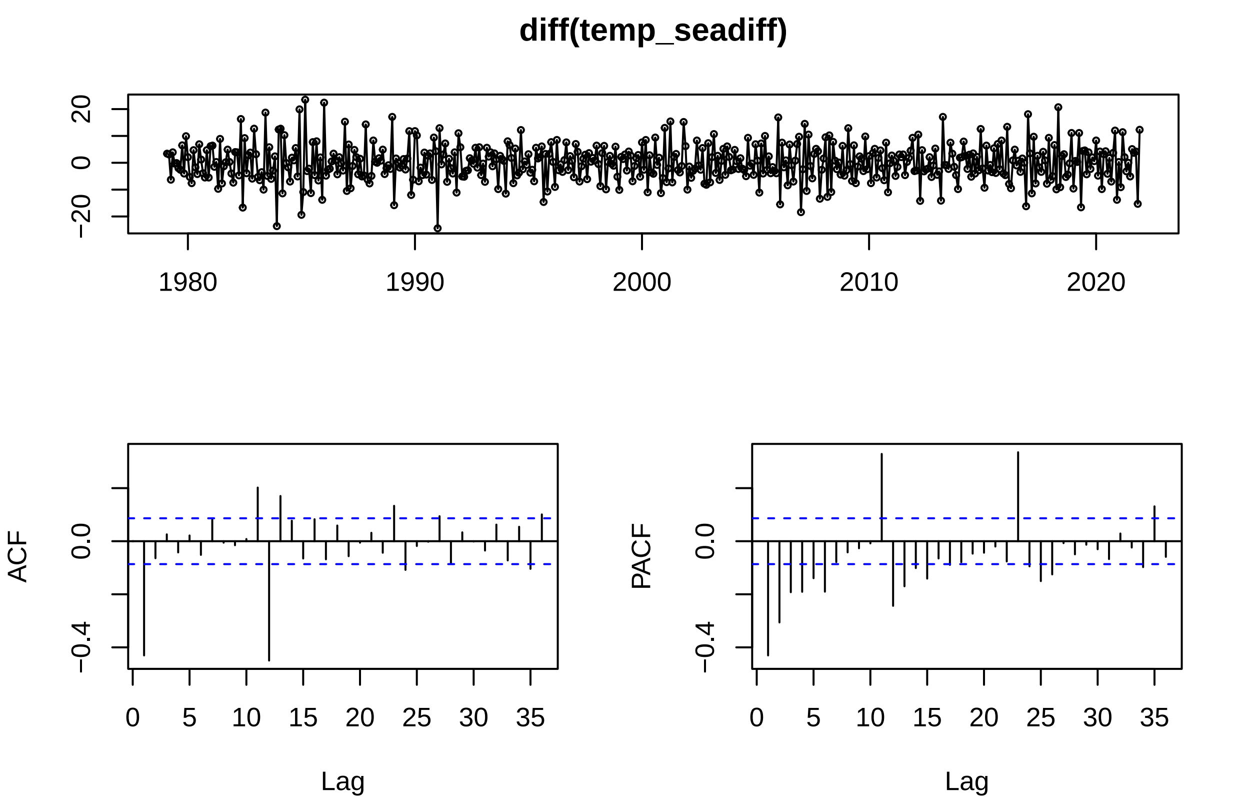

```
# One order regular differencing with precipitation
acf(precipitation.ts, lag.max=100)
```




```
precip_seadiff <- diff(precipitation.ts, lag=12)
tsdisplay(precip_seadiff)
```




```
tsdisplay(diff(temp_seadiff))
```



```r
ndiffs(temperature.ts)
```

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

```r
nsdiffs(temperature.ts)
```

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

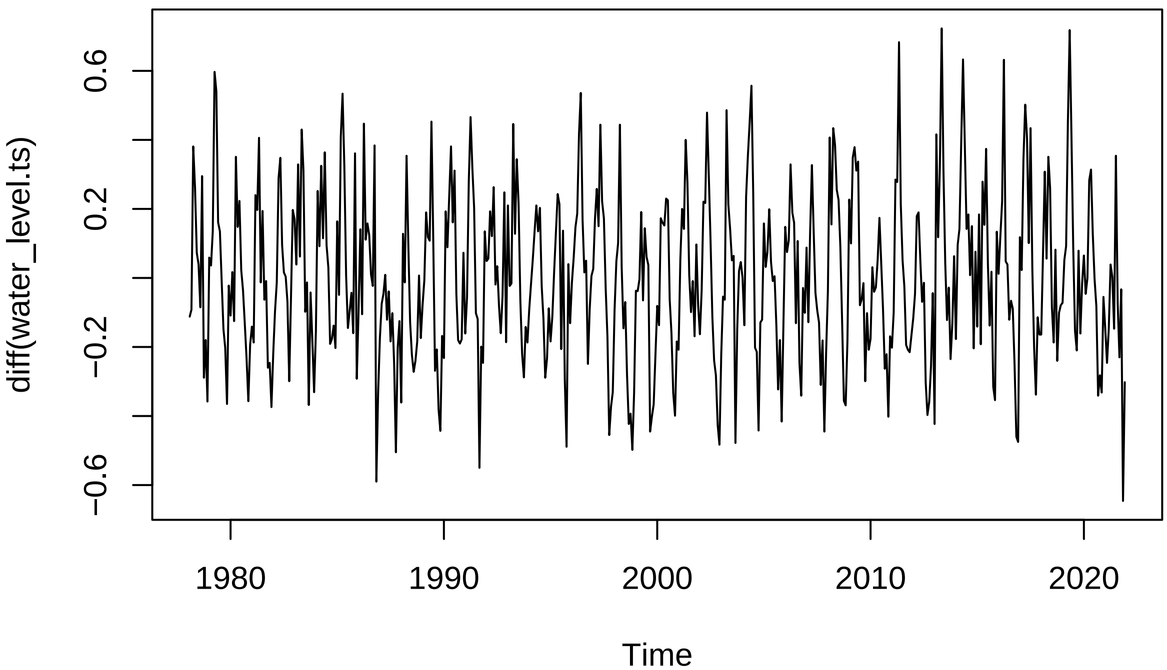
```r
ndiffs(precipitation.ts)
```

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

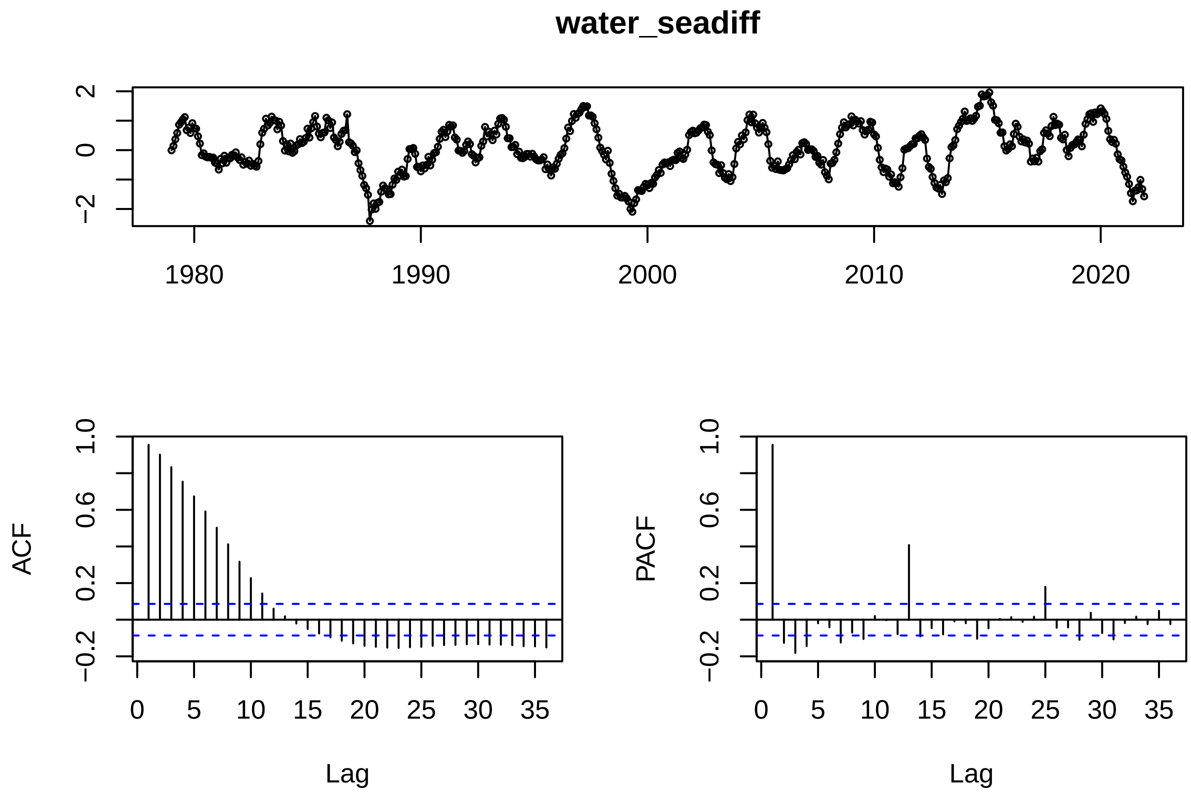
```r
nsdiffs(precipitation.ts)
```

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

```r
plot(diff(water_level.ts))
```




```r
tsdisplay(water_seadiff)
```




```r
kpss.test(water_seadiff) # large p-value, series is level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  water_seadiff
## KPSS Level = 0.19833, Truncation lag parameter = 6, p-value = 0.1
```

```r
adf.test(water_seadiff) # small p-value, series is level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  water_seadiff
## Dickey-Fuller = -6.2712, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
ndiffs(water_level.ts) # 1 order of differencing required
```

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

```r
nsdiffs(water_level.ts) # 1 order of seasonal differencing required
```

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

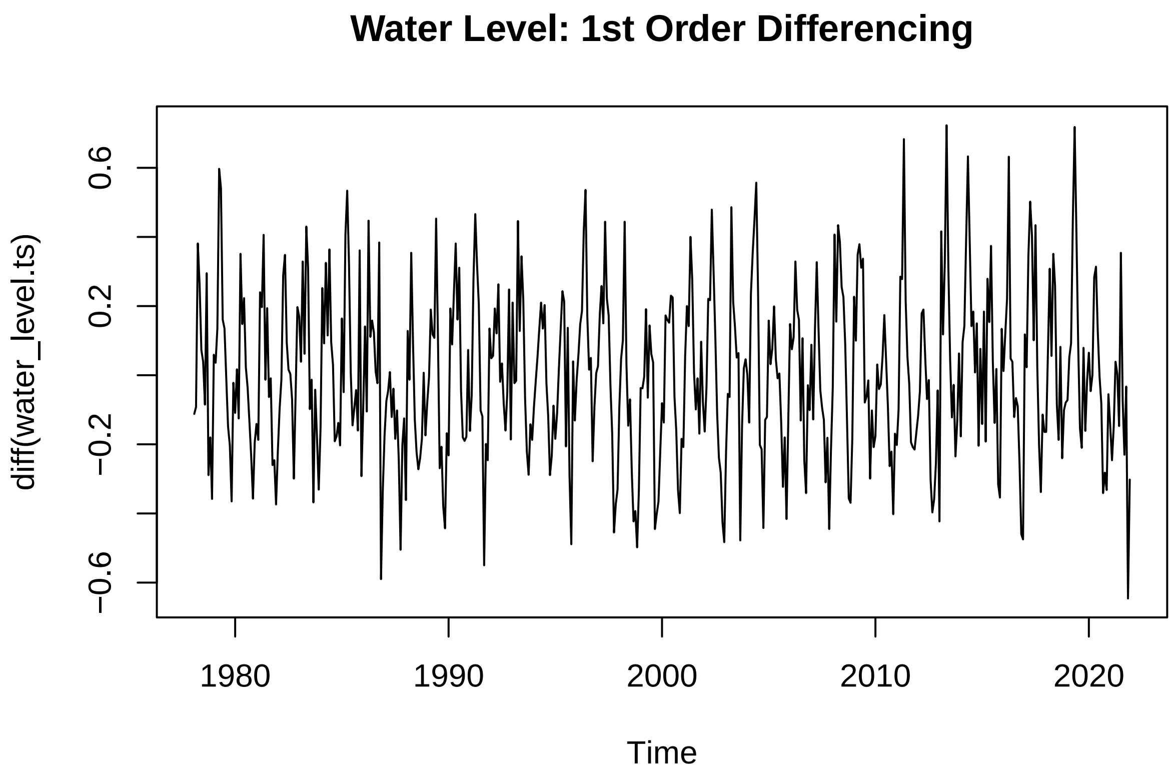
```r
ndiffs(water_seadiff)
```

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

```r
nsdiffs(water_diff)
```

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

```r
# First order differencing
plot(diff(water_level.ts), main="Water Level: 1st Order Differencing")
```



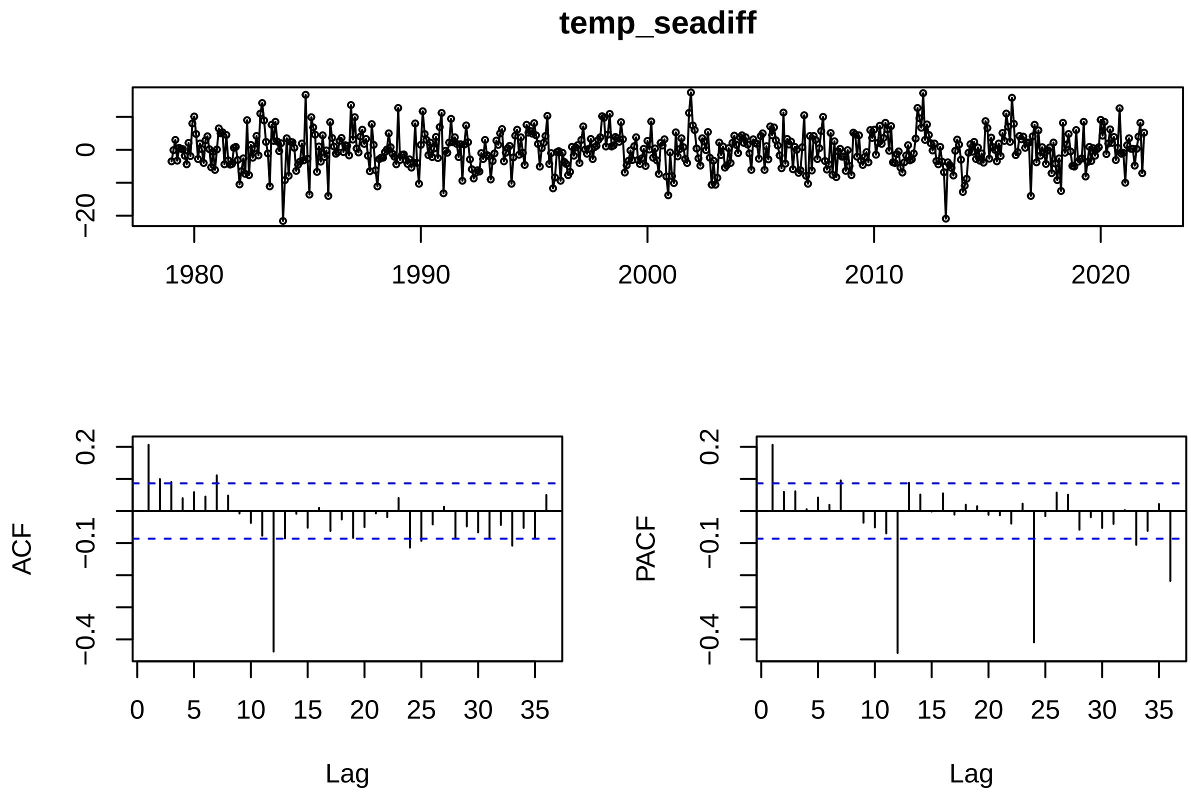

```r
kpss.test(diff(water_level.ts)) # large p-value, series is level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(water_level.ts)
## KPSS Level = 0.037031, Truncation lag parameter = 6, p-value = 0.1
```

```r
adf.test(diff(water_level.ts)) # small p-value, series is level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(water_level.ts)
## Dickey-Fuller = -9.0884, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
temp_seadiff <- diff(temperature.ts, lag=12)
tsdisplay(temp_seadiff)
```




```r
kpss.test(temp_seadiff) # large p-value, series is level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  temp_seadiff
## KPSS Level = 0.018831, Truncation lag parameter = 6, p-value = 0.1
```

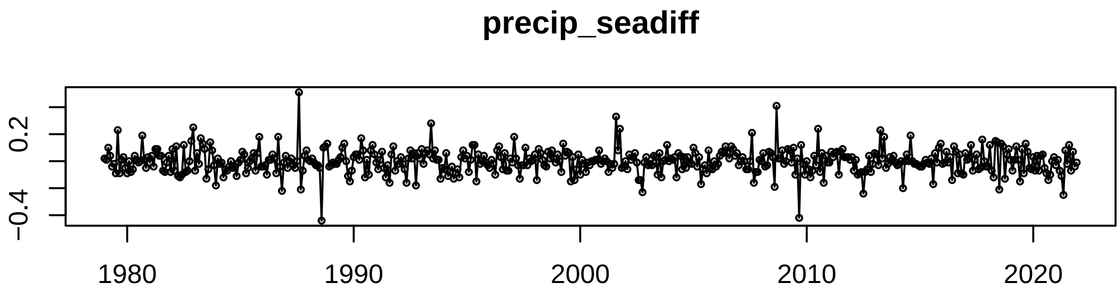
```r
adf.test(temp_seadiff) # small p-value, series is level stationary
```

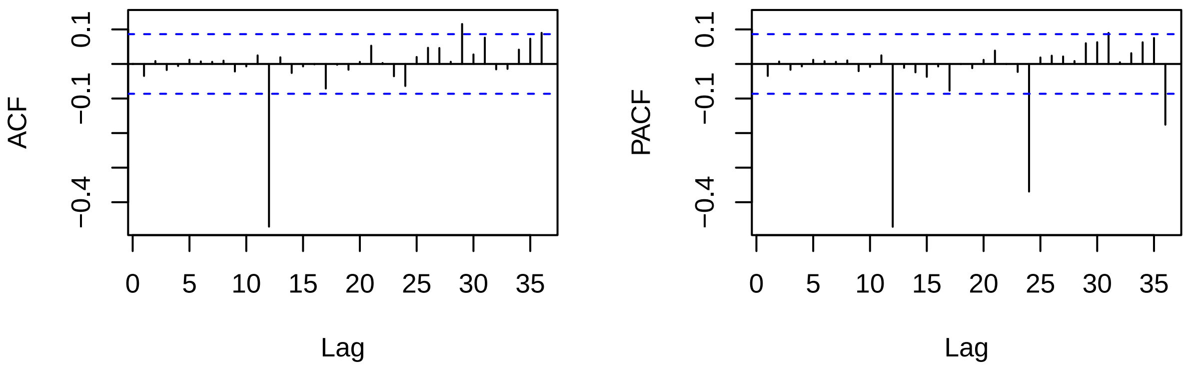
```
##
##  Augmented Dickey-Fuller Test
##
## data:  temp_seadiff
## Dickey-Fuller = -6.3588, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
nsdiffs(temperature.ts) # 1
```

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

```r
precip_seadiff <- diff(precipitation.ts, lag=12)
tsdisplay(precip_seadiff)
```






```
kpss.test(precip_seadiff) # large p-value, series is level stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  precip_seadiff
## KPSS Level = 0.023824, Truncation lag parameter = 6, p-value = 0.1
```

```
adf.test(precip_seadiff) # small p-value, series is level stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  precip_seadiff
## Dickey-Fuller = -7.4748, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
nsdiffs(precipitation.ts) # 0
```

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

STEP 5: Train/Test split

```
# train time period: 1/1978 - 12/2016
# test time period: 1/2017 - 12/2021

water_train <- window(water_level.ts, start=c(1978,1), end=c(2016,12))
water_test <- window(water_level.ts, start=c(2017,1), end=c(2021,12))
```

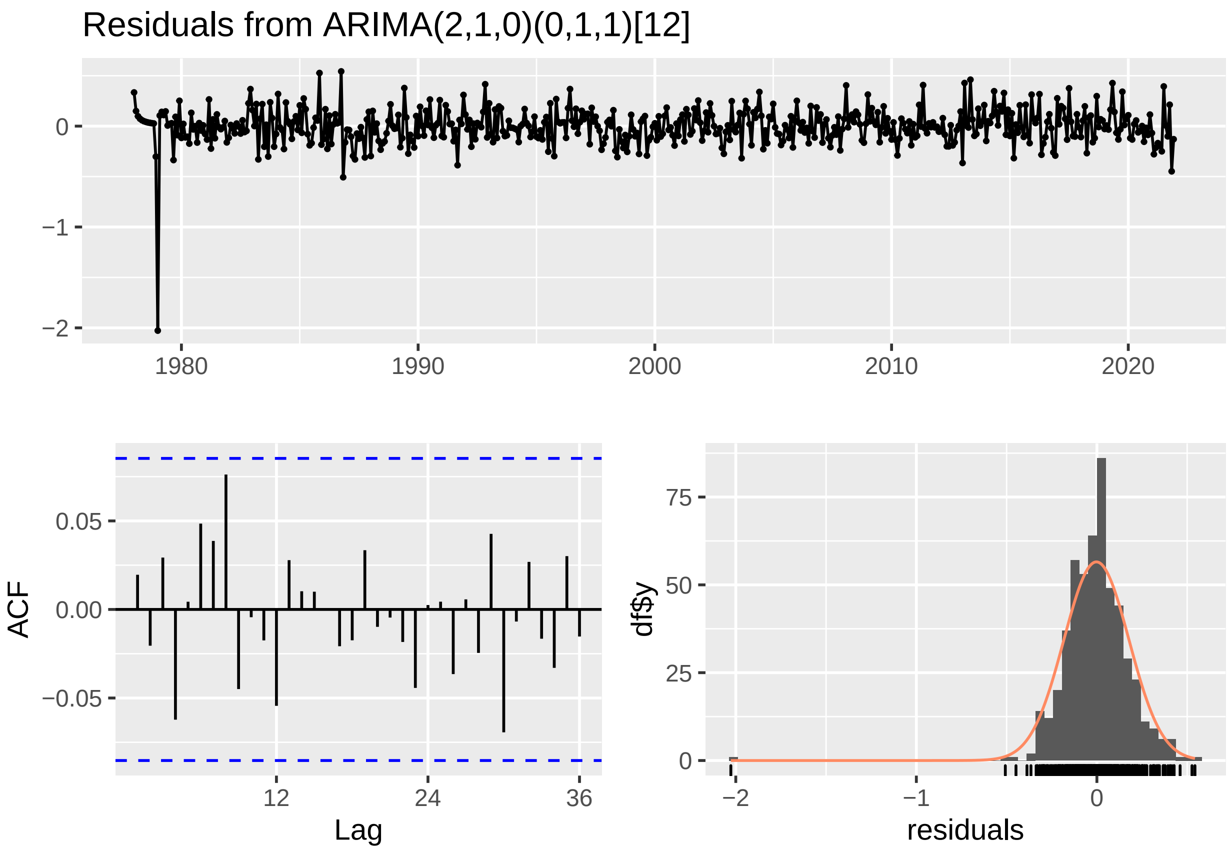
```r
temp_train <- window(temperature.ts, start=c(1978,1), end=c(2016,12))
temp_test <- window(temperature.ts, start=c(2017,1), end=c(2021,12))

precip_train <- window(precipitation.ts, start=c(1978,1), end=c(2016,12))
precip_test <- window(precipitation.ts, start=c(2017,1), end=c(2021,12))

df_train <-data.frame(water_train, temp_train, precip_train)
df_test <- data.frame(water_test, temp_test, precip_test)

# forecast horizon
h <- as.integer(length(water_test))
```

STEP 6: Model & Forecast: sARIMA

```r
arima.seasonal <- Arima(water_level.ts, order = c(2, 1, 0), seasonal = c(0, 1, 1))
arima.seasonal
```

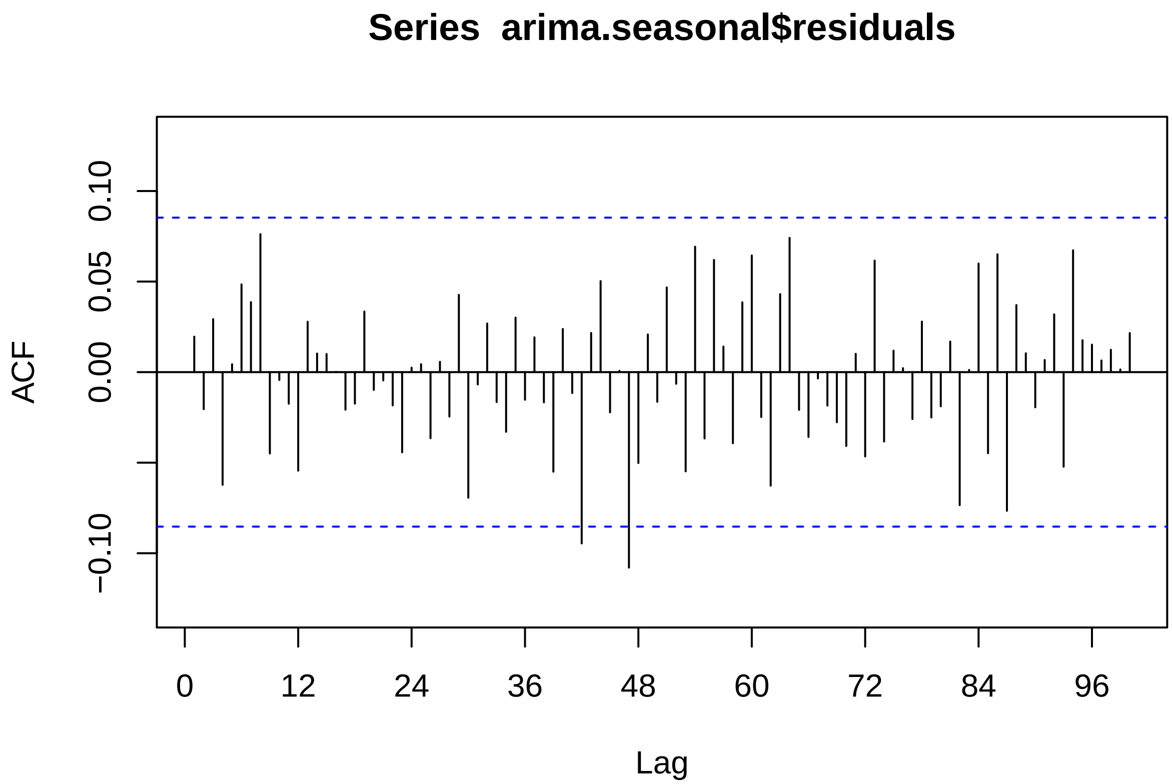
```
## Series: water_level.ts
## ARIMA(2,1,0)(0,1,1)[12]
##
## Coefficients:
##          ar1     ar2     sma1
##       0.1398  0.1651  -0.9999
## s.e.  0.0434  0.0438   0.0340
##
## sigma^2 = 0.03361:  log likelihood = 196.98
## AIC=-385.96   AICc=-385.89   BIC=-368.99
```

```r
checkresiduals(arima.seasonal)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)(0,1,1)[12]
## Q* = 13.912, df = 21, p-value = 0.8734
##
## Model df: 3.   Total lags used: 24
```

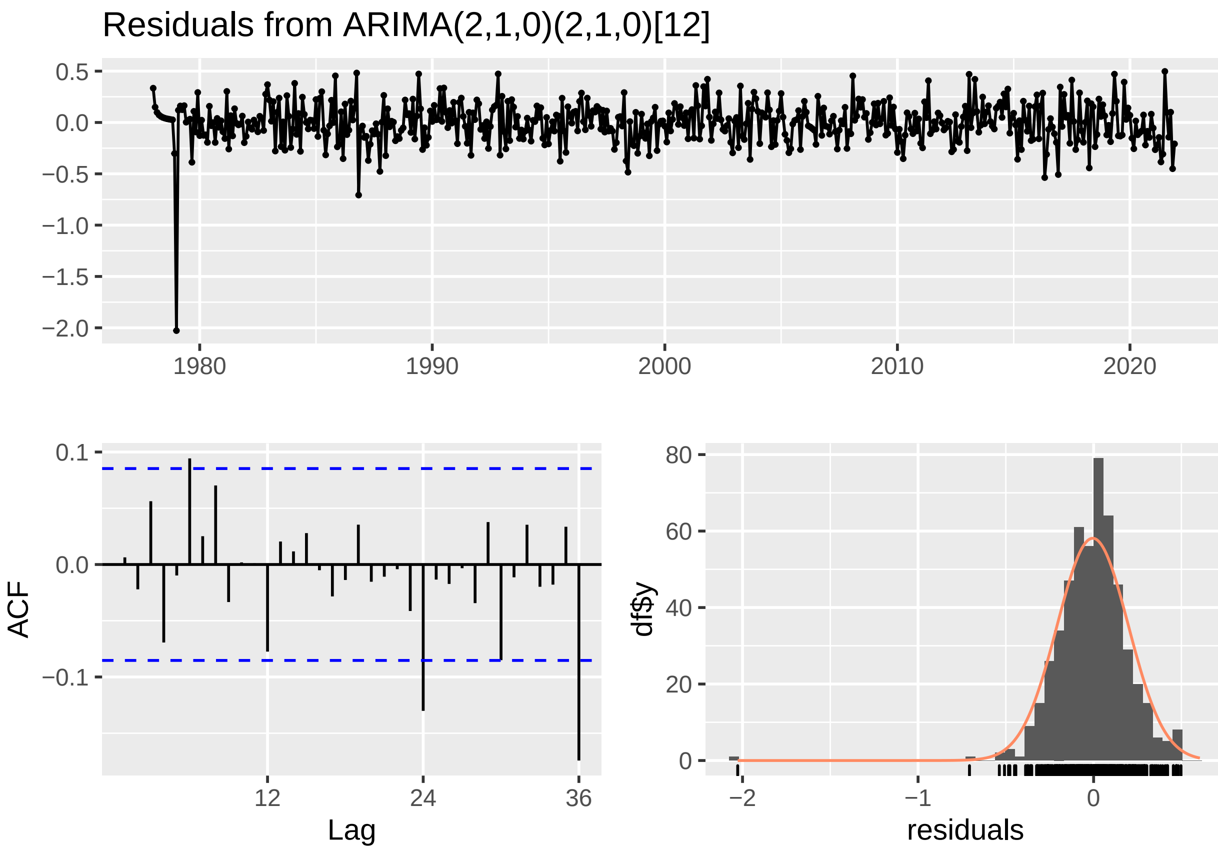
```
Acf(arima.seasonal$residuals, 100)
```




```r
arima.seasonal.2 <- Arima(water_level.ts, order = c(2, 1, 0), seasonal = c(2, 1, 0))
arima.seasonal.2
```

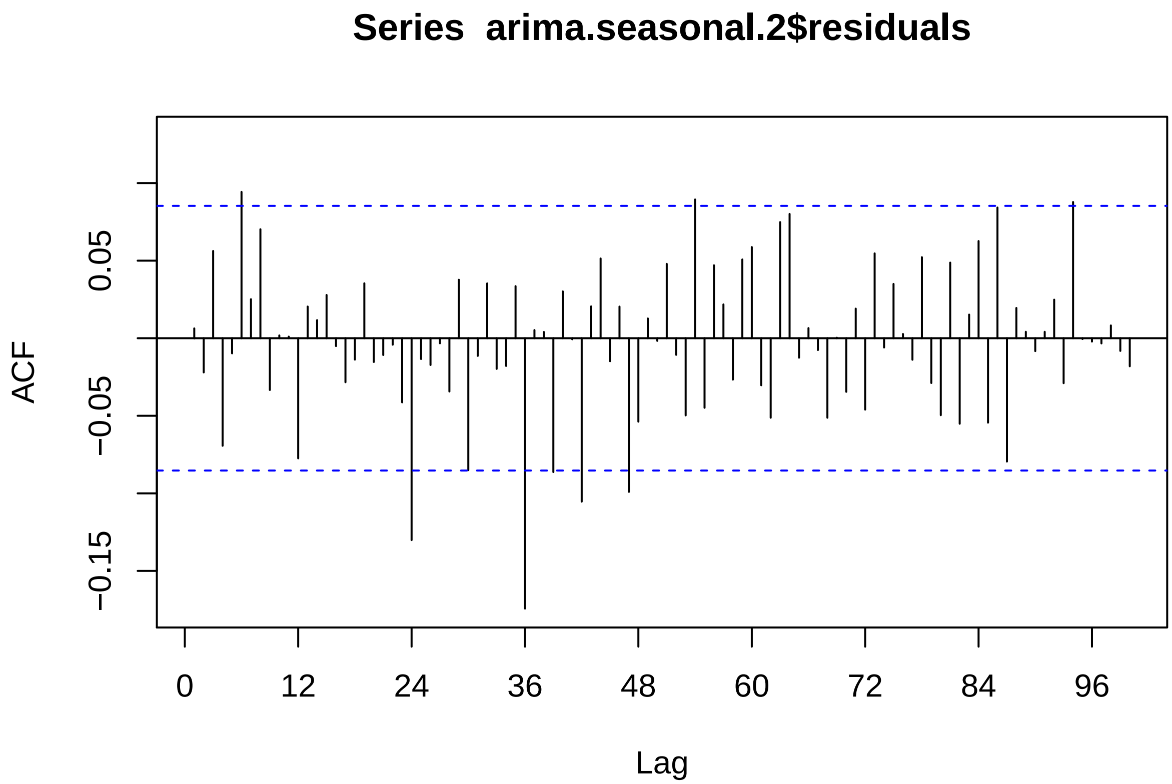
```
## Series: water_level.ts
## ARIMA(2,1,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     ar2     sar1     sar2
##       0.1447  0.1869  -0.6765  -0.3097
## s.e.  0.0436  0.0435   0.0429   0.0427
##
## sigma^2 = 0.0427:  log likelihood = 137.64
## AIC=-265.28   AICc=-265.17   BIC=-244.06
```

```r
checkresiduals(arima.seasonal.2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)(2,1,0)[12]
## Q* = 28.735, df = 20, p-value = 0.0931
##
## Model df: 4.   Total lags used: 24
```

```
Acf(arima.seasonal.2$residuals, 100)
```

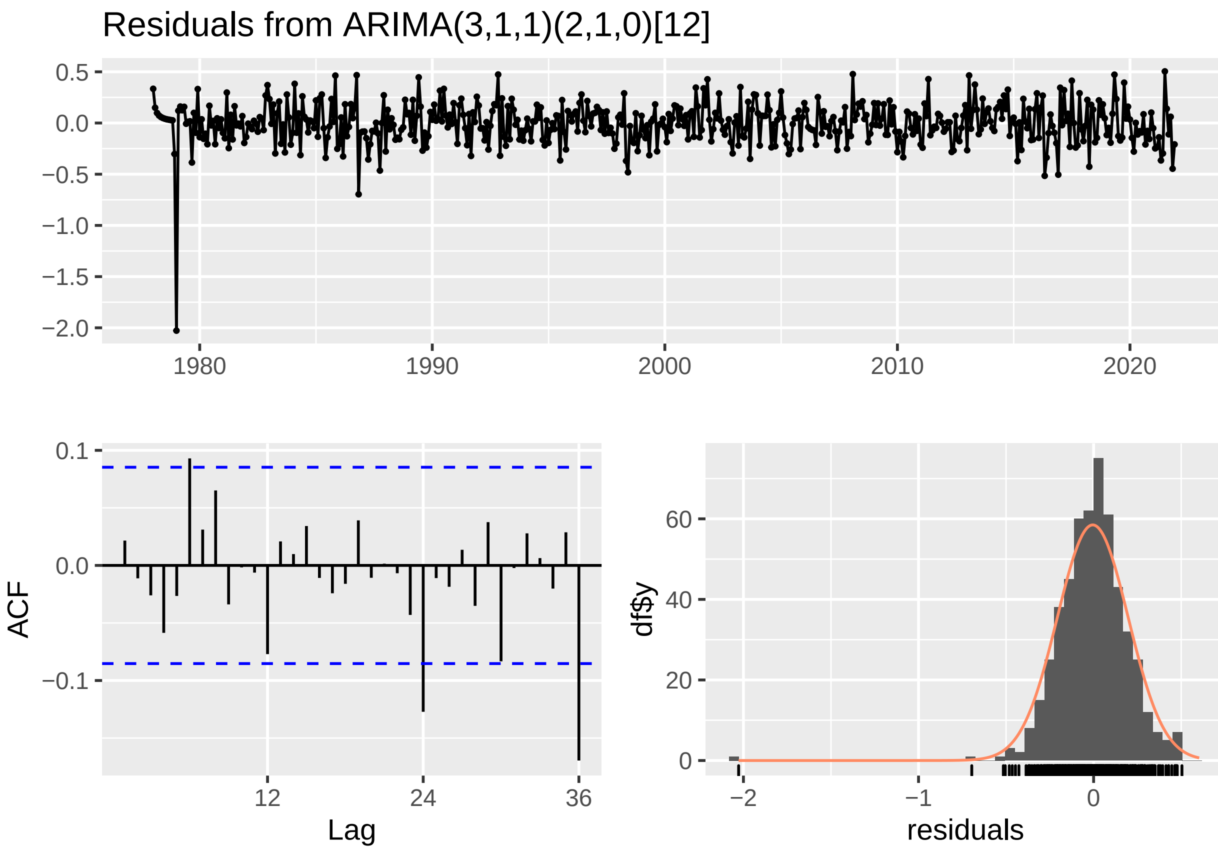


auto.arima()

```
arima.seasonal.3 <- auto.arima(water_level.ts, seasonal = TRUE, d = 1, D = 1)
arima.seasonal.3
```

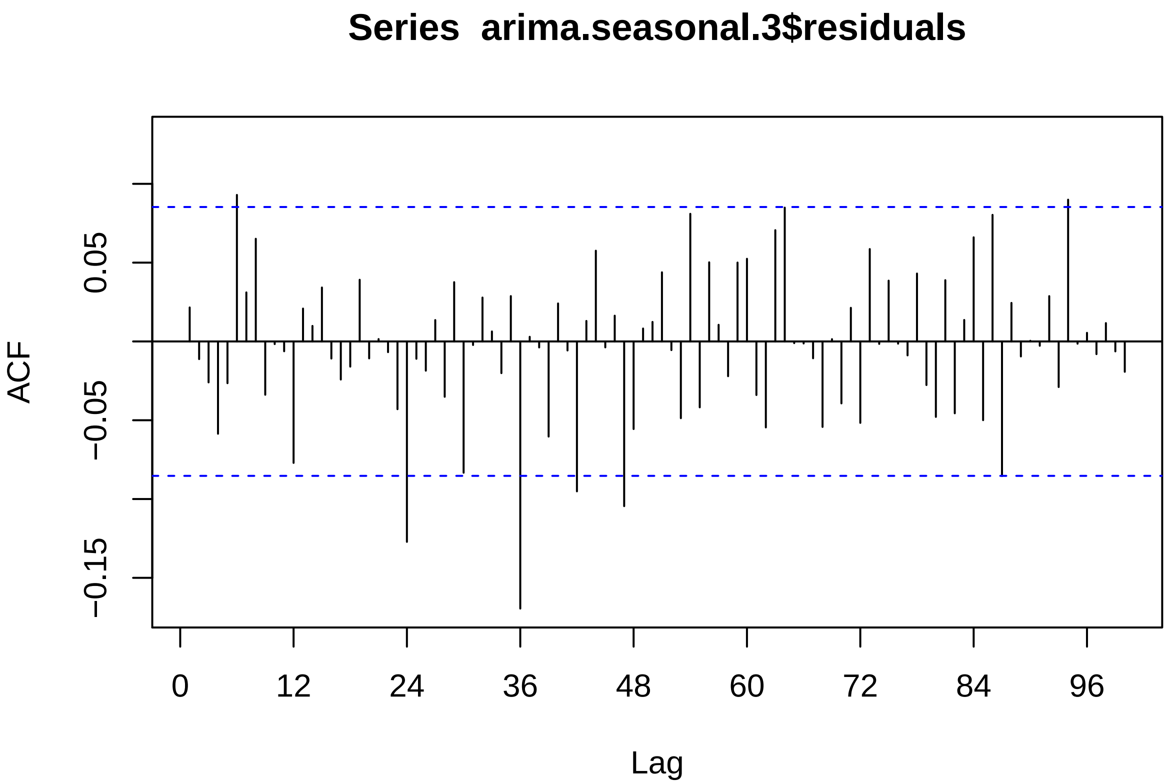
```
## Series: water_level.ts
## ARIMA(3,1,1)(2,1,0)[12]
##
## Coefficients:
##          ar1     ar2     ar3     ma1     sar1     sar2
##      -0.1216  0.2094  0.1409  0.2503  -0.6769  -0.3166
## s.e.  0.2651  0.0581  0.0612  0.2649   0.0429   0.0427
##
## sigma^2 = 0.04254:  log likelihood = 140.06
## AIC=-266.12   AICc=-265.9   BIC=-236.41
```

```
checkresiduals(arima.seasonal.3)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,1,1)(2,1,0)[12]
## Q* = 26.573, df = 18, p-value = 0.08736
##
## Model df: 6.   Total lags used: 24
```

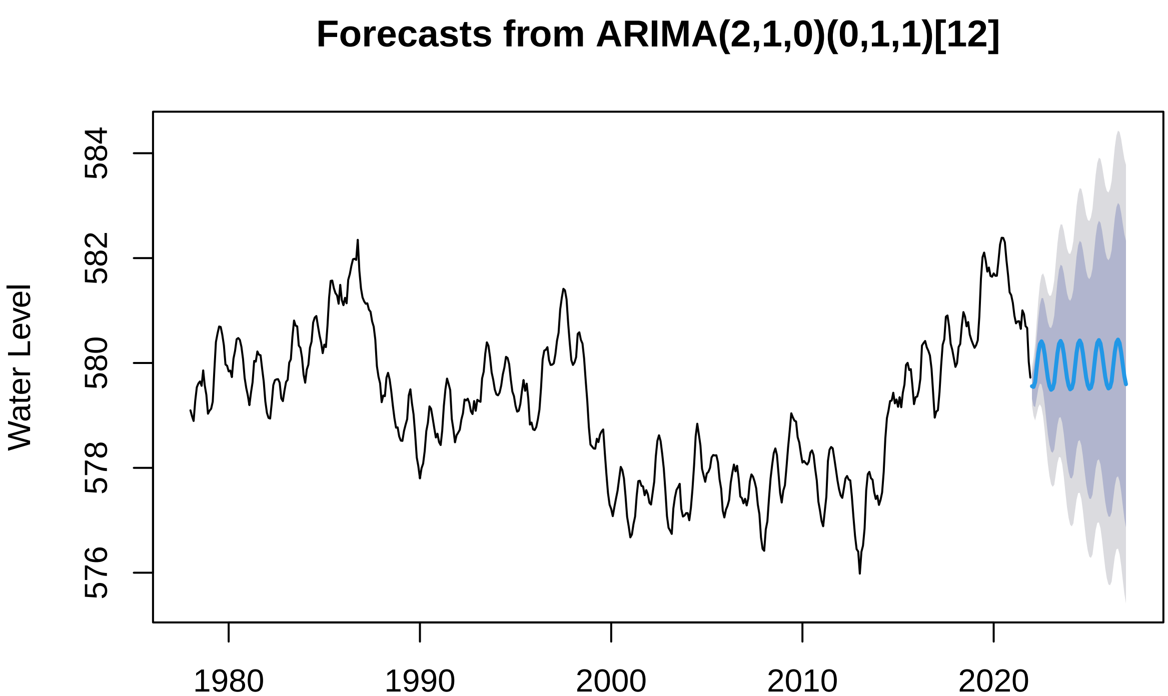
```
Acf(arima.seasonal.3$residuals, 100)
```



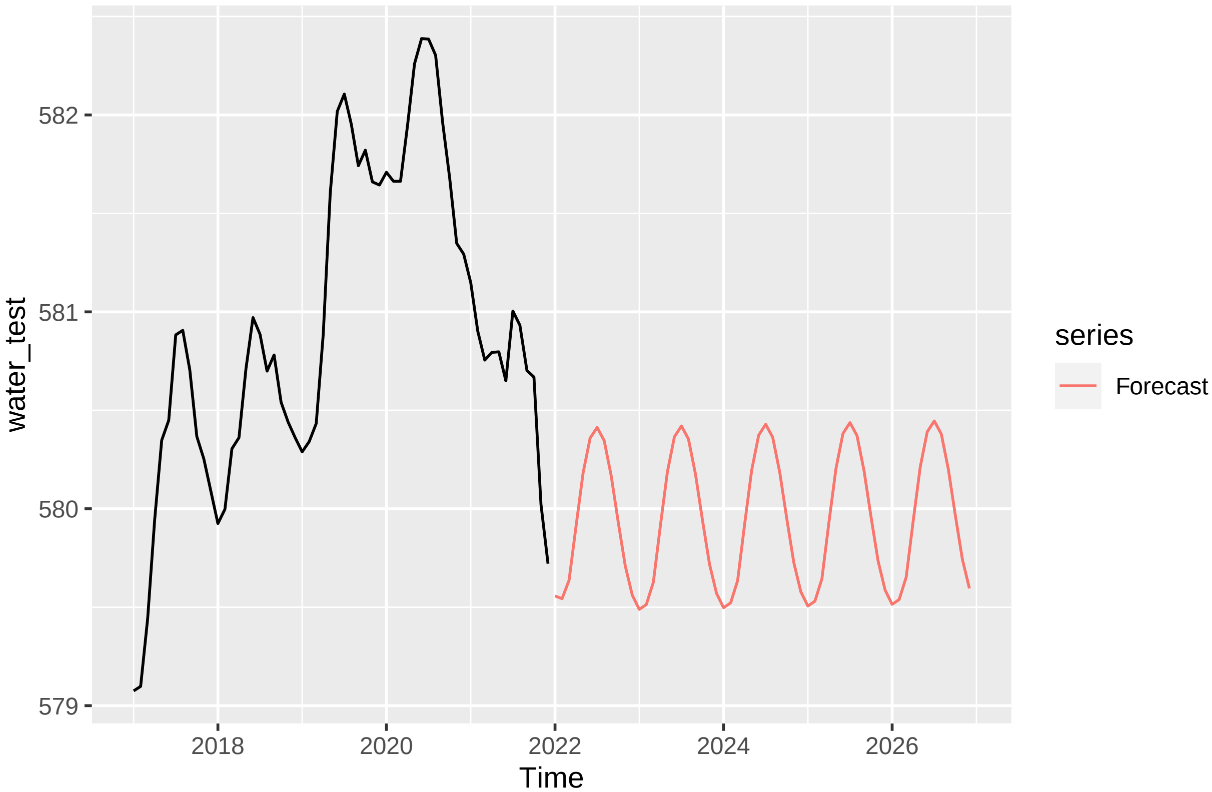
###

Best model is ARIMA(2,1,0)(0,1,1) ## sARIMA Forecast

```
arima.seasonal.forecast <- forecast(arima.seasonal, h=h)
plot(arima.seasonal.forecast, ylab="Water Level")
```




```
autoplot(water_test) + autolayer(arima.seasonal.forecast$mean, series="Forecast")
```

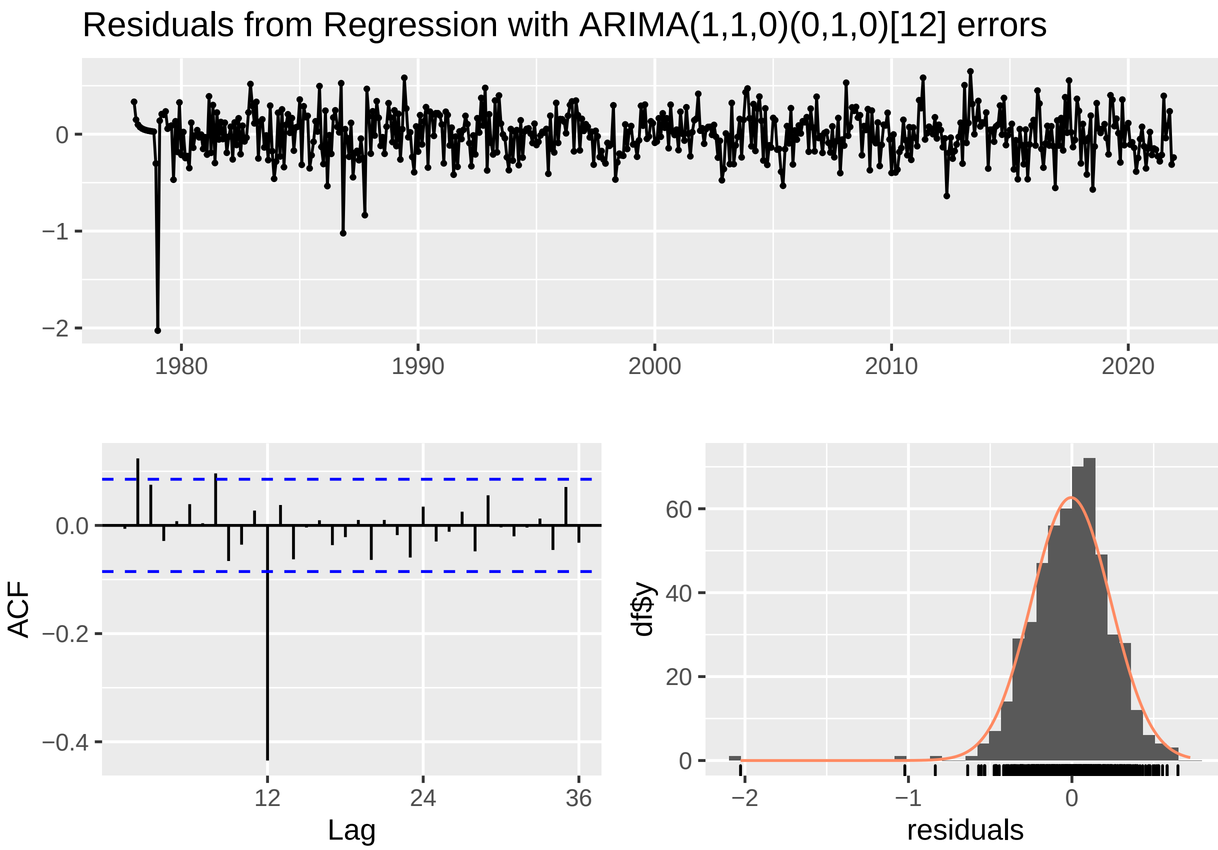


STEP 7: Model & Forecast: regression with ARIMA errors

```
xreg <- cbind(temperature.ts, precipitation.ts)
arima.1 <- Arima(water_level.ts, xreg = xreg, order = c(1, 1, 0), seasonal = c(0, 1, 0))
arima.1
```

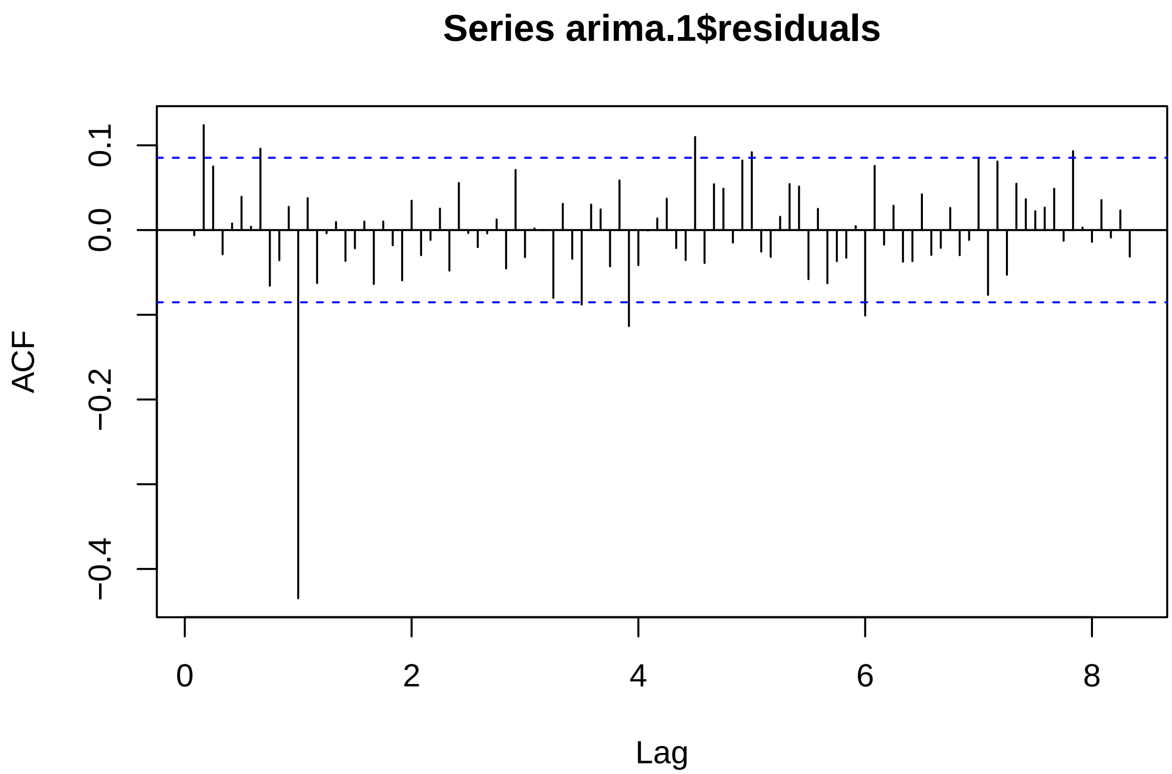
```
## Series: water_level.ts
## Regression with ARIMA(1,1,0)(0,1,0)[12] errors
##
## Coefficients:
##          ar1  temperature.ts  precipitation.ts
##       0.1260          0.0025           -0.0648
## s.e.  0.0438          0.0014            0.0722
##
## sigma^2 = 0.06085:  log likelihood = 30.39
## AIC=-52.78   AICc=-52.7   BIC=-35.8
```

```
checkresiduals(arima.1)
```

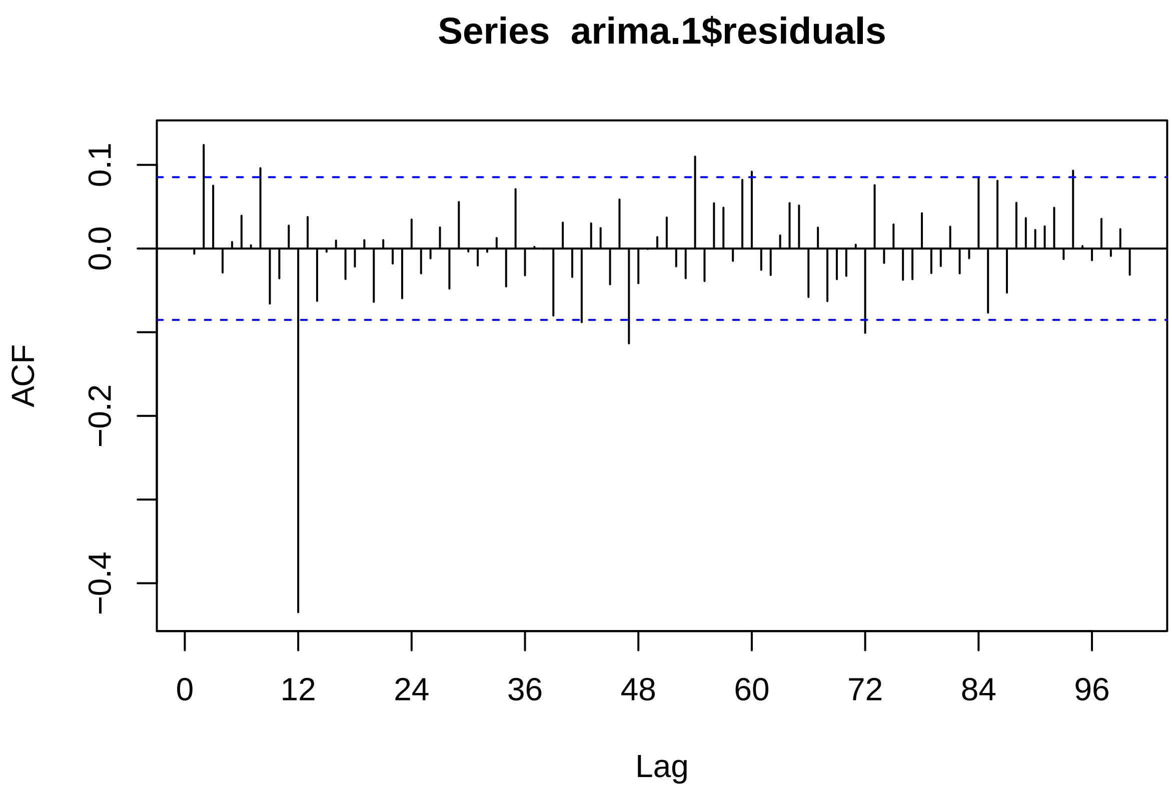



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,0)(0,1,0)[12] errors
## Q* = 132.44, df = 21, p-value < 2.2e-16
##
## Model df: 3.   Total lags used: 24
```

```
acf(arima.1$residuals, 100)
```



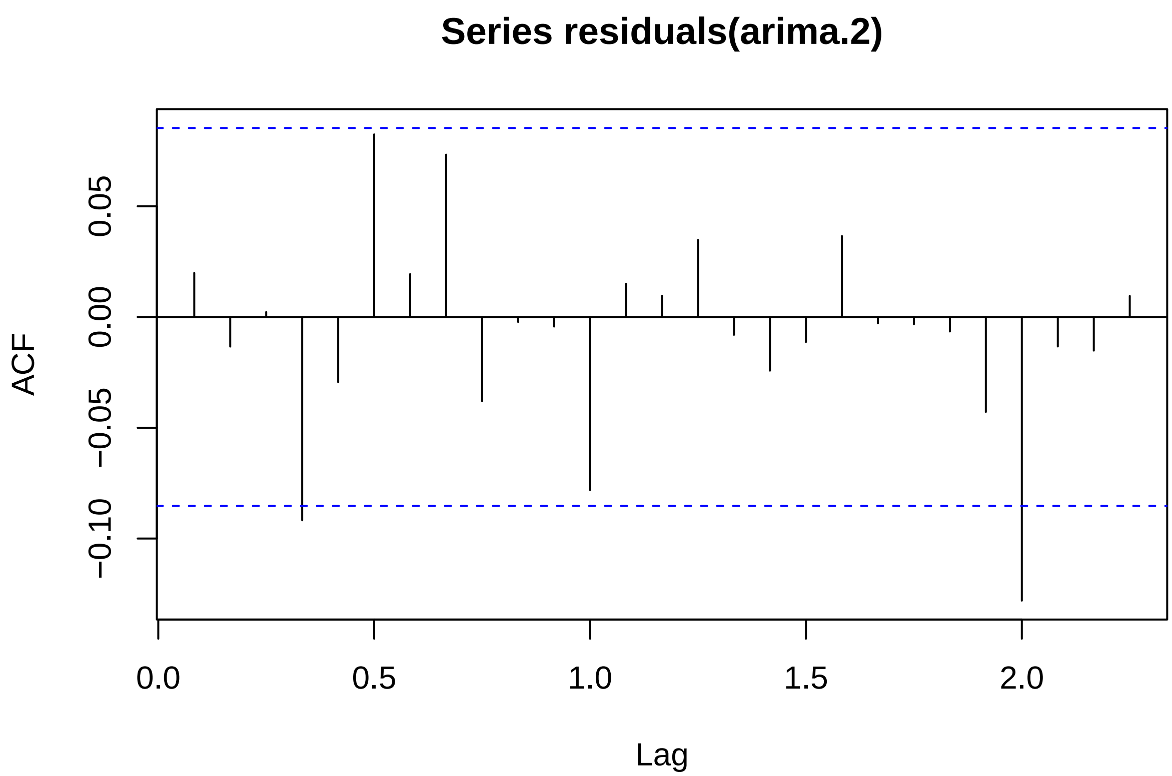

```
Acf(arima.1$residuals, 100)
```



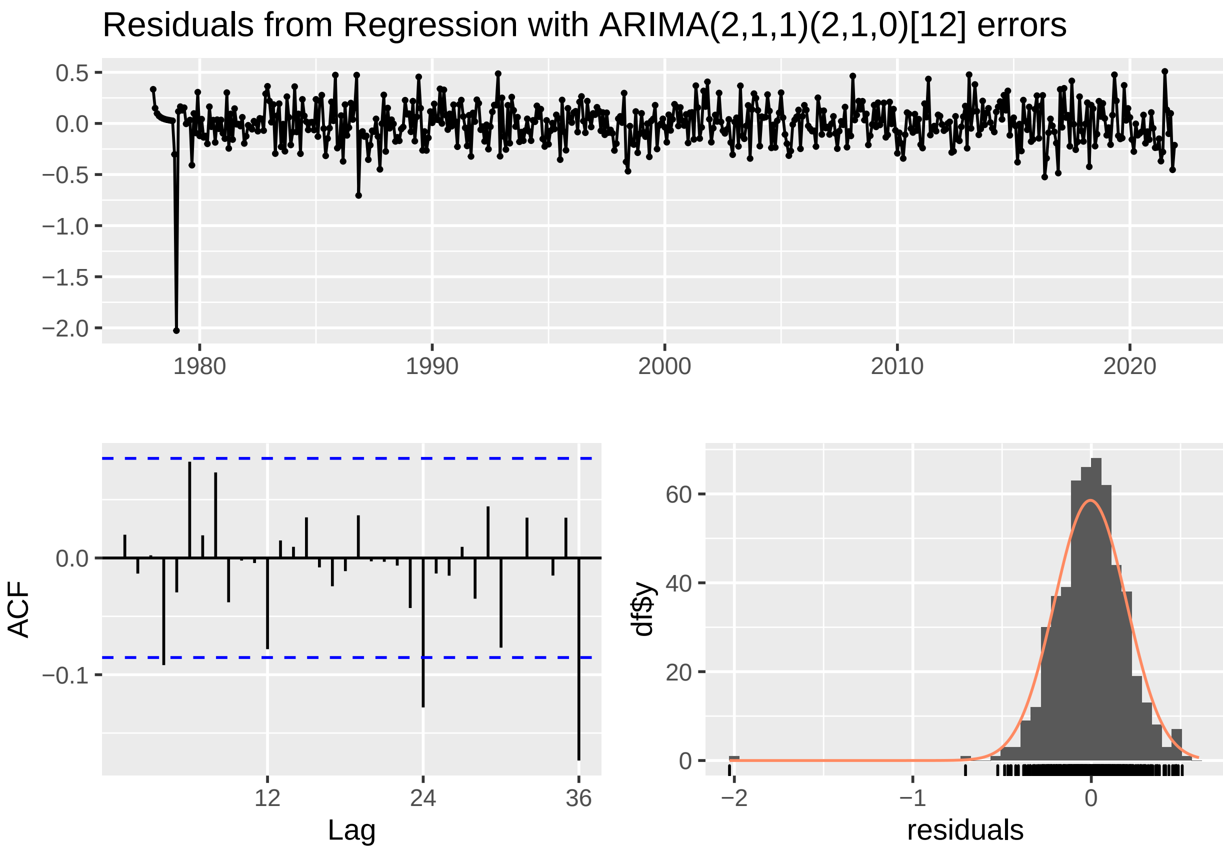

```
arima.2 <- auto.arima(water_level.ts, xreg = xreg, D = 1, d = 1)
arima.2
```

```
## Series: water_level.ts
## Regression with ARIMA(2,1,1)(2,1,0)[12] errors
##
## Coefficients:
##          ar1     ar2      ma1     sar1     sar2  temperature.ts
##       0.5234  0.1190  -0.3948  -0.6762  -0.3094          0.0010
## s.e.  0.2420  0.0758   0.2445   0.0432   0.0429          0.0013
##       precipitation.ts
##                -0.0575
## s.e.            0.0722
##
## sigma^2 = 0.04267:  log likelihood = 139.73
## AIC=-263.45   AICc=-263.17   BIC=-229.5
```

```
acf(residuals(arima.2))
```

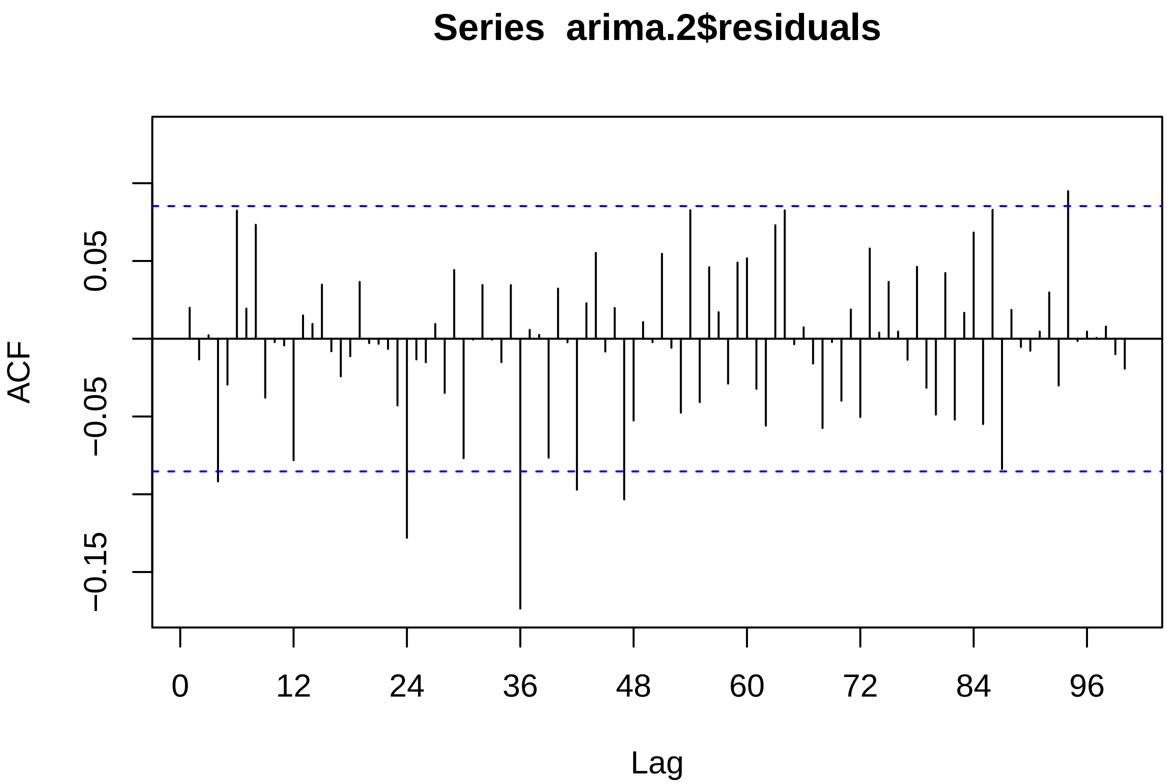



```
checkresiduals(arima.2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(2,1,1)(2,1,0)[12] errors
## Q* = 28.251, df = 17, p-value = 0.04209
##
## Model df: 7.   Total lags used: 24
```

```
Acf(arima.2$residuals, 100)
```

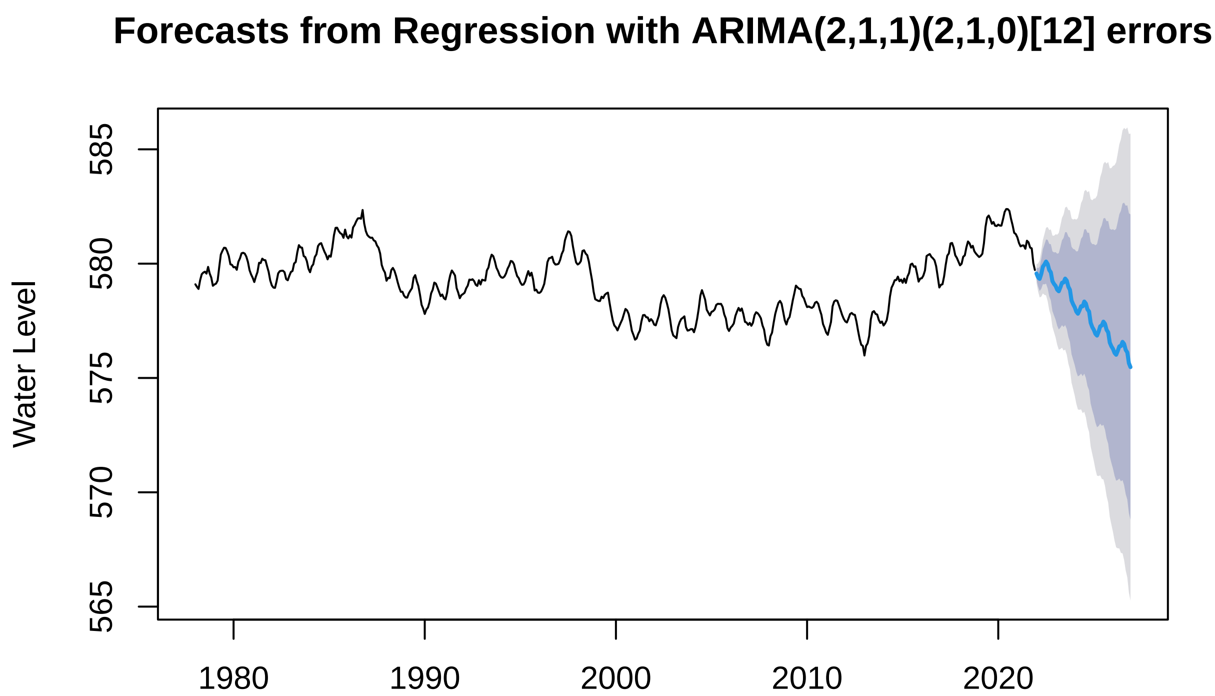


###

Regression with ARIMA(2,1,1)(2,1,0) is the best model.

Forecast

```
## Forecast regressions with seasonal naive method
temperature.fcst <- snaive(temperature.ts, 60)$mean
precipitation.fcst <- snaive(precipitation.ts, 60)$mean
x_reg.fcst <- cbind(temperature.fcst, precipitation.fcst)
arima.reg.forecast <- forecast(arima.2, xreg = x_reg.fcst, h=h)
plot(arima.reg.forecast, ylab="Water Level")
```



STEP 7: Model & Forecast: Spectral Analysis

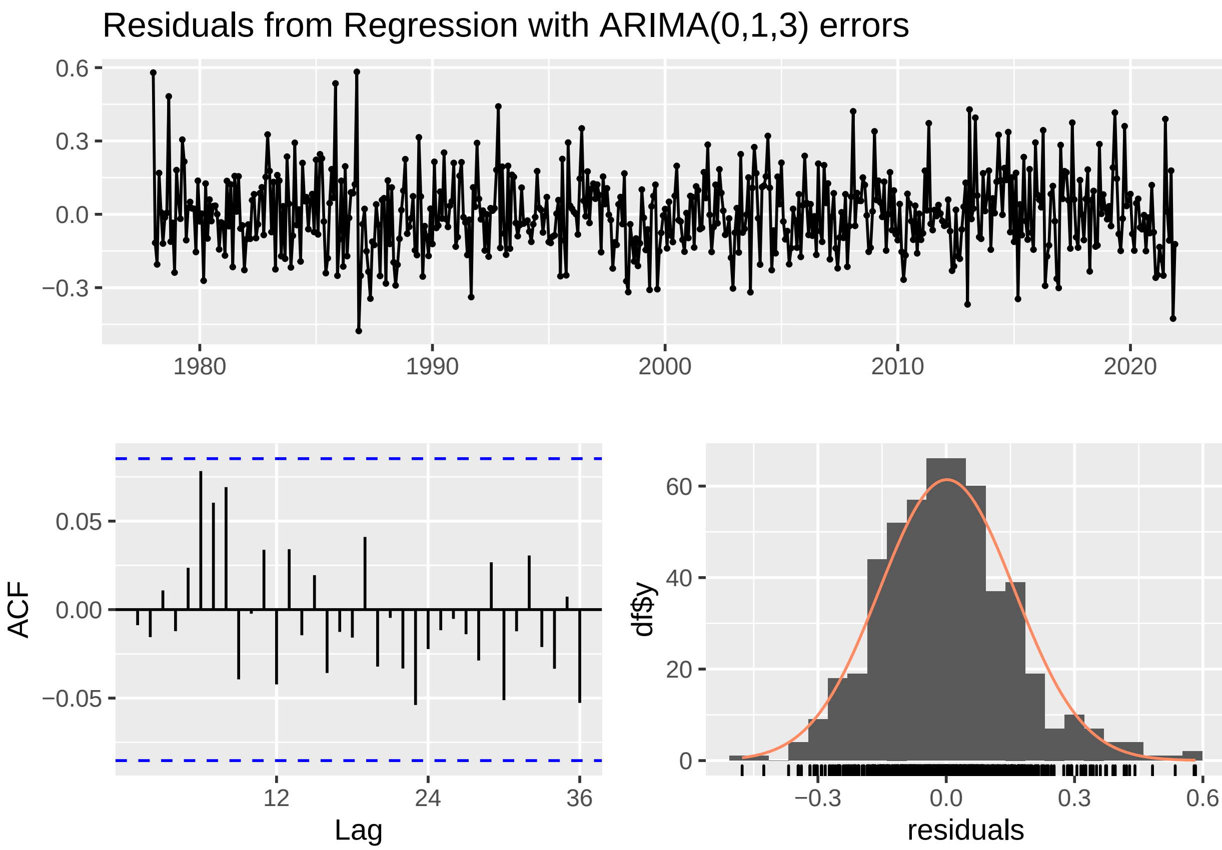
```
# Find best k
AIC = rep(NA, 6)
K = c()

for (k in 1:6){
  harmonics <- fourier(water_level.ts, K = k)
  fit <- auto.arima(water_level.ts, xreg = harmonics, seasonal = FALSE)
  aic <- AIC(fit)
  AIC[k] <- aic
}

AIC # Best k is 3
```

```
## [1] -432.3166 -439.0049 -442.2156 -439.4609 -436.3445 -433.9042
```

```
harmonics <- fourier(water_level.ts, K = 3)
spectral.1 <- auto.arima(water_level.ts, xreg = harmonics, seasonal = FALSE)
spectral.1
```

```
## Series: water_level.ts
## Regression with ARIMA(0,1,3) errors
##
## Coefficients:
##          ma1     ma2     ma3    S1-12    C1-12   S2-12   C2-12   S3-12   C3-12
##       0.1340  0.1931  0.1080  -0.2530  -0.4032  0.0037  0.0292  0.0151  0.0022
## s.e.  0.0436  0.0458  0.0426   0.0234   0.0234  0.0087  0.0087  0.0055  0.0055
##
## sigma^2 = 0.02542:  log likelihood = 231.11
## AIC=-442.22   AICc=-441.79   BIC=-399.54
```

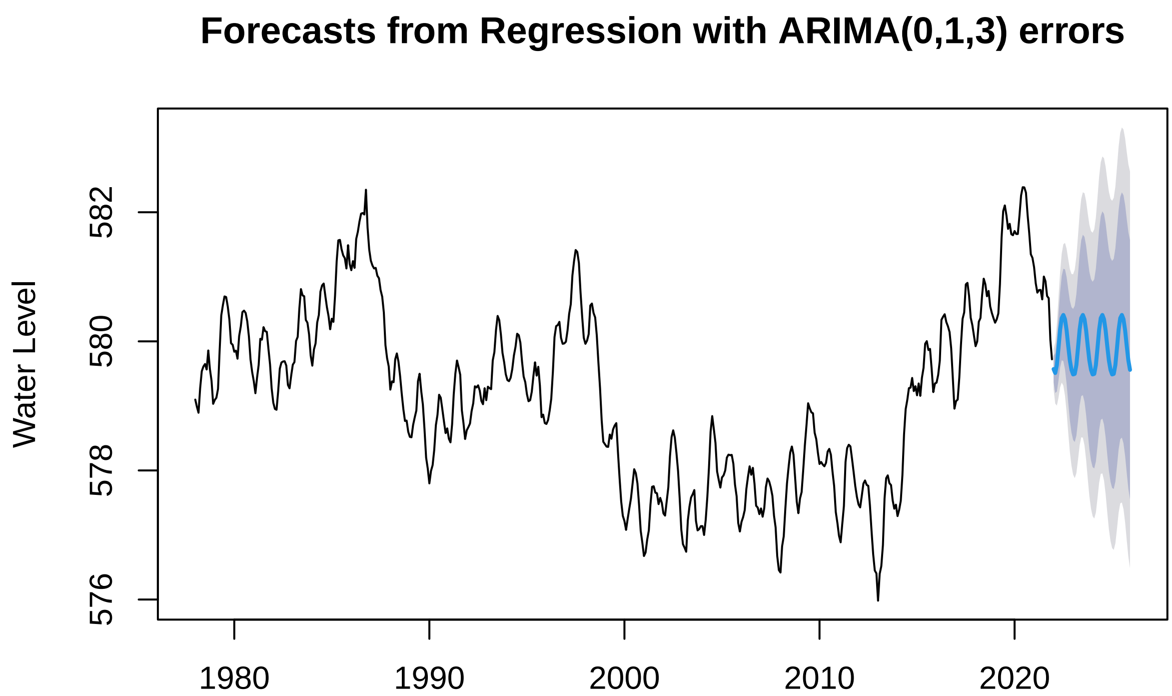
```
checkresiduals(spectral.1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,3) errors
## Q* = 16.744, df = 15, p-value = 0.3344
##
## Model df: 9.   Total lags used: 24
```

Forecast

```
spectral.forecast <- forecast(spectral.1, xreg = fourier(water_level.ts, 3, 48))
plot(spectral.forecast, ylab="Water Level")
```



STEP 7: Model & Forecast: VAR

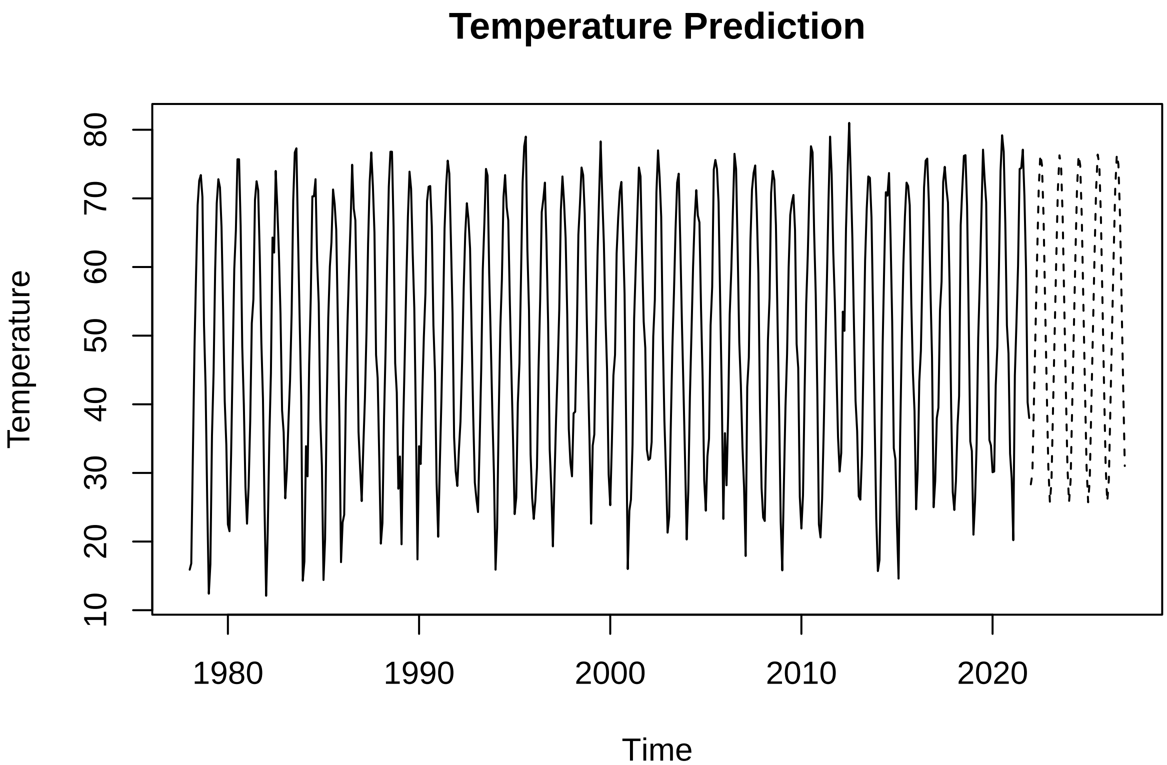
```
var.1<- VAR(y=data.frame(water_level.ts, temperature.ts, precipitation.ts), p=3, type='both', season=12)
summary(var.1)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: water_level.ts, temperature.ts, precipitation.ts
## Deterministic variables: both
## Sample size: 525
## Log Likelihood: -443.679
## Roots of the characteristic polynomial:
## 0.983 0.6805 0.3678 0.3678 0.3509 0.3509 0.1746 0.1746 0.1443
## Call:
## VAR(y = data.frame(water_level.ts, temperature.ts, precipitation.ts),
##     p = 3, type = "both", season = 12L)
##
##
## Estimation results for equation water_level.ts:
## ===============================================
## water_level.ts = water_level.ts.l1 + temperature.ts.l1 + precipitation.ts.l1 + water_level.ts.l2 + te
##
##                        Estimate Std. Error t value Pr(>|t|)
## water_level.ts.l1     1.039e+00  4.468e-02  23.248  < 2e-16 ***
## temperature.ts.l1    -4.745e-04  1.765e-03  -0.269   0.7882
## precipitation.ts.l1   6.141e-01  1.050e-01   5.848 8.97e-09 ***
## water_level.ts.l2     1.262e-01  6.562e-02   1.923   0.0550 .
## temperature.ts.l2    -7.230e-03  1.782e-03  -4.058 5.73e-05 ***
## precipitation.ts.l2 -5.587e-02  1.085e-01  -0.515   0.6068
## water_level.ts.l3    -1.740e-01  4.416e-02  -3.941 9.25e-05 ***
```

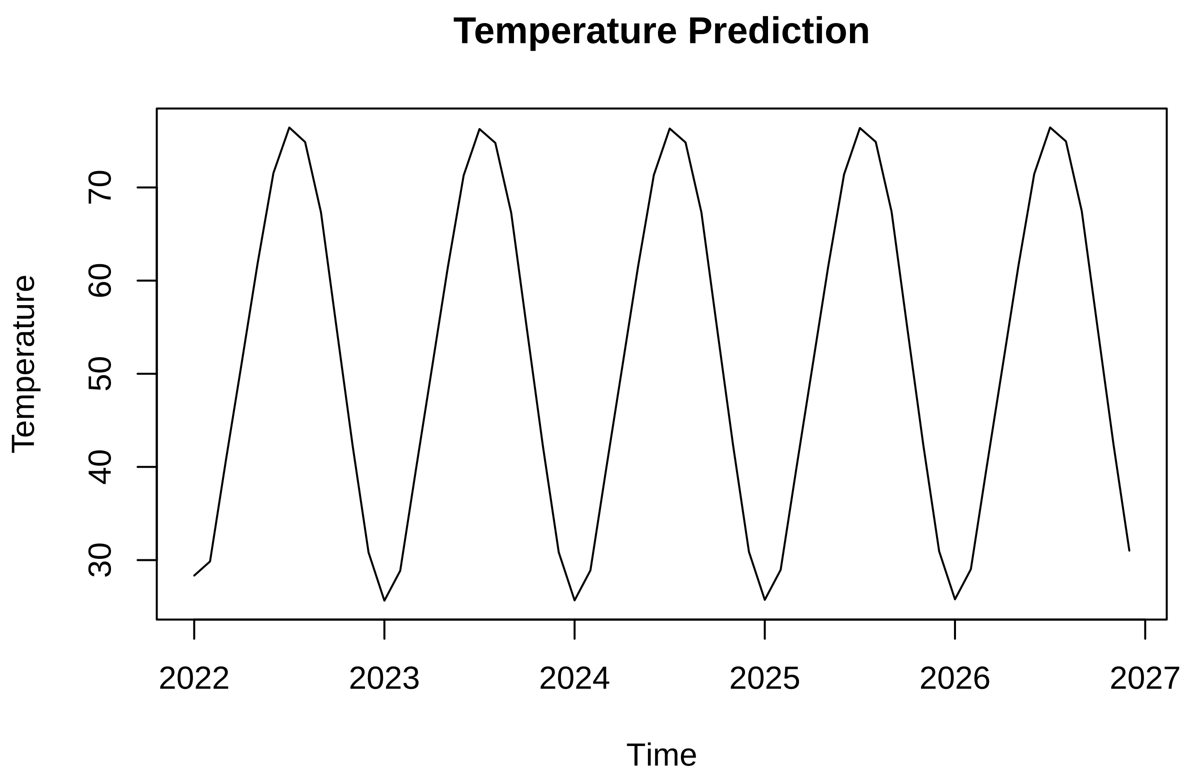
```
## temperature.ts.l3   -5.811e-03  1.784e-03  -3.257   0.0012 **
## precipitation.ts.l3  2.280e-02  1.067e-01   0.214   0.8309
## const                5.899e+00  2.892e+00   2.040   0.0419 *
## trend                6.697e-05  4.677e-05   1.432   0.1527
## sd1                 -8.950e-02  4.464e-02  -2.005   0.0455 *
## sd2                 -1.563e-01  6.525e-02  -2.395   0.0170 *
## sd3                 -1.909e-01  8.032e-02  -2.376   0.0179 *
## sd4                 -5.360e-02  8.486e-02  -0.632   0.5279
## sd5                 -6.804e-03  7.942e-02  -0.086   0.9318
## sd6                  1.235e-02  7.262e-02   0.170   0.8650
## sd7                  4.595e-02  7.292e-02   0.630   0.5289
## sd8                  8.498e-02  7.461e-02   1.139   0.2553
## sd9                  7.628e-02  7.154e-02   1.066   0.2868
## sd10                 9.038e-02  6.079e-02   1.487   0.1377
## sd11                 4.878e-02  4.322e-02   1.129   0.2596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.1489 on 503 degrees of freedom
## Multiple R-Squared: 0.9888,  Adjusted R-squared: 0.9884
## F-statistic:  2124 on 21 and 503 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation temperature.ts:
## ===============================================
## temperature.ts = water_level.ts.l1 + temperature.ts.l1 + precipitation.ts.l1 + water_level.ts.l2 + te
##
##                       Estimate Std. Error t value Pr(>|t|)
## water_level.ts.l1    -2.157129   1.126725  -1.915 0.056122 .
## temperature.ts.l1     0.181260   0.044513   4.072 5.41e-05 ***
## precipitation.ts.l1  -3.589981   2.648019  -1.356 0.175796
## water_level.ts.l2     0.444905   1.654720   0.269 0.788140
## temperature.ts.l2     0.065205   0.044930   1.451 0.147327
## precipitation.ts.l2   1.992853   2.736119   0.728 0.466738
## water_level.ts.l3     1.833793   1.113586   1.647 0.100235
## temperature.ts.l3     0.028888   0.044989   0.642 0.521083
## precipitation.ts.l3   5.169932   2.691697   1.921 0.055335 .
## const               -35.697747  72.927274  -0.489 0.624703
## trend                 0.004553   0.001179   3.860 0.000128 ***
## sd1                  -1.764208   1.125854  -1.567 0.117745
## sd2                   3.858997   1.645431   2.345 0.019400 *
## sd3                  15.427548   2.025574   7.616 1.30e-13 ***
## sd4                  24.642854   2.139930  11.516  < 2e-16 ***
## sd5                  33.267016   2.002768  16.611  < 2e-16 ***
## sd6                  40.310150   1.831212  22.013  < 2e-16 ***
## sd7                  41.988573   1.838952  22.833  < 2e-16 ***
## sd8                  38.098442   1.881505  20.249  < 2e-16 ***
## sd9                  29.885949   1.804086  16.566  < 2e-16 ***
## sd10                 17.944715   1.533052  11.705  < 2e-16 ***
## sd11                  7.973788   1.089910   7.316 1.02e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##
## Residual standard error: 3.754 on 503 degrees of freedom
## Multiple R-Squared: 0.9585,  Adjusted R-squared: 0.9568
## F-statistic: 553.7 on 21 and 503 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation precipitation.ts:
## =================================================
## precipitation.ts = water_level.ts.l1 + temperature.ts.l1 + precipitation.ts.l1 + water_level.ts.l2 +
##
##                       Estimate Std. Error t value Pr(>|t|)
## water_level.ts.l1    9.443e-03  1.955e-02   0.483  0.62932
## temperature.ts.l1   -5.319e-04  7.724e-04  -0.689  0.49140
## precipitation.ts.l1 -8.808e-03  4.595e-02  -0.192  0.84807
## water_level.ts.l2   -2.877e-03  2.871e-02  -0.100  0.92024
## temperature.ts.l2   -6.792e-05  7.797e-04  -0.087  0.93061
## precipitation.ts.l2 -2.907e-02  4.748e-02  -0.612  0.54069
## water_level.ts.l3   -6.406e-03  1.932e-02  -0.331  0.74042
## temperature.ts.l3   -2.713e-05  7.807e-04  -0.035  0.97229
## precipitation.ts.l3 -2.169e-02  4.671e-02  -0.464  0.64259
## const                4.153e-02  1.266e+00   0.033  0.97383
## trend                2.362e-05  2.046e-05   1.154  0.24892
## sd1                 -2.543e-02  1.954e-02  -1.302  0.19359
## sd2                 -2.412e-02  2.855e-02  -0.845  0.39870
## sd3                 -9.964e-03  3.515e-02  -0.283  0.77694
## sd4                  3.201e-02  3.713e-02   0.862  0.38912
## sd5                  5.161e-02  3.475e-02   1.485  0.13818
## sd6                  5.415e-02  3.178e-02   1.704  0.08900 .
## sd7                  6.005e-02  3.191e-02   1.882  0.06042 .
## sd8                  9.214e-02  3.265e-02   2.822  0.00496 **
## sd9                  4.597e-02  3.131e-02   1.468  0.14262
## sd10                 4.686e-02  2.660e-02   1.762  0.07875 .
## sd11                 2.668e-02  1.891e-02   1.410  0.15903
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.06514 on 503 degrees of freedom
## Multiple R-Squared: 0.1643,  Adjusted R-squared: 0.1294
## F-statistic: 4.709 on 21 and 503 DF,  p-value: 7.396e-11
##
##
##
## Covariance matrix of residuals:
##                  water_level.ts temperature.ts precipitation.ts
## water_level.ts         0.022159       0.040694         0.002404
## temperature.ts         0.040694      14.092090         0.004995
## precipitation.ts       0.002404       0.004995         0.004243
##
## Correlation matrix of residuals:
##                  water_level.ts temperature.ts precipitation.ts
## water_level.ts          1.00000        0.07282          0.24787
## temperature.ts          0.07282        1.00000          0.02043
## precipitation.ts        0.24787        0.02043          1.00000
```

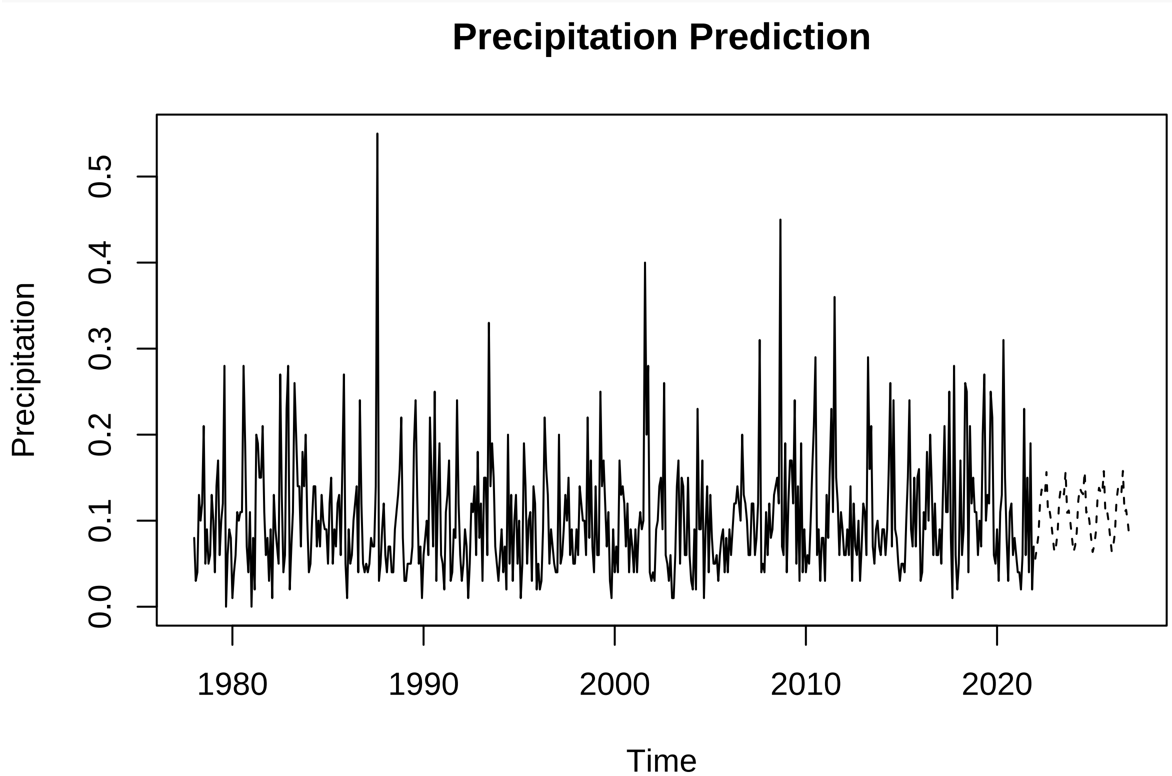
```
preds<- predict(var.1, n.ahead=60)

# Temperature
temp_pred<- ts(preds$fcst$temp[,1], start=c(2022,1), fr=12)
ts.plot(cbind(temperature.ts, temp_pred), lty=1:2, main="Temperature Prediction", ylab="Temperature")
```



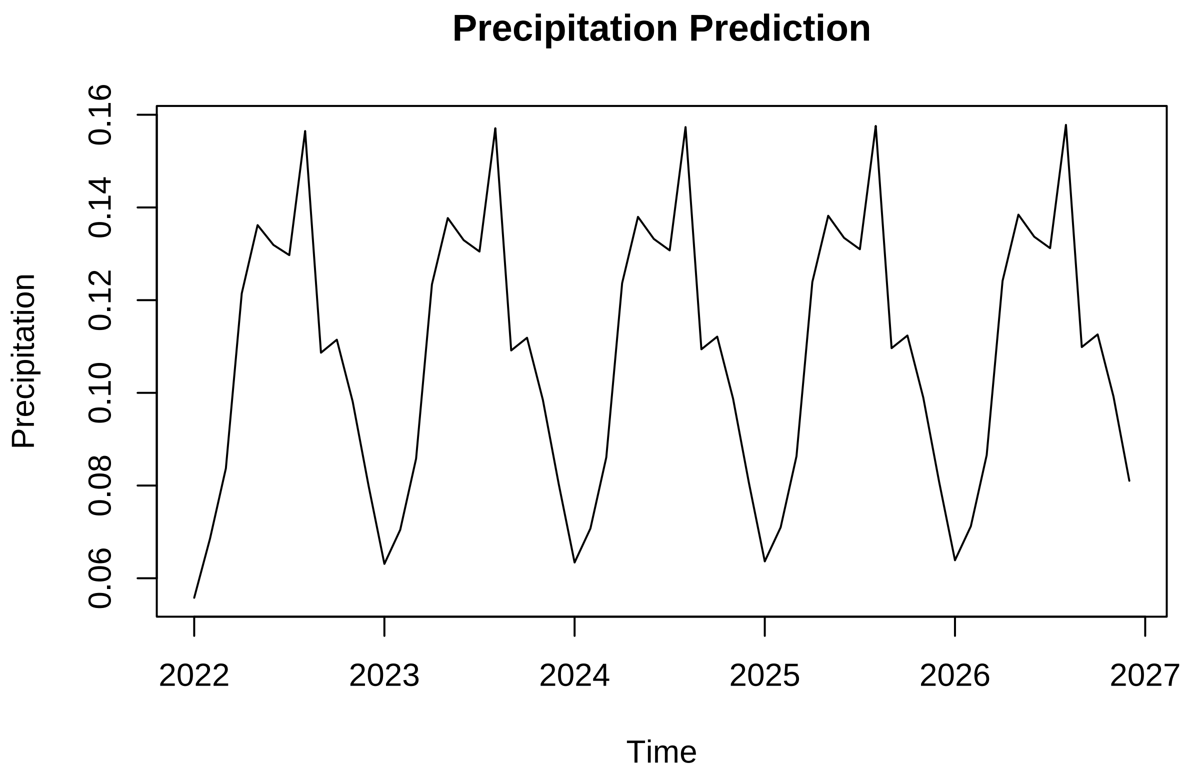

```
ts.plot(temp_pred, main="Temperature Prediction", ylab="Temperature")
```



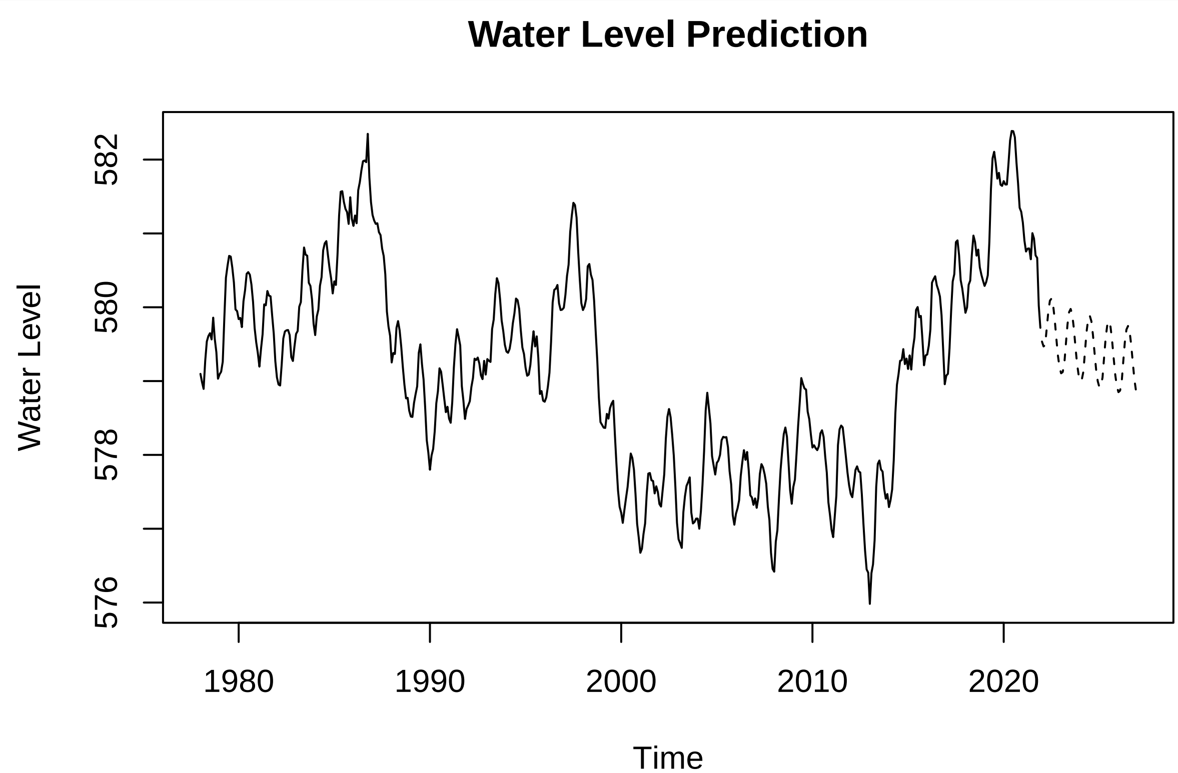

```
# Precipitation
precip_pred<- ts(preds$fcst$precip[,1], start=c(2022,1), fr=12)
ts.plot(cbind(precipitation.ts, precip_pred), lty=1:2, main="Precipitation Prediction", ylab="Precipita
```



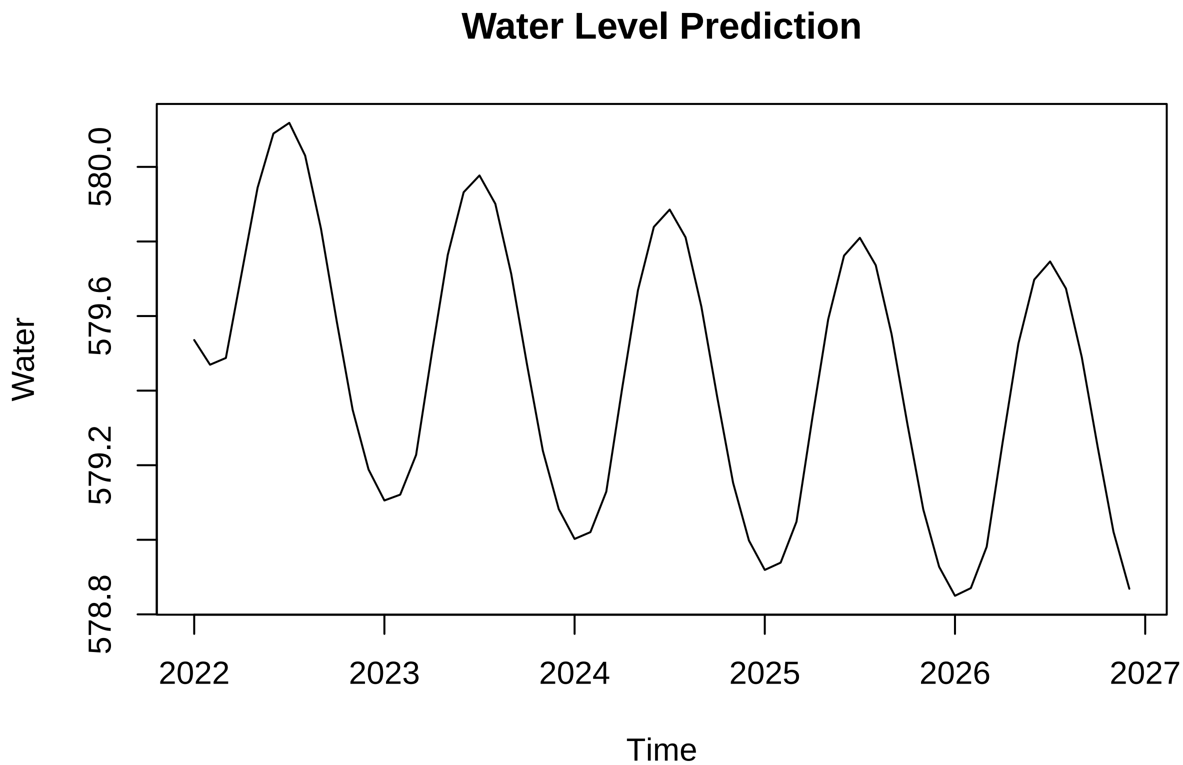
```r
ts.plot(precip_pred, main="Precipitation Prediction", ylab="Precipitation")
```



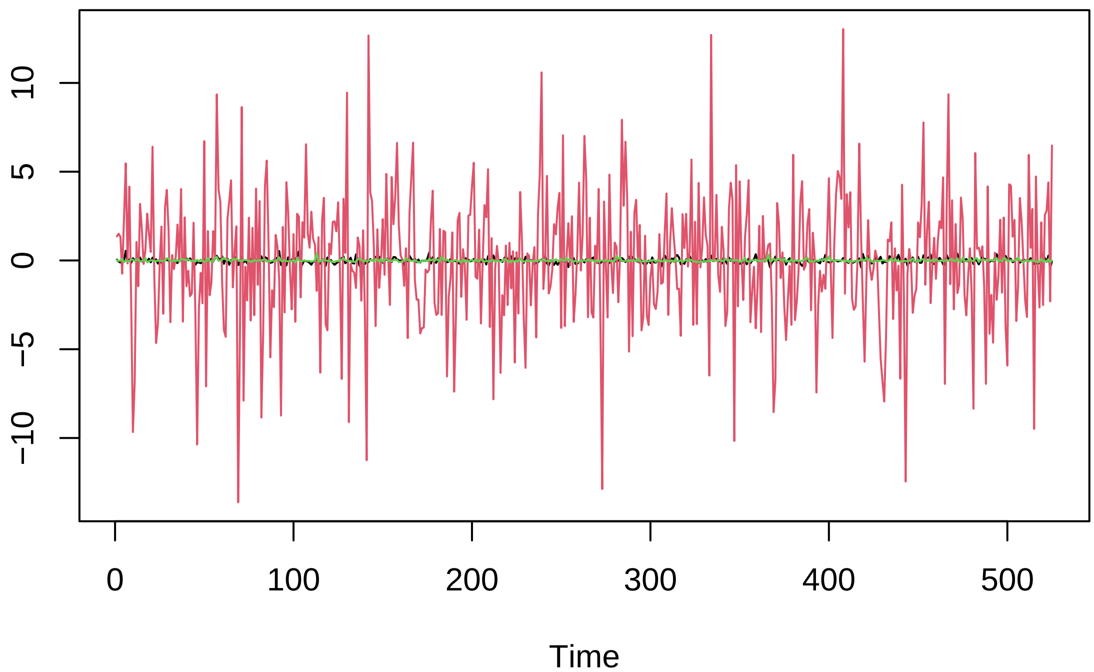

```r
# Water Level
water_pred<- ts(preds$fcst$water_level.ts[,1], start=c(2022,1), fr=12)
ts.plot(cbind(water_level.ts, water_pred), lty=1:2, main="Water Level Prediction", ylab="Water Level")
```



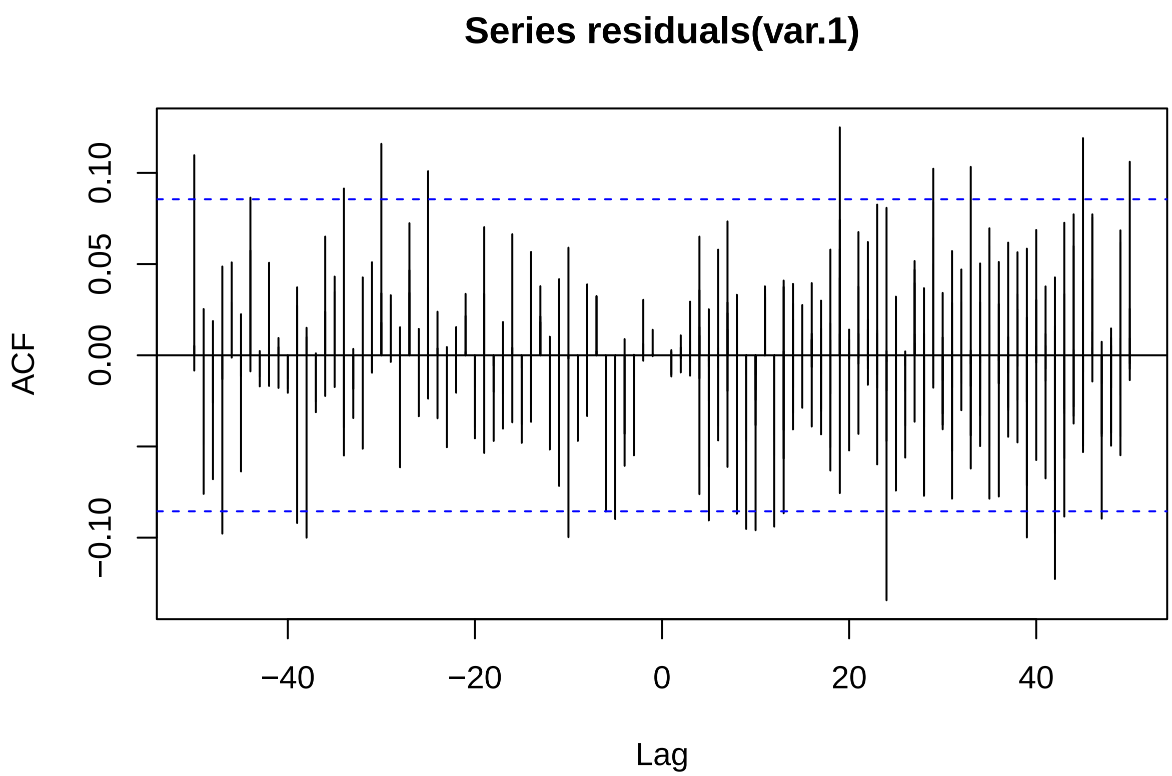
```r
ts.plot(water_pred, main="Water Level Prediction", ylab="Water")
```



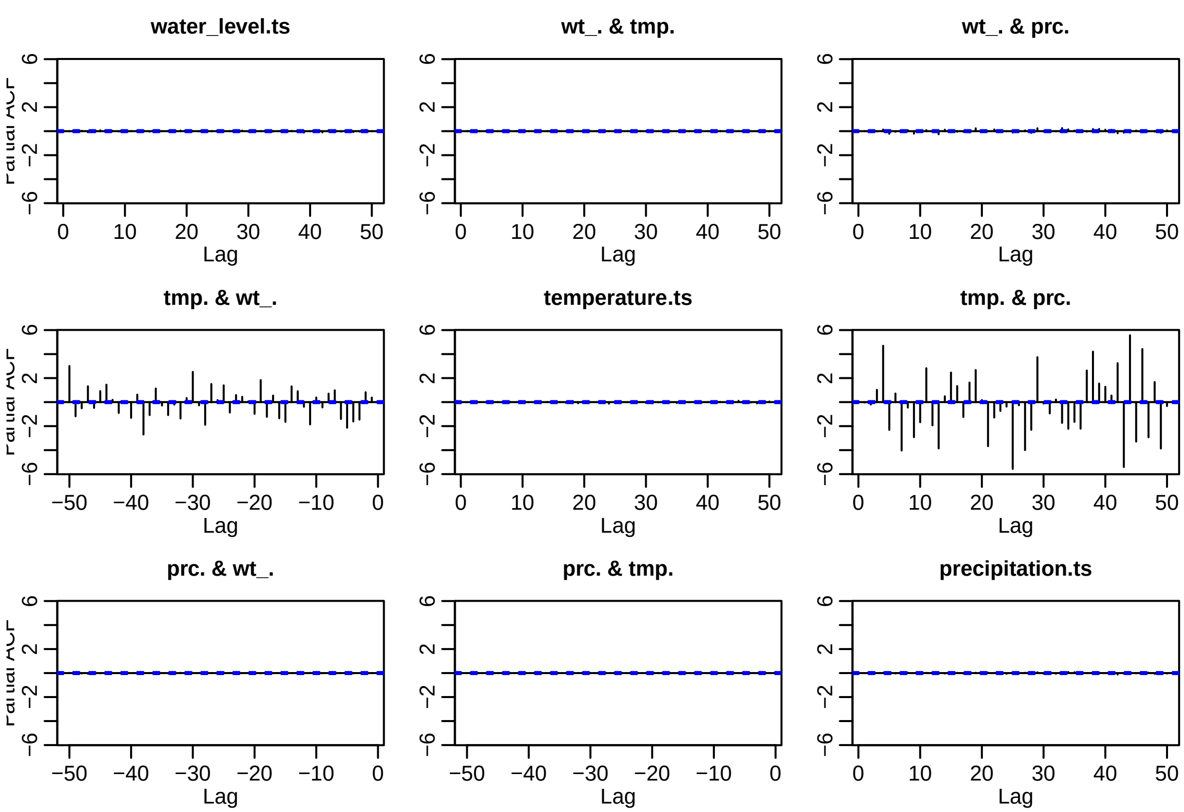

```r
ts.plot(residuals(var.1), col=1:3)
```




```r
acf(residuals(var.1), 50)
```




```
pacf(residuals(var.1), 50)
```



STEP 8: Model Testing

Cross-Valitation with rolling window

```
k <- 408 # minimum data length for fitting a model
n <- 528 # Number of data points

p <- 12 ### Period
H <- 60 # Forecast Horizon
```

```
st <- tsp(water_level.ts)[1]+(k-2)/p #  gives the start time in time units,


error.seasonal.arima.1 <- matrix(NA,n-k,H)
error.seasonal.arima.2 <- matrix(NA,n-k,H)
error.arima.1 <- matrix(NA,n-k,H)
error.arima.2 <- matrix(NA,n-k,H)
error.spectral <- matrix(NA,n-k,H)


mape.seasonal.arima.1 <- matrix(NA,n-k,H)
mape.seasonal.arima.2 <- matrix(NA,n-k,H)
mape.arima.1 <- matrix(NA,n-k,H)
mape.arima.2 <- matrix(NA,n-k,H)
mape.spectral <- matrix(NA,n-k,H)

rmse_1.seasonal.arima.1 <- matrix(NA,n-k,H)
rmse_1.seasonal.arima.2 <- matrix(NA,n-k,H)
rmse_1.arima.1 <- matrix(NA,n-k,H)
rmse_1.arima.2 <- matrix(NA,n-k,H)
rmse_1.spectral <- matrix(NA,n-k,H)
rmse_1.var <- matrix(NA,n-k,H)
rmse_1.base <- matrix(NA,n-k,H)

mape_1.seasonal.arima.1 <- matrix(NA,n-k,H)
mape_1.seasonal.arima.2 <- matrix(NA,n-k,H)
mape_1.arima.1 <- matrix(NA,n-k,H)
mape_1.arima.2 <- matrix(NA,n-k,H)
mape_1.spectral <- matrix(NA,n-k,H)
mape_1.var <- matrix(NA,n-k,H)
```

```
defaultW <- getOption("warn")
options(warn = -1)


for(i in 1:60)
{

  ### One Month rolling forecasting
  # Expanding Window
  train_1 <- window(water_level.ts, end=st + i/p)  ## Window Length: k+i
  train_1.temperature <- window(temperature.ts, end=st + i/p)  ## Window Length: k+i
  train_1.precipitation <- window(precipitation.ts, end=st + i/p)  ## Window Length: k+i
  train_1.xreg <- cbind(train_1.temperature, train_1.precipitation)
  train_1.temperature.fcst <- snaive(train_1.temperature, 60)$mean
  train_1.precipitation.fcst <- naive(train_1.precipitation, 60)$mean
```

```
  train_1.xreg.fcst <- cbind(train_1.temperature.fcst, train_1.precipitation.fcst)
  train_1.harmonics <- fourier(train_1, K = 3)

  # Sliding Window - keep the training window of fixed length.
  # The training set always consists of k observations.
  train_2 <- window(water_level.ts, start=st+(i-k+1)/p, end=st+i/p) ## Window Length: k

  # Test dataset
  test <- window(water_level.ts, start=st + (i+1)/p, end=st + (i+H)/p) ## Window Length: H

  fit_1.seasonal.arima.1 <- Arima(train_1, order=c(2,1,0), seasonal=list(order=c(0,1,1), period=p),
              include.drift=TRUE,  method="ML")
  fcast_1.seasonal.arima.1 <- forecast(fit_1.seasonal.arima.1, h=H)


  fit_1.arima.2 <- Arima(train_1, xreg = train_1.xreg, order=c(2,1,1), seasonal=list(order=c(2,1,0), pe
              include.drift=TRUE,  method="ML")
  fcast_1.arima.2 <- forecast(fit_1.arima.2, h=H, xreg = train_1.xreg.fcst)

  fit_1.spectral <- auto.arima(train_1, xreg = train_1.harmonics, seasonal = FALSE)

  fcast_1.spectral <- forecast(fit_1.spectral, xreg = fourier(train_1, 3, H))

  fit_1.var <- VAR(y=data.frame(train_1, train_1.temperature, train_1.precipitation), p=3, type='both',

  fcast_1.var <- predict(fit_1.var, n.ahead = 60)
  water_pred.var<- fcast_1.var$fcst$train_1[,1]

  waterSnaive <- snaive(train_1, h=60)

  rmse_1.seasonal.arima.1[i,1:length(test)] <- (fcast_1.seasonal.arima.1[['mean']]-test)^2
  rmse_1.arima.2[i,1:length(test)] <- (fcast_1.arima.2[['mean']]-test)^2
  rmse_1.spectral[i,1:length(test)] <- (fcast_1.spectral[['mean']]-test)^2
  rmse_1.var[i,1:length(test)] <- (water_pred.var-test)^2
  rmse_1.base[i, 1:length(test)] <- accuracy(waterSnaive$mean, test)[,2] # RMSE
}
```

```
dev.new(width=6, height=6,pointsize=12)
plot(1:60, colMeans(rmse_1.seasonal.arima.1,na.rm=TRUE), type="l",col=1,xlab="Iterations", ylab="RMSE",
lines(1:60, colMeans(rmse_1.arima.2,na.rm=TRUE), type="l",col=2)
lines(1:60, colMeans(rmse_1.spectral,na.rm=TRUE), type="l",col=3)
lines(1:60, colMeans(rmse_1.var,na.rm=TRUE), type="l",col=4)
#lines(1:60, colMeans(rmse_1.base,na.rm=TRUE), type="l",col=5)
legend("topleft",legend=c("Seasonal ARIMA", 'Regression with ARIMA Errors', "Spectral Analysis", "VAR")
```

Cross-Valitation with expanding window

```
rmse_2.seasonal.arima.1 <- matrix(NA,n-k,H)
rmse_2.seasonal.arima.2 <- matrix(NA,n-k,H)
rmse_2.arima.1 <- matrix(NA,n-k,H)
rmse_2.arima.2 <- matrix(NA,n-k,H)
rmse_2.spectral <- matrix(NA,n-k,H)
rmse_2.var <- matrix(NA,n-k,H)
```

```r
rmse_2.base <- matrix(NA,n-k,H)

mape_2.seasonal.arima.1 <- matrix(NA,n-k,H)
mape_2.seasonal.arima.2 <- matrix(NA,n-k,H)
mape_2.arima.1 <- matrix(NA,n-k,H)
mape_2.arima.2 <- matrix(NA,n-k,H)
mape_2.spectral <- matrix(NA,n-k,H)
mape_2.var <- matrix(NA,n-k,H)
```

```r
defaultW <- getOption("warn")
options(warn = -1)


for(i in 1:60)
{
  train_2 <- window(water_level.ts, start=st+(i-k+1)/p, end=st+i/p)  ## Window Length: k+i
  train_2.temperature <- window(temperature.ts, start=st+(i-k+1)/p, end=st+i/p)  ## Window Length: k+i
  train_2.precipitation <- window(precipitation.ts, start=st+(i-k+1)/p, end=st+i/p)  ## Window Length: 
  train_2.xreg <- cbind(train_2.temperature, train_2.precipitation)
  train_2.temperature.fcst <- snaive(train_2.temperature, 60)$mean
  train_2.precipitation.fcst <- naive(train_2.precipitation, 60)$mean
  train_2.xreg.fcst <- cbind(train_2.temperature.fcst, train_2.precipitation.fcst)
  train_2.harmonics <- fourier(train_2, K = 3)

  # Sliding Window - keep the training window of fixed length.
  # The training set always consists of k observations.
  train_2 <- window(water_level.ts, start=st+(i-k+1)/p, end=st+i/p) ## Window Length: k

  # Test dataset
  test <- window(water_level.ts, start=st + (i+1)/p, end=st + (i+H)/p) ## Window Length: H


  fit_2.seasonal.arima.1 <- Arima(train_2, order=c(2,1,0), seasonal=list(order=c(1,1,0), period=p),
              include.drift=TRUE,  method="ML")
  fcast_2.seasonal.arima.1 <- forecast(fit_2.seasonal.arima.1, h=H)


  fit_2.arima.2 <- Arima(train_2, xreg = train_2.xreg, order=c(2,1,0), seasonal=list(order=c(2,0,0), pe
              include.drift=TRUE,  method="ML")
  fcast_2.arima.2 <- forecast(fit_2.arima.2, h=H, xreg = train_2.xreg.fcst)

  fit_2.spectral <- auto.arima(train_2, xreg = train_2.harmonics, seasonal = FALSE)

  fcast_2.spectral <- forecast(fit_2.spectral, xreg = fourier(train_2, 3, H))

  fit_2.var <- VAR(y=data.frame(train_2, train_2.temperature, train_2.precipitation), p=3, type='both', 

  fcast_2.var <- predict(fit_2.var, n.ahead = 60)
  water_pred.var<- fcast_2.var$fcst$train_2[,1]

  waterSnaive <- snaive(train_2, h=60)

  rmse_2.seasonal.arima.1[i,1:length(test)] <- (fcast_2.seasonal.arima.1[['mean']]-test)^2
  rmse_2.arima.2[i,1:length(test)] <- (fcast_2.arima.2[['mean']]-test)^2
```

```
  rmse_2.spectral[i,1:length(test)] <- (fcast_2.spectral[['mean']]-test)^2
  rmse_2.var[i,1:length(test)] <- (water_pred.var-test)^2
  rmse_2.base[i, 1:length(test)] <- accuracy(waterSnaive$mean, test)[,2] # RMSE
}
```

```
dev.new(width=6, height=6,pointsize=12)
plot(1:60, colMeans(rmse_2.seasonal.arima.1,na.rm=TRUE), type="l",col=1,xlab="Iterations", ylab="RMSE",
lines(1:60, colMeans(rmse_2.arima.2,na.rm=TRUE), type="l",col=2)
lines(1:60, colMeans(rmse_2.spectral,na.rm=TRUE), type="l",col=3)
lines(1:60, colMeans(rmse_2.var,na.rm=TRUE), type="l",col=4)
#lines(1:60, colMeans(rmse_2.base,na.rm=TRUE), type="l",col=5)
legend("topleft",legend=c("Seasonal ARIMA", 'Regression with ARIMA Errors (auto.arima)', "Spectral Analy
```