Week 5 - Autoregressive & Moving Average Models

Jonathan Thong

2023-03-24

Simulating and Estimating AR and MA Models

Let's start by simulating an AR series using the **arima.sim()** function. To do this, we will need to input a set of AR coefficients. To ensure that our coefficients are such that our model is stationary, we will use the **generateAR()** function in the **DREGAR** package

```r
rm(list=ls())
if("DREGAR" %in% rownames(installed.packages()) == FALSE) install.packages("DREGAR")
library(DREGAR)
```

Let's suppose that we want to simulate from an AR(5). We first generate a set of stationary coefficients which we store in the object **ar.coef** :

```r
ar.order <- 5
ar.coef <- generateAR(n = ar.order)
```

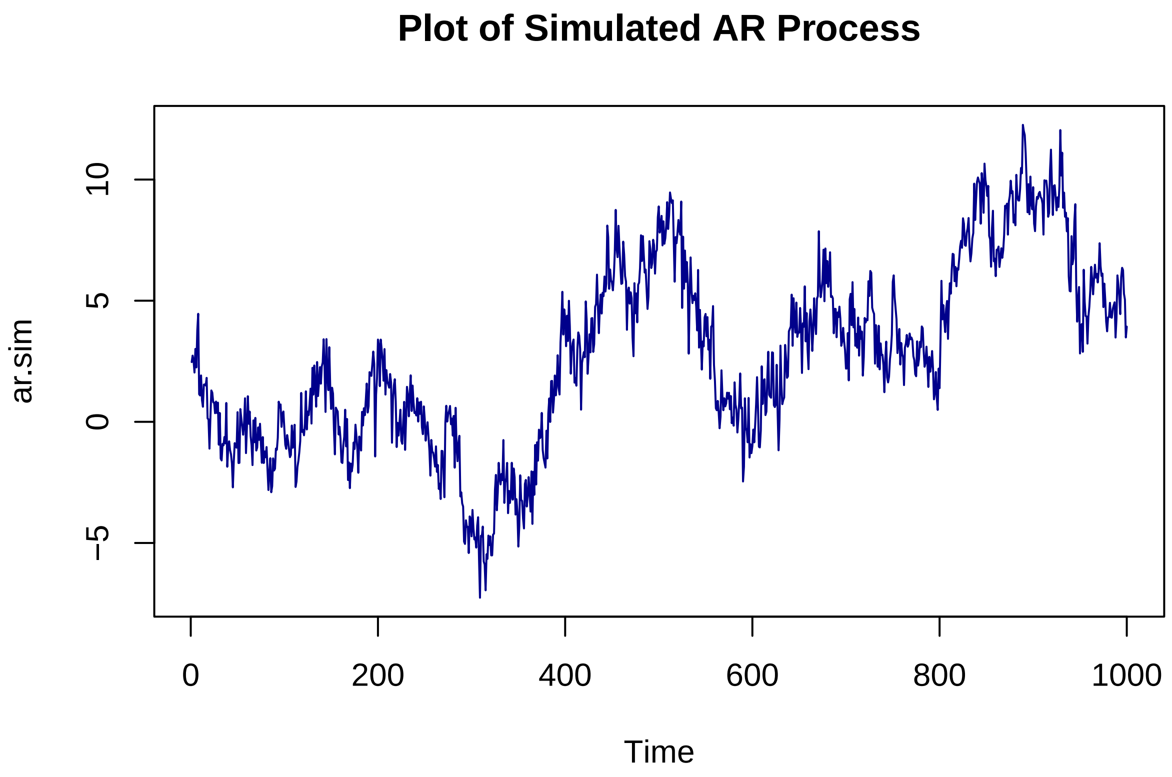
```
##  please wait ...
```

```r
ar.coef
```

```
## [1] 0.53020433 0.16202103 0.12355394 0.09399246 0.07397852
```

Then,we can use these coefficients as inputs into our **arima.sim()** function. Let's simulate 1000 observations:

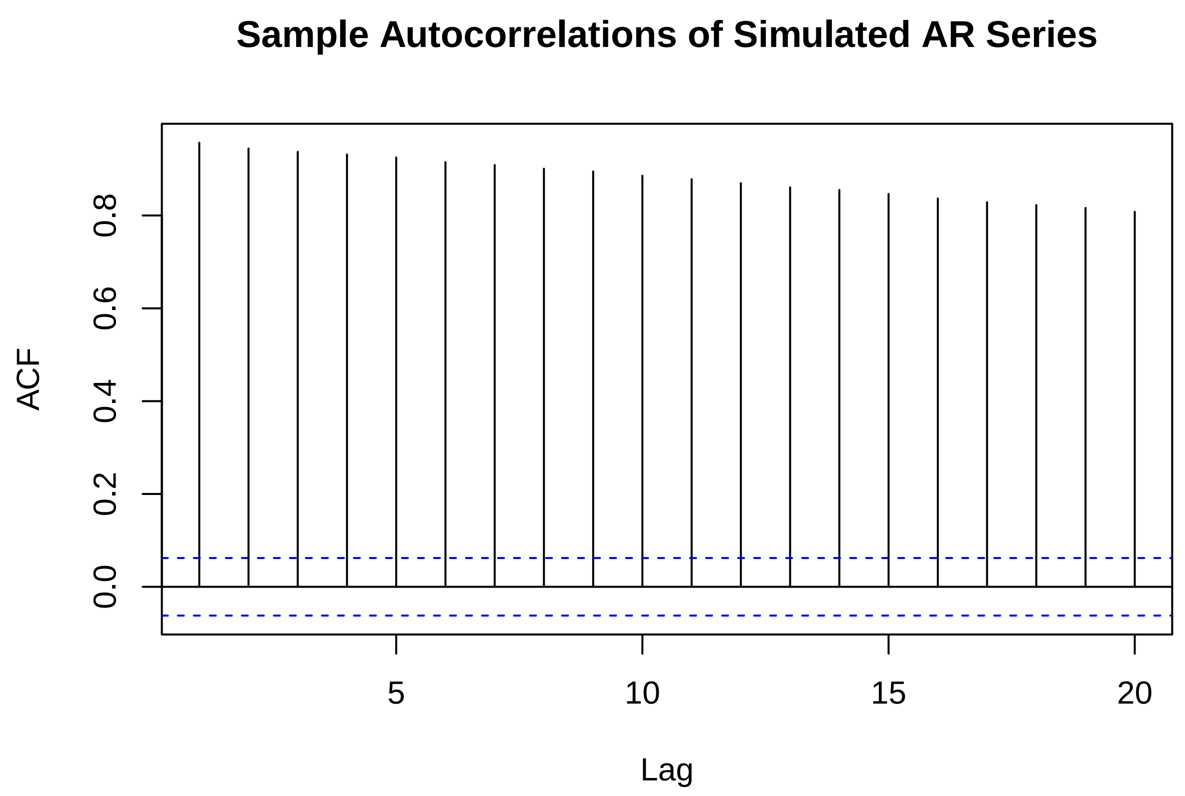
```r
T = 1000
ar.sim <- arima.sim(n = T, list(ar = ar.coef))
plot(ar.sim,
     main = "Plot of Simulated AR Process",
     col = "blue4")
```



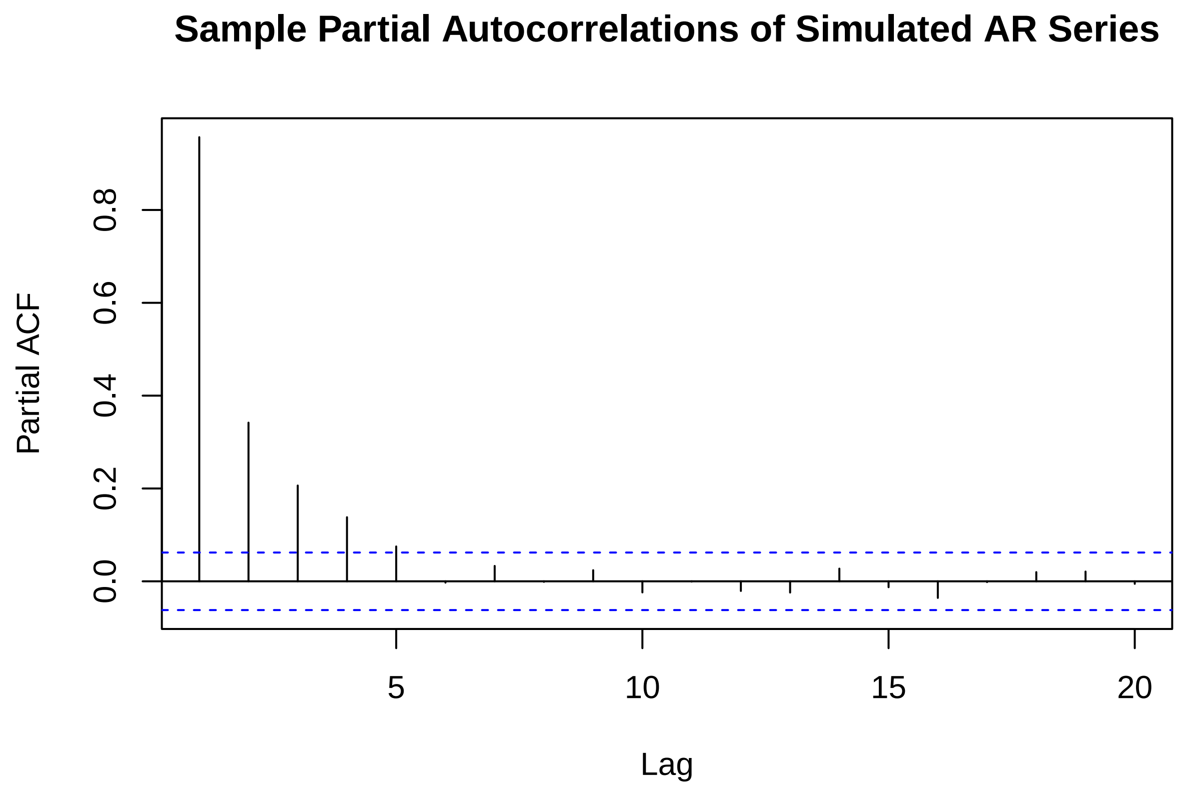
Now let's have a look at the sample autocorrelations and partial autocorrelations.

```
ar.acf <- acf(ar.sim, plot = FALSE)
ar.pacf <- pacf(ar.sim, plot = FALSE)

plot(ar.acf[1:20], main = "Sample Autocorrelations of Simulated AR Series")
```



```
plot(ar.pacf[1:20], main = "Sample Partial Autocorrelations of Simulated AR Series")
```



Let's now fit an AR(5) model to our data using the **Arima()** function from the **forecast** package. To do this, we need to specify the order of model in the argument of the function via the inputs **c(p,d,q)** where **p** denotes the order of the AR component, **d** denotes the order of integration (we will discuss this in a later lecture) and **q** denotes the order of the MA component.

```
library(forecast)
```

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

```
ar.mod1 <- Arima(ar.sim, order = c(5,0,0), include.mean = FALSE)
summary(ar.mod1)
```

```
## Series: ar.sim
## ARIMA(5,0,0) with zero mean
##
## Coefficients:
##          ar1     ar2     ar3     ar4     ar5
##       0.5200  0.1714  0.1164  0.0998  0.0801
## s.e.  0.0315  0.0354  0.0356  0.0354  0.0316
##
## sigma^2 = 1.005:  log likelihood = -1420.45
## AIC=2852.89   AICc=2852.97   BIC=2882.34
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE     MASE
## Training set 0.03754006 0.9998436 0.7821971 -11.35022 94.47374 0.889629
##                       ACF1
## Training set -0.001189451
```

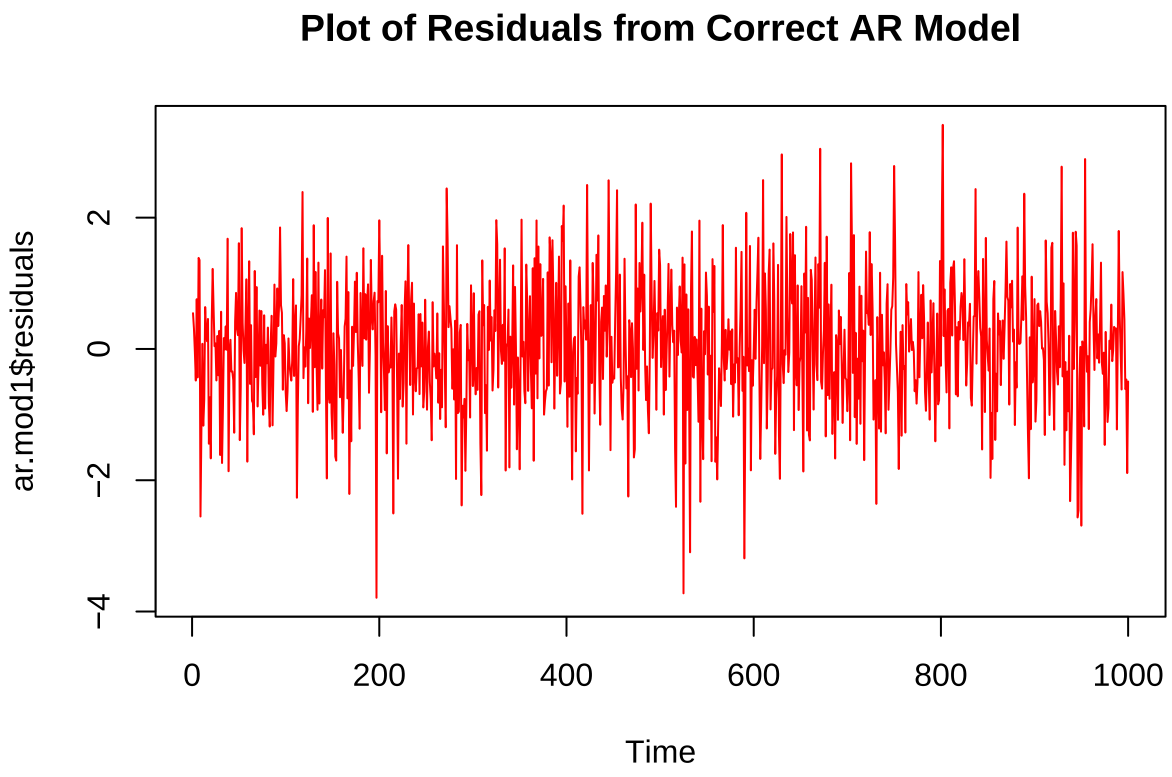
Let's compare our the true model coefficients with the estimates:

```
coef.compare1 <- data.frame(true = ar.coef, estimate = ar.mod1$coef)
coef.compare1
```

```
##           true   estimate
## ar1 0.53020433 0.51995227
## ar2 0.16202103 0.17136690
## ar3 0.12355394 0.11643306
## ar4 0.09399246 0.09979314
## ar5 0.07397852 0.08013919
```

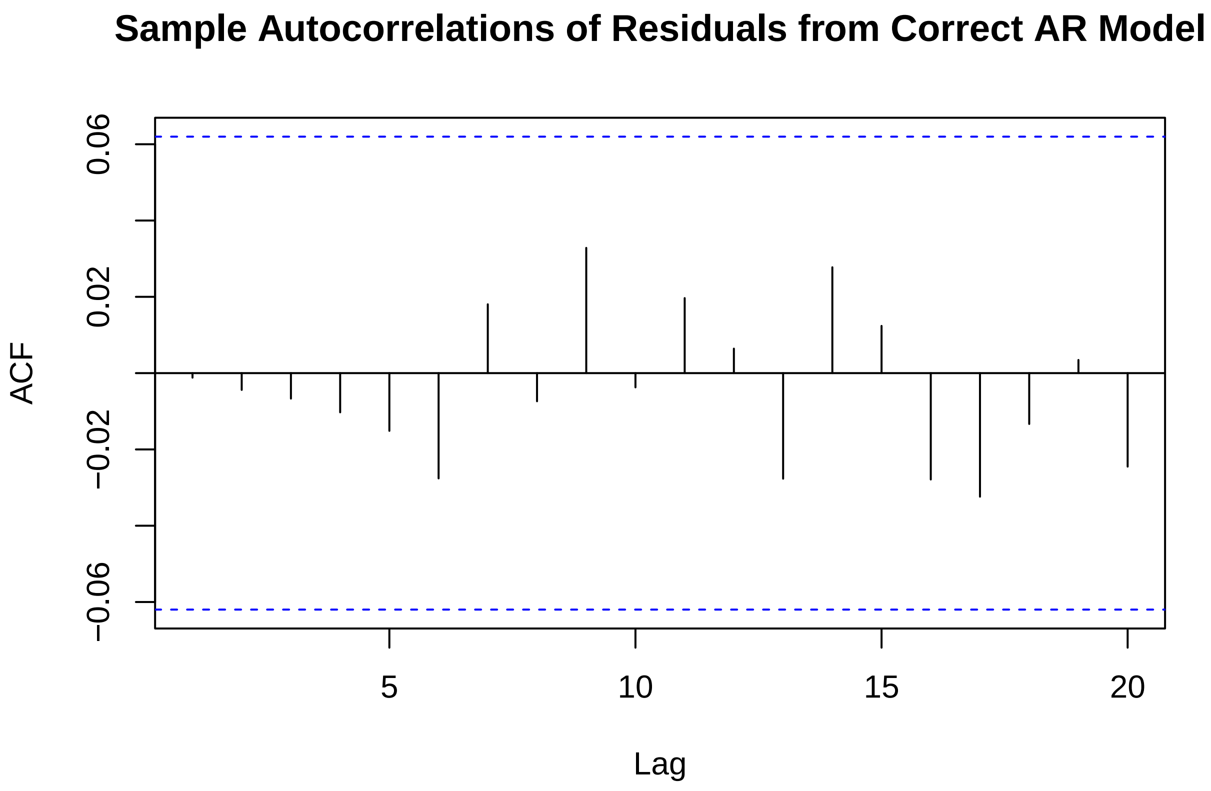
Let's have a look at the residuals from our estimated model:

```
plot(ar.mod1$residuals,
     main = "Plot of Residuals from Correct AR Model",
     col = "red")
```

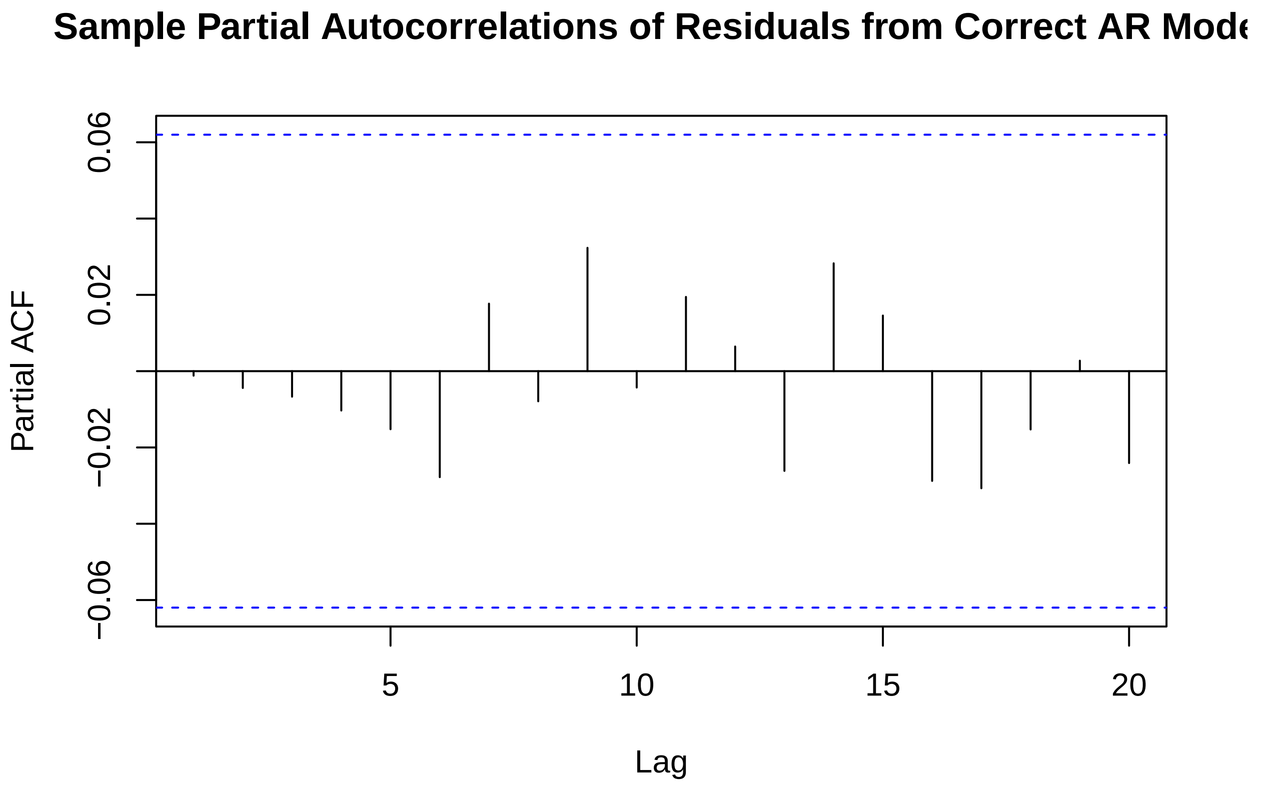


```
ar.res.acf1 <- acf(ar.mod1$residuals, plot = FALSE)
ar.res.pacf1 <- pacf(ar.mod1$residuals, plot = FALSE)

plot(ar.res.acf1[1:20], main = "Sample Autocorrelations of Residuals from Correct AR Model")
```



```
plot(ar.res.pacf1[1:20], main = "Sample Partial Autocorrelations of Residuals from Correct AR Model")
```



We can perform our portmanteau tests to confirm that the residuals have no dependence structure:

```
m = sqrt(T)

Box.test(ar.mod1$residuals, lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  ar.mod1$residuals
## X-squared = 14.034, df = 31.623, p-value = 0.9971
```

```
Box.test(ar.mod1$residuals, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ar.mod1$residuals
## X-squared = 14.34, df = 31.623, p-value = 0.9964
```

Now let's suppose that we fit an incorrectly specified model to our data. Let's say that we try to fit an AR(2) model:

```
ar.mod2 <- Arima(ar.sim, order = c(2,0,0), include.mean = FALSE)
summary(ar.mod2)
```

```
## Series: ar.sim
## ARIMA(2,0,0) with zero mean
##
## Coefficients:
```

```
##          ar1     ar2
##       0.6345  0.3463
## s.e.  0.0296  0.0296
##
## sigma^2 = 1.077:  log likelihood = -1456.59
## AIC=2919.19   AICc=2919.21   BIC=2933.91
##
## Training set error measures:
##                      ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.05460815 1.036763 0.8145056 -14.95612 117.9202 0.9263751
##                     ACF1
## Training set -0.07543683
```

The first thing that we notice is that the AIC from our wrongly specified model is larger than the correctly specified one.

```
ar.mod1$aic
```

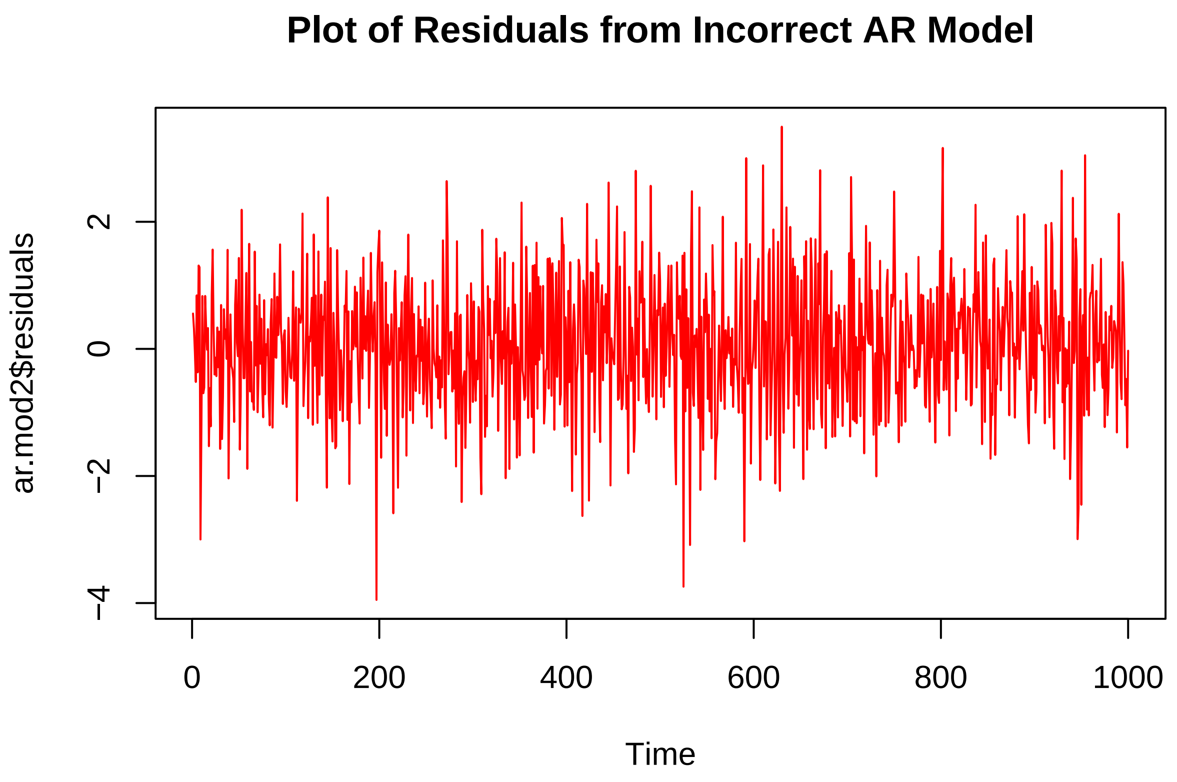
```
## [1] 2852.89
```

```
ar.mod2$aic
```

```
## [1] 2919.188
```

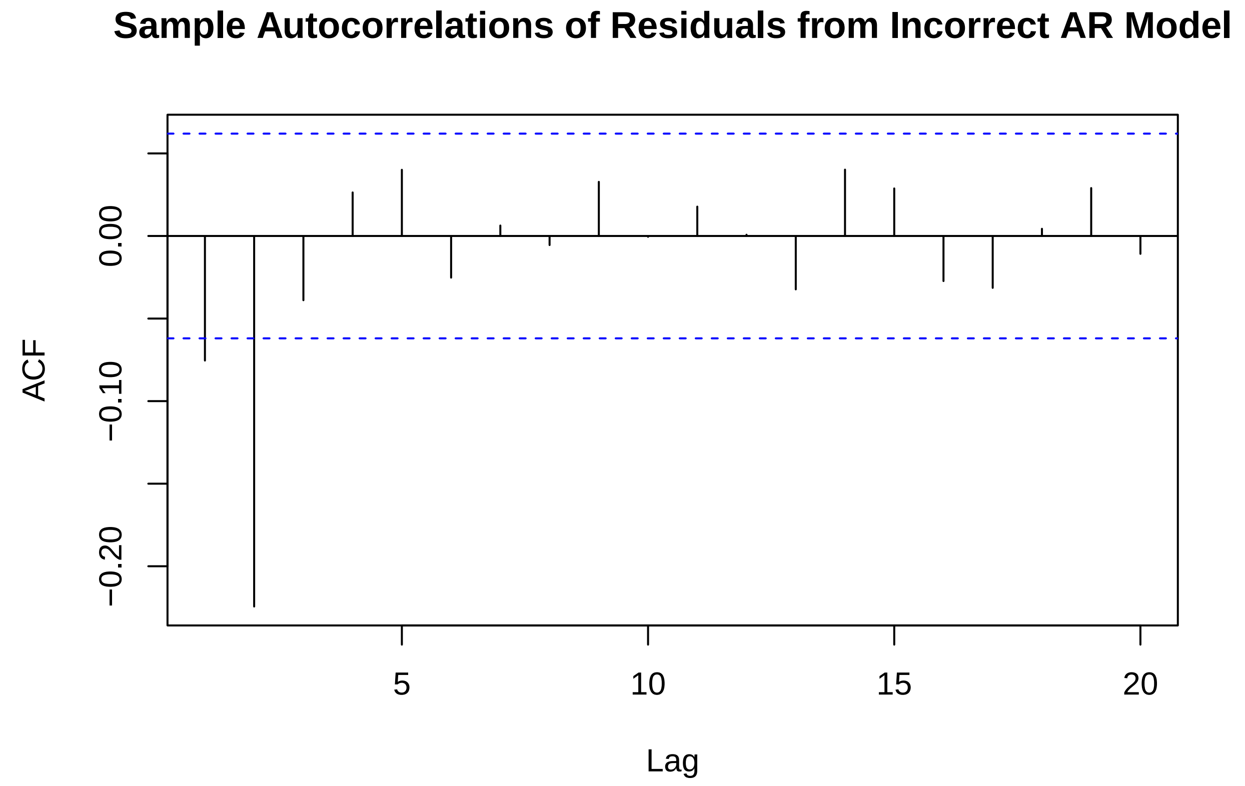
If we have a look at the residuals from the incorrect model and their associated sample autocorrelations and partial autocorrelations we can clearly see that there exists some dependence:

```
plot(ar.mod2$residuals,
     main = "Plot of Residuals from Incorrect AR Model",
     col = "red")
```

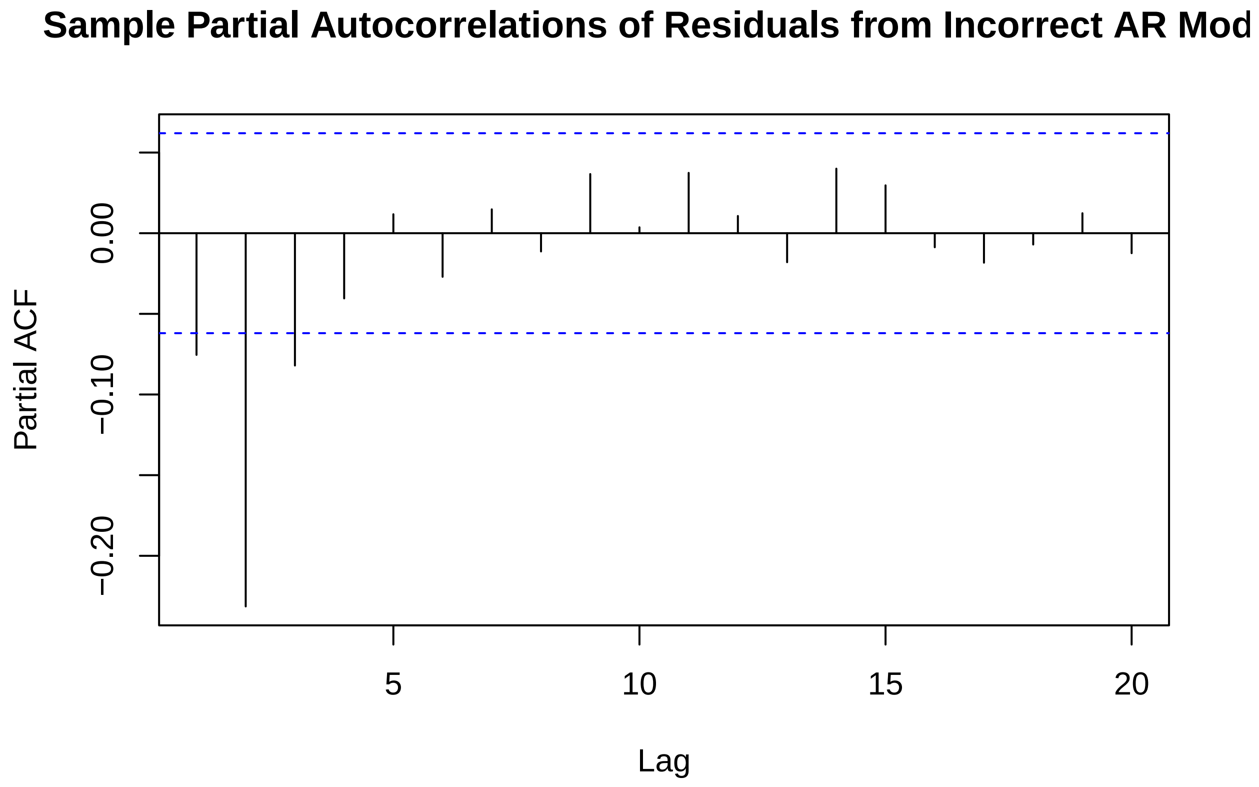


```
ar.res.acf2 <- acf(ar.mod2$residuals, plot = FALSE)
ar.res.pacf2 <- pacf(ar.mod2$residuals, plot = FALSE)
```

```
plot(ar.res.acf2[1:20], main = "Sample Autocorrelations of Residuals from Incorrect AR Model")
```



```
plot(ar.res.pacf2[1:20], main = "Sample Partial Autocorrelations of Residuals from Incorrect AR Model")
```



We can verify our visual impression using our portmanteau tests:

```
Box.test(ar.mod2$residuals, lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  ar.mod2$residuals
## X-squared = 78.4, df = 31.623, p-value = 7.56e-06
```

```
Box.test(ar.mod2$residuals, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ar.mod2$residuals
## X-squared = 79.084, df = 31.623, p-value = 6.079e-06
```

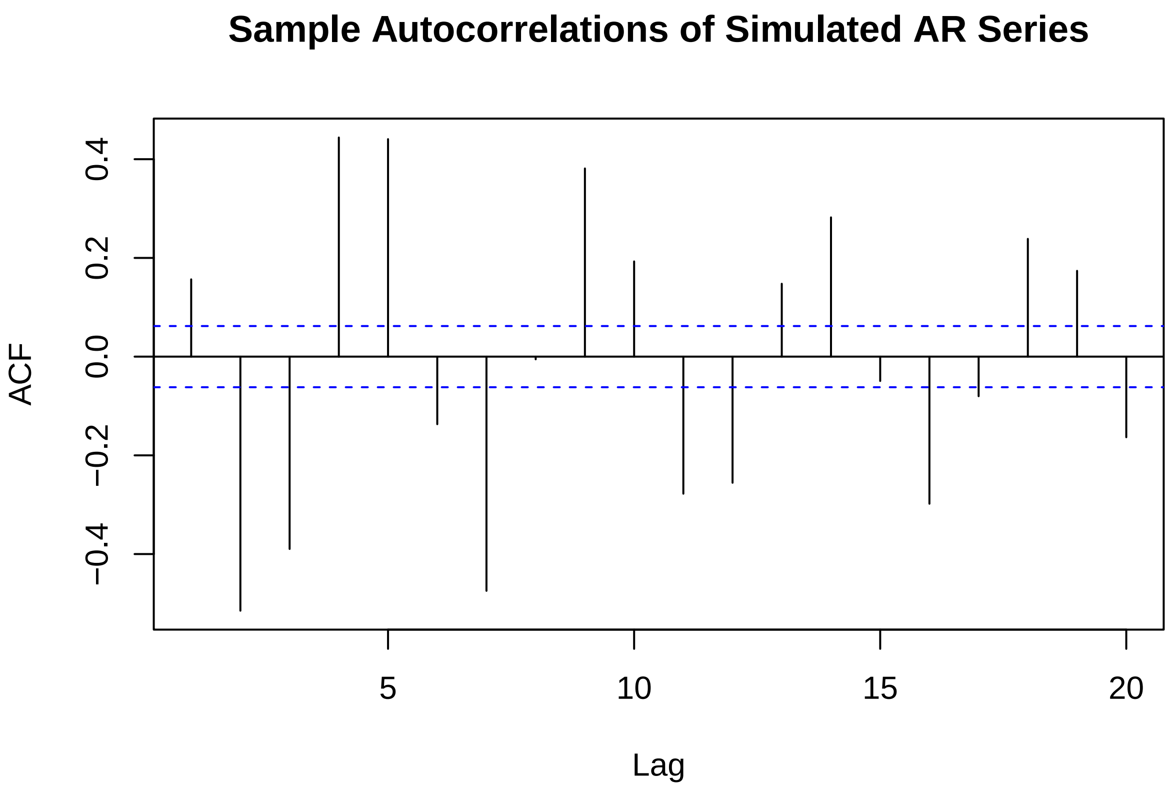
Yule-Walker Equations

Let's see how we can compute AR coefficient estimates using Yule-Walker equations. Let's start by simulating some observations from an AR(4):

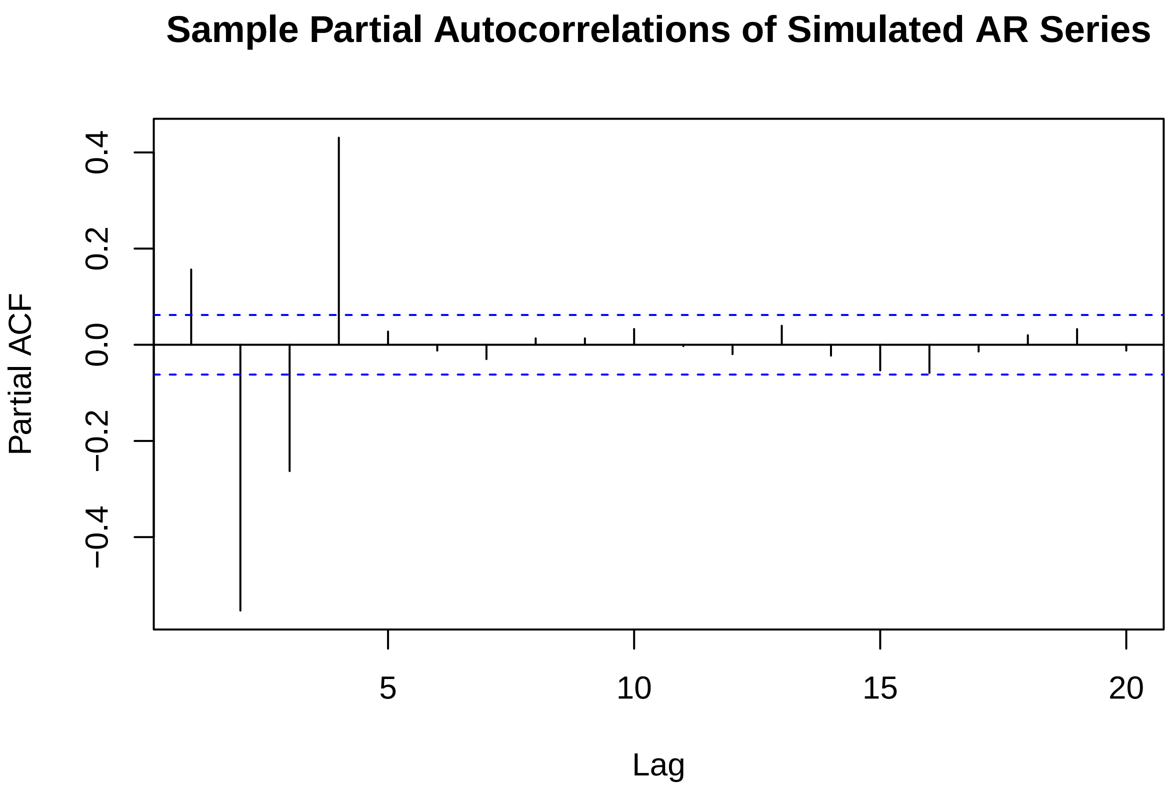
```
z <- arima.sim(n = T, list(ar=c(0.2, -0.3, -0.3, 0.4)))

z.acf <- acf(z, plot = FALSE)
z.pacf <- pacf(z, plot = FALSE)

plot(z.acf[1:20], main = "Sample Autocorrelations of Simulated AR Series")
```



```r
plot(z.pacf[1:20], main = "Sample Partial Autocorrelations of Simulated AR Series")
```



We want to create the following vectors and matrices:

$$\rho = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_p \end{bmatrix}, \qquad P = \begin{bmatrix} \rho_0 & \rho_1 & \cdots & \rho_{p-1} \\ \rho_1 & \rho_0 & \cdots & \rho_{p-2} \\ \vdots & \ddots & \ddots & \vdots \\ \rho_{p-1} & \rho_{p-2} & \cdots & \rho_0 \end{bmatrix}$$

First we define the matrices P and *rho*. To create P we will need to use the **VecRot()** function in the **DescTools** package:

```r
if("DescTools" %in% rownames(installed.packages()) == FALSE) install.packages("DescTools")
library(DescTools)
```

```
##
## Attaching package: 'DescTools'

## The following object is masked from 'package:forecast':
##
##     BoxCox
```

```r
p <- 4 #Order of the AR

Rho <- z.acf$acf[1:(p+1)]

P <- matrix(0,p,p)
for(i in 1:p){
  P[i,] = VecRot(Rho[1:p],k=(i-1))
}
```

```r
P[lower.tri(P)] <- 0

P <- P + t(P) - diag(p)

rho <- Rho[2:(p+1)]
```

Then, we compute

$$\Phi = P^{-1}\rho$$

```r
Pinv <- solve(P)

Phi <- Pinv %*% rho
```

Let's now compare these estimates with the ones produced by the **Arima()** function:

```r
ar.mod3 <- Arima(z, order = c(4,0,0), include.mean = FALSE)

coef.compare2 <- data.frame(true = c(0.2,-0.2,-0.2,0.4),yulewalker = Phi, arma = ar.mod3$coef)

coef.compare2
```

```
##     true yulewalker       arma
## ar1  0.2  0.2110482  0.2156473
## ar2 -0.2 -0.2783801 -0.2762637
## ar3 -0.2 -0.3048806 -0.3041951
## ar4  0.4  0.4306292  0.4366286
```