Nov 8 - Notes

Hongdou Li

11/8/2018

Recap:

- $\{Y_t\} \sim ARMA(p,q)$ if :

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q}$$
$$\phi(B) Y_t = \theta(B) \epsilon_t$$

where $\{\epsilon_t\} \sim WN(0, \sigma^2)$ and

  - $\phi(Z) = 1 - \phi_1 Z - \phi_2 Z^2 - ... - \phi_p Z^p$
  - $\theta(Z) = 1 + \theta_1 Z + \theta_2 Z^2 + ... + \theta_q Z^q$

- An ARMA(p,q) model is stationary **iff** the zeros of $\phi(Z)$ lie outside the unit circle in the complex plane.
- An ARMA(p,q) model is invertible **iff** the zeros of $\theta(Z)$ lie outside the unit-circle in the complex plane.

Estimating ARMA(p,q) models

Goal: Use observed data $\{y_1, y_2 ... y_n\}$we want to estimate all of the parameters:$\phi_1, \phi_2 ... \phi_p, \theta_1, \theta_2 ... \theta_q, \sigma$. We'll discuss two methods of estimation. They are (1)** Maximum Likelihood estimation**, and (2)** Least Squares Estimation**.

ML Estimation

- To do this, we need to make assumption about distribution of the errors, and hence the data.

$\overrightarrow{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ . \\ . \\ \epsilon_n \end{bmatrix} \sim MVN(\overrightarrow{0}, \sigma^2 I)$ where I is nxn $\overrightarrow{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ . \\ . \\ Y_n \end{bmatrix} \sim MVN(\overrightarrow{0}, \Gamma)$

where $\Gamma = \begin{bmatrix} \gamma(0) & \gamma(1) & \gamma(2) & ... & \gamma(n-1) \\ \gamma(1) & \gamma(0) & \gamma(1) & ... & \gamma(n-2) \\ \gamma(2) & \gamma(1) & \gamma(0) & ... & \gamma(n-3) \\ . & & & & \\ & & & ... & \gamma(0) \end{bmatrix}$

with distributional assumption, we can write the likelihood function as:

$L(\phi_1 ... \phi_p, \theta_!, ... \theta_q, \sigma) = \frac{1}{(2\pi)^{n/2} |\Gamma|^{1/2}} e^{-1/2 \bar{y}^T \Gamma^{-1} \bar{y}}$

$|\Gamma|^{1/2}$ is the determinant

Where $\bar{y} = (y_1, y_2, ... y_n)^T$ we want to maximize the function and specifically determine the parameter values at which the maximum is attained.

LS Estimation

We want the values of $\phi_1, \phi_2 ... \phi_p, \theta_1, \theta_2 ... \theta_q, \sigma$ that minimize error specifically sum of squared error:

$S(\phi_1, ..., \phi_p, \theta_1 ... \theta_q) = \sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2$

Notice, This is not a function of θ. We estimate θ after having estimated the ϕ's and θ's to be:

$$\hat{\sigma} = \sqrt{\frac{S(\hat{\phi}, \hat{\theta})}{n-p-q}}$$

where $S(\hat{\phi}, \hat{\theta})$ sum of squared residuals (residuals are like estimated value of errors)

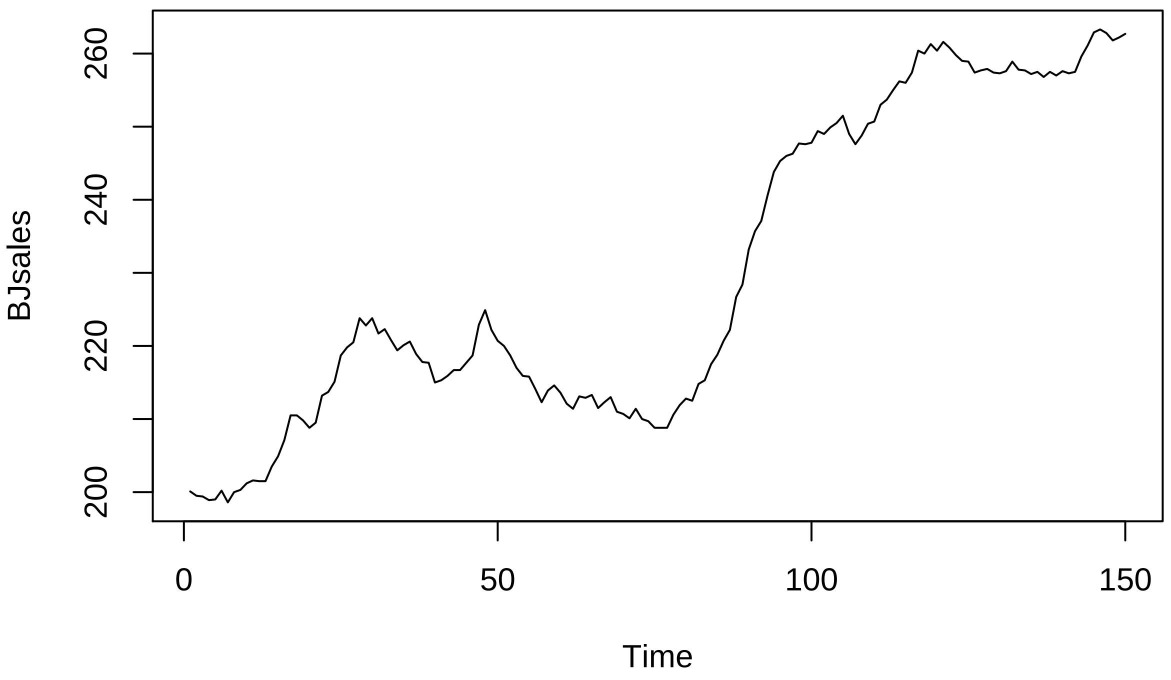
Once a model has been fit to data, we should verify that it follows the assumptions we assume that it does. Specifically we assume that $\{\epsilon_t\} \sim WN(0, \sigma^2)$. We can think of residuals $\{e_t\}$ as sample observations of $\{\epsilon_t\}$ and so we expect the residuals to behave in the same way. Specifically we check whether $\{e_t\}$:

- have zero=mean
- have constant variance
- are uncorrelated
- follow a normal distribution <- (if we used MLEstimation)

In general, we can view model-fitting as a multi-stage process:

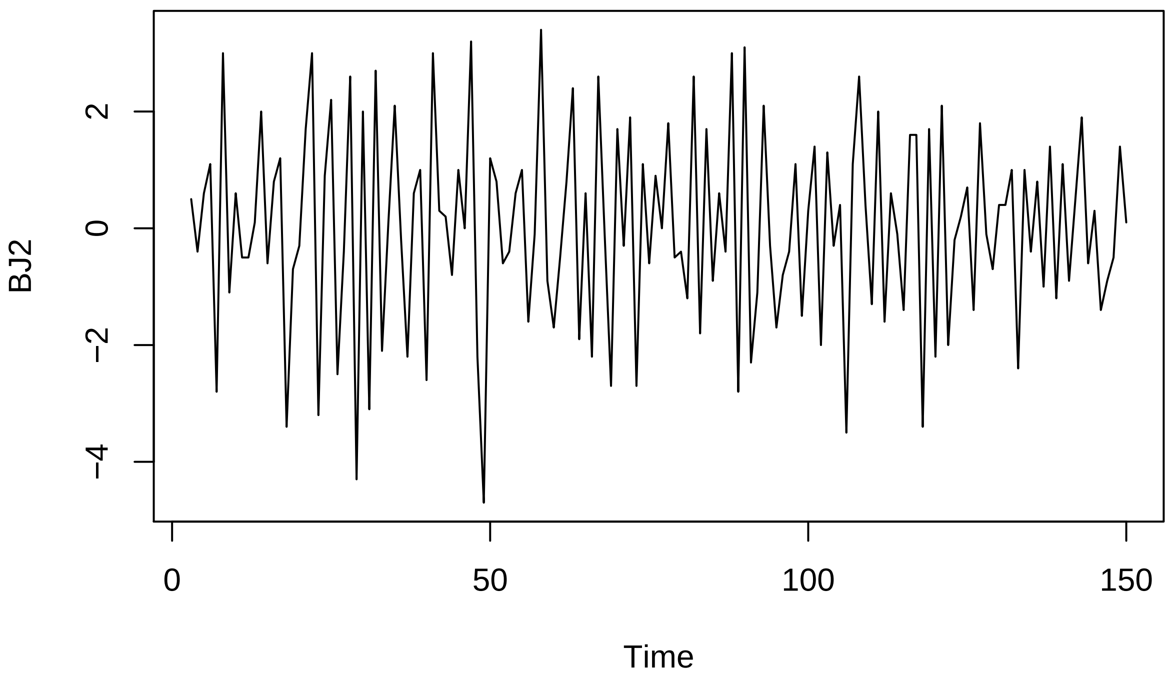
1. Order Selection
2. Model Fitting
3. Verification
4. Forecasting

```r
plot(BJsales)
```

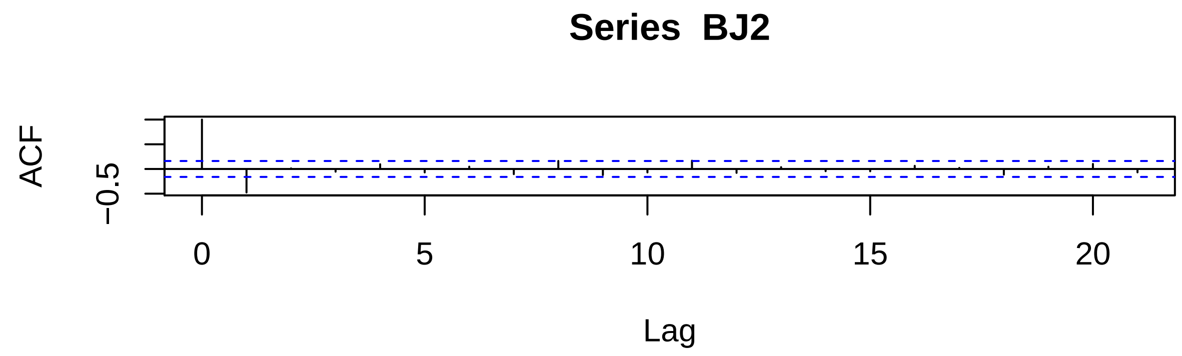


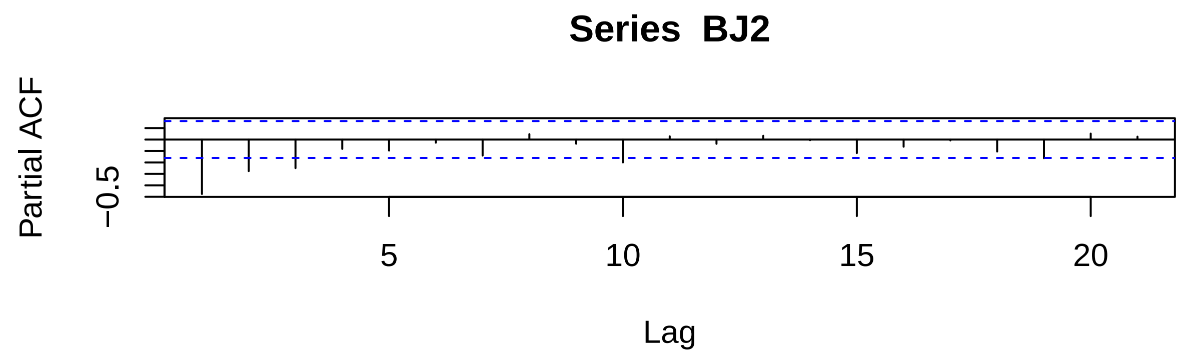
you want to git rid of trend inoder to get the stationary data. after taking the first difference there is still trend. so we take the second difference.

```
#Instead we'll use the twice-differenced version of this time series.
BJ2 <- diff(diff(BJsales))
plot(BJ2)
```



```
par(mfrow=c(2,1))
acf(BJ2)
pacf(BJ2)
```





lookingat this, there is no exponential decay in both ACF and there seems exponential decay in PACF. MA(1) model will fit the model well.

Or one can say that there is no exponential decay in PACF. so that might be ARMA(3,1)

```
fit.ar <- arima(BJ2, order=c(3,0,0),include.mean=FALSE) #AR(3)
fit.ar
```

```
##
## Call:
## arima(x = BJ2, order = c(3, 0, 0), include.mean = FALSE)
##
## Coefficients:
##           ar1      ar2      ar3
##       -0.6738  -0.4264  -0.2482
## s.e.   0.0795   0.0902   0.0790
##
## sigma^2 estimated as 1.915:  log likelihood = -258.38,  aic = 524.76
```

```
fit.ma <- arima(BJ2, order=c(0,0,1),include.mean=FALSE) #MA(1)
fit.ma
```

```
##
## Call:
## arima(x = BJ2, order = c(0, 0, 1), include.mean = FALSE)
##
## Coefficients:
##           ma1
##       -0.7480
## s.e.   0.0662
##
## sigma^2 estimated as 1.866:  log likelihood = -256.57,  aic = 517.14
```

```
fit.arma <- arima(BJ2, order=c(3,0,1),include.mean=FALSE) #ARMA(3,1)
fit.arma
```

```
##
## Call:
## arima(x = BJ2, order = c(3, 0, 1), include.mean = FALSE)
##
## Coefficients:
##          ar1     ar2     ar3      ma1
##       0.2340  0.1786  0.1167  -1.0000
## s.e.  0.0817  0.0824  0.0817   0.0224
##
## sigma^2 estimated as 1.784:  log likelihood = -254.71,  aic = 519.42
```

log-likelihood is the smaller the better.

```
fitls.ar <- arima(BJ2, order=c(3,0,0), method="CSS") #AR(3)
fitls.ar
```

```
##
## Call:
## arima(x = BJ2, order = c(3, 0, 0), method = "CSS")
##
## Coefficients:
##           ar1      ar2      ar3  intercept
##       -0.6776  -0.4325  -0.2529     0.0019
## s.e.   0.0796   0.0907   0.0798     0.0486
##
## sigma^2 estimated as 1.952:  part log likelihood = -259.48
```

```
fitls.ma <- arima(BJ2, order=c(0,0,1), method="CSS") #MA(1)
fitls.ma
```

```
##
## Call:
## arima(x = BJ2, order = c(0, 0, 1), method = "CSS")
##
## Coefficients:
##           ma1  intercept
##       -0.7482     0.0041
## s.e.   0.0642     0.0288
##
## sigma^2 estimated as 1.87:  part log likelihood = -256.31
```

```
fitls.arma <- arima(BJ2, order=c(3,0,1), method="CSS") #ARMA(3,1)
fitls.arma
```

```
##
## Call:
## arima(x = BJ2, order = c(3, 0, 1), method = "CSS")
##
## Coefficients:
##          ar1     ar2     ar3      ma1  intercept
##       0.2684  0.1723  0.1192  -1.0424    -0.0062
## s.e.  0.0694  0.0739  0.0667   0.0030     0.0003
##
## sigma^2 estimated as 1.718:  part log likelihood = -250.05
```

we don't have protection from overfitting (AIC) when we do least square estimation.

```
#We can fit any number of models of different orders and use the output information to select
#the model with the best fit.
m1<-arima(BJ2,order=c(1,0,0))
m2<-arima(BJ2,order=c(2,0,0))
m3<-arima(BJ2,order=c(3,0,0))
m4<-arima(BJ2,order=c(4,0,0))
m5<-arima(BJ2,order=c(0,0,1))
m6<-arima(BJ2,order=c(0,0,2))
m7<-arima(BJ2,order=c(1,0,1))
m8<-arima(BJ2,order=c(2,0,1))
m9<-arima(BJ2,order=c(3,0,1))
m10<-arima(BJ2,order=c(4,0,1))
m11<-arima(BJ2,order=c(1,0,2))
m12<-arima(BJ2,order=c(2,0,2))
m13<-arima(BJ2,order=c(3,0,2))
m14<-arima(BJ2,order=c(4,0,2))

sigma2<-c(m1$sigma2,m2$sigma2,m3$sigma2,m4$sigma2,m5$sigma2,m6$sigma2,m7$sigma2,m8$sigma2,m9$sigma2,m10
loglik<-c(m1$loglik,m2$loglik,m3$loglik,m4$loglik,m5$loglik,m6$loglik,m7$loglik,m8$loglik,m9$loglik,m10
AIC<-c(m1$aic,m2$aic,m3$aic,m4$aic,m5$aic,m6$aic,m7$aic,m8$aic,m9$aic,m10$aic,m11$aic,m12$aic,m13$aic,m
d <- data.frame(pq = c("(1,0)","(2,0)","(3,0)","(4,0)","(0,1)","(0,2)","(1,1)","(2,1)","(3,1)","(4,1)",
d
```

```
##       pq   sigma2    loglik      AIC
## 1  (1,0) 2.215049 -268.9796 543.9593
## 2  (2,0) 2.044968 -263.1467 534.2933
## 3  (3,0) 1.914904 -258.3804 526.7609
## 4  (4,0) 1.901316 -257.8680 527.7360
## 5  (0,1) 1.865771 -256.5647 519.1295
## 6  (0,2) 1.863648 -256.4948 520.9896
## 7  (1,1) 1.863253 -256.4831 520.9661
## 8  (2,1) 1.851161 -256.1343 522.2685
## 9  (3,1) 1.783163 -254.6874 521.3748
## 10 (4,1) 1.748282 -253.1107 520.2215
## 11 (1,2) 1.765652 -253.7712 517.5425
## 12 (2,2) 1.849999 -255.9894 523.9788
## 13 (3,2) 1.762841 -253.6767 521.3534
## 14 (4,2) 1.748334 -253.1103 522.2206
```

```
# Order this by sigma2
#d[order(d$sigma2),]

# Order this by loglik
#d[order(-d$loglik),]

# Order this by AIC
#d[order(d$AIC),]
```

More on order selection

- We use $\hat{\sigma}^2$, $logL(\hat{\phi}, \hat{\theta}, \hat{\sigma})$ and AIC to help choose an optimal model (ie., the best p and q). all three of them are smaller better.
- Note AIC includes a penalty for having an overly complicated model, and hence protect us from over-fitting. The other metrics don't do this.
- $AIC = -2logL(\hat{\phi}, \hat{\theta}, \hat{\sigma}) + 2(p+q+1)$ (Akaike Information Criterion)
- corrected version of AIC: $AICC = -2logL(\hat{\phi}, \hat{\theta}, \hat{\sigma}) + \frac{2(p+q+1)n}{n-p-q-2}$

* Note that we cannotcalculate these information criterion if we do LSEstimation.

We can formally compare "nested" models using a **likelihood ratio test** (LRT)

$$H_0\text{: reduced and full models fit the data equally well}$$

vs.

$$H_1\text{: the full model fits the data better than the reduced model}$$

$$D = -2log[\frac{L(\text{reduced model})}{L(\text{full model})}]$$

Where m_F = # parameters in full model

m_R = # parameters in reduced model

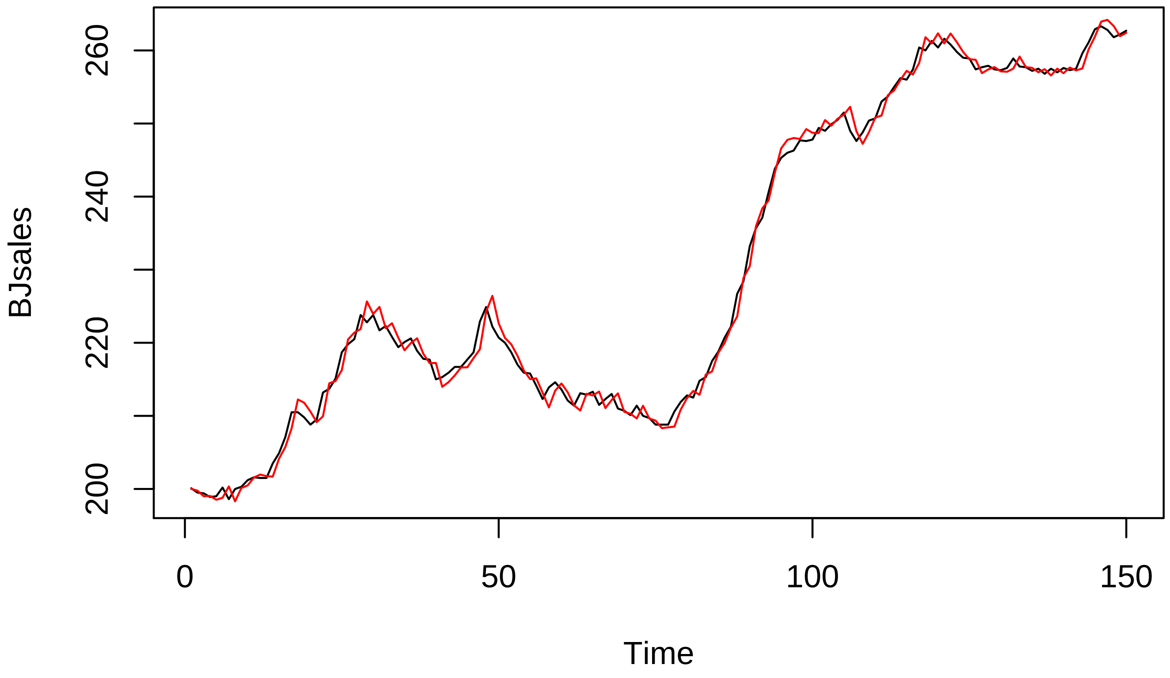
* Large values of D provide evidence against H_0.

```
# Compare MA(1) with ARMA(1,2)
D <- -2*(m5$loglik - m11$loglik)
pval <- 1-pchisq(D,2)
print(c("Test Statistic:", round(D, 4), "P-value:", round(pval, 4)))
```

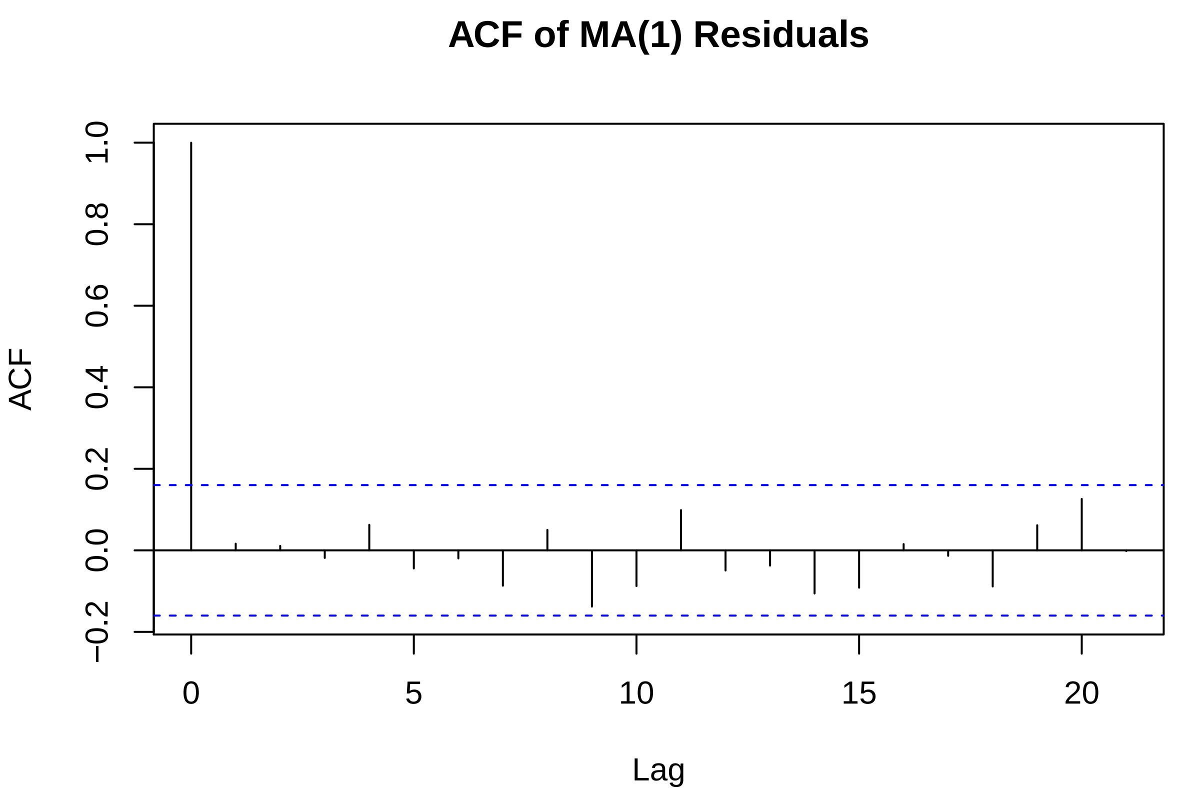
```
## [1] "Test Statistic:" "5.587"           "P-value:"        "0.0612"
```

for the lab, We may choose to use MA(1) because it is simpler.

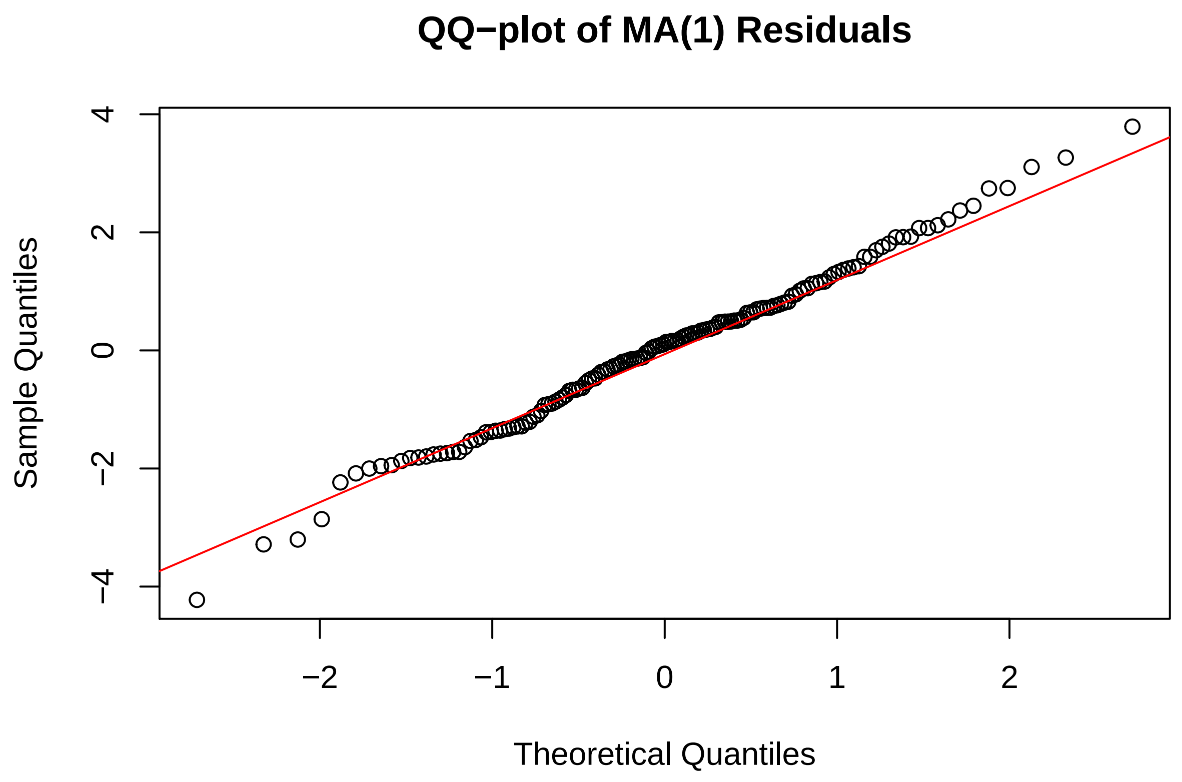
```
# Fit the MA(0,1) model to the data:
MA1 <- arima(BJsales, order = c(0,2,1)) # 2 means we take 2 differences
MA1_fit <- BJsales - MA1$residuals
par(mfrow=c(1,1))
plot(BJsales)
points(x = 1:150, y = MA1_fit, type = "l", col = "red")
```



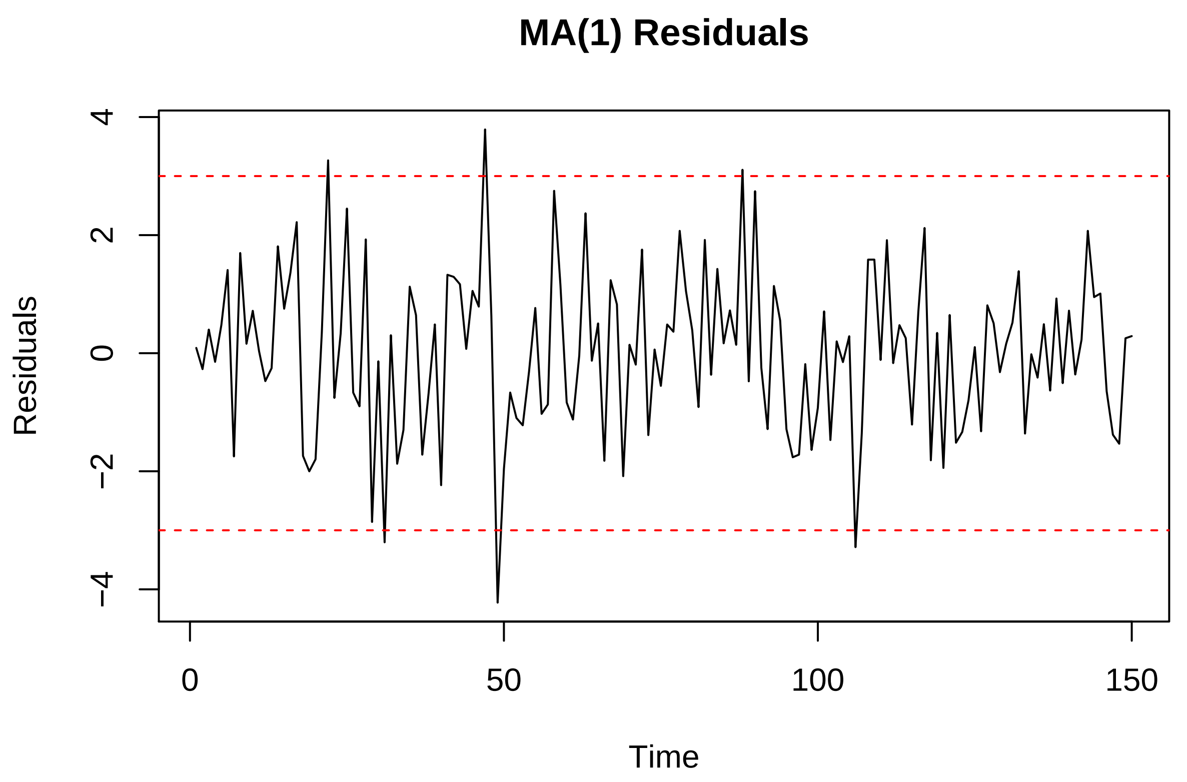
```
# Once we choose a model, we perform Step #3: Verification. That is, we perform model diagnostics
# using the residuals in a manner similar to OLS.
par(mfrow = c(1,1))
acf(MA1$residuals, main = "ACF of MA(1) Residuals")
```



```r
qqnorm(MA1$residuals, main = "QQ-plot of MA(1) Residuals")
qqline(MA1$residuals, col = "red")
```



```r
ts.plot(MA1$residuals, main = "MA(1) Residuals", ylab = "Residuals")
abline(h = c(-3,3), col = "red", lty = 2)
```



Augmented Dickey-Fuller Test to test the stationary

```
#install.packages('aTSA')
library(aTSA)
```

```
##
## Attaching package: 'aTSA'

## The following object is masked from 'package:graphics':
##
##     identify
```

```
adf.test(BJsales)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 3.52   0.990
## [2,]   1 2.45   0.990
## [3,]   2 1.94   0.986
## [4,]   3 1.71   0.978
## [5,]   4 1.42   0.960
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -0.172   0.935
## [2,]   1 -0.478   0.880
## [3,]   2 -0.664   0.814
## [4,]   3 -0.837   0.754
## [5,]   4 -1.010   0.693
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -0.986   0.937
## [2,]   1 -1.316   0.861
## [3,]   2 -1.606   0.739
## [4,]   3 -1.789   0.662
## [5,]   4 -2.077   0.541
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```