Nov 13 Notes

Hongdou Li

11/13/2018

Non-Stationary Models

Idea: We can transform a non-stationary time series so that the new time series is stationary and hence can be modeled with an ARMA model.

When the "Transfermation" involves **Differencing** what we're really doing is fitting an ARIMA model to the original time series.

ARIMA (Autoregressive Integrated Moving Averages)

An ARIMA model models a non-stationary time series which after finitely many differences becomes stationary.

If d is a non-negative integer then $\{Y_t\} \sim ARIMA(p, d, q)$ if

$$X_t = (1 - B)^d Y_t \sim ARIMA(p, q)$$

This definition implies that an ARIMA(p,d,q) model can be represented as follows:

$$\phi(B)X_t = \theta(B)\epsilon_t$$

$$\phi(B)(1 - B)^d Y_t = \theta(B)\epsilon_t$$

what is differencing actually doing?

Short answer: Eliminating trend

How? Notation :

$\nabla = (1 - B)$

$\nabla Y_t = (1 - B)Y_t = Y_t - Y_{t-1}$

$\nabla^2 Y_t = (1 - B)^2 Y_t = (1 - 2B + B^2)Y_t = Y_t - 2Y_{t-1} + Y_{t-2}$

Remark If $\{Y_t\}$ exhibits polynomial trend of the form $m_t = \sum_{i=0}^{d} \alpha_i t^i$

then $\nabla^d Y_t$ will not have a trend component.

- Example1:

$Y_t = c + bt + \epsilon_t$

$$\begin{aligned}
\nabla Y_t &= (1 - B)Y_t \\
&= (1 - B)(c + bt + \epsilon_t) \\
&= (c - Bc) + (bt - bBt) + (\epsilon_t - B\epsilon_t) \\
&= (c - c) + (bt - b(t - 1)) + (\epsilon_t - \epsilon_{t-1}) \\
&= b + (\epsilon_t - \epsilon_{t-1}) \\
&= b + \nabla\epsilon_t
\end{aligned}$$

- Example2:

$Y_t = c + bt + at^2 + \epsilon_t$

$$\begin{aligned}\nabla^2 Y_t &= (1-B)^2 Y_t \\ &= (1 - 2B + B^2)Y_t \\ &= (1 - 2B + B^2)(c + bt + at^2 + \epsilon_t) \\ &= (c - 2c + c) + b(t - 2(t-1) + (t-2)) + a(t^2 - 2(t-1)^2 + (t-2)^2) + D^2\epsilon_t \\ &= 0 + b(t - 2t + 2 + t - 2) + a(t^2 - 2(t^2 - 2t + 1) + (t^2 - 4t + 4)) + \nabla^2\epsilon_t \\ &= 2a + \nabla^2\epsilon_t\end{aligned}$$

- Example 1 Revisted:

$\nabla Y_t = b + \nabla\epsilon_t$

$$\begin{aligned}\nabla^2 Y_t &= \nabla b + \nabla^2\epsilon_t \\ &= \nabla^2\epsilon_t \\ &= \epsilon_t - 2\epsilon_{t-1} + \epsilon_{t-2}\end{aligned}$$

So how do I choose d? In other words, how many times must I difference my data before the result is stationary?

We can determine this informally with plots or formally with hypothesis tests:

- Graphically:
  - Plots of the time series => a lack of trend and constant variance informally indicates stationary.
  - ACF plots of time series => stationarity is indicated by rapid decay as opposed to slow decay.
- Formally: Use "Unit root tests" such as **Augmented Dickey-Fuller**

Augmented Dickey-Fuller(ADF) Test

H_0: Time series is not stationary

(roots of generating function are on/inside the unit circle)

vs.

H_1: Time series is stationary

(roots of generating function are outside the unit circle)

Idea: Fit an AR(p) model to the data and obtain the estimates $\hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3, ..., \hat{\phi}_p$ and determine whether these are consistent with the stationarity conditions.

The order p can be determined by the user, but automated implementations of this test choose by default as a large a value of p as can be accomodated by the sample size.

Example:

$\{Y_t\} \sim AR(1) : Y_t = \phi Y_{t-1} + \epsilon_t$

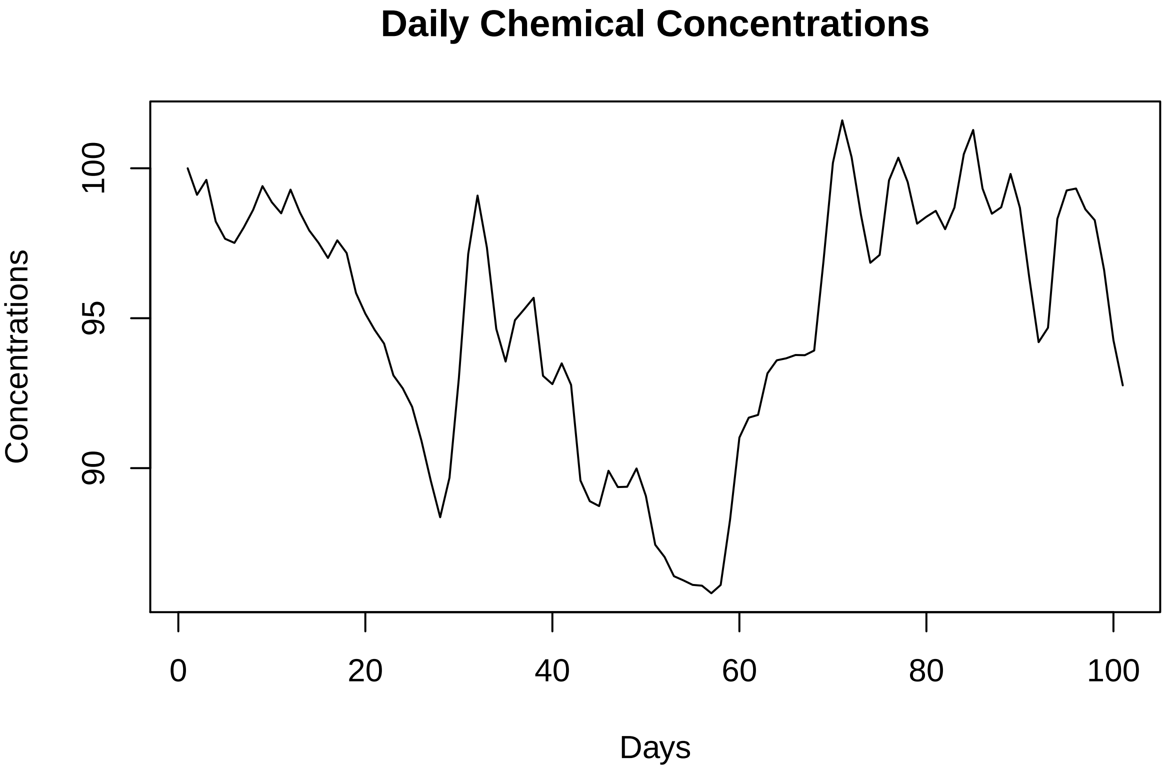
stationarycondition: $|\phi| < 1$ is $\hat{\phi}$ consistent with this condition.

(a nonstationary ARMA model is ARIMA)

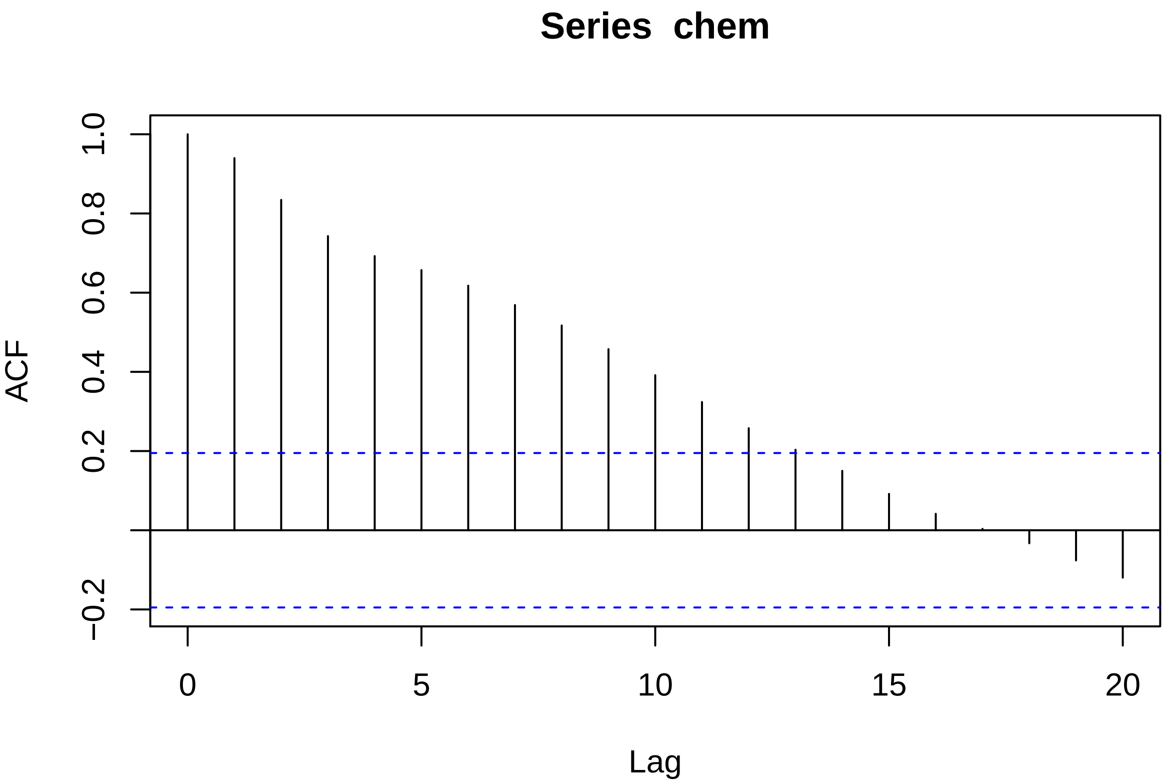
Thusm d is the number of iterations of the ADF test with increasingly differenced data before the H_0 is rejected.

```
library(forecast)
library(tseries)
```

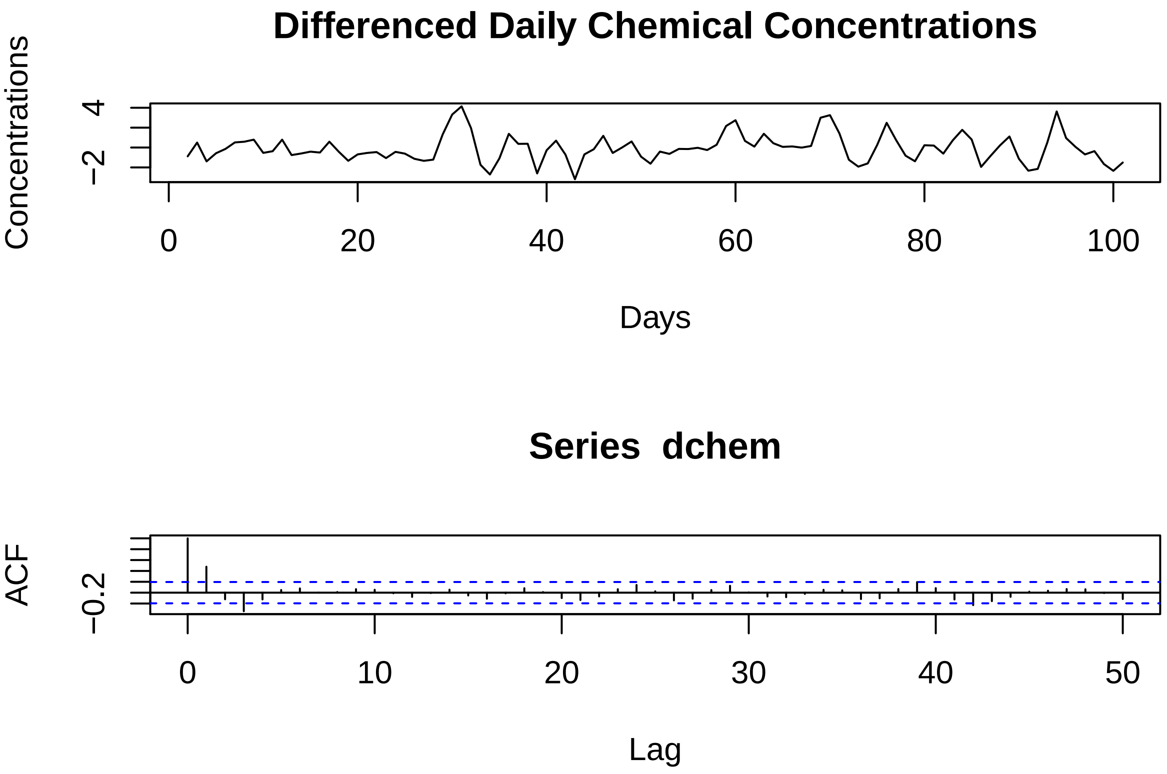
```
chem <- read.csv("chemical.csv", header = F)
chem <- ts(chem$V1)
plot(chem, main = "Daily Chemical Concentrations", ylab = "Concentrations", xlab = "Days")
```



```
# Check whether ordinary differencing is necessary
acf(chem) #it seems necessary
```



```r
# Apply ordinary differencing
dchem <- diff(chem)
par(mfrow = c(2,1))
ts.plot(dchem, main = "Differenced Daily Chemical Concentrations", ylab = "Concentrations", xlab = "Day
acf(dchem, lag.max = 50)
```



```
# This seems stationary. Let's check with ADF test:
ndiffs(x = chem, test = "adf")
```

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

```
ndiffs(x = dchem, test = "adf")
```

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

```
adf.test(x = chem, alternative = "stationary")
```

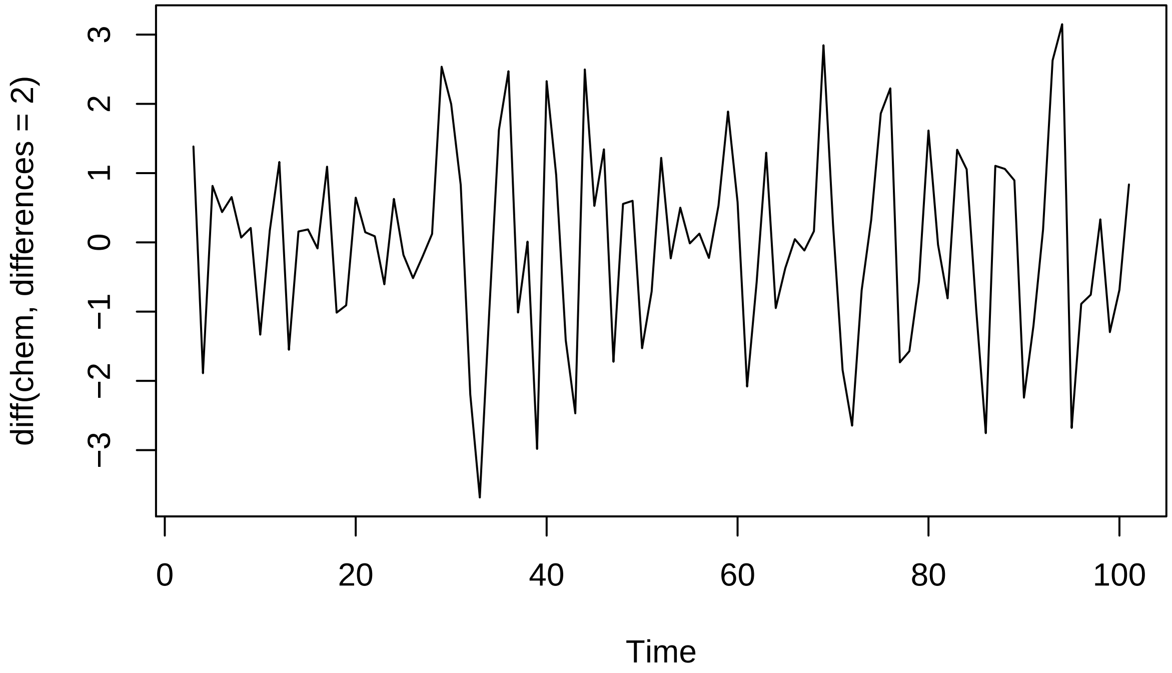
```
##
##  Augmented Dickey-Fuller Test
##
## data:  chem
## Dickey-Fuller = -1.9696, Lag order = 4, p-value = 0.5893
## alternative hypothesis: stationary
```

```
adf.test(x = dchem, alternative = "stationary")
```

```
## Warning in adf.test(x = dchem, alternative = "stationary"): p-value smaller
## than printed p-value
```

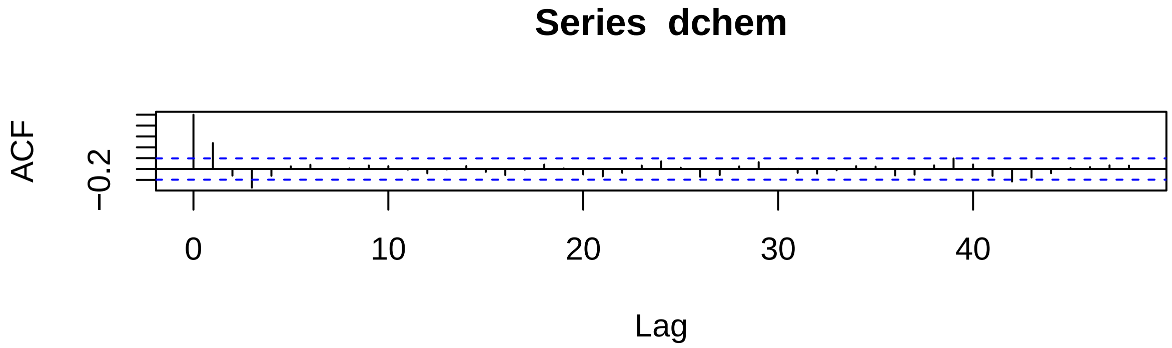
```
##
##  Augmented Dickey-Fuller Test
##
## data:  dchem
## Dickey-Fuller = -4.6141, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

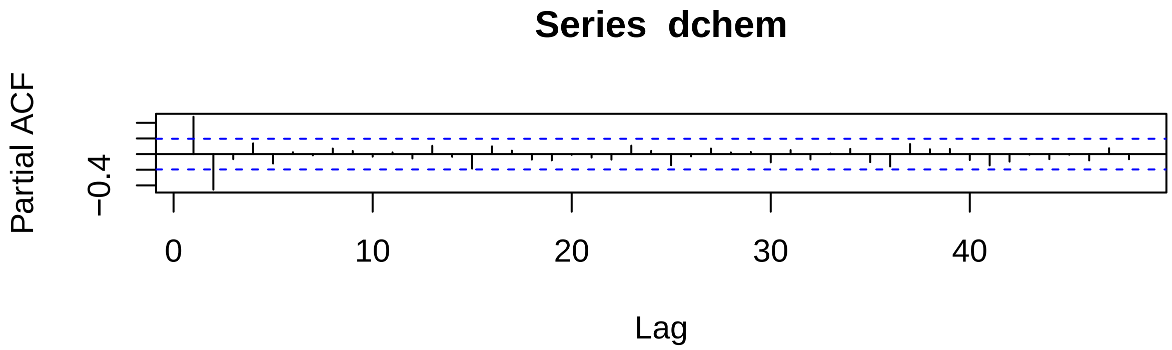
```
plot(diff(chem,differences = 2))
```



```
# So let's use ARIMA to model this. First we need to pick orders p and q.
# We do that with ACF and PACF plots:
par(mfrow=c(2,1))
acf(dchem, lag.max = 48)
```

```r
pacf(dchem, lag.max = 48)
```





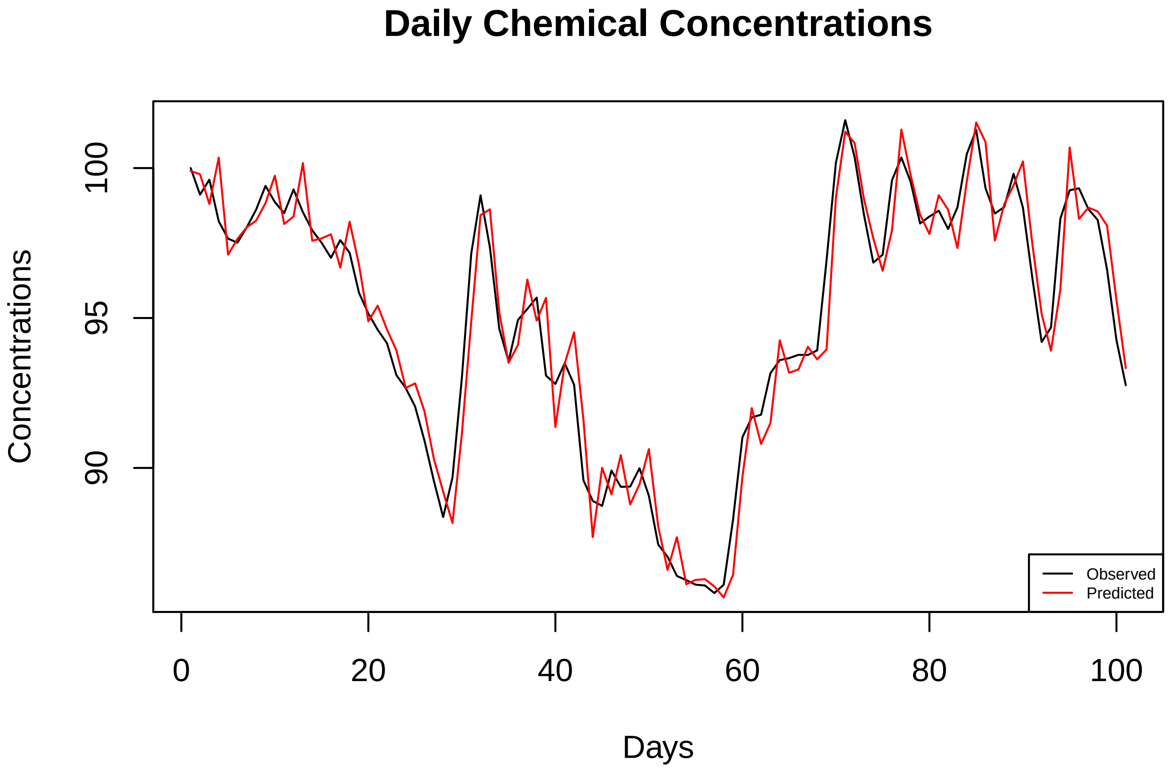
```r
m <- arima(x = chem, order = c(2,1,3))
summary(m)
```

```
##
## Call:
## arima(x = chem, order = c(2, 1, 3))
##
## Coefficients:
##          ar1      ar2     ma1     ma2      ma3
##       0.5324  -0.3412  0.1635  0.0626  -0.1819
## s.e.  0.2441   0.2540  0.2519  0.2849   0.1796
##
## sigma^2 estimated as 1.115:  log likelihood = -147.78,  aic = 307.56
##
## Training set error measures:
##                       ME     RMSE       MAE         MPE      MAPE
## Training set -0.04703029 1.050942 0.8556846 -0.05294365 0.9031028
##                   MASE         ACF1
## Training set 0.8194308 -0.003084045
```

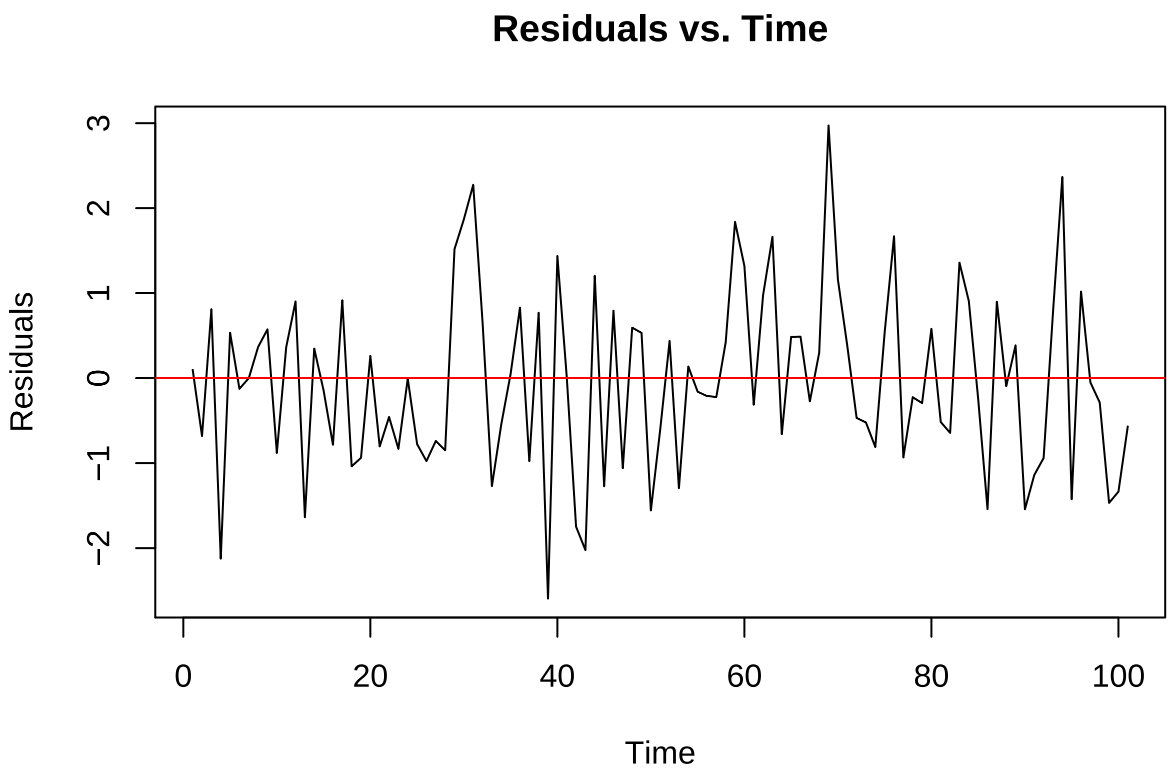
```r
# What would auto.arima() have chosen?
auto.arima(chem)
```

```
## Series: chem
## ARIMA(2,1,0)
##
## Coefficients:
##          ar1      ar2
##        0.6994  -0.4592
## s.e.   0.0888   0.0893
##
## sigma^2 estimated as 1.173:  log likelihood=-149.21
## AIC=304.41   AICc=304.66   BIC=312.23
```

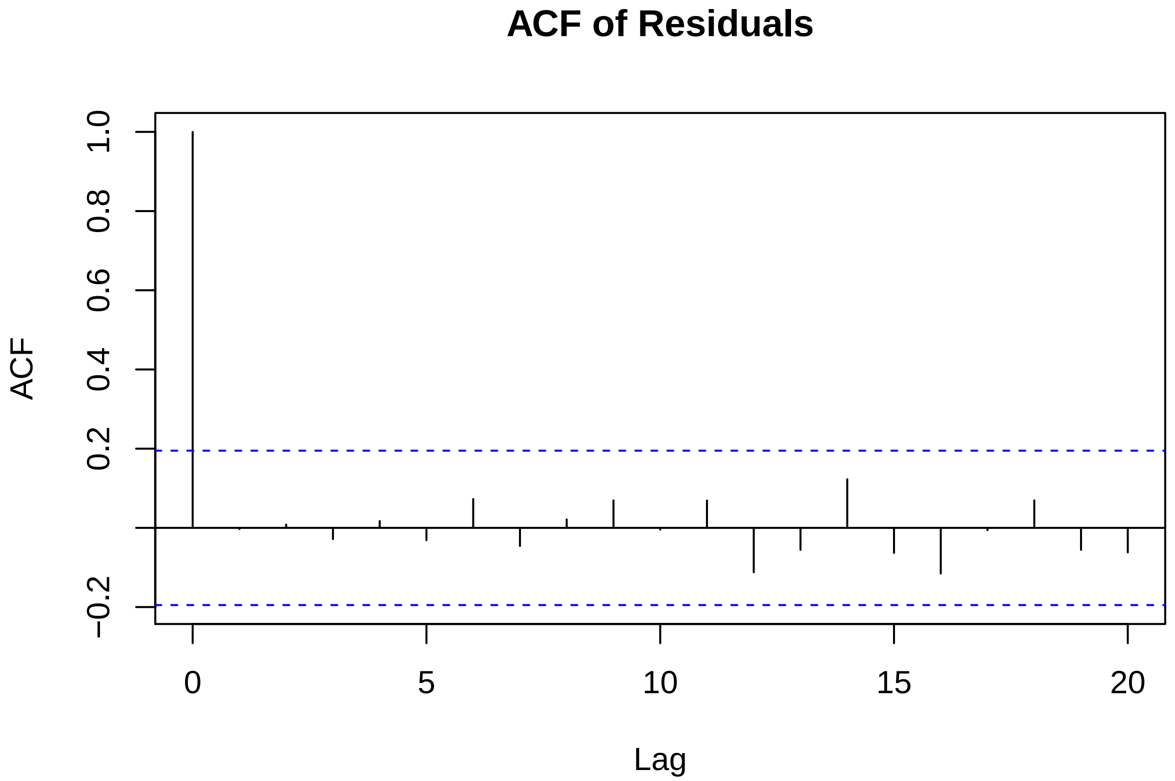
```
# Let's visualize how well this model fits the data:
f <- chem - m$residuals #fitted values
par(mfrow=c(1,1))
plot(chem, type = "l", main = "Daily Chemical Concentrations", ylab = "Concentrations", xlab = "Days")
points(f, type = "l", col = "red")
legend("bottomright", legend = c("Observed", "Predicted"), lty = 1, col = c("black", "red"), cex = 0.5)
```



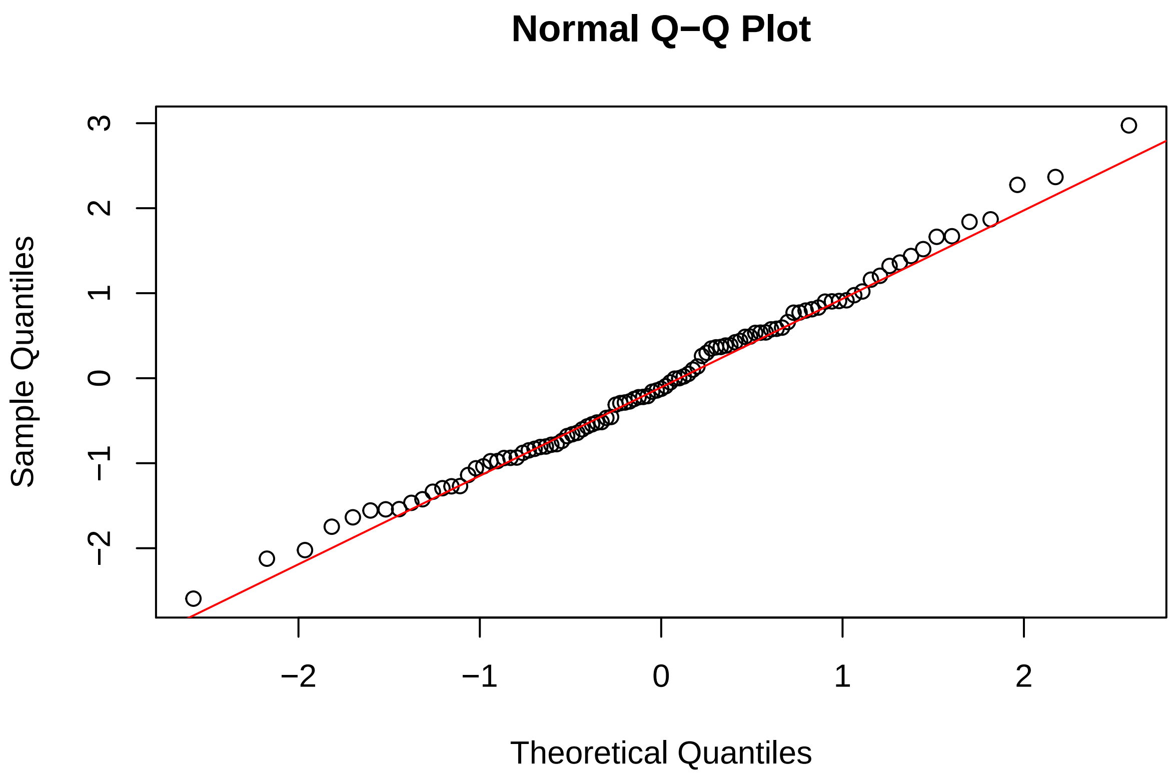
```
# Residual Diagnostics:
plot(m$residuals, main = "Residuals vs. Time", ylab = "Residuals")
abline(h = 0, col = "red")
```



```
acf(m$residuals, main = "ACF of Residuals")
```



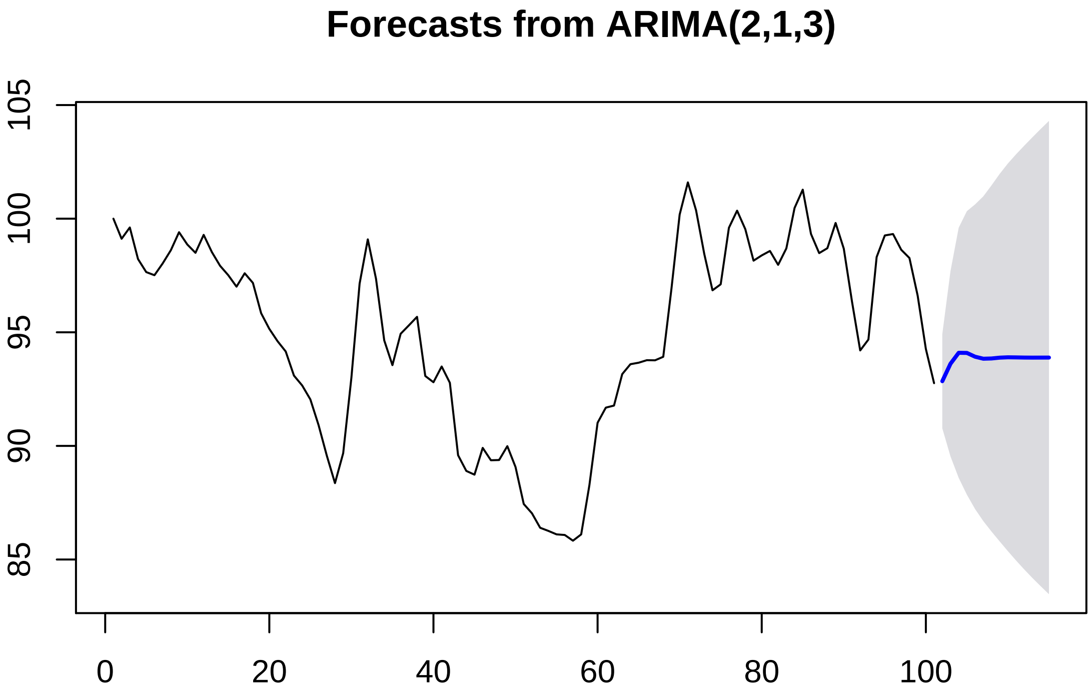
```
qqnorm(m$residuals)
qqline(m$residuals, col = "red")
```



```
shapiro.test(m$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  m$residuals
## W = 0.99331, p-value = 0.9029
```

```
# Forecast
par(mfrow = c(1,1))
plot(forecast(object = m, h = 14, level = 0.95))
```


Forecasts from ARIMA(2,1,3)
105
100
95
90
85
0
20
40
60
80
100