```
-------------------------------------------------
Nice way to plot two ts
M1 <- glm(y ~ y1 + y2, family = quasi(link = "log", variance = "mu"))
ts.plot(ts(M1$y),ts(M1$fitted.values),xlab="Weeks",ylab="Counts",type="l",col=c(1,
legend('topright', col=c(1,2), c('original','QLM'),pch=c('-','-'))
-------------------------------------------------


Introduction to ARIMA Modeling: Box-Jenkins
===========================================


Definition of ARIMA(p,d,q) by Box-Jenkins
-----------------------------
BX_{t} = X_{t-1}
phi(B) = 1-phi_1(B)-phi_2(B^2)-.....-phi_p(B^p)
theta(B) = 1-theta_1(B)-theta_2(B^2)-.....-theta_q(B^q)
Del^d = (1-B)^d
phi(B)Del^dX_{t} = theta(B)a_{t}

But R uses for ARMA:

X[t] = a[1]X[t-1] + ... + a[p]X[t-p] + e[t] + b[1]e[t-1] + ... + b[q]e[t-q]
The MA coefficients differ in sign from those of S-PLUS.

Simulation of ARIMA
-----------------------
To simulate: arima.sim(list(order=c(p,d,q), ar=c(), ma=c()), n=)


par(mfrow=c(3,2))
x1 <- arima.sim(list(order=c(0,0,1), ma=0.9), n=200)
ts.plot(x1, main="ARIMA(0,0,1): ma=0.9; r1 > 0", cex=1)
acf1 <- acf(x1)
                                                        #Fig. 1
x2 <- arima.sim(list(order=c(0,0,1), ma=-0.9), n=200)
ts.plot(x2, main="ARIMA(0,0,1): ma=-0.9; r1 < 0",cex=1)
acf2 <- acf(x2)
```

```
acf2$acf                                                    #Fig. 1
```

```
> acf1$acf

, , 1
              [,1]
 [1,]  1.000000000
 [2,]  0.493742794 <------True r1 = 0.9/(1+0.81) = 0.4972376 OK!
 [3,] -0.028493954
 [4,] -0.011159535
 [5,]  0.037564103
 [6,] -0.017510740
 [7,] -0.094138481
 [8,] -0.035939552
 [9,]  0.022602530
```

```
> acf2$acf

, , 1
              [,1]
 [1,]  1.000000000
 [2,] -0.448156476 <------True r1 = -0.9/(1+0.81) = -0.4972376 OK!
 [3,] -0.087718487
 [4,]  0.011382800
 [5,] -0.008069823
 [6,]  0.061927669
 [7,]  0.004534838
 [8,] -0.084459193
 [9,]  0.089120433
```

```
In R:
x4 <- arima.sim(list(order=c(1,0,1), ar=-0.6, ma=-0.9), n=200)
ts.plot(x4, main="ARIMA(1,0,1): ar=-0.6, ma=-0.9", cex=1)   #Fig. 1
acf(x4)
```

```
Nonstationary TS:
------------------
x <- arima.sim(list(order=c(1,1,2), ar=0.7, ma=c(0.3,-.4)), n=200)
ts.plot(x, main="ARIMA(1,1,2): ar=0.7,ma=(.3,-.4)")
acf1 <- acf(x)                                          #Fig. 2

acf1$acf  ## Decays Slowly

, , 1
           [,1]
 [1,] 1.0000000
 [2,] 0.9881594
 [3,] 0.9699393
 [4,] 0.9446668
 [5,] 0.9144189
 [6,] 0.8820849
 [7,] 0.8487893
 [8,] 0.8157008
 [9,] 0.7832231


#Rendering x stationary by a 1st diff.
y <- diff(x)
ts.plot(y, main="First Difference")  ## Fig 2'
acf2 <- acf(y)

> acf2$acf ## Decays faster

, , 1
             [,1]
 [1,]  1.00000000
 [2,]  0.64053637
 [3,]  0.60118890
 [4,]  0.43465883
 [5,]  0.23894198
 [6,]  0.14328928
 [7,] -0.03305205
```

```
 [8,] -0.06615476
 [9,] -0.12769663


#Nonstationary x
x <- arima.sim(list(order=c(1,2,2), ar=0.7, ma=c(0.3,-.4)), n=200)
tsplot(x, main="ARIMA(1,2,2): ar=0.7,ma=(.3,-.4)")
acf(x)                                                  #Fig. 3


#Rendering x stationary by a 2nd difference
y <- diff(diff(x))
ts.plot(y, main="2nd Difference")
acf(y)

#Clipping the nonstat. x at the mean                    #Fig. 4
xc <- (x >= mean(x))*1 #
ts.plot(xc, main="Clip ARIMA(1,2,2) at Mean")
acf(xc)

#Clipping the stationary y at the mean
yc <- (y >= 0.0)*1 #clipping.
ts.plot(yc, main="Clip 2nd Difference")
acf(yc)


AR Fitting
----------------
Use:
ar(x, aic=T, order.max=<<see below>>, method="yule-walker")

Note: "burg" alternative method.

x1 <- arima.sim(list(order=c(1,0,0), ar=0.576), n=250)
fitAR <- ar(x1, aic=T, order.max=3, method="yule-walker")

OR: alternatively
fitAR <- ar(x1, aic=T, order.max=3, method="burg")
```

```
##In R
>  names(fitAR)
 [1] "order"        "ar"           "var.pred"     "x.mean"       "aic"
 [6] "n.used"       "order.max"    "partialacf"   "resid"        "method"
[11] "series"       "frequency"    "call"         "asy.var.coef"

> fitAR$ar
[1] 0.5318699
> fitAR$order
[1] 1

> fitAR$aic
        0         1         2         3
81.129985  0.000000  1.903630  3.594618  ##Good: Order is 1.



Example of AR Fit: Lynx Data
-----------------------------
> ts.plot(lynx) #Large numbers
> x <- log(lynx) ##Better use log
> ts.plot(x, main="Lynx Data")                          #Fig. 5
> acf(x)

> LynxFit <- ar(x, aic=T, order.max=15, method="burg")

###With R
> LynxFit$order
[1] 12
>  LynxFit$aic  ##Order  is 12!!!
           0            1            2            3            4            5
223.04472702 112.50531175  21.76572817  22.12811323  19.18028908  18.95102637
           6            7            8            9           10           11
 20.38414926  15.94634460  15.92058937  16.38731060  12.85283412   0.09105204
          12           13           14           15
  0.00000000   1.67017612   3.64946413   5.49169944
```

```
More general fitting for ARIMA:
-----------------------------------
Use:


arima(x, order = c(0L, 0L, 0L),
           seasonal = list(order = c(0L, 0L, 0L), period = NA),
           xreg = NULL, include.mean = TRUE,
           transform.pars = TRUE,
           fixed = NULL, init = NULL,
           method = c("CSS-ML", "ML", "CSS"), n.cond,
           SSinit = c("Gardner1980", "Rossignol2011"),
           optim.method = "BFGS",
           optim.control = list(), kappa = 1e6)


method: fitting method: maximum likelihood or minimize conditional
          sum-of-squares.  The default (unless there are missing
          values) is to use conditional-sum-of-squares to find starting
          values, then maximum likelihood.

x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), n=500)

###With R
arimaFIT <- arima(x,order=c(1,1,1))
> arimaFIT

Call:
arima(x = x, order = c(1, 1, 1))

Coefficients:
         ar1     ma1
      0.5479  0.7294
s.e.  0.0418  0.0379
```

```
sigma^2 estimated as 1.138:  log likelihood = -742.62,  aic = 1491.23

> names(arimaFIT)
 [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
 [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
[13] "model"

noise <- rnorm(500,0,2) #####Input innovation sequence. Var=4!!!

> x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), innov=noise,n=500)
> length(x)
[1] 501
> mean(x)
[1] 2.635337
> mean(diff(x))
[1] 0.1746507  <------Sample mean after differencing is usually close to 0.


##With R
> x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), n=500)

FIT.arima <- arima(x,order=c(1,1,1))
> names(FIT.arima)
 [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
 [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
[13] "model"

>FIT.arima

Call:
arima(x = x, order = c(1, 1, 1))

Coefficients:
         ar1     ma1
      0.4229  0.7941
s.e.  0.0440  0.0287

sigma^2 estimated as 0.9427:  log likelihood = -695.6,  aic = 1397.2
```

```
> FIT.arima$model
$phi
[1] 0.4229153

$theta
[1] 0.794122

$Delta
[1] 1

$Z
[1] 1 0 1

$a
[1]   -1.273199    1.441105 -126.833289

$P
              [,1]          [,2]          [,3]
[1,]  0.000000e+00  0.000000e+00 -2.721902e-28
[2,]  0.000000e+00  1.202922e-16 -1.493925e-27
[3,] -2.721902e-28 -1.493925e-27  2.721902e-28

$T
          [,1] [,2] [,3]
[1,] 0.4229153    1    0
[2,] 0.0000000    0    0
[3,] 1.0000000    0    1

$V
         [,1]      [,2] [,3]
[1,] 1.000000 0.7941220    0
[2,] 0.794122 0.6306297    0
[3,] 0.000000 0.0000000    0

$h
[1] 0
```

```
$Pn
             [,1]      [,2]         [,3]
[1,] 1.000000e+00 0.7941220 1.609039e-27
[2,] 7.941220e-01 0.6306297 0.000000e+00
[3,] 1.609039e-27 0.0000000 2.721902e-28

>
FIT.arima <- arima(x,order=c(1,1,1), method="CSS")


> FIT.arima

Call:
arima(x = x, order = c(1, 1, 1), method = "CSS")

Coefficients:
         ar1     ma1
      0.4988  0.7919
s.e.  0.0408  0.0268

sigma^2 estimated as 0.9085:  part log likelihood = -685.48


############## FORECASTTING #################

Can get forecasts in two ways. First, from a package forecast which we
need to download. Second (much simpler) use function predict() from R.
Both functions are applied to an ARIMA model.

1. install.packages('forecast')
2. library(forecast)
3. help(forecast) to see functions inside

# fit an ARIMA model of order P, D, Q
fit <- arima(my_ts, order=c(p, d, q))

x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), n=500)
```

```
M <- arima(x, order = c(1,1,1))

> names(M)
 [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
 [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
[13] "model"


> M

Call:
arima(x = x, order = c(1, 1, 1))

Coefficients:
         ar1     ma1
      0.5297  0.7951
s.e.  0.0402  0.0268

sigma^2 estimated as 1.049:  log likelihood = -722.44,  aic = 1450.87


> acf(M$residuals) ##Perfect.
> cpgram(M$residuals) ##Perfect.



# predict next 5 observations
library(forecast)

> forecast(M, 5)
    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
502       69.40920 68.09665 70.72176 67.40183 71.41658
503       70.36049 67.03866 73.68233 65.28019 75.44080
504       70.86442 65.68591 76.04294 62.94457 78.78428
505       71.13137 64.29654 77.96620 60.67840 81.58433
506       71.27278 62.96702 79.57855 58.57021 83.97535
```

```
> plot(forecast(M, 5))

x_fit <- fitted(M) ##### Got from "forecast".
ts.plot(x_fit[1:100],col=1)
lines(x[1:100],col=2)

FFF <- forecast(M, 5)
> names(FFF)
 [1] "method"    "model"     "level"     "mean"      "lower"     "upper"
 [7] "x"         "xname"     "fitted"    "residuals"

> head(FFF$lower)
          80%      95%
[1,] 68.09665 67.40183
[2,] 67.03866 65.28019
[3,] 65.68591 62.94457
[4,] 64.29654 60.67840
[5,] 62.96702 58.57021

> head(FFF$upper)
          80%      95%
[1,] 70.72176 71.41658
[2,] 73.68233 75.44080
[3,] 76.04294 78.78428
[4,] 77.96620 81.58433
[5,] 79.57855 83.97535

##Get the same from function predict() from R:
> predict(M, n.ahead = 5)
$pred
Time Series:
Start = 502
End = 506
Frequency = 1
[1] 69.40920 70.36049 70.86442 71.13137 71.27278

$se
```

```
Time Series:
Start = 502
End = 506
Frequency = 1
[1] 1.024190 2.592041 4.040817 5.333244 6.481022

#Check:
> 69.40920+1.96*1.024190
[1] 71.41661 is Hi 95.




Now with S-Plus version Version 6.2.1
---------------------------------------

---Demonstrate use of AIC----

Simulate ARIMA(1,1,1):

noise <- rnorm(500,0,1) #####Input innovation sequence.

x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), innov=noise)
> length(x)
[1] 500
> mean(x)
[1] -3.854367
> mean(diff(x))
[1] -0.011483163  <------Sample mean after differencing is usually close
                         to 0.

FIT.arima <- arima.mle(x,model=list(order=c(1,1,1)), aic=T)


> FIT.arima
Call: arima.mle(x = x, model = list(order = c(1, 1, 1)), aic = T)
```

```
Method:  Maximum Likelihood
Model :  1 1 1

Coefficients:
        AR : 0.52268  <--- OK
        MA : 0.79855  <--- OK

Variance-Covariance Matrix:
            ar(1)       ma(1)
ar(1) 0.006509314 0.004048008
ma(1) 0.004048008 0.003244918

Optimizer has  converged
Convergence Type: relative function convergence
AIC: 1404.16014  <---- For the TRUE model.



(*) Now, fit an incorrect models to the x ARIMA(1,1,1)
    time series to see the change in AIC.

a. ARIMA(1,0,1)
> FIT.arima <- arima.mle(x,model=list(order=c(1,0,1)), aic=T)
> FIT.arima
Call: arima.mle(x = x, model = list(order = c(1, 0, 1)), aic = T)
Method:  Maximum Likelihood
Model :  1 0 1

Coefficients:
        AR : 0.98807
        MA : 0.25118

Variance-Covariance Matrix:
             ar(1)        ma(1)
ar(1) 4.949131e-05 0.0000616756
ma(1) 6.167560e-05 0.0019544328

Optimizer has  converged
```

```
Convergence Type: relative function convergence
AIC: 1432.3785 <---- AIC increases correctly!!!


b. ARIMA(2,1,1)
> FIT.arima <- arima.mle(x,model=list(order=c(2,1,1)), aic=T)
> FIT.arima
Call: arima.mle(x = x, model = list(order = c(2, 1, 1)), aic = T)
Method:  Maximum Likelihood
Model :  2 1 1

Coefficients:
        AR : 0.5033 -0.08625  <--- Not bad.
        MA : 0.74494          <--- Not bad.

Variance-Covariance Matrix:
            ar(1)       ar(2)       ma(1)
ar(1) 0.007333389 0.001369452 0.005524831
ar(2) 0.001369452 0.002983941 0.002375868
ma(1) 0.005524831 0.002375868 0.005720039

Optimizer has  converged
Convergence Type: relative function convergence
AIC: 1401.14944 <--- Minimum AIC for the models considered.

c. ARIMA(2,1,2)
> FIT.arima <- arima.mle(x,model=list(order=c(2,1,2)), aic=T)
> FIT.arima
Call: arima.mle(x = x, model = list(order = c(2, 1, 2)), aic = T)
Method:  Maximum Likelihood
Model :  2 1 2

Coefficients:
        AR : 0.35987 -0.00884 <--- 0.35987 Different from 0.5
        MA : 0.60263 0.11552  <--- 0.60263 Different from 0.8

Variance-Covariance Matrix:
           ar(1)      ar(2)      ma(1)      ma(2)
```

```
ar(1)  0.3767927 -0.2079405  0.3749790 -0.3064355
ar(2) -0.2079405  0.1225396 -0.2073151  0.1746923
ma(1)  0.3749790 -0.2073151  0.3751774 -0.3065823
ma(2) -0.3064355  0.1746923 -0.3065823  0.2552010

Optimizer has NOT converged
Due to: iteration limit
AIC: 1403.16533 <------ Smaller than 1404.16014 but close!!!

d. ARIMA(0,1,2)
> FIT.arima <- arima.mle(x,model=list(order=c(0,1,2)), aic=T)
> FIT.arima
Call: arima.mle(x = x, model = list(order = c(0, 1, 2)), aic = T)
Method:  Maximum Likelihood
Model :  0 1 2

Coefficients:
        MA : 0.26244 0.23801 <--- ???

Variance-Covariance Matrix:
              ma(1)         ma(2)
ma(1)  0.0018904860 -0.0006511165
ma(2) -0.0006511165  0.0018904860

Optimizer has  converged
Convergence Type: relative function convergence
AIC: 1408.89487  <---- AIC increases correctly!!!

e. ARIMA(1,5,1)
> FIT.arima <- arima.mle(x,model=list(order=c(1,5,1)), aic=T)
> FIT.arima
Call: arima.mle(x = x, model = list(order = c(1, 5, 1)), aic = T)
Method:  Maximum Likelihood
Model :  1 5 1

Coefficients:
        AR : -0.73575  <--- ???
        MA : 0.99981
```

```
Variance-Covariance Matrix:
             ar(1)        ma(1)
ar(1) 9.285431e-04 1.988077e-07
ma(1) 1.988077e-07 7.522832e-07

Optimizer has NOT converged
Due to: iteration limit
AIC: 2611.08966  <---- AIC increases correctly!!!



Diagnostics plots for an ARIMA model:
--------------------------------------
Residuals, Acf, P-values of the Ljung-Box Portmanteau goodness of fit
stat. as a function of number of lags of resid. acf. (Chi sq. df=K-p-q)

noise <- rnorm(500,0,1) ##### Input innovation sequence.
x <- arima.sim(list(order=c(1,1,1), ar=0.5, ma=0.8), innov=noise)
FIT.arima <- arima.mle(x,model=list(order=c(1,1,1)), aic=T)

> arima.diag(FIT.arima)   ## Residual Plots       #Fig. 6


> FIT.arima
Call: arima.mle(x = x, model = list(order = c(1, 1, 1)), aic = T)
Method:  Maximum Likelihood
Model :  1 1 1

Coefficients:
        AR : 0.49838
        MA : 0.7625

Variance-Covariance Matrix:
            ar(1)       ma(1)
ar(1) 0.008315946 0.005614589
ma(1) 0.005614589 0.004631286
```

```
Optimizer has  converged
Convergence Type: relative function convergence
AIC: 1428.14553


Partial ACF: Show with AR(3)
------------------------------
> x <- arima.sim(model=list(ar=c(0.4,-0.5, 0.6)), n=500) + 5 ##Mean=5!!!

This is equivalent to
> x <- arima.sim(model=list(ar=c(0.4,-0.5, 0.6)), n=500); x <- x+5

So the new TS is:

Y(t)=X(t)+5=.4*Y(t-1)-.5*Y(t-2)+.6*Y(t-3)+5(1-.4+.5-.6)+e(t)

> length(x)
[1] 500
> mean(x)
[1] 5.050258


> fit <- arima.mle(x, model=list(order=c(3,0,0))) ##Does not give the mean!!!
> fit
Call: arima.mle(x = x, model = list(order = c(3, 0, 0)))
Method:  Maximum Likelihood
Model :  3 0 0

Coefficients:
        AR : 0.63276 -0.43587 0.79396  #<----True=(.4,-.5,.6)

Variance-Covariance Matrix:
              ar(1)         ar(2)         ar(3)
ar(1)  0.0007437031 -0.0005768467 -0.0001338365
ar(2) -0.0005768467  0.0011670440 -0.0005768467
ar(3) -0.0001338365 -0.0005768467  0.0007437031

Optimizer has  converged
```

```
Convergence Type: relative function convergence
AIC: 1443.5506


> acf(x, type="partial")
Call: acf(x = x, type = "partial")

Partial Correlation matrix:
   lag       x
 1   1  0.2001
 2   2 -0.3221
 3   3  0.6220  <--- OK, 3rd order
 4   4  0.0004
 5   5 -0.0199
 6   6 -0.0061
 7   7  0.0587
 8   8  0.0167
 9   9 -0.0169
10  10  0.0039
11  11  0.0115
12  12  0.0275
13  13 -0.0179
14  14 -0.0139
15  15  0.0256
16  16  0.0248
17  17  0.0120

par(oma=c(0,0,4,0))


> diag <- arima.diag(fit)  ### Get a lot of output!

> acf(x, type="partial")    ### Partial ACF             #Fig. 7
> diag <- arima.diag(fit)   ### Residials, ACF, PACF, Ljung-Box p-val
                                of residuals.
> mtext("Fitting AR(3)", outer=T, cex=1.2)
```

```
Forecasting: Use above AR(3) data
-----------------------------------
use: arima.forecast()

> fit$model ## AR(3) from above
$order:
[1] 3 0 0

$ar:
[1]  0.63276 -0.43587 0.79396

$ndiff:
[1] 0


FORECAST <- arima.forecast(x, n=10,model=fit$model)
> FORECAST
$mean:
 [1] 7.136579 4.077151 3.967921 6.402856 5.487476 3.861708 5.184801 5.915272
 [9] 4.529178 4.447229

$std.err: ##Does not include the effect of estimating the mean and the
            parameters!!!
 [1] 1.030036 1.202789 1.203561 1.318992 1.561450 1.581482 1.599695 1.748794
 [9] 1.826640 1.840161


> mean(FORECAST$mean)
[1] 5.101017
```