STAT 720 sp 2019 Lec 06

Karl Gregory

2/15/2019

This lecture will make use of the `tscourse` package, which is installed with the following R code:

```
library(devtools)
devtools::install_github("gregorkb/tscourse")
library(tscourse)
```

Estimation of ARMA coefficients

Let $\{X_t, t \in \mathbb{Z}\}$ be the zero-mean causal invertible ARMA(p, q) process satisfying

$$X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}, \tag{1}$$

where $\{Z_t, t \in \mathbb{Z}\}$ is WN$(0, \sigma^2)$. We consider estimating $\phi_1, \ldots, \phi_p$, $\theta_1, \ldots, \theta_q$, and σ^2 based on a length-n realization $X_1, \ldots, X_n$ of the time series. We first consider estimation in the AR(p) and MA(q) models as special cases.

Yule-Walker estimation of AR(p) parameters

Let $\{X_t, t \in \mathbb{Z}\}$ be the zero-mean causal AR(p) process satisfying

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + Z_t, \tag{2}$$

where $\{Z_t, t \in \mathbb{Z}\}$ is WN$(0, \sigma^2)$, and define

$$\boldsymbol{\Gamma}_p = (\gamma(i-j))_{1 \leq i,j \leq p} \quad \text{and} \quad \boldsymbol{\gamma}_p = (\gamma(1), \ldots, \gamma(p))^T,$$

where $\gamma(\cdot)$ is the autocovariance function of the process $\{X_t, t \in \mathbb{Z}\}$. We discuss the Yule-Walker method of estimating the parameters $\phi_1, \ldots, \phi_p$ and σ^2.

The Yule-Walker equations

Let $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_p)^T$. Then we have the relations

$$\boldsymbol{\Gamma}_p \boldsymbol{\phi} = \boldsymbol{\gamma}_p, \quad \text{and} \quad \sigma^2 = \gamma(0) - \boldsymbol{\phi}^T \boldsymbol{\gamma}_p,$$

and these are called the Yule-Walker equations. We can derive the Yule-Walker equations as follows: Multiplying both sides of (2) by X_{t-j} and taking expectations gives

$$X_{t-j} X_t = \phi_1 X_{t-j} X_{t-1} + \cdots + \phi_p X_{t-j} X_{t-p} + X_{t-j} Z_t$$

for $j = 1, \ldots, p$ and

$$X_t^2 = \phi_1 X_t X_{t-1} + \cdots + \phi_p X_t X_{t-p} + X_t Z_t$$

for $j = 0$. We now take expectations on both sides of these equations, noting that because of causality we have $X_{t-j}Z_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j} Z_t$ for some sequence $\{\psi_j, j = 0, 1, 2, \dots\}$, so that $\mathbb{E}X_{t-j}Z_t = 0$ for $j = 1, \dots, p$ and $\mathbb{E}X_t Z_t = \sigma^2$. This gives

$$\gamma(j) = \phi_1\gamma(j-1) + \cdots + \phi_p\gamma(j-p)$$

for $j = 1, \dots, p$, which can be written as $\boldsymbol{\Gamma}_p\boldsymbol{\phi} = \boldsymbol{\gamma}_p$, and

$$\gamma(0) = \phi_1\gamma(1) + \cdots + \phi_p\gamma(p) + \sigma^2$$

for $j = 0$, from which we have $\sigma^2 = \gamma(0) - \boldsymbol{\phi}^T\boldsymbol{\gamma}_p$.

The Yule-Walker estimators

The Yule-Walker estimators $\hat{\boldsymbol{\phi}}$ of $\boldsymbol{\phi}$ and $\hat{\sigma}^2$ of σ^2 are the values of $\boldsymbol{\phi}$ and σ^2 which satisfy the Yule-Walker equations when the true autocovariance function $\gamma(\cdot)$ is replaced by the sample autocovariance function $\hat{\gamma}_n(\cdot)$. Defining $\hat{\boldsymbol{\Gamma}}_p = (\hat{\gamma}_n(i-j))_{1\le i,j\le p}$ and $\hat{\boldsymbol{\gamma}}_p = (\hat{\gamma}_n(1), \dots, \hat{\gamma}_n(p))^T$, the Yule-Walker estimators $\hat{\boldsymbol{\phi}}$ and $\hat{\sigma}^2$ are defined by the equations

$$\hat{\boldsymbol{\Gamma}}_p\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\gamma}}_p \quad \text{and} \quad \hat{\sigma}^2 = \hat{\gamma}(0) - \hat{\boldsymbol{\phi}}^T\hat{\boldsymbol{\gamma}}_p. \tag{3}$$

Using the Durbin-Levinson algorithm to compute the Yule-Walker estimators

Recall that the Durbin-Levinson algorithm computes the vectors $\mathbf{a}_1, \mathbf{a}_2, \dots$ which satisfy the equations $\boldsymbol{\Gamma}_1\mathbf{a}_1 = \boldsymbol{\gamma}_1, \boldsymbol{\Gamma}_2\mathbf{a}_2 = \boldsymbol{\gamma}_2, \dots$ Therefore to obtain the Yule-Walker estimator $\hat{\boldsymbol{\phi}}$ of $\boldsymbol{\phi}$, we simply need to plug the sample autocovariance function into the Durbin-Levinson algorithm, and it will produce the vector $\hat{\boldsymbol{\phi}}$ which satisfies $\hat{\boldsymbol{\Gamma}}_p\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\gamma}}_p$. Then we compute $\hat{\sigma}^2$ according to (3). The R code below carries this out on a simulated data set.

```
phi <- c(.8,.3,-.2)
# make sure the process is causal and stationary by checking whether the polynomial
# phi() has any roots with modulus less than or equal to 1:
min(Mod(polyroot(c(1,-phi)))) <= 1
```

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

```
n <- 500
B <- 1000
p <- length(phi)
X0 <- numeric(B+n)
X0[1:p] <- 0

for(i in (p+1):(B+n))
{

    X0[i] <- sum(phi * X0[(i-1):(i-p)]) + rnorm(1)

}

X <- X0[-c(1:B)]

gamma.hat <- sample.acf(X)$gamma.hat
gamma.0.hat <- gamma.hat[1]
gamma.n.hat <- gamma.hat[-1]
DL.1step.out <- DL.1step(X-mean(X),gamma.0.hat,gamma.n.hat)
```

```r
# get YW estimator from the DL alg.
phi.hat <- DL.1step.out$Phi[1+p,p:1]
sigma.sq.hat <- gamma.0.hat - sum(gamma.n.hat[1:p]*phi.hat)
```

The above code generated a time series of length $n = 500$ from an AR(3) time series with $\boldsymbol{\phi} = (0.8, 0.3, -0.2)^T$ and $\sigma^2 = 1$. The Yule-Walker estimators of $\boldsymbol{\phi}$ and σ^2 were $\hat{\boldsymbol{\phi}} = (0.8134496, 0.3084123, -0.2123889)^T$ and $\hat{\sigma}^2 = 1.0213094$.

Asymptotic distribution of Yule-Walker estimators for AR(p) model

Under the model in (2), we have

$$\sqrt{n}(\hat{\boldsymbol{\phi}} - \boldsymbol{\phi}) \to \text{Normal}(\mathbf{0}, \sigma^2 \boldsymbol{\Gamma}_p^{-1}) \quad \text{in distribution}$$

and

$$\hat{\sigma}^2 \to \sigma^2 \quad \text{in probability}$$

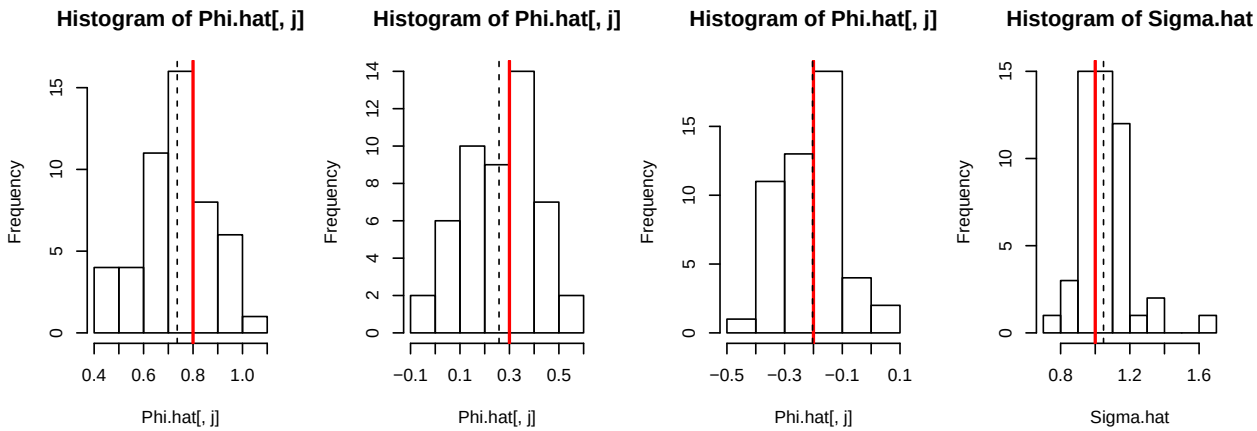
as $n \to \infty$ (see Theorem 8.1.1 of B&D Theory). The following R code runs a simulation which produces many realizations of the Yule-Walker estimators.

```r
phi <- c(.8,.3,-.2)
min(Mod(polyroot(c(1,-phi)))) <= 1
```

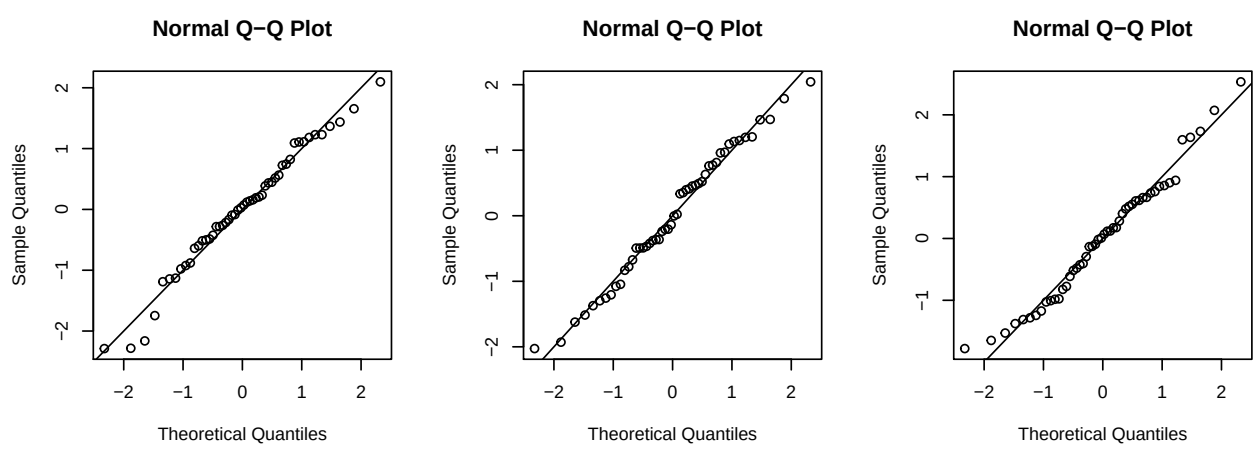
```
## [1] FALSE
```

```r
n <- 50
B <- 500
p <- length(phi)
X0 <- numeric(B+n)

S <- 50
Phi.hat <- matrix(NA,S,p)
Sigma.hat <- numeric(S)
for(s in 1:S)
{

    X0[1:p] <- 0

    for(i in (p+1):(B+n))
    {

        X0[i] <- sum(phi * X0[(i-1):(i-p)]) + rnorm(1)

    }

    X <- X0[-c(1:B)]

    gamma.hat <- sample.acf(X)$gamma.hat
    gamma.0.hat <- gamma.hat[1]
    gamma.n.hat <- gamma.hat[-1]
    DL.1step.out <- DL.1step(X-mean(X),gamma.0.hat,gamma.n.hat)

    Phi.hat[s,] <- DL.1step.out$Phi[1+p,p:1]
    Sigma.hat[s] <- sqrt( gamma.0.hat - sum(gamma.n.hat[1:p]*Phi.hat[s,] ))
```

```r
}

par(mfrow=c(1,4))
for(j in 1:p)
{
    hist(Phi.hat[,j])
    abline(v=phi[j],col="red",lwd=2)
    abline(v=mean(Phi.hat[,j]),lty=2)
}
hist(Sigma.hat)
abline(v=1,col="red",lwd=2)
abline(v = mean(Sigma.hat),lty=2)
```




```r
par(mfrow=c(1,3))
for(j in 1:p)
{
    qqnorm(scale(Phi.hat[,j]))
    abline(0,1)

}
```



Innovations estimation of MA(q) parameters

For any time series $\{X_t, t \in \mathbb{Z}\}$ with autocovariances given by $\kappa(i,j) = \mathbb{E}X_iX_j$, $i,j \in \mathbb{Z}$, recall that the innovations algorithm based on $\kappa(\cdot,\cdot)$ returns $\{\theta_{m,1},\ldots,\theta_{m,m}, m = 1,2,\ldots,n\}$ such that the one-step-ahead

predictions $\hat{X}_1, \ldots, \hat{X}_{n+1}$ which minimize the expected squared prediction error (MSEP) can be recursively defined as

$$\hat{X}_{m+1} = \begin{cases} 0, & m = 0 \\ \sum_{i=1}^{m} \theta_{m,i}(X_{m+1-i} - \hat{X}_{m+1-i}), & m = 1, \ldots, n. \end{cases}$$

Moreover, innovations algorithm returns the MSEPs $v_0, v_1, \ldots, v_n$ of the predictors $\hat{X}_1, \ldots, \hat{X}_{n+1}$.

Now, let $\{\hat{\theta}_{m,1}, \ldots, \hat{\theta}_{m,m}, m = 1, \ldots, n\}$ and $\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_n$ be the values corresponding to $\{\theta_{m,1}, \ldots, \theta_{m,m}, m = 1, 2, \ldots, n\}$ and $v_0, v_1, \ldots, v_n$ returned by the innovations algorithm when it is based on the sample autocovariances, that is, under $\kappa(i,j) = \hat{\gamma}_n(i-j)$ for $i, j \in \mathbb{Z}$. It turns out that we can find among the values $\{\hat{\theta}_{m,1}, \ldots, \hat{\theta}_{m,m}, m = 1, \ldots, n\}$ consistent estimators of the parameters of an MA(q) model. In particular, we have the following result:

Asymptotic distribution of innovations estimators for MA(q) model

Let

$$X_t = Z_t + \theta_1 Z_t + \cdots + \theta_q Z_{t-q},$$

where $\{Z_t, t \in \mathbb{Z}\}$ is WN$(0, \sigma^2)$ and $\mathbb{E}Z_t^4 < \infty$ and suppose that $\{X_t, t \in \mathbb{Z}\}$ is invertible. Define the $q \times 1$ vectors $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_q)^T$ and $\hat{\boldsymbol{\theta}}_m = (\hat{\theta}_{m,1}, \ldots, \hat{\theta}_{m,q})^T$ for $m = 1, \ldots, n$. Then for any sequence $m = m(n)$ such that $m \to \infty$ and $m/n^{1/3} \to 0$ as $n \to \infty$ we have

$$\sqrt{n}(\hat{\boldsymbol{\theta}}_m - \boldsymbol{\theta}) \to \text{Normal}(\mathbf{0}, \mathbf{A}) \quad \text{in distribution}$$

as $n \to \infty$, where

$$\mathbf{A} = \left(\sum_{r=1}^{\min(i,j)} \theta_{i-r}\theta_{j-r} \right)_{1 \le i,j \le q}, \quad (\text{defining } \theta_0 := 1).$$

Moreover, $\hat{v}_m \to \sigma^2$ in probability as $n \to \infty$.

Innovations-based estimation of MA(q) model in practice

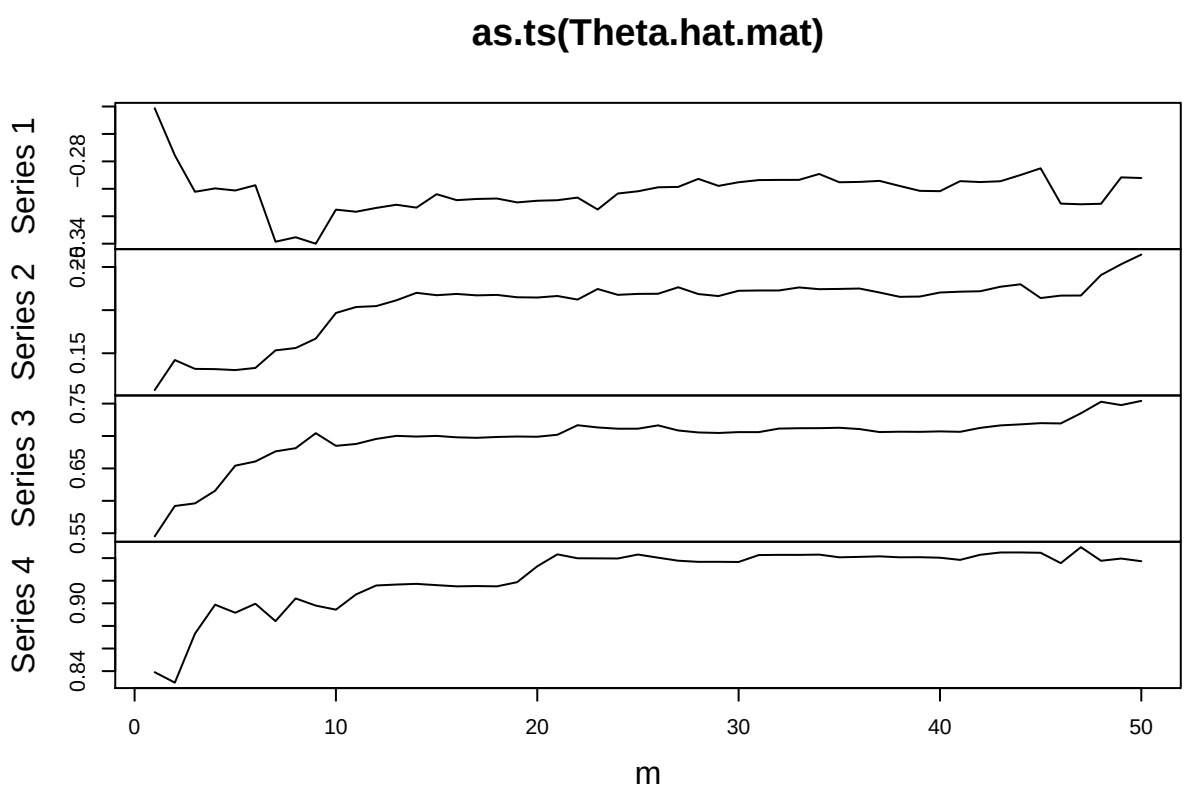
Applying the above result in practice requires that we choose a value of m. One way to choose a value of m is to plot the estimates $\hat{\theta}_{m,1}, \ldots, \hat{\theta}_{m,q}$ for $m = q, \ldots, \lfloor n/4 \rfloor$, for example, and to choose, say, the smallest values of m for which the estimates appear to have stabilized. The following R code carries this out on a simulated data set.

```
# generate MA(q) data:
n <- 200
theta <- c(.8,.6,.1,-.3)

# make sure process is invertible by checking whether the polynomial theta() has
# any roots with modulus less than or equal to 1.
min(Mod(polyroot(c(1,theta)))) <= 1
```

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

```
q <- length(theta)
Z <- rnorm(n+q,0,1)
X <- numeric(n)

for( t in 1:n)
{
    ind <- q + t:(t-q)
    X[t] <- sum( c(1,theta) * Z[ind] )
```

```r
}

# compute sample autocovariances and build matrix of autocovariances
gamma.hat <- sample.acf(X,max.lag=n)$gamma.hat

K.hat <- matrix(NA,n+1,n+1)
for(j in 1:(n+1))
    for(i in 1:(n+1))
    {

        K.hat[i,j] <- c(gamma.hat)[1+abs(i-j)]

    }

# run the innovations algorithm to get theta values
innov.hstep.out <- innov.hstep(X-mean(X),h=1,K.hat)
Theta <- innov.hstep.out$Theta

# extract from Theta matrix the values of interest
Theta.hat.mat <- matrix(NA,floor(n/4),q)
for(i in 1:floor(n/4))
{
    ind <- i:(i+q-1)
    Theta.hat.mat[i,] <- Theta[q+i,ind]
}

# plot columns of Theta.hat.mat against m
plot(as.ts(Theta.hat.mat),xlab="m")
```



```r
# choose  m = 40
m <- 40
theta.hat <- Theta.hat.mat[m,q:1]
sigma.hat <- sqrt(innov.hstep.out$v[1+m])
```

In the above, a length-200 realization of an MA(4) process with moving average coefficients $\boldsymbol{\theta} = (0.8, 0.6, 0.1, -0.3)^T$ is generated and the innovations estimator $\hat{\boldsymbol{\theta}}_m$ of $\boldsymbol{\theta}$ is computed, yielding

$$\hat{\boldsymbol{\theta}}_m = (0.9403631, 0.7071278, 0.2204064, -0.3017284)^T$$

under the choice $m = 40$. The innovations estimator $\hat{v}_m$ of $\sigma^2 = 1$ is 0.9876714.

Preliminary estimation of ARMA(p, q) parameters

For a causal invertible ARMA(p, q) process $\{X_t, t \in \mathbb{Z}\}$, our strategy for estimating the parameters $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ will be the following: Firstly, we exploit the causality property, which allows us to express the time series as a MA(∞) model with some coefficients $\psi_0, \psi_1, \psi_2, \ldots$, of which we estimate $\psi_1, \ldots, \psi_{p+q}$ using the innovations algorithm approach, treating the data as having come from a MA$(p+q)$ model (the invertibility property makes this possible). Secondly, we transform our estimates of $\psi_0, \psi_1, \ldots, \psi_{p+q}$ into estimates of $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$, which we can do by noting that the coefficients of the MA(∞) representation of the time series satisfy $\psi_0 = 1$ and

$$\psi_j = \theta_j + \sum_{l=1}^{\min(j,p)} \phi_l \psi_{j-l} \text{ for } j = 1, \ldots, q+p,$$

where $\theta_j := 0$ for $j > q$ and $\psi_j := 0$ for $j < 0$ (see Lec 05 notes). We find that the above equations for $j = q+1, \ldots, q+p$ can be expressed as

$$\begin{bmatrix} \psi_{q+1} \\ \psi_{q+2} \\ \vdots \\ \psi_{q+p} \end{bmatrix} = \begin{bmatrix} \psi_q & \psi_{q-1} & \cdots & \psi_{q+1-p} \\ \psi_{q+1} & \psi_q & \cdots & \psi_{q+2-p} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{q+p-1} & \psi_{q+p-2} & \cdots & \psi_q \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix},$$

so that we can obtain $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ from $\psi_1, \ldots, \psi_{p+q}$ by

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix} = \begin{bmatrix} \psi_q & \psi_{q-1} & \cdots & \psi_{q+1-p} \\ \psi_{q+1} & \psi_q & \cdots & \psi_{q+2-p} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{q+p-1} & \psi_{q+p-2} & \cdots & \psi_q \end{bmatrix}^{-1} \begin{bmatrix} \psi_{q+1} \\ \psi_{q+2} \\ \vdots \\ \psi_{q+p} \end{bmatrix},$$

and then

$$\theta_j = \psi_j - \sum_{l=1}^{\min(j,p)} \phi_l \psi_{j-l} \text{ for } j = 1, \ldots, q.$$

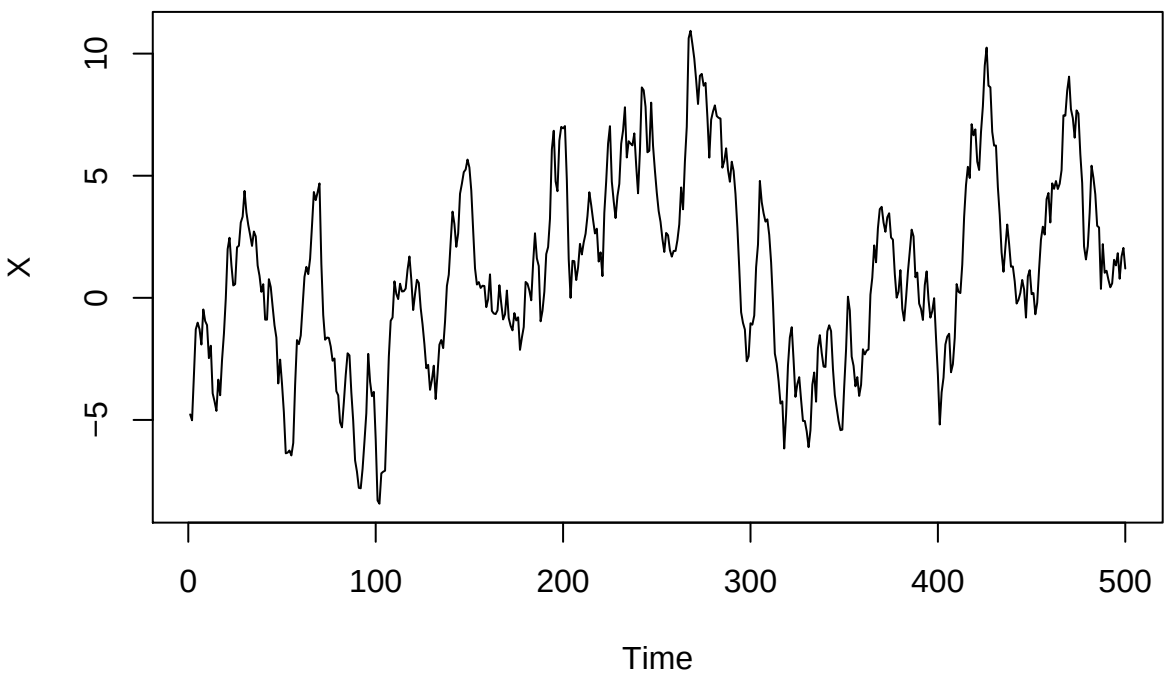
Simulating data from an ARMA(p, q) process

Before we can do simulations to illustrate preliminary estimation of ARMA(p, q) models, we need to be able to generate realizations of ARMA(p, q) processes. We will exploit the MA(∞) representation of the process, which we owe to the assumption of causality. The preceding section gave a recipe for recursively computing the coefficients in the MA(∞) representation of the time series in terms of the ARMA(p, q) parameters $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$; for given values of these parameters, we simply find the coefficients $\psi_0, \psi_1, \psi_2, \ldots$ of the MA(∞) representation according to this recipe, truncate them such that $\psi_j = 0$ for $j > T$ for some

large T, and generate a realization from the corresponding MA(T) process. The following R code defines a function which carries out these steps.

```
# first define a function to give the coefficients of
# a truncated MA(inf) representation of the ARMA(p,q) series.
ARMAtoMAinf <- function(phi=NULL,theta=NULL,trun=500)
{

    if(length(phi)==0)
    {
        q <- length(theta)
        psi <- numeric(trun)
        psi[1:(q+1)] <- c(1,theta)

    } else if(length(phi)>0)
    {
        # check to see if the time series is causal:
        minroot <- min(Mod(polyroot(c(1,-phi))))
        if( minroot < 1)
            stop("The ARMA process specified is not causal.")

        p <- length(phi)
        q <- length(theta)

        # set theta_j = 0 for j > q
        theta.0 <- c(theta,rep(0,trun-q))

        # set psi_j = 0 for j < 0
        psi.0 <- numeric(trun+p)
        psi.0[p] <- 1 # this is psi_0

        for(j in 1:trun)
        {

            psi.0[p+j] <- theta.0[j] + sum( phi[1:p] * psi.0[(p+j-1):j] )

        }

        # take away zeroes at beginning
        psi <- psi.0[p:(p+trun)]

    }

    return(psi)

}

# now define a function to generate ARMA(p,q) data
get.ARMA.data <- function(phi,theta,sigma,n,trun=500)
{

    # check to see if the time series is causal:
    minroot <- min(Mod(polyroot(c(1,-phi))))
    if( minroot < 1)
```

```r
        stop("The ARMA process specified is not causal.")

    minroot <- min(Mod(polyroot(c(1,theta))))
    if( minroot < 1)
        stop("The ARMA process specified is not invertible.")

    psi <- ARMAtoMAinf(phi,theta,trun=trun)

    Z <- rnorm(n+trun,0,sigma)
    X <- numeric(n)

    for( t in 1:n)
    {
        ind <- trun + t:(t-trun)
        X[t] <- sum( psi * Z[ind] )
    }

    return(as.ts(X))

}

phi <- c(.4,.4,.1)
theta <- c(.8,.3)
sigma <- 1
n <- 500

X <- get.ARMA.data(phi,theta,sigma,n)
plot(X)
```



Implementation of preliminary ARMA(p, q) estimation

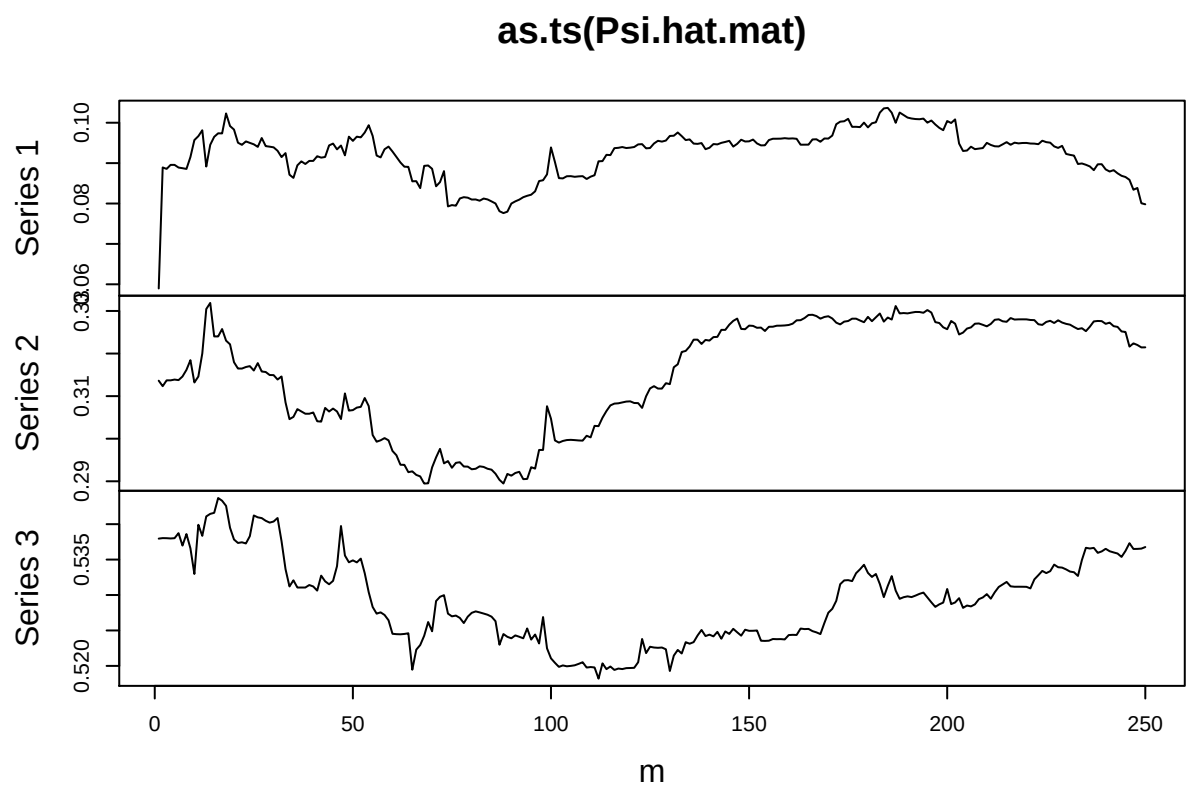
In order to carry out preliminary estimation of ARMA(p, q) models, we define the following function in R which transforms a set of moving average coefficients $\psi_0, \psi_1, \ldots, \psi_{p+q}$ into the corresponding coefficients $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ of the equivalent causal ARMA(p, q) model:

```
MAtoARMA <- function(psi,p,q)
{

    Psi.qp <- matrix(0,p,p)
    for( i in 1:p)
        for(j in 1:p)
        {
            if(q+i-j > 0)
            {
                Psi.qp[i,j] <- psi[1+q+i-j]

            } else if(q+i-j == 0){

                Psi.qp[i,j] <- 1

            }
        }

    phi <- as.numeric(solve( Psi.qp) %*% psi[1+(q+1):(q+p)])

    psi.0 <- c(rep(0,p-1),psi)
    theta <- numeric()

    for( j in 1:q)
    {

        theta[j] <- psi.0[p+j] - sum( phi[1:p] * psi.0[(p+j-1):j] )

    }

    output <- list( phi = phi,
                    theta = theta)

    return(output)

}
```

Now the following R code generates some ARMA(p, q) data and computes the preliminary estimators as described:

```
# generate data
phi <- c(.2)
theta <- c(.3,.2)
sigma <- 1
n <- 500
X <- get.ARMA.data(phi,theta,sigma,n)

# get sample autocovariances
gamma.hat <- sample.acf(X,max.lag=n)$gamma.hat
```

```r
K.hat <- matrix(NA,n+1,n+1)
for(j in 1:(n+1))
    for(i in 1:(n+1))
    {

        K.hat[i,j] <- c(gamma.hat)[1+abs(i-j)]

    }

# run the innovations algorithm on centered data
innov.hstep.out <- innov.hstep(X-mean(X),h=1,K.hat)
Psi <- innov.hstep.out$Theta

# extract the values of interest
p <- length(phi)
q <- length(theta)
Psi.hat.mat <- matrix(NA,floor(n/2),p+q)
for(i in 1:floor(n/2))
{
    ind <- i:(i+p+q-1)
    Psi.hat.mat[i,] <- Psi[p+q+i,ind]
}

# plot columns of Psi.hat.mat against m
plot(as.ts(Psi.hat.mat),xlab="m")
```




```r
# choose m
m <- 200
```

```r
psi.hat <- c(1,Psi.hat.mat[m,(p+q):1])

# pull true values of psi for comparison
psi <- ARMAtoMAinf(phi,theta,p+q)

# print estimates and true values of MA parameters
psi.hat
```

```
## [1] 1.0000000 0.5308491 0.3257188 0.1004210
```

```r
psi
```

```
## [1] 1.00 0.50 0.30 0.06
```

```r
# convert the estimates to the ARMA coefficients
MAtoARMA(psi.hat,p,q)
```

```
## $phi
## [1] 0.3083059
##
## $theta
## [1] 0.2225433 0.1620549
```

```r
MAtoARMA(psi,p,q) # check to make sure function works :)
```

```
## $phi
## [1] 0.2
##
## $theta
## [1] 0.3 0.2
```

Maximum likelihood estimation of ARMA(p, q) parameters

We now assume that $\{X_t, t \in \mathbb{Z}\}$ is a Gaussian process with mean zero and with covariances $\kappa(i,j) = \mathbb{E}X_iX_j$, $i,j \in \mathbb{Z}$. Consider a length-n realization $X_1, \ldots, X_n$ of the time series and define the vector $\mathbf{X}_n = (X_1, \ldots, X_n)^T$ as well as the matrix $\mathbf{\Gamma}_n = \mathbb{E}\mathbf{X}_n\mathbf{X}_n^T$, which is the matrix of covariances among $X_1, \ldots, X_n$. Assuming that $\mathbf{\Gamma}_n$ is nonsingular, the likelihood function in $\mathbf{\Gamma}_n$ based on $X_1, \ldots, X_n$ is given by

$$L_n(\mathbf{\Gamma}_n; \mathbf{X}_n) = (2\pi)^{-n/2}|\mathbf{\Gamma}_n|^{-1/2} \exp\left[-\frac{1}{2}\mathbf{X}_n^T\mathbf{\Gamma}_n^{-1}\mathbf{X}_n\right].$$

The first obstacle to maximum likelihood estimation in time series models is the evaluation of the likelihood function itself, as it involves taking the inverse of the matrix $\mathbf{\Gamma}_n$, which can be computationally burdensome. However, we find that the output of the innovations algorithm can be used to bypass direct inversion of $\mathbf{\Gamma}_n$.

Fast evaluation of the Gaussian likelihood

Let $\mathbf{\Theta}_n$ be the matrix

$$\mathbf{\Theta}_n = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ \theta_{1,1} & 0 & 0 & \ldots & 0 & 0 \\ \theta_{2,2} & \theta_{2,1} & 0 & \ldots & 0 & 0 \\ \theta_{3,3} & \theta_{3,2} & \theta_{3,1} & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ \theta_{n-1,n-1} & \theta_{n-1,n-2} & \theta_{n-1,n-3} & \ldots & \theta_{n-1,1} & 0 \end{bmatrix},$$

where $\{\theta_{m,1}, \ldots, \theta_{m,m}, m = 1 \ldots, n-1\}$ are the values returned by the innovations algorithm based on $\mathbf{\Gamma}_n$ which satisfy

$$\hat{X}_{m+1} = \begin{cases} 0, & m = 0 \\ \sum_{i=1}^{m} \theta_{m,i}(X_{m+1-i} - \hat{X}_{m+1-i}), & m = 1, \ldots, n-1. \end{cases}$$

That is, defining $\hat{\mathbf{X}}_n = (\hat{X}_1, \ldots, \hat{X}_n)^T$, let $\mathbf{\Theta}_n$ be the matrix such that

$$\hat{\mathbf{X}}_n = \mathbf{\Theta}_n(\mathbf{X}_n - \hat{\mathbf{X}}_n).$$

Moreover, let $\mathbf{D}_n = \mathrm{diag}(v_0, v_1, \ldots, v_{n-1})$, where $v_0, v_1, \ldots, v_{n-1}$ are the MSEPs of the predictors $\hat{X}_1, \ldots, \hat{X}_n$. Then we find that $\mathbf{\Gamma}_n$ admits the factorization

$$\mathbf{\Gamma}_n = (\mathbf{\Theta}_n + \mathbf{I}_n)\mathbf{D}_n(\mathbf{\Theta}_n + \mathbf{I}_n)^T,$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix. This gives

$$\mathbf{\Gamma}_n^{-1} = [(\mathbf{\Theta}_n + \mathbf{I}_n)^T]^{-1}\mathbf{D}_n^{-1}(\mathbf{\Theta}_n + \mathbf{I}_n)^{-1},$$

where

$$\mathbf{D}_n^{-1} = \mathrm{diag}(v_0^{-1}, v_1^{-1}, \ldots, v_{n-1}^{-1}).$$

$$\begin{aligned} \mathbf{X}_n &= \mathbf{X}_n - \hat{\mathbf{X}}_n + \hat{X}_n \\ &= \mathbf{X}_n - \hat{\mathbf{X}}_n + \mathbf{\Theta}(\mathbf{X}_n - \hat{\mathbf{X}}_n) \\ &= (\Theta_n + \mathbf{I}_n)(\mathbf{X}_n - \hat{\mathbf{X}}_n). \end{aligned}$$

The above allows us to write

$$\begin{aligned} \mathbf{X}_n^T\mathbf{\Gamma}_n^{-1}\mathbf{X}_n &= [(\Theta_n + \mathbf{I}_n)(\mathbf{X}_n - \hat{\mathbf{X}}_n)]^T\mathbf{\Gamma}_n^{-1}(\Theta_n + \mathbf{I}_n)(\mathbf{X}_n - \hat{\mathbf{X}}_n) \\ &= (\mathbf{X}_n - \hat{\mathbf{X}}_n)^T(\Theta_n + \mathbf{I}_n)^T[(\mathbf{\Theta}_n + \mathbf{I}_n)^T]^{-1}\mathbf{D}_n^{-1}(\mathbf{\Theta}_n + \mathbf{I}_n)^{-1}(\Theta_n + \mathbf{I}_n)(\mathbf{X}_n - \hat{\mathbf{X}}_n) \\ &= (\mathbf{X}_n - \hat{\mathbf{X}}_n)^T\mathbf{D}_n^{-1}(\mathbf{X}_n - \hat{\mathbf{X}}_n) \\ &= \sum_{j=1}^{n}(X_j - \hat{X}_j)^2/v_{j-1}. \end{aligned}$$

So the innovations algorithm can be used to bypass the inversion of $\mathbf{\Gamma}_n$. In addition we have

$$|\mathbf{\Gamma}_n| = |(\mathbf{\Theta}_n + \mathbf{I}_n)|^2|\mathbf{D}_n| = \prod_{j=1}^{n} v_{j-1}.$$

We may therefore re-express the likelihood function as

$$L_n(\mathbf{\Gamma}_n; \mathbf{X}_n) = (2\pi)^{-n/2}|v_0 \cdots v_{n-1}|^{-1/2} \exp\left[-\frac{1}{2}\sum_{j=1}^{n}(X_j - \hat{X}_j)^2/v_{j-1}\right].$$

Maximizing the Gaussian likelihood for an ARMA(p, q) process

If $X_1, \ldots, X_n$ come from an ARMA(p, q) process as in (1), the matrix $\mathbf{\Gamma}_n$ of covariances among $X_1, \ldots, X_n$ has is a function of $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_p)^T$ and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_q)^T$ and σ^2, so we may write $\mathbf{\Gamma}_n = \mathbf{\Gamma}_n(\boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2)$. Then the likelihood function becomes a function of $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and σ^2, so that it may be written as

$$L_n(\boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2; \mathbf{X}_n) = (2\pi)^{-n/2}|v_0 \cdots v_{n-1}|^{-1/2} \exp\left[-\frac{1}{2}\sum_{j=1}^{n}(X_j - \hat{X}_j)^2/v_{j-1}\right],$$

where the predicted values $\hat{X}_1, \ldots, \hat{X}_n$ and the MSEPs $v_0, \ldots, v_{n-1}$ involve $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and σ^2. Some properties of the ARMA(p,q) model can be cleverly used in order to evaluate and maximize the likelihood function quickly. See Sections 5.3 and 8.7 of B&D Theory or details. Rather than writing our own code for maximizing the ARMA(p,q) likelihood, we will use the R function `arima()`. The R code below computes the maximum likelihood estimators of $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and σ^2 on the data generated in the previous chunk of R code:

```
arima(x = X - mean(X),order=c(p,0,q),include.mean=FALSE)
```

```
##
## Call:
## arima(x = X - mean(X), order = c(p, 0, q), include.mean = FALSE)
##
## Coefficients:
##          ar1     ma1     ma2
##       0.2861  0.2495  0.1619
## s.e.  0.1339  0.1326  0.0747
##
## sigma^2 estimated as 1.051:  log likelihood = -722.15,  aic = 1452.3
```

Asymptotic distribution of maximum likelihood estimators

Suppose $X_1, \ldots, X_n$ is a length-n realization of the causal invertible ARMA(p,q) process in (1), with $\{Z_t, t \in \mathbb{Z}\} \sim \text{IID}(0, \sigma^2)$. Let $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\theta}}$, and $\hat{\sigma}^2$ be the maximum likelihood estimators of $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and σ^2, respectively, and define the $(p+q) \times 1$ vectors $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\phi}}^T, \hat{\boldsymbol{\theta}}^T)^T$ and $\boldsymbol{\beta} = (\boldsymbol{\phi}^T, \boldsymbol{\theta}^T)^T$. Then

$$\sqrt{n}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \to \text{Normal}(\mathbf{0}, V(\boldsymbol{\beta})) \quad \text{in distribution} \tag{4}$$

as $n \to \infty$, where

$$V(\boldsymbol{\beta}) = \sigma^2 \begin{bmatrix} \mathbb{E}\mathbf{U}\mathbf{U}^T & \mathbb{E}\mathbf{U}\mathbf{V}^T \\ \mathbb{E}\mathbf{V}\mathbf{U}^T & \mathbb{E}\mathbf{V}\mathbf{V}^T \end{bmatrix}^{-1},$$

where $\mathbf{U} = (U_p, \ldots, U_1)^T$ and $\mathbf{V} = (V_q, \ldots, V_1)^T$, where $\{U_t, t \in \mathbb{Z}\}$ and $\{V_t, t \in \mathbb{Z}\}$ are the autoregressive processes defined by

$$\phi(B)U_t = Z_t \quad \text{and} \quad \theta(B)V_t = Z_t \quad \text{for all } t \in \mathbb{Z}.$$

Example: Asymptotic distribution of MLEs in ARMA(1,1) model

Let $\{X_t, t \in \mathbb{Z}\}$ be the time series defined by

$$X_t - \phi X_{t-1} = Z_t + \theta Z_{t-1}$$

for $\{Z_t, t \in \mathbb{Z}\} \sim \text{IID}(0, \sigma^2)$, where $|\phi| < 1$ and $|\theta| < 1$ (so that the process is causal and invertible). The asymptotic covariance matrix of $\hat{\boldsymbol{\beta}} = (\hat{\phi}, \hat{\theta})^T$ is found as follows: Let $\{U_t, t \in \mathbb{Z}\}$ and $\{V_t, t \in \mathbb{Z}\}$ be the time series defined by

$$U_t = \phi U_{t-1} + Z_t \quad \text{and} \quad V_t = -\theta V_{t-1} + Z_t \quad \text{for all } t \in \mathbb{Z}.$$

Then we have the MA(∞) representations

$$U_t = \sum_{j=0}^{\infty} \phi^j Z_{t-j} \quad \text{and} \quad V_t = \sum_{j=0}^{\infty} (-\theta)^j Z_{t-j} \quad \text{for all } t \in \mathbb{Z}.$$

Now, noting that $p = q = 1$ so that $\mathbf{U}$ and $\mathbf{V}$ are 1×1 vectors, that is scalars, we compute the expectations

$$
\begin{aligned}
\mathbb{E}U^2 &= \mathbb{E}\left(\sum_{j=0}^{\infty}\phi^j Z_{t-j}\right)^2 = \mathbb{E}\sum_{j=0}^{\infty}\sum_{j=0}^{\infty}\phi^j\phi^k Z_{t-j}Z_{t-k} = \sum_{j=0}^{\infty}\phi^{2j}\sigma^2 = \frac{\sigma^2}{1-\phi^2} \\
\mathbb{E}UV &= \mathbb{E}\left(\sum_{j=0}^{\infty}\phi^j Z_{t-j}\right)\left(\sum_{k=0}^{\infty}(-\theta)^k Z_{t-k}\right) = \sum_{j=0}^{\infty}(-\theta\phi)^j\sigma^2 = \frac{\sigma^2}{1+\theta\phi} \\
\mathbb{E}V^2 &= \mathbb{E}\left(\sum_{j=0}^{\infty}(-\theta)^j Z_{t-j}\right)^2 = \mathbb{E}\sum_{j=0}^{\infty}\sum_{j=0}^{\infty}(-\theta)^j(-\theta)^k Z_{t-j}Z_{t-k} = \sum_{j=0}^{\infty}\theta^{2j}\sigma^2 = \frac{\sigma^2}{1-\theta^2}.
\end{aligned}
$$

Then

$$
\begin{aligned}
V(\boldsymbol{\beta}) &= \sigma^2\begin{bmatrix} \sigma^2(1-\phi^2) & \sigma^2(1+\theta\phi) \\ \sigma^2(1+\theta\phi) & \sigma^2(1-\theta^2) \end{bmatrix}^{-1} \\
&= \frac{1}{(1-\phi^2)^{-1}(1-\theta^2)^{-1}-(1+\theta\phi)^{-2}}\begin{bmatrix} (1-\phi^2) & -(1+\theta\phi) \\ -(1+\theta\phi) & (1-\theta^2) \end{bmatrix} \\
&= \frac{(1-\phi^2)(1-\theta^2)(1+\theta\phi)^2}{(1+\theta\phi)^2-(1-\phi^2)(1-\theta^2)}\begin{bmatrix} (1-\phi^2) & -(1+\theta\phi) \\ -(1+\theta\phi) & (1-\theta^2) \end{bmatrix} \\
&= \frac{(1+\theta\phi)}{(\phi+\theta)^2}\begin{bmatrix} (1-\theta^2)(1+\theta\phi) & -(1-\phi^2)(1-\theta^2) \\ -(1-\phi^2)(1-\theta^2) & (1-\phi^2)(1+\theta\phi) \end{bmatrix}.
\end{aligned}
$$

so that

$$
\sqrt{n}\left(\begin{bmatrix} \hat{\phi} \\ \hat{\theta} \end{bmatrix} - \begin{bmatrix} \phi \\ \theta \end{bmatrix}\right) \to \text{Normal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \frac{(1+\theta\phi)}{(\phi+\theta)^2}\begin{bmatrix} (1-\theta^2)(1+\theta\phi) & -(1-\phi^2)(1-\theta^2) \\ -(1-\phi^2)(1-\theta^2) & (1-\phi^2)(1+\theta\phi) \end{bmatrix}\right)
$$

in distribution as $n \to \infty$.

Confidence intervals for ARMA(p, q) coefficients

We can construct confidence intervals for the ARMA(p, q) coefficients based on the asymptotic distribution of the maximum likelihood estimators. Under the settings which ensure asymptotic Normality in (4) of the maximum likelihood estimators, the endpoints of an asymptotic $(1-\alpha)100\%$ confidence interval for the coefficient in position j of the vector $\boldsymbol{\beta} = (\phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q)^T$ are given by

$$
\hat{\beta}_j \pm z_\alpha\sqrt{v_{jj}(\hat{\boldsymbol{\beta}})/n},
$$

where $v_{jj}(\boldsymbol{\beta})$ is the jth diagonal element of the matrix $V(\hat{\boldsymbol{\beta}})$ and z_α is the upper α quantile of the standard Normal distribution.

The following R code runs a simulation which assesses the coverage of this interval in the ARMA$(1, 1)$ case.

```
phi <- .5
theta <- .3
sigma <- 1
n <- 500
alpha <- .05

S <- 500
```

```r
beta.hat <- matrix(NA,S,p+q)
sigma.hat <- numeric()
lo.ci.phi <- up.ci.phi <- lo.ci.theta <- up.ci.theta <- numeric()
for( s in 1:S )
{

  X <- get.ARMA.data(phi,theta,sigma,n)
  arima.out <- arima(x = X - mean(X),order=c(1,0,1),include.mean=FALSE)
    phi.hat <- arima.out$coef[1]
    theta.hat <- arima.out$coef[2]

    v11.hat <- (1-theta.hat^2) * (1+theta.hat*phi.hat)^2 / (theta.hat+phi.hat)^2
    v22.hat <- (1-phi.hat^2) * (1+theta.hat*phi.hat)^2 / (theta.hat+phi.hat)^2

    lo.ci.phi[s] <- phi.hat - qnorm(1-alpha/2) * sqrt(v11.hat/n)
    up.ci.phi[s] <- phi.hat + qnorm(1-alpha/2) * sqrt(v11.hat/n)

    lo.ci.theta[s] <- theta.hat - qnorm(1-alpha/2) * sqrt(v22.hat/n)
    up.ci.theta[s] <- theta.hat + qnorm(1-alpha/2) * sqrt(v22.hat/n)

    sigma.hat[s] <- sqrt(arima.out$sigma2)

}

cov.phi <- mean( (lo.ci.phi < phi) & (up.ci.phi > phi) )
cov.theta <- mean( (lo.ci.theta < theta) & (up.ci.theta > theta) )

avg.width.phi <- mean(up.ci.phi - lo.ci.phi)
avg.width.theta <- mean(up.ci.theta - lo.ci.theta)

cov.phi
```

```
## [1] 0.958
```

```r
cov.theta
```

```
## [1] 0.888
```

```r
avg.width.phi
```

```
## [1] 0.2404043
```

```r
avg.width.theta
```

```
## [1] 0.2194922
```

Strikingly, the variance σ^2 of the sequence $\{Z_t, t \in \mathbb{Z}\}$ cancels out in the asymptotic covariance matrix of the maximum likelihood estimators of $\hat{\boldsymbol{\phi}}$ and $\hat{\boldsymbol{\theta}}$. If σ is changed in the code above, the average width of the confidence intervals does not change.

Making forecasts after estimating ARMA(p, q) parameters

Now we consider using innovations algorithm to make h-step-ahead forecasts after fitting an ARMA(p, q) model to some data. Recall that the innovations algorithm makes predictions based on the covariances among the random variables $X_1, \ldots, X_n + h$, so we the covariances among these random variables according to the fitted ARMA(p, q) model. If $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\theta}}$, and $\hat{\sigma}^2$ are the maximum likelihood estimators of $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and σ^2 based on

$X_1, \ldots, X_n$, then we can find the coefficients $\{\hat{\psi}_j, j = 0, 1, 2 \ldots\}$ of the the MA(∞) representation of the ARMA(p, q) time series with coefficients $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\theta}}$. Then we can use these coefficients to define the autocovariance function of the fitted ARMA(p, q) process as

$$\hat{\gamma}(h; \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}, \hat{\sigma}^2) = \hat{\sigma}^2 \sum_{j=0}^{\infty} \hat{\psi}_j \hat{\psi}_{j+|h|}$$

These covariances are the only input required by the innovations algorithm.

There is a more efficient algorithm for computing h-step-ahead predictions for ARMA(p, q) models described in Section 5.3 of B&D Theory.

The following R code defines a function which returns values of the autocovariances function of an ARMA(p, q) process. Another function is defined which uses this output to compute the h-step ahead predictions based on the innovations algorithm. The use of the function is illustrated in getting the h-step-ahead predictions based on an ARMA(p, q) model fitted to a simulated data set. The predictions and the MSEPs are compared to those returned by the `predict.Arima()` function.

```
# function to get the acvf of an ARMA(p,q) process:
ARMAacvf <- function(phi=NULL,theta=NULL,sigma=1,max.lag=12,trun=500)
{
    # check to see if the time series is causal:
    if(length(phi)>0)
    {
    minroot <- min(Mod(polyroot(c(1,-phi))))
    if( minroot < 1)
        stop("The ARMA process specified is not causal.")
    }

    psi <- ARMAtoMAinf(phi,theta,trun)

    gamma.0 <- sigma^2 * sum(psi^2)
    ARMAacvf <- numeric(max.lag+1)
    ARMAacvf[1] <- gamma.0

    for(h in 1:max.lag)
    {

        ARMAacvf[1+h] <- sigma^2 * sum( psi[1:(trun-h)] * psi[(1+h):trun])

    }

    return(ARMAacvf)

}

# function to get h-step-ahead predictions based on an ARMA(p,q) model
ARMA.hstep <- function(X,h,phi,theta,sigma)
{

    X.cent <- X - mean(X)
    n <- length(X)
    gamma.hat <- ARMAacvf(phi,theta,sigma,max.lag=n+h)
    gamma.0 <- gamma.hat[1]
    gamma.n <- gamma.hat[-1]
```

```r
    K <- matrix(0,n+h,n+h)
    for(j in 1:(n+h-1))
        for(i in (j+1):(n+h))
        {

            K[i,j] <- c(gamma.0,gamma.n)[1+abs(i-j)]

        }

    K <- K + t(K) + diag(rep(gamma.0),n+h)

    # Next part inefficient: Speed up someday with 5.3.9 of B&D Theory
    innov.hstep.out <- innov.hstep(X.cent,h,K)
    X.pred <- innov.hstep.out$X.pred + mean(X)
    v <- innov.hstep.out$v

    p <- length(phi)
    q <- length(theta)
    ll <- -(n/2)*log(2*pi) - (1/2) * sum( log( v[1:n] ))
          - (1/2) * sum( (X - X.pred[1:n])^2/v[1:n])
    aic <- -2*ll + 2 * ( p + q + 1 ) # plus 1 for the variance

    output <- list( X.pred = X.pred,
                    v = v,
                    aic = aic)

    return(output)

}

# specify model parameters and generate data
phi <- c(.5,.1)
theta <- c(.3,.6)
p <- length(phi)
q <- length(theta)
sigma <- 1
n <- 50
X <- get.ARMA.data(phi,theta,sigma,n)

# fit arma model on centered data and store output:
arima.out <- arima(X-mean(X),order=c(p,0,q),include.mean=FALSE)
phi.hat <- arima.out$coef[1:p]
theta.hat <- arima.out$coef[(p+1):(p+q)]
sigma.hat <- sqrt(arima.out$sigma2)

h <- 10

ARMA.hstep.out <- ARMA.hstep(X,h,phi.hat,theta.hat,sigma.hat)

# get forecasts from predict.Arima function:
arima.predict.out <- predict(arima.out,n.ahead=h)

# compare output:
```

```r
ARMA.hstep.out$X.pred[-c(1:n)]
```

```
##  [1]  0.6539551 -0.5567909  0.3213523  0.2807885  0.3090928  0.3077359
##  [7]  0.3086483  0.3086030  0.3086324  0.3086309
```

```r
as.numeric(arima.predict.out$pred) + mean(X)
```

```
##  [1]  0.6539551 -0.5567909  0.3213523  0.2807885  0.3090928  0.3077359
##  [7]  0.3086483  0.3086030  0.3086324  0.3086309
```

```r
sqrt(ARMA.hstep.out$v[-c(1:n)])
```

```
##  [1] 0.7571372 0.8767088 1.1550595 1.1551304 1.1553828 1.1553829 1.1553832
##  [8] 1.1553832 1.1553832 1.1553832
```

```r
as.numeric(arima.predict.out$se)
```

```
##  [1] 0.7571372 0.8767088 1.1550595 1.1551304 1.1553828 1.1553829 1.1553832
##  [8] 1.1553832 1.1553832 1.1553832
```

Missing data issues

We consider how to estimate the ARMA coefficients when the time series has missing values. In addition we consider imputation or prediction of the missing values.

Estimation with missing values

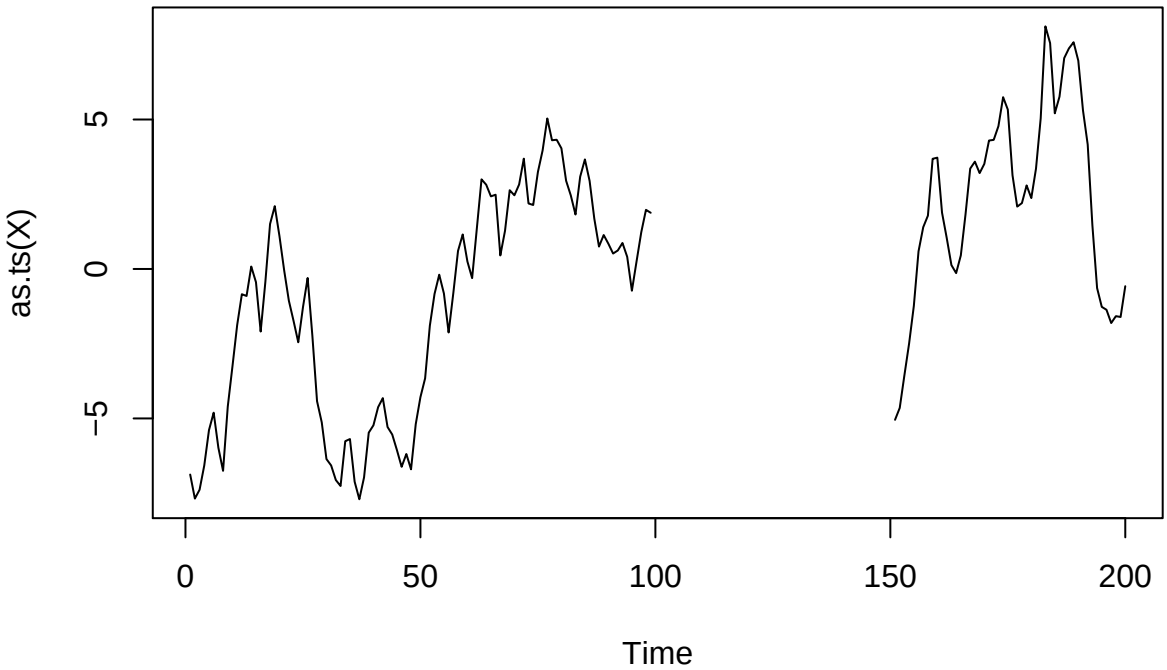
Suppose the time series $\{X_t : t \in \mathbb{Z}\}$ is a causal ARMA(p, q) process with mean zero, and suppose that instead of observing $X_1, \ldots, X_n$, we observe only the set of random variables $\{X_i, i \in \mathcal{O}\}$, where $\mathcal{O} \subset \{1, \ldots, n\}$, so that values at some time points are missing. Let $\mathcal{M} = \{1, \ldots, n\} \setminus \mathcal{O}$ denote the set of indices for which we do not observe the time series. We can obtain maximum likelihood estimates for the ARMA parameters by maximizing the likelihood function based on the observed data. Define the vector $\mathbf{X}_{\mathcal{O}} = (X_j, j \in \mathcal{O})^T$ and let $\mathbf{\Gamma}_{\mathcal{O}}$ be the matrix formed by keeping the rows and columns of the matrix $\mathbf{\Gamma}_n$ which correspond to the set of indices $\mathcal{O}$ of the observed data. Then the likelihood function based on the observed data $\{X_i, i \in \mathcal{O}\}$ is given by

$$L_{\mathcal{O}}(\mathbf{\Gamma}_{\mathcal{O}}; \mathbf{X}_{\mathcal{O}}) = (2\pi)^{-|\mathcal{O}|/2} |\mathbf{\Gamma}_{\mathcal{O}}|^{-1/2} \exp\left[-\frac{1}{2}\mathbf{X}_{\mathcal{O}}^T \mathbf{\Gamma}_{\mathcal{O}}^{-1} \mathbf{X}_{\mathcal{O}}\right],$$

where $\mathbf{\Gamma}_{\mathcal{O}}$ is a function of the ARMA parameters $\phi_1, \ldots, \phi_p$ and $\theta_1, \ldots, \theta_q$ and the white noise variance σ^2. To find the maximum likelihood estimators of these parameters based on the non-missing data $\{X_i, i \in \mathcal{O}\}$, we simply need to maximize this function. We will not spend time deriving an algorithm to maximize this; the `arima()` function in R will maximize the likelihood based on the observed data when the input vector contains missing values. The following R code illustrates this:

```r
# generate some data:
phi <- c(.9)
theta <- c(.7)
n <- 200
sigma <- 1
X <- get.ARMA.data(phi,theta,sigma,n)

# make a chunk in the middle missing:
M <- 100:150
O <- c(1:n)[-M]
```

```r
X[M] <- NA
plot(as.ts(X))
```




```r
# now fit an ARMA(p,q) model to the incomplete data
p.hat <- 1
q.hat <- 1
arima.out <- arima(X-mean(X,na.rm=TRUE),order=c(p.hat,0,q.hat),include.mean=FALSE)
arima.out
```

```
##
## Call:
## arima(x = X - mean(X, na.rm = TRUE), order = c(p.hat, 0, q.hat), include.mean = FALSE)
##
## Coefficients:
##          ar1     ma1
##       0.9273  0.6362
## s.e.  0.0289  0.0693
##
## sigma^2 estimated as 0.8352:  log likelihood = -201.41,  aic = 408.83
```

"Predicting" or imputing missing values

After obtaining the maximum likelihood estimates of the ARMA parameters based on the observed data, we may wish to predict or impute the missing values. Note that we can use both past and future values surrounding a time point for which the data are missing to "predict" its value.

For any $i \in \mathcal{M}$, we will predict the missing value X_i with the projection $P_{\overline{\text{sp}}\{X_j, j \in \mathcal{O}\}} X_i$, which is given by

$$P_{\overline{\text{sp}}\{X_j, j \in \mathcal{O}\}} X_i = \sum_{j \in \mathcal{O}} a_j X_j,$$

where $\{a_j, j \in \mathcal{O}\}$ are chosen to minimize the mean squared error of prediction (MSEP); that is, $\{a_j, j \in \mathcal{O}\}$

satisfy

$$\mathbb{E}\left[X_i - \sum_{j\in\mathcal{O}} a_j X_j\right]^2 = \min_{b_j, j\in\mathcal{O}} \mathbb{E}\left[X_i - \sum_{j\in\mathcal{O}} b_j X_j\right]^2.$$

Keep in mind that we have assumed that the time series has mean zero, so there is no intercept term. Defining the vector $\boldsymbol{\gamma}_{i,\mathcal{O}} = (\gamma(i-j), j \in \mathcal{O})^T$, we can find the vector $\mathbf{a} = (a_j, j \in \mathcal{O})^T$ by setting the derivative of the MSEP with respect to $\mathbf{a}$ equal to zero. We have

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{a}} \mathbb{E}(X_i - \mathbf{a}^T \mathbf{X}_{\mathcal{O}})^2 &= -2\mathbb{E}\mathbf{X}_{\mathcal{O}}(X_i - \mathbf{a}^T\mathbf{X}_{\mathcal{O}}) \\
&= -2(\mathbb{E}\mathbf{X}_{\mathcal{O}} X_i - \mathbb{E}\mathbf{X}_{\mathcal{O}}\mathbf{X}_{\mathcal{O}}^T\mathbf{a}) \\
&= -2\boldsymbol{\gamma}_{i,\mathcal{O}} + 2\boldsymbol{\Gamma}_{\mathcal{O}}\mathbf{a},
\end{aligned}$$

so the values $\{a_j, j \in \mathcal{O}\}$ are found according to

$$\mathbf{a} = \boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}}.$$

Under this choice of $\{a_j, j \in \mathcal{O}\}$ the MSEP is

$$\begin{aligned}
\mathbb{E}\left[X_i - (\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}})^T\mathbf{X}_{\mathcal{O}}\right]^2 &= \mathbb{E}\left[X_i^2 - 2\boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\mathbf{X}_{\mathcal{O}}X_i + \boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\mathbf{X}_{\mathcal{O}}\mathbf{X}_{\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}}\right] \\
&= \gamma(0) - 2\boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}} + \boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\Gamma}_{\mathcal{O}}\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}} \\
&= \gamma(0) - \boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}}.
\end{aligned}$$

If the time series is Gaussian, we could use the above to construct a 95% prediction interval for X_i as

$$(\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}})^T\mathbf{X}_{\mathcal{O}} + 1.96\sqrt{\gamma(0) - \boldsymbol{\gamma}_{i,\mathcal{O}}^T\boldsymbol{\Gamma}_{\mathcal{O}}^{-1}\boldsymbol{\gamma}_{i,\mathcal{O}}}.$$

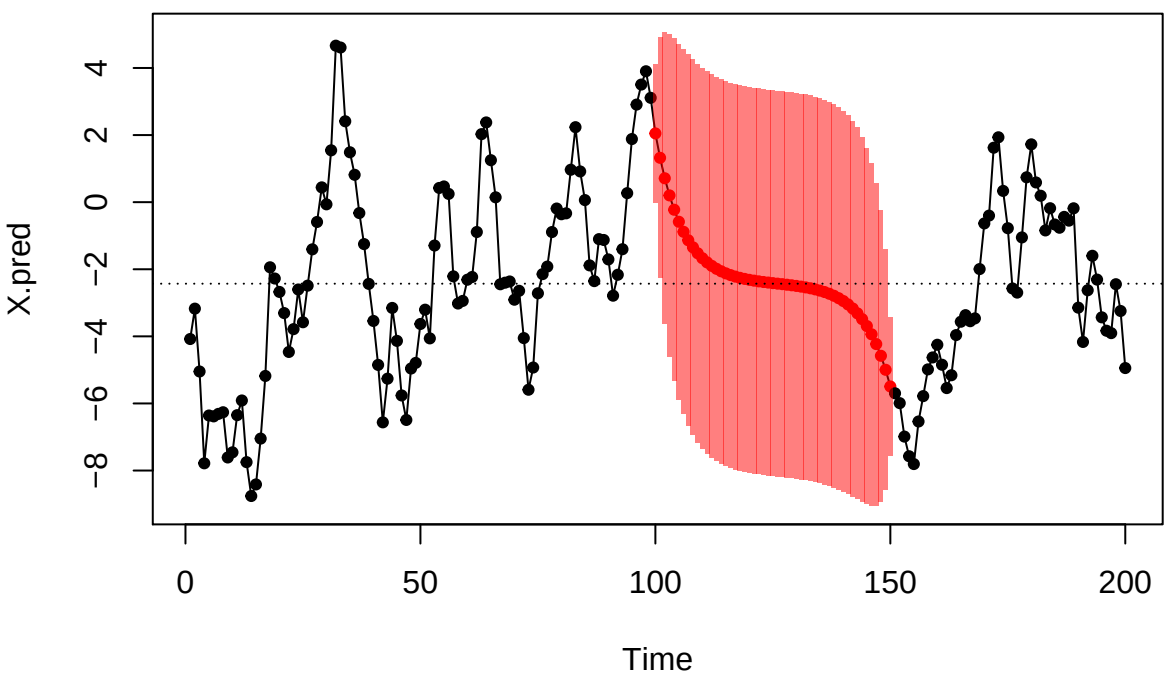
This can be done for each $i \in \mathcal{M}$. The following R code implements this on a data set with missing values by (i) using the `arima()` function to get the maximum likelihood estimators of the ARMA coefficients based on the observed data, then (ii) using the coefficients of the fitted ARMA model to construct an estimate $\hat{\boldsymbol{\Gamma}}_{\mathcal{O}}$ of the covariance matrix $\boldsymbol{\Gamma}_{\mathcal{O}}$, and then (iii) computing the predictions and constructing prediction intervals for each $i \in \mathcal{M}$ as described in this section.

Since we have assumed in these calculations that the time series has mean zero, one should center the time series by subtracting the mean of the observed data before implementing the above. Then, having obtained predictions for the centered time series, one adds to these predictions the mean of the observed data to obtain the predictions for the uncentered time series. This is done the code below.

```
ARMAimpute <- function(X,phi,theta,sigma,plot=TRUE)
{

    n <- length(X)
    X.bar <- mean(X,na.rm=TRUE)
    X.cent <- X - X.bar

    M <- which(is.na(X))
    O <- c(1:n)[-M]

    g.hat <- ARMAacvf(phi,theta,sigma,max.lag=n-1)
    G.hat <- matrix(NA,n,n)
```

```r
    for(i in 1:n)
        for(j in 1:n)
        {

            G.hat[i,j] <- g.hat[1 + abs(i-j)]

        }

    G.hat.O <- G.hat[O,O]

    X.pred <- X
    v.pred <- lo.pred <- up.pred <- as.numeric(X)

    for(i in 1:length(M))
    {

        g.hat.i.O <- G.hat[O,M[i]]
        G.hat.0.inv <- solve(G.hat.O)
        a.i <- G.hat.0.inv %*% g.hat.i.O
        X.pred[M[i]] <- sum( a.i * X.cent[O]) + X.bar
        v.pred[M[i]] <- G.hat[1,1] - t(g.hat.i.O) %*% G.hat.0.inv %*% g.hat.i.O
        lo.pred[M[i]] <- X.pred[M[i]] - 1.96 * sqrt(v.pred[M[i]])
        up.pred[M[i]] <- X.pred[M[i]] + 1.96 * sqrt(v.pred[M[i]])

    }

    if(plot==TRUE)
    {

        plot(X.pred,ylim=range(up.pred,lo.pred,X.pred))
        points(X.pred,type="p",pch=19,cex=.7,col=ifelse(1:n %in% O,"black","red"))
        for(i in 1:length(M))
        {
            y.poly <- c(lo.pred[M[i]],lo.pred[M[i]],up.pred[M[i]],up.pred[M[i]])
            x.poly <- c(M[i]-1/2,M[i]+1/2,M[i]+1/2,M[i]-1/2)
            polygon(x=x.poly,y=y.poly,col=rgb(1,0,0,.5),border=NA)

        }

        abline( h=X.bar,lty=3)

    }


    output <- list(X.pred = X.pred,
                   lo.pred = lo.pred,
                   up.pred = up.pred,
                   M = M,
                   O = O)

}

# generate data
```

```r
phi <- c(.9)
theta <- c(.7)
n <- 200
sigma <- 1
X <- get.ARMA.data(phi,theta,sigma,n)

# a chunk in the middle missing:
M <- 100:150

# missing at random:
# M <- sample(1:n,30)

O <- c(1:n)[-M]
X[M] <- NA

p.hat <- 1
q.hat <- 1
arima.out <- arima(X-mean(X,na.rm=TRUE),order=c(p.hat,0,q.hat),include.mean=FALSE)
phi.hat <- arima.out$coef[1:p.hat]
theta.hat <- arima.out$coef[-c(1:p.hat)]
sigma.hat <- sqrt(arima.out$sigma2)

ARMAimpute(X,phi.hat,theta.hat,sigma.hat,plot=TRUE)
```



The dashed line is positioned at the mean of the observed data points. We see that the further away we are from the observed data, the closer the predictions are to the mean.