Regression with ARMA Errors (cf. ARIMAX)

MX

June 2025

Intro

This document demonstrates the simulation and estimation of regression models with ARIMA errors. We also compare different R packages and estimation methods.

The general model structure is:

$$y_t = \beta_0 + \beta_1 x_t + z_t$$

where z_t follows an ARIMA process

Example 1: Regression with AR(1) Errors

Data Generation

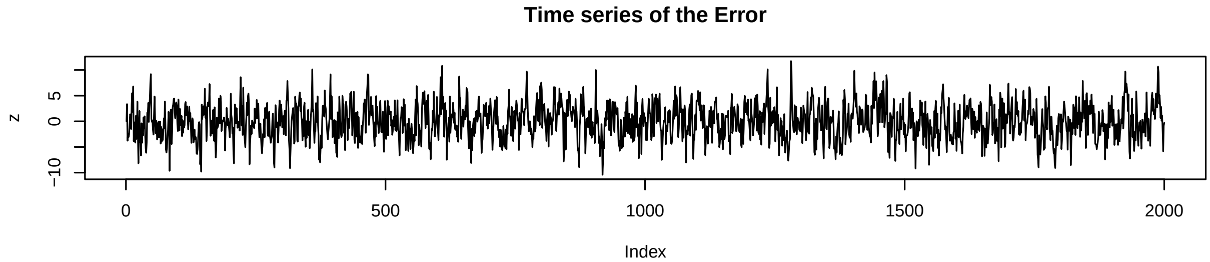
We start by simulating a regression model where the error term follows an AR(1) process:

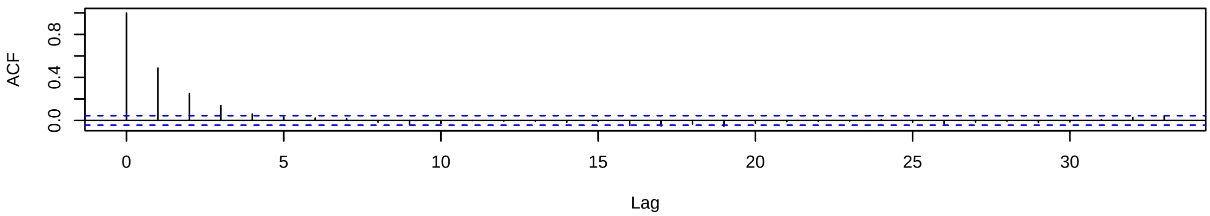
$$z_t = 0.5 z_{t-1} + e_t, \quad e_t \sim WN(0, 3^2)$$

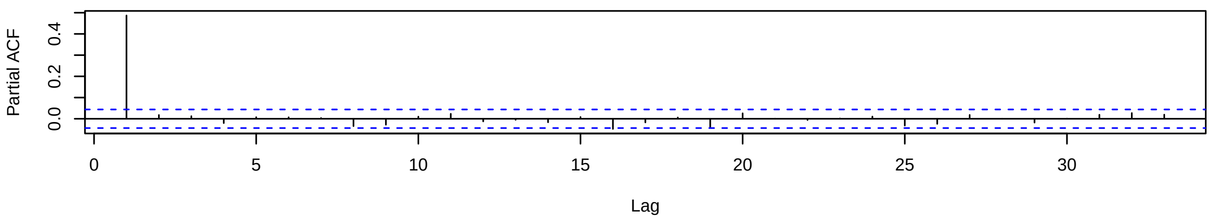
```
set.seed(1989)
n = 2000

phi = 0.5
z = rep(0, n)
e = rnorm(n, mean=0, sd=3)
for (t in 2:n){
  z[t] = phi * z[t-1] + e[t]
}
```

AR(1) looks like this:

```
par(mfcol=c(3,1))
plot(z, type = "l", main = "Time series of the Error")
acf(z, main = "")
pacf(z, main = "")  # spike at 1, looks correct
```





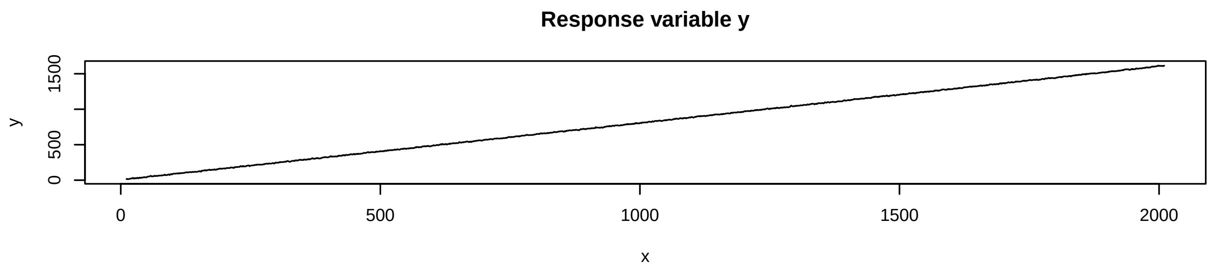


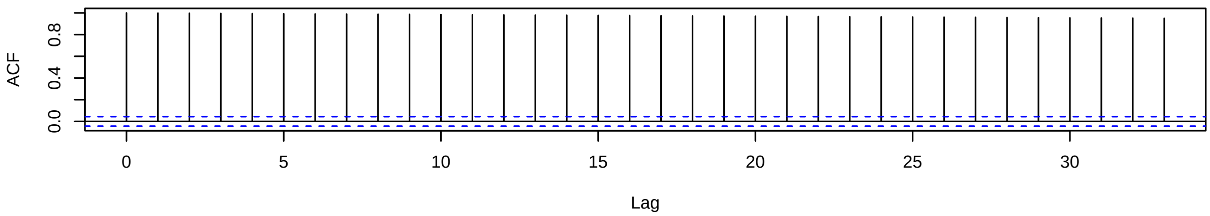
Now we generate the dependent variable:

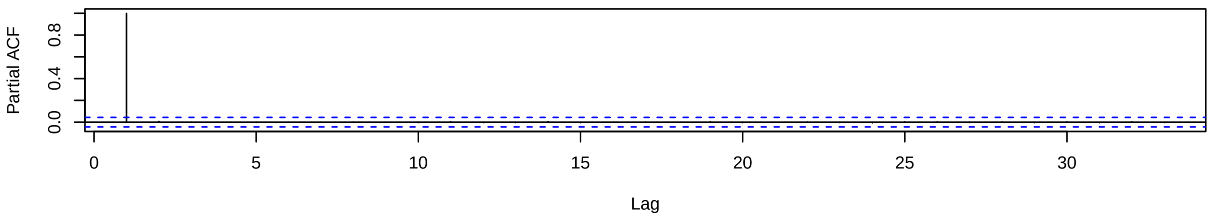
$$y_t = 6 + 0.8x_t + z_t$$

```
x = c(11:(10+n))
beta_0 = 6
beta_1 = 0.8
y = beta_0 + beta_1 * x + z
```

```
par(mfcol=c(3,1))
plot(y~x, type = "l", xlab = "x", main = "Response variable y")
acf(y, main = "")
pacf(y, main = "")  # spike at 1
```







Model Estimation Comparison

Method 1: Simultaneous Estimation with arima() The preferred approach estimates all parameters simultaneously:

```
arima(y, order = c(1,0,0), xreg=x, include.mean = T) # arima() in base R
```

```
##
## Call:
## arima(x = y, order = c(1, 0, 0), xreg = x, include.mean = T)
##
## Coefficients:
##          ar1  intercept       x
##       0.4866     5.8686  0.8001
## s.e.  0.0195     0.2608  0.0002
##
## sigma^2 estimated as 8.841:  log likelihood = -5017.45,  aic = 10042.89
```

```
cat('\n True values: beta_0, which is the "intercept" in the output:', beta_0, ' beta_1 or "x" in the o
```

```
##
##  True values: beta_0, which is the "intercept" in the output: 6  beta_1 or "x" in the output 0.8
```

```r
cat('\n phi or "ar1"', phi, '\t sigma^2 is 3^2')
```

```
##
##  phi or "ar1" 0.5     sigma^2 is 3^2
```

Estimates are close to the true values.

Note, include.mean controls whether the 'intercept' in the output or beta_0 is estimated.

Method 2: Sequential Estimation using lm() Use lm() to fit y first, then use arima() to fit the residuals. We expect this sequential estimation to be inferior than the above, as the standard errors for the regression coefficients from lm() are incorrect because the errors are assumed to be independent when they are autocorrelated.

```r
fit11 = lm(y ~ x)
sumary(fit11)
```

```
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept) 5.86860883 0.15351639   38.228 < 2.2e-16
## x           0.80009574 0.00013191 6065.515 < 2.2e-16
##
## n = 2000, p = 2, Residual SE = 3.40587, R-Squared = 1
```

```r
arima(resid(fit11), order=c(1,0,0), include.mean=T)
```

```
##
## Call:
## arima(x = resid(fit11), order = c(1, 0, 0), include.mean = T)
##
## Coefficients:
##          ar1  intercept
##       0.4866     0.0000
## s.e.  0.0195     0.1295
##
## sigma^2 estimated as 8.841:  log likelihood = -5017.45,  aic = 10040.89
```

The parameter estimates from the regression step are similar, but the standard errors are understated.

Alternative R Packages

```r
library(forecast)
Arima(y, order = c(1,0,0), xreg=x, include.mean=T)
```

Using forecast::Arima()

```
## Series: y
## Regression with ARIMA(1,0,0) errors
##
```

```
## Coefficients:
##          ar1  intercept    xreg
##       0.4866     5.8686  0.8001
## s.e.  0.0195     0.2608  0.0002
##
## sigma^2 = 8.855:  log likelihood = -5017.45
## AIC=10042.89   AICc=10042.91   BIC=10065.3
```

The `forecast::Arima()` function gives identical results to base R's `arima()` except for sigma^2. Note, include.mean controls whether the 'intercept' in the output or *beta*0 is estimated.

Using forecast::auto.arima() Let's see what `forecast::auto.arima()` selects:

```
fit21 = auto.arima(y, xreg=x,
                   seasonal = TRUE, stepwise = F, trace = TRUE,
                   allowdrift = T, allowmean = T,
                   lambda = NULL, approximation = F)
```

```
##
##  Regression with ARIMA(0,0,0) errors : 11677.65
##  Regression with ARIMA(0,0,0) errors : 10581.77
##  Regression with ARIMA(0,0,1) errors : 10832.96
##  Regression with ARIMA(0,0,1) errors : 10147.07
##  Regression with ARIMA(0,0,2) errors : 10585.35
##  Regression with ARIMA(0,0,2) errors : 10076.1
##  Regression with ARIMA(0,0,3) errors : 10454.38
##  Regression with ARIMA(0,0,3) errors : 10049.96
##  Regression with ARIMA(0,0,4) errors : 10395.8
##  Regression with ARIMA(0,0,4) errors : 10047.91
##  Regression with ARIMA(0,0,5) errors : 10367.98
##  Regression with ARIMA(0,0,5) errors : 10049.47
##  Regression with ARIMA(1,0,0) errors : 10314.44
##  Regression with ARIMA(1,0,0) errors : 10042.91
##  Regression with ARIMA(1,0,1) errors : 10253.38
##  Regression with ARIMA(1,0,1) errors : 10044.27
##  Regression with ARIMA(1,0,2) errors : Inf
##  Regression with ARIMA(1,0,2) errors : 10046.09
##  Regression with ARIMA(1,0,3) errors : Inf
##  Regression with ARIMA(1,0,3) errors : 10047.13
##  Regression with ARIMA(1,0,4) errors : Inf
##  Regression with ARIMA(1,0,4) errors : 10049
##  Regression with ARIMA(2,0,0) errors : 10275.58
##  Regression with ARIMA(2,0,0) errors : 10044.3
##  Regression with ARIMA(2,0,1) errors : Inf
##  Regression with ARIMA(2,0,1) errors : Inf
##  Regression with ARIMA(2,0,2) errors : Inf
##  Regression with ARIMA(2,0,2) errors : 10047.21
##  Regression with ARIMA(2,0,3) errors : Inf
##  Regression with ARIMA(2,0,3) errors : Inf
##  Regression with ARIMA(3,0,0) errors : 10248.5
##  Regression with ARIMA(3,0,0) errors : 10046.01
##  Regression with ARIMA(3,0,1) errors : Inf
##  Regression with ARIMA(3,0,1) errors : 10047.22
```

```
##  Regression with ARIMA(3,0,2) errors : Inf
##  Regression with ARIMA(3,0,2) errors : Inf
##  Regression with ARIMA(4,0,0) errors : 10238.44
##  Regression with ARIMA(4,0,0) errors : 10047.17
##  Regression with ARIMA(4,0,1) errors : Inf
##  Regression with ARIMA(4,0,1) errors : 10049.11
##  Regression with ARIMA(5,0,0) errors : 10221.17
##  Regression with ARIMA(5,0,0) errors : 10049.07
##
##
##
##  Best model: Regression with ARIMA(1,0,0) errors
```

```r
summary(fit21)
```

```
## Series: y
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept    xreg
##       0.4866     5.8686  0.8001
## s.e.  0.0195     0.2608  0.0002
##
## sigma^2 = 8.855:  log likelihood = -5017.45
## AIC=10042.89   AICc=10042.91   BIC=10065.3
##
## Training set error measures:
##                         ME     RMSE      MAE        MPE      MAPE      MASE
## Training set -8.833442e-05 2.973434 2.382752 -0.0702387 0.7295079 0.8419134
##                      ACF1
## Training set -0.008366473
```

`auto.arima()` correctly selects ARIMA(1,0,0) based on AICc, but it could be wrong.

Note, set stepwise = F for exhaustive search, trace=TRUE for detailed reporting, allowmean=T for including the 'intercept' in the output or β_0, lambda=NULL for no transformation, approximation=F for no approximation.

Remark on `allowdrift` When the model is not differenced, `allowdrift` parameter is ignored. Suppose the error z_t has an ARIMA(1,1,1) structure:

$$y_t = 6 + 0.8x_t + z_t \text{where } (1 - \phi B)(1 - B)z_t = (1 + \theta B)e_t$$

To estimate this regression model with an ARIMA error, the model needs to be differenced:

$$y_t = 6 + 0.8x_t + z_t, \text{ and } y_{t-1} = 6 + 0.8x_{t-1} + z_{t-1}$$

$$\Delta y_t = 0.8\Delta x_t + \Delta z_t, \text{where } \Delta z_t \text{ follows ARMA(1,1): } \Delta z_t = \phi \Delta z_{t-1} + e_t + \theta e_{t-1}$$

In this situtaion `allowdrift=T` means estimating an additional δ:

$$\Delta y_t = \delta + 0.8\Delta x_t + \Delta z_t$$

This corresponds to a linear trend δt in the level model:

$$y_t = 6 + 0.8x_t + \delta t + z_t$$

```r
library(TSA)
arimax(y, order=c(1,0,0), xreg = x)
```

Using TSA::arimax()

```
##
## Call:
## arimax(x = y, order = c(1, 0, 0), xreg = x)
##
## Coefficients:
##          ar1  intercept    xreg
##       0.4866     5.8686  0.8001
## s.e.  0.0195     0.2608  0.0002
##
## sigma^2 estimated as 8.841:  log likelihood = -5017.45,  aic = 10040.89
```

The `arimax()` function produces results identical to base R's `arima()`. Note: `arimax()` with `xtransf` and `transfer` arguments is used for transfer function models with dynamic regressors, not simple regression-style covariates.

Example 2: Regression with ARMA(1,1) Errors

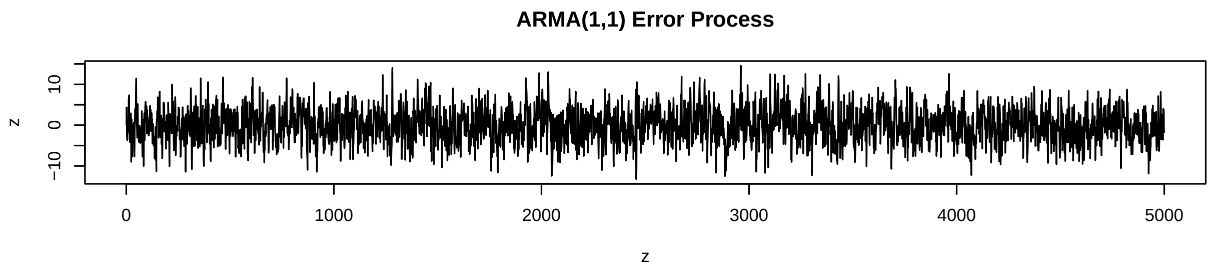
Data Generation

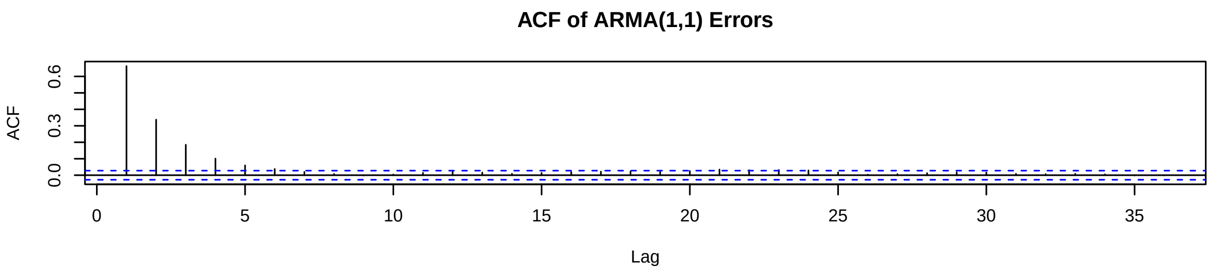
Now consider a more complex error structure with both AR and MA components:

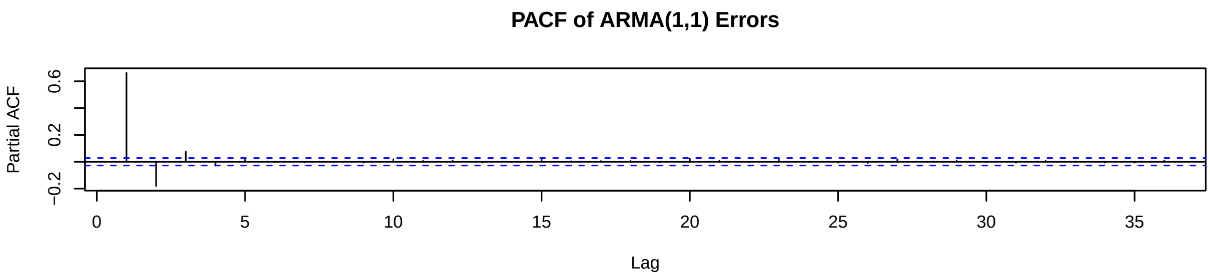
$$z_t = 0.5z_{t-1} + e_t + 0.3e_{t-1}, \quad e_t \sim WN(0, 3^2)$$

```r
rm(list = ls())
set.seed(1989)
n = 5000

phi = 0.5
theta = 0.3
z = rep(0, n)
e = rnorm(n, mean=0, sd=3)
for (t in 2:n){
  z[t] = phi * z[t-1] + e[t] + theta * e[t-1]
}
```

```r
par(mfcol=c(3,1))
plot(z, type = "l", xlab = "z", main = "ARMA(1,1) Error Process")
acf(z, main = "ACF of ARMA(1,1) Errors")
pacf(z, main = "PACF of ARMA(1,1) Errors")
```

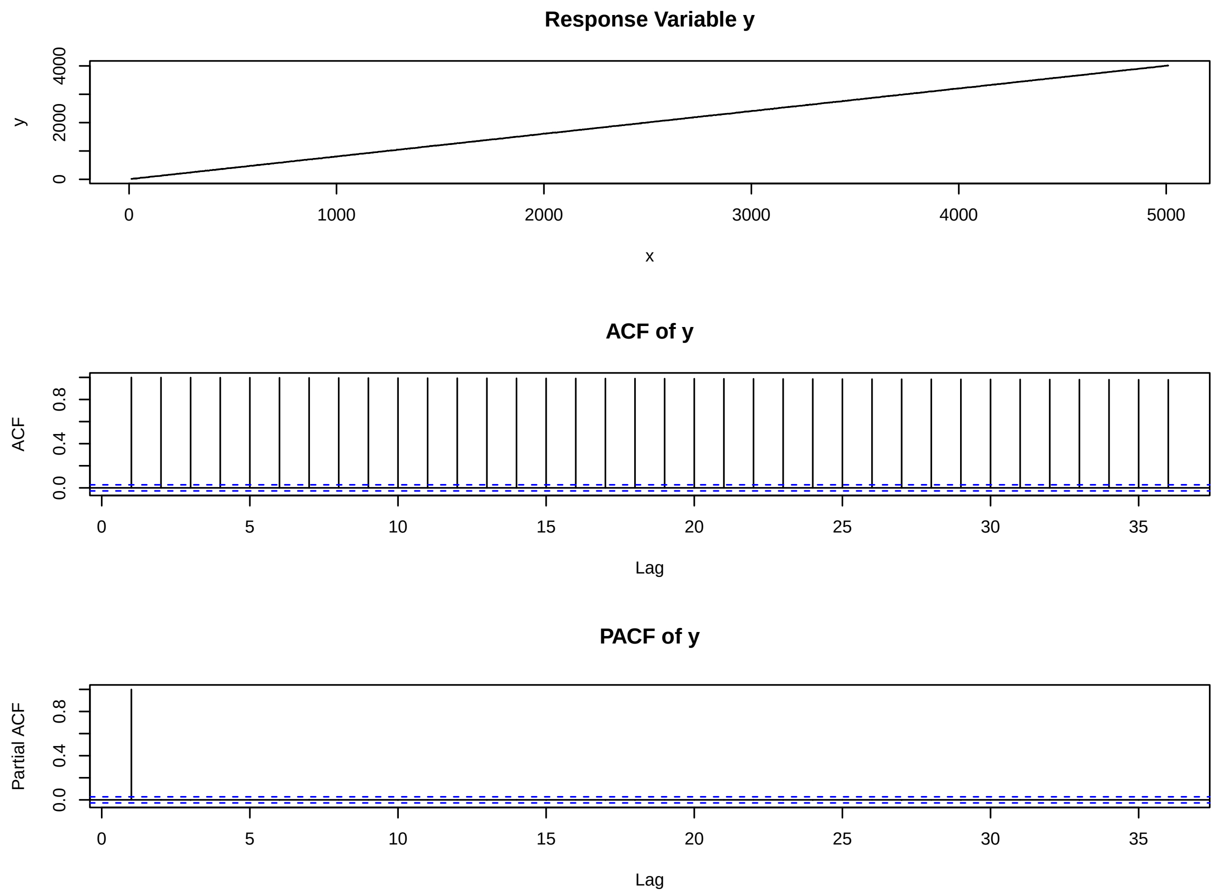







```
x = c(11:(10+n))
beta_0 = 6
beta_1 = 0.8
y = beta_0 + beta_1 * x + z
```

```
par(mfcol=c(3,1))
plot(y~x, type = "l", xlab = "x", main = "Response Variable y")
acf(y, main = "ACF of y")
pacf(y, main = "PACF of y")
```



Model Estimation for ARMA(1,1) Errors

```r
arima(y, order = c(1,0,1), xreg=x)
```

stats: arima()

```
##
## Call:
## arima(x = y, order = c(1, 0, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1  intercept   xreg
##       0.4957  0.3070     6.0258  8e-01
## s.e.  0.0180  0.0201     0.2186  1e-04
##
## sigma^2 estimated as 8.843:  log likelihood = -12544.11,  aic = 25096.22
```

The estimates are reasonably close to true values: - Intercept: (true: 6) - x coefficient:(true: 0.8) - AR(1) coefficient: (true: 0.5) - MA(1) coefficient: (true: 0.3)

```r
library(forecast)
Arima(y, order = c(1,0,1), xreg=x, include.mean=T)
```

forecast::Arima()

```
## Series: y
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1     ma1  intercept   xreg
##       0.4957  0.3070     6.0258  8e-01
## s.e.  0.0180  0.0201     0.2186  1e-04
##
## sigma^2 = 8.85:  log likelihood = -12544.11
## AIC=25098.22   AICc=25098.23   BIC=25130.8
```

```r
fit21 = auto.arima(y, xreg=x,
                   seasonal = TRUE, stepwise = F, trace = F,
                   allowdrift = T, allowmean = T,
                   lambda = NULL, approximation = F)
summary(fit21)
```

```
## Series: y
## Regression with ARIMA(2,0,1) errors
##
## Coefficients:
##          ar1     ar2     ma1  intercept   xreg
##       0.3866  0.0810  0.4125     6.0235  8e-01
## s.e.  0.0610  0.0444  0.0583     0.2236  1e-04
##
## sigma^2 = 8.846:  log likelihood = -12542.5
## AIC=25097   AICc=25097.02   BIC=25136.1
##
## Training set error measures:
##                        ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.0001560642 2.972772 2.373215 -0.03130488 0.346894 0.8713108
##                      ACF1
## Training set 1.306728e-05
```

This time `auto.arima()` fails to identify the ARIMA(1,0,1) structure.

```r
library(TSA)
arimax(y, order=c(1,0,1), xreg = x)
```

TSA:arimax()

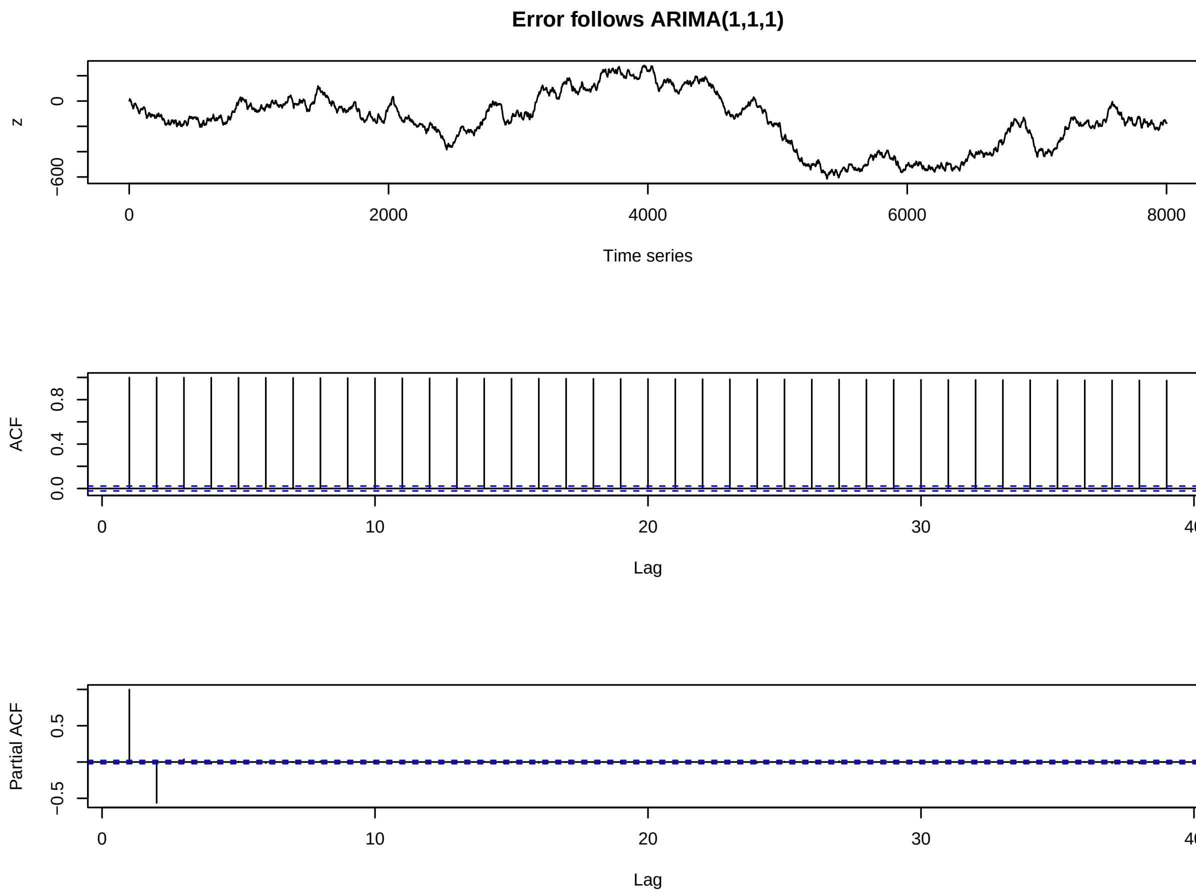
```
##
```

```
## Call:
## arimax(x = y, order = c(1, 0, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1  intercept   xreg
##       0.4957  0.3070     6.0258  8e-01
## s.e.  0.0180  0.0201     0.2186  1e-04
##
## sigma^2 estimated as 8.843:  log likelihood = -12544.11,  aic = 25096.22
```

Again, `arimax()` produces identical results to base R.

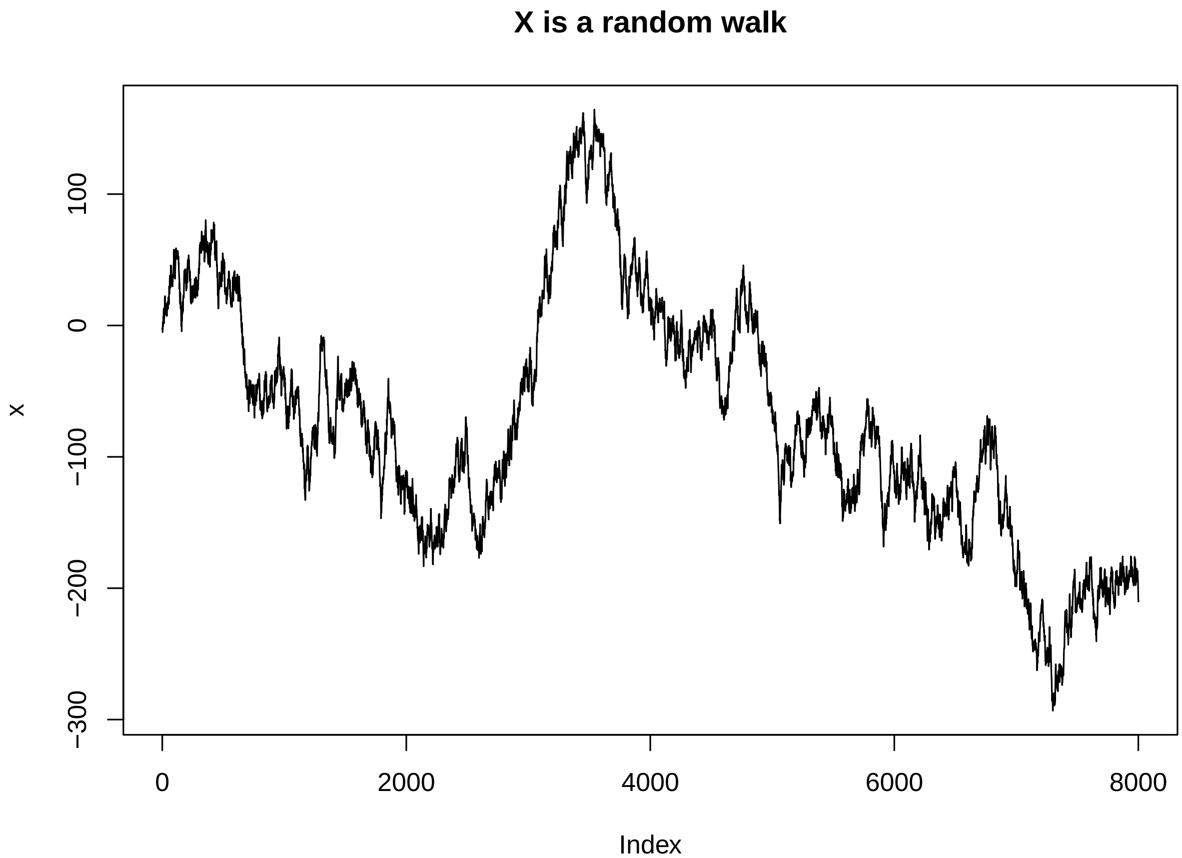
Example 3: Regression with ARIMA(1,1,1) Errors

```r
rm(list =ls())
set.seed(1989)
params = list(order = c(1,1,1), ar = 0.5, ma = 0.3)
n = 8000
z = arima.sim(model = params, n = n, sd = 3)  #length of z = 8001
par(mfcol = c(3,1))
plot(z, type='l', xlab = 'Time series', main='Error follows ARIMA(1,1,1)')
acf(z, main ='')
pacf(z, main='')
```

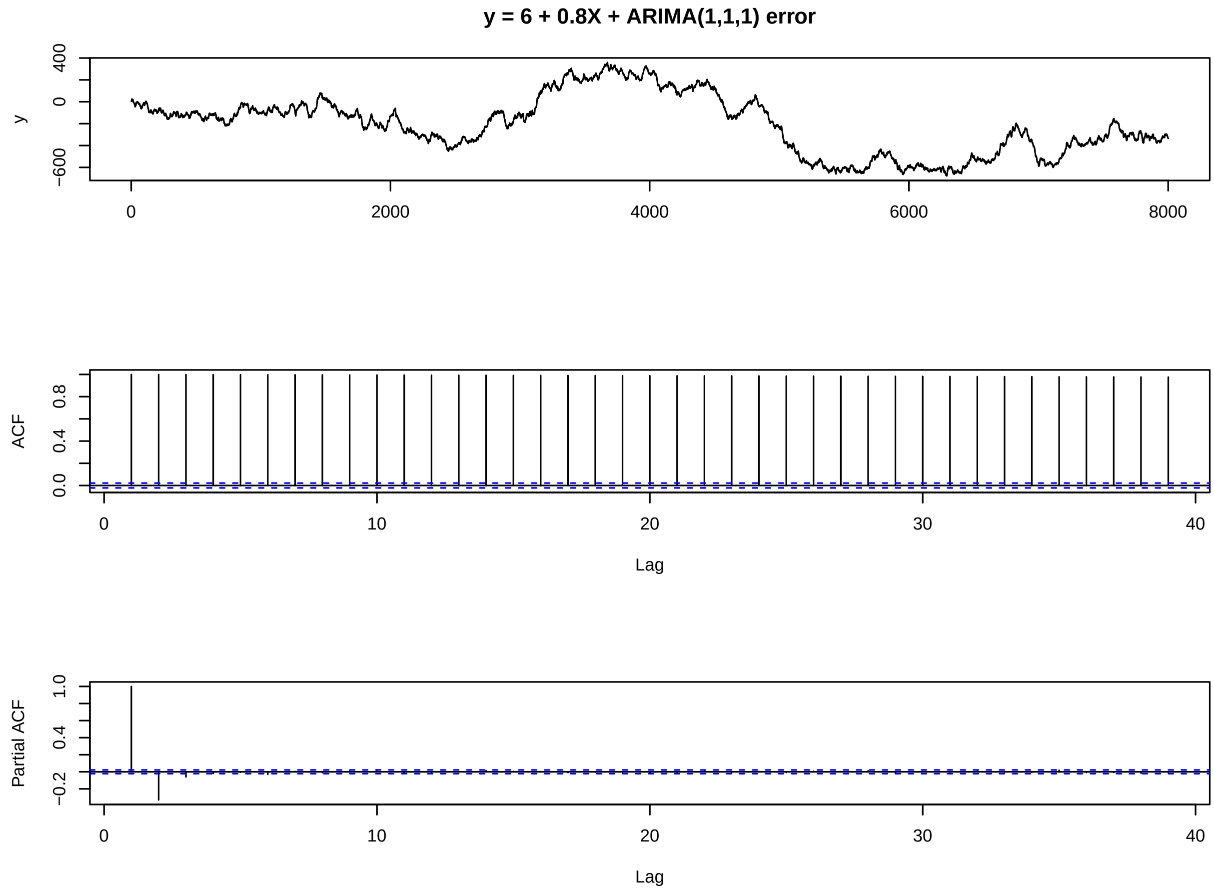


Data Generation

```r
set.seed(1)
x = rnorm(mean=0, sd=4, n=n+1)
x = cumsum(x)   # X is random walk, non-stationary
plot(x, type='l', main = 'X is a random walk')
```




```
beta_0 = 6
beta_1 = 0.8
y = beta_0 + beta_1* x + z

par(mfcol = c(3,1))
plot(y, type='l', xlab = '', main='y = 6 + 0.8X + ARIMA(1,1,1) error')
acf(y, main ='')
pacf(y, main='')
```




```r
arima(y, xreg=x, order=c(1,1,1))
```

Model estimation for ARIMA(1,1,1) errors

```
##
## Call:
## arima(x = y, order = c(1, 1, 1), xreg = x)
##
## Coefficients:
##          ar1     ma1    xreg
##       0.5073  0.2899  0.7937
## s.e.  0.0140  0.0157  0.0063
##
## sigma^2 estimated as 8.884:  log likelihood = -20088.92,  aic = 40183.84
```

```r
arima(diff(y), xreg=diff(x), order = c(1,0,1), include.mean = F)
```

```
##
## Call:
## arima(x = diff(y), order = c(1, 0, 1), xreg = diff(x), include.mean = F)
```

```
##
## Coefficients:
##          ar1     ma1    xreg
##       0.5073  0.2899  0.7937
## s.e.  0.0140  0.0157  0.0063
##
## sigma^2 estimated as 8.884:  log likelihood = -20088.92,  aic = 40183.84
```

Identical results. It appears that differencing is applied to all variables–including X–in the regression model before the model is estimated. Refer to: https://f0nzie.github.io/hyndman-bookdown-rsuite/regression-with-arima-errors-in-r.html.

```
forecast::auto.arima(y, xreg=x, seasonal = TRUE, stepwise = F, trace = F, lambda = NULL, approximation =
```

```
## Series: y
## Regression with ARIMA(1,1,1) errors
##
## Coefficients:
##          ar1     ma1    xreg
##       0.5073  0.2899  0.7937
## s.e.  0.0140  0.0157  0.0063
##
## sigma^2 = 8.888:  log likelihood = -20088.92
## AIC=40185.84   AICc=40185.85   BIC=40213.79
```

Summary

1. **Simultaneous estimation** (using `arima()`, `Arima()`, or `arimax()`) is statistically superior to sequential estimation because it properly accounts for autocorrelated errors

2. **Different R packages** (`stats`, `forecast`, `TSA`) generally produce very similar results for the same model specification.

3. **Automatic model selection** with `auto.arima()` is helpful but cannot guarantee finding the true pattern.

4. **TBD**: "The R function Arima() will fit a regression model with ARIMA errors if the argument xreg is used. The order argument specifies the order of the ARIMA error model. If differencing is specified, then the differencing is applied to all variables in the regression model before the model is estimated." [1] However one can further confirm this by checking the source code.