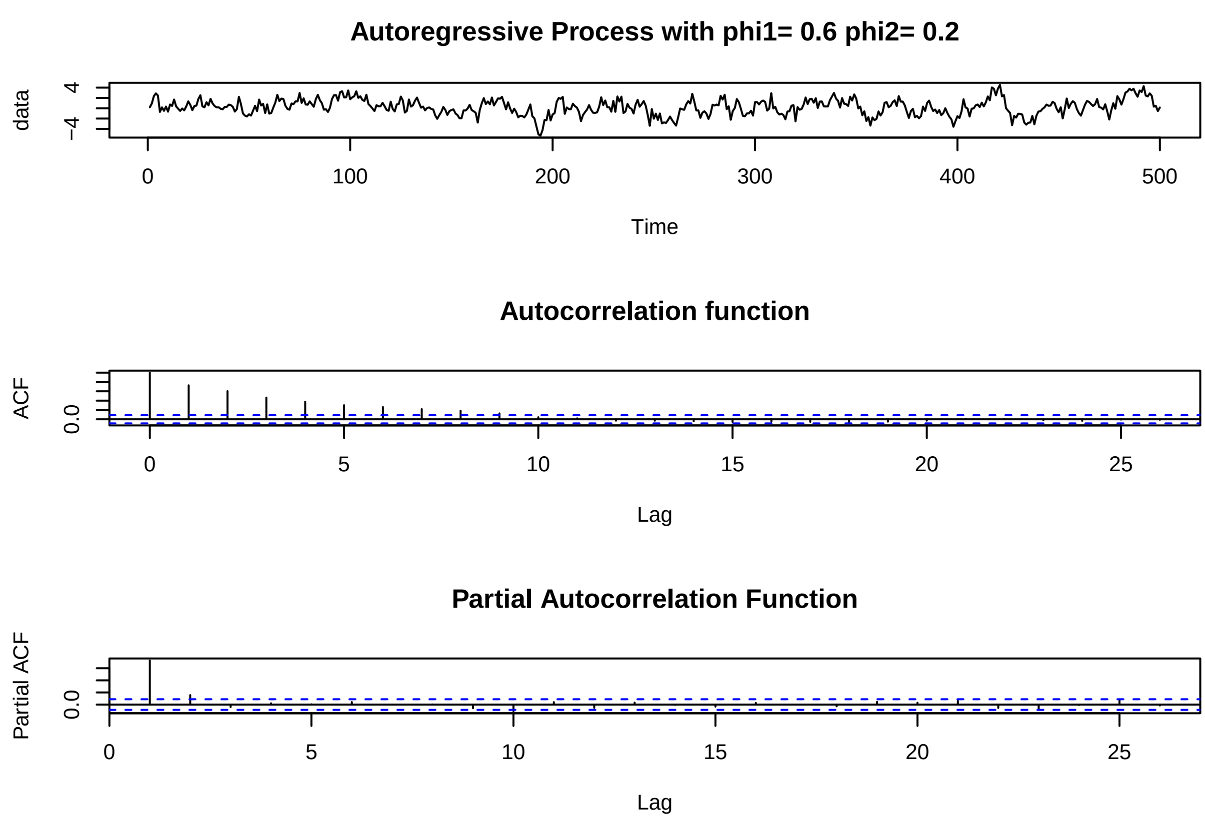
week-4.R

Ahmed

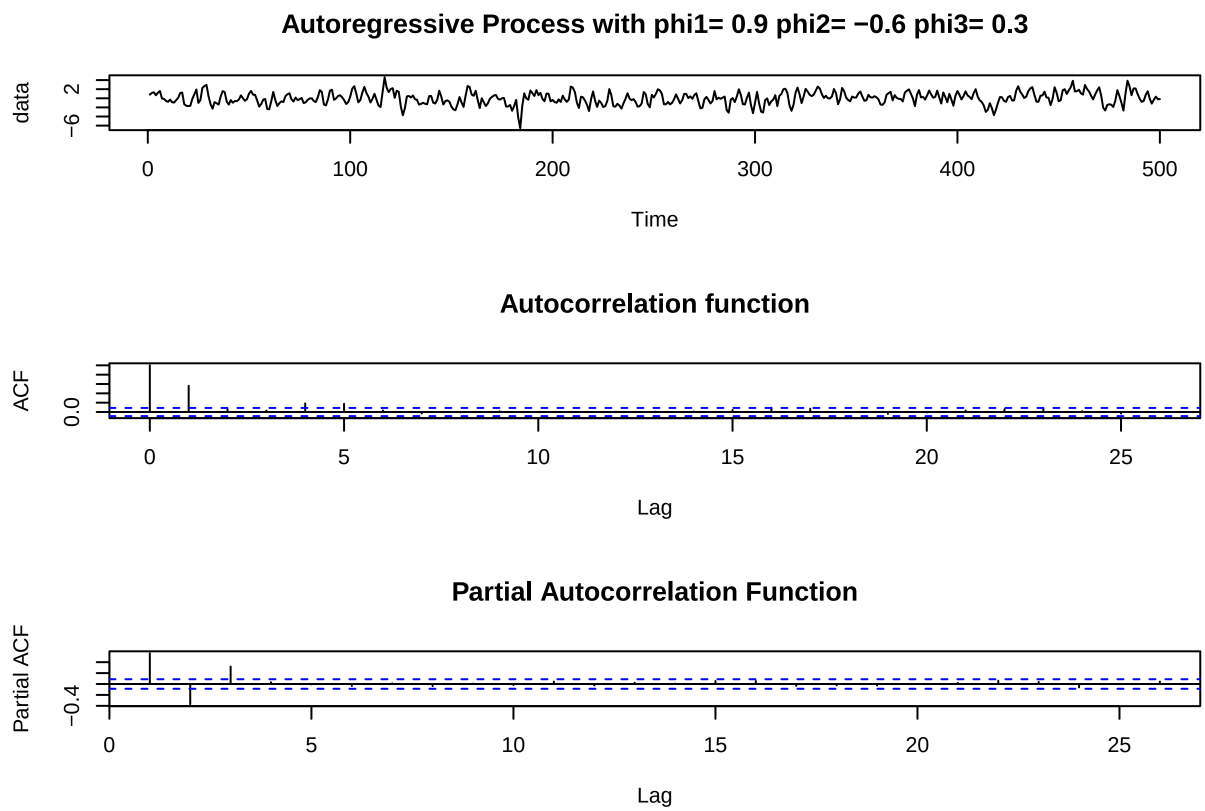
2023-04-16

```r
#AR(2) simulation
rm(list=ls(all=TRUE))
par(mfrow=c(3,1))
phi1=0.6
phi2=0.2
data=arima.sim(n=500,list(ar=c(phi1,phi2)))
plot(data,main=
       paste("Autoregressive Process with phi1=",phi1,"phi2=",phi2))
acf(data,main="Autocorrelation function")
acf(data,type = "partial",main="Partial Autocorrelation Function")
```

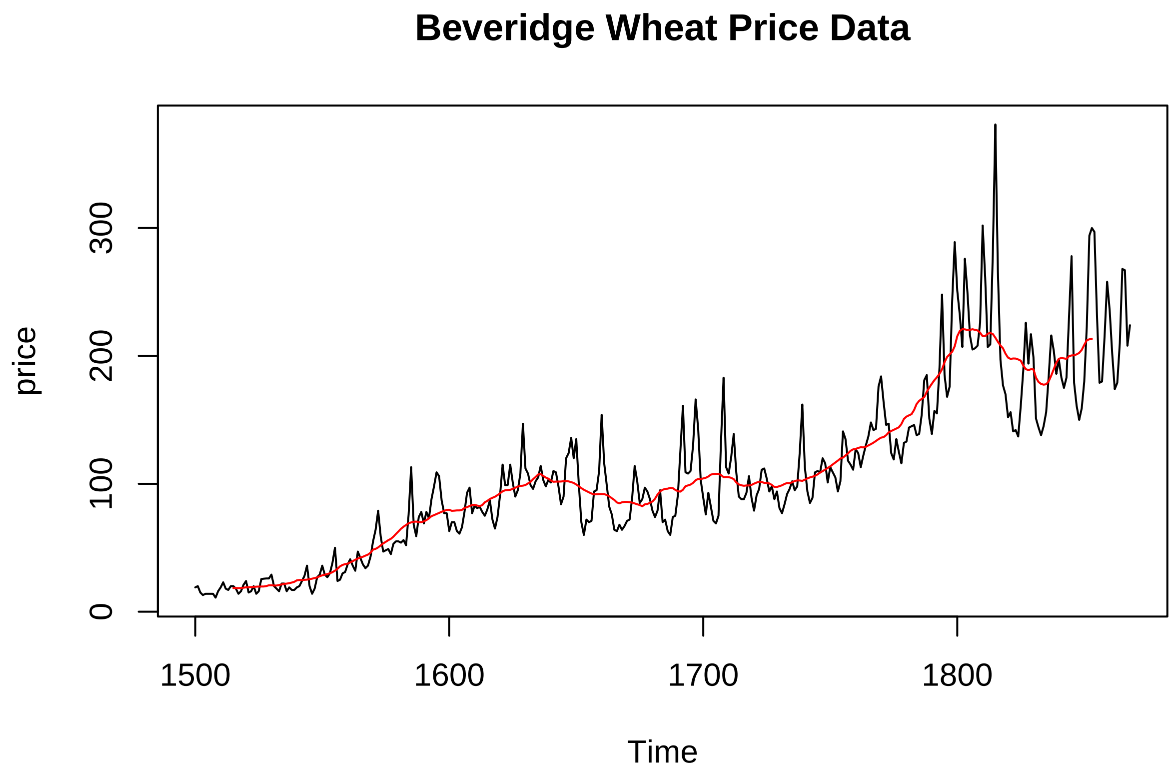



```r
#AR(3) simulation
rm(list=ls(all=TRUE))
```

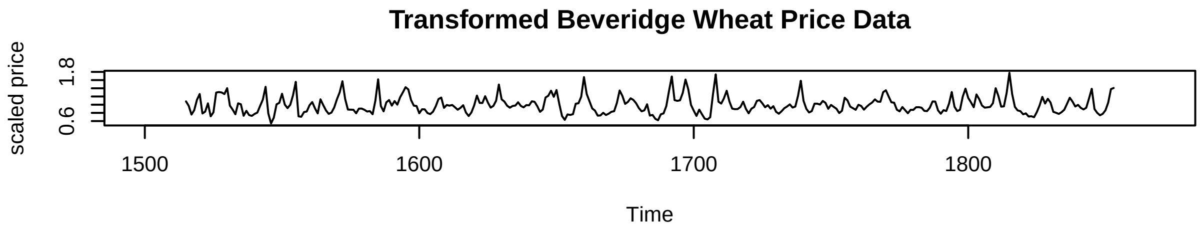
```
phi1 = 0.9
phi2 = -0.6
phi3= 0.3
data=arima.sim(n=500,list(ar=c(phi1,phi2,phi3)))
plot(data,main=
       paste("Autoregressive Process with phi1=",phi1,"phi2=",phi2,"phi3=",phi3))
acf(data,main="Autocorrelation function")
acf(data,type = "partial",main="Partial Autocorrelation Function")
```

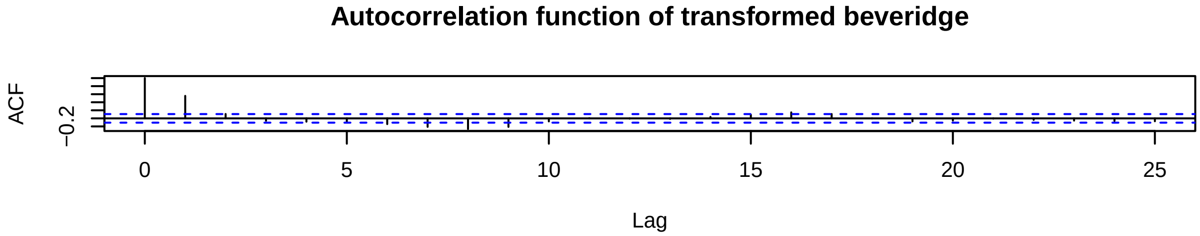


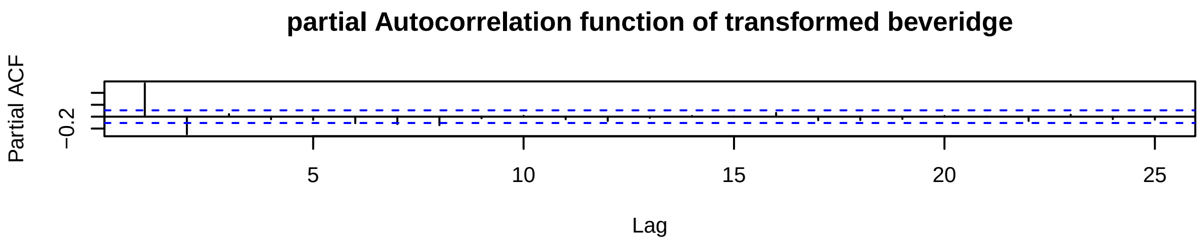

```
# Beveridge_wheat price data set
par(mfrow=c(1,1))
data = read.table("beveridge_wheat.txt",header = TRUE)
beveridge = ts(data[,2],start=1500)
plot(beveridge,ylab="price",main="Beveridge Wheat Price Data")
beveridgeMA=filter(beveridge,rep(1/31,31),side=2)
lines(beveridgeMA,col="red")
```




```
par(mfrow=c(3,1))
y = beveridge/beveridgeMA
plot(y,ylab="scaled price",
     main="Transformed Beveridge Wheat Price Data")
acf(na.omit(y),
    main="Autocorrelation function of transformed beveridge")
acf(na.omit(y),type="partial",
    main="partial Autocorrelation function of transformed beveridge")
```







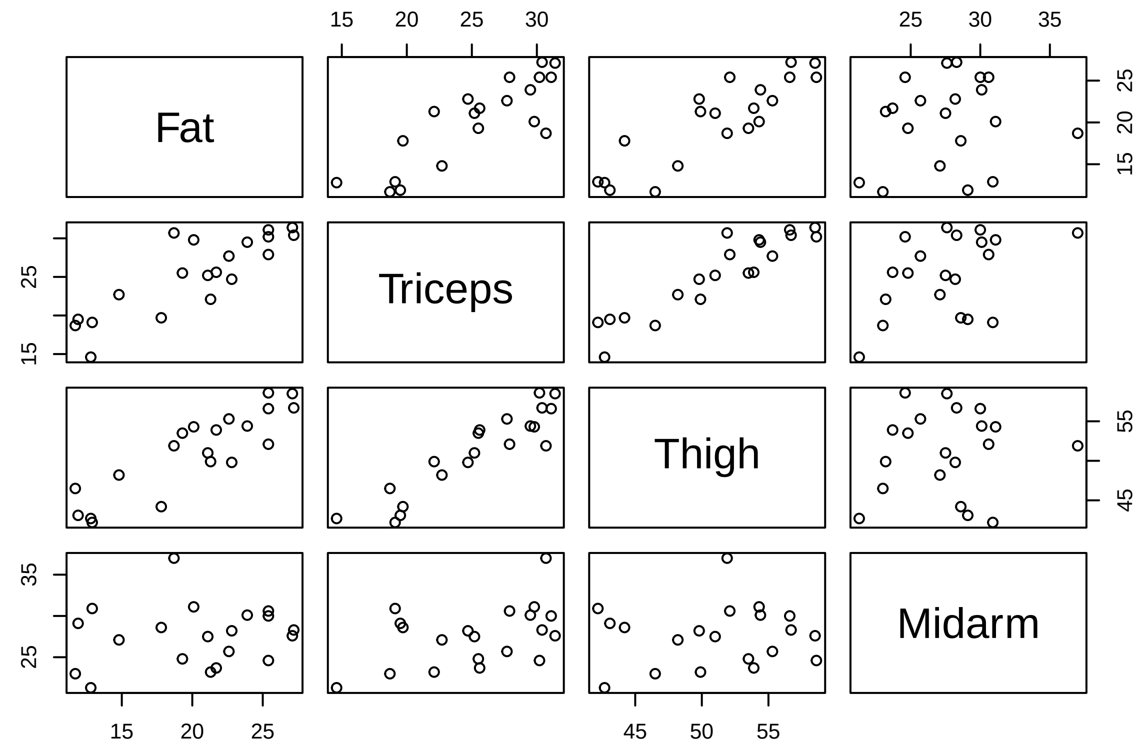

```r
ar(na.omit(y),order.max = 5)
```

```
##
## Call:
## ar(x = na.omit(y), order.max = 5)
##
## Coefficients:
##       1        2
##  0.7232  -0.2949
##
## Order selected 2  sigma^2 estimated as  0.027
```

```r
library(isdals)
```

```
## Warning: package 'isdals' was built under R version 4.2.3
```

```r
data("bodyfat")
attach(bodyfat)
pairs(cbind(Fat,Triceps,Thigh,Midarm))
```




```
cor(cbind(Fat,Triceps,Thigh,Midarm))
```

```
##                Fat   Triceps     Thigh    Midarm
## Fat      1.0000000 0.8432654 0.8780896 0.1424440
## Triceps  0.8432654 1.0000000 0.9238425 0.4577772
## Thigh    0.8780896 0.9238425 1.0000000 0.0846675
## Midarm   0.1424440 0.4577772 0.0846675 1.0000000
```

```
# partial correlation manually
fatHat=predict(lm(Fat~Thigh))
tricepsHat = predict(lm(Triceps~Thigh))
cor((Fat-fatHat),(Triceps-tricepsHat))
```

```
## [1] 0.1749822
```

```
# partial function
library(ppcor)
```

```
## Warning: package 'ppcor' was built under R version 4.2.3
```

```
## Loading required package: MASS
```

```r
pcor(cbind(Fat,Triceps,Thigh))$estimate
```

```
##               Fat  Triceps     Thigh
## Fat     1.0000000 0.1749822 0.4814109
## Triceps 0.1749822 1.0000000 0.7130120
## Thigh   0.4814109 0.7130120 1.0000000
```

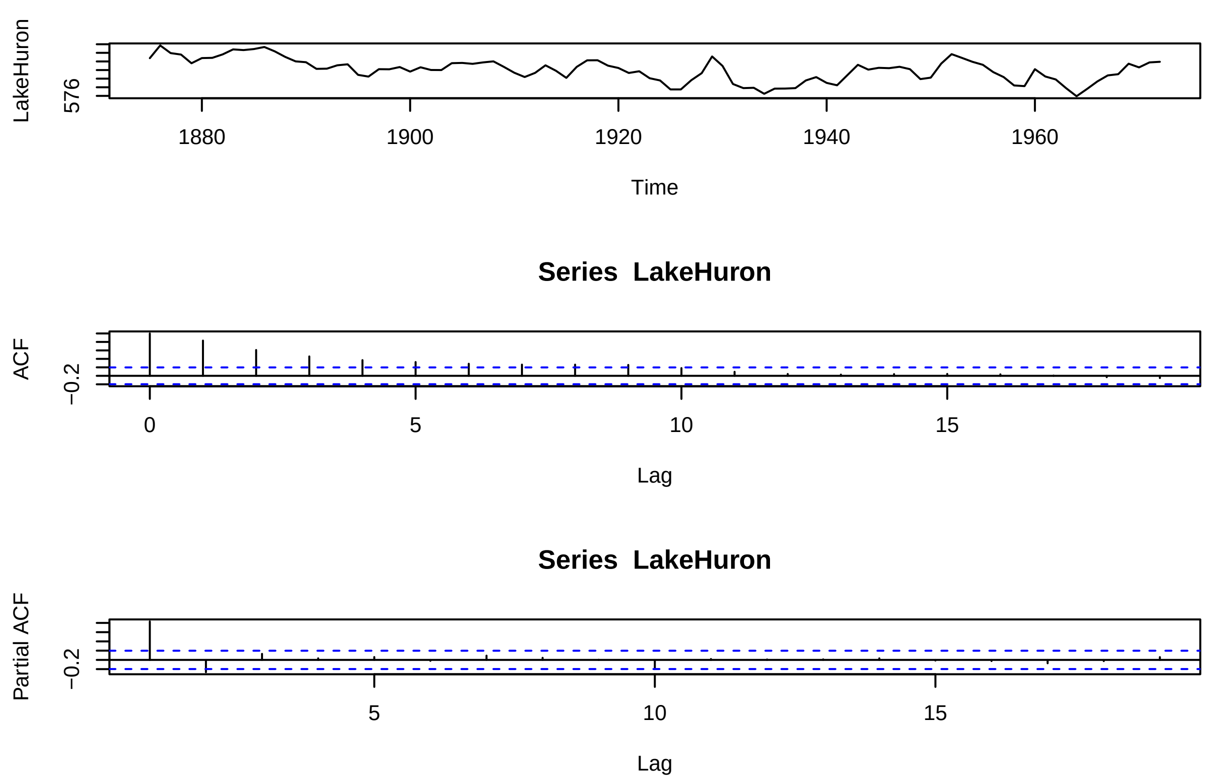
```r
pcor(cbind(Fat,Triceps,Thigh,Midarm))$estimate
```

```
##                Fat  Triceps      Thigh     Midarm
## Fat      1.0000000 0.3381500 -0.2665991 -0.3240520
## Triceps  0.3381500 1.0000000  0.9963725  0.9955918
## Thigh   -0.2665991 0.9963725  1.0000000 -0.9926612
## Midarm  -0.3240520 0.9955918 -0.9926612  1.0000000
```

```r
fatHat=predict(lm(Fat~Thigh+Midarm))
tricepsHat = predict(lm(Triceps~Thigh+Midarm))
cor((Fat-fatHat),(Triceps-tricepsHat))
```

```
## [1] 0.33815
```

```r
par( mfrow=c(3,1) )
plot(LakeHuron)
acf(LakeHuron)
acf(LakeHuron, type="partial")
```



```
attach(attitude);
rcl = cbind(rating, complaints, learning);
cor(rcl)
```

```
##                rating complaints  learning
## rating      1.0000000  0.8254176 0.6236782
## complaints 0.8254176  1.0000000 0.5967358
## learning   0.6236782  0.5967358 1.0000000
```

```
attach(attitude);
```

```
## The following objects are masked from attitude (pos = 3):
##
##     advance, complaints, critical, learning, privileges, raises, rating
```

```
rating.hat = predict( lm( rating ~ learning) )
complaints.hat = predict( lm( complaints~learning) )
cor((rating-rating.hat),(complaints-complaints.hat))
```

```
## [1] 0.7225924
```

```
(pacf(complaints.hat))
```

```
##
## Partial autocorrelations of series 'complaints.hat', by lag
##
##      1      2      3      4      5      6      7      8      9     10     11
##  0.013  0.165 -0.223 -0.232 -0.225 -0.173  0.079 -0.127 -0.088 -0.159  0.162
##     12     13     14
##  0.154  0.082 -0.107
```

```
#simulate ar(2) lab
set.seed(2017)
#model parameters that we will estimate
sigma=4
phi=NULL
phi[1:2]=c(1/3,1/2)
phi
```

```
## [1] 0.3333333 0.5000000
```

```
n=10000
#simulate process
ar.process=arima.sim(n,model=list(ar=c(1/3,1/2)), sd=4)
ar.process[1:5]
```

```
## [1] 4.087685 5.598492 3.019295 2.442354 5.398302
```

```r
#find 2nd & 3rd sample autocorrelation
r=NULL
r[1:2]=acf(ar.process, plot=F)$acf[2:3]
r
```

```
## [1] 0.6814103 0.7255825
```

```r
#matrix R
R=matrix(1,2,2) # matrix of dimension 2 by 2, with entries all 1's.
R
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1
```

```r
# edit R
R[1,2]=r[1] # only diagonal entries are edited
R[2,1]=r[1] # only diagonal entries are edited
R
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.6814103
## [2,] 0.6814103 1.0000000
```

```r
#b column vector on right
b=matrix(r,nrow=2,ncol=1)# b- column vector with no entries
b
```

```
##           [,1]
## [1,] 0.6814103
## [2,] 0.7255825
```

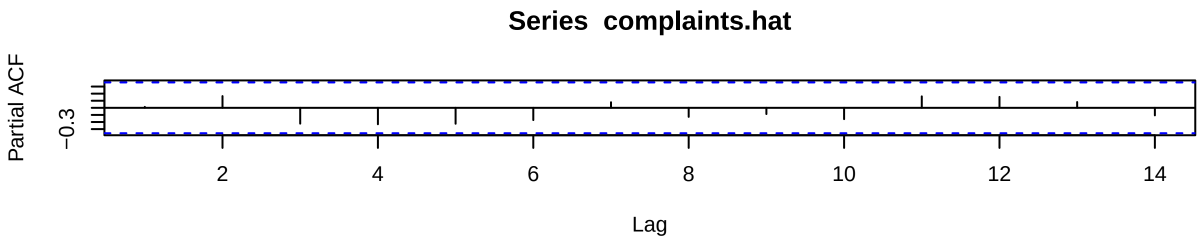
```r
#solve Rx=b
phi.hat=matrix(c(solve(R,b)[1,1], solve(R,b)[2,1]),2,1)
phi.hat
```

```
##           [,1]
## [1,] 0.3490720
## [2,] 0.4877212
```

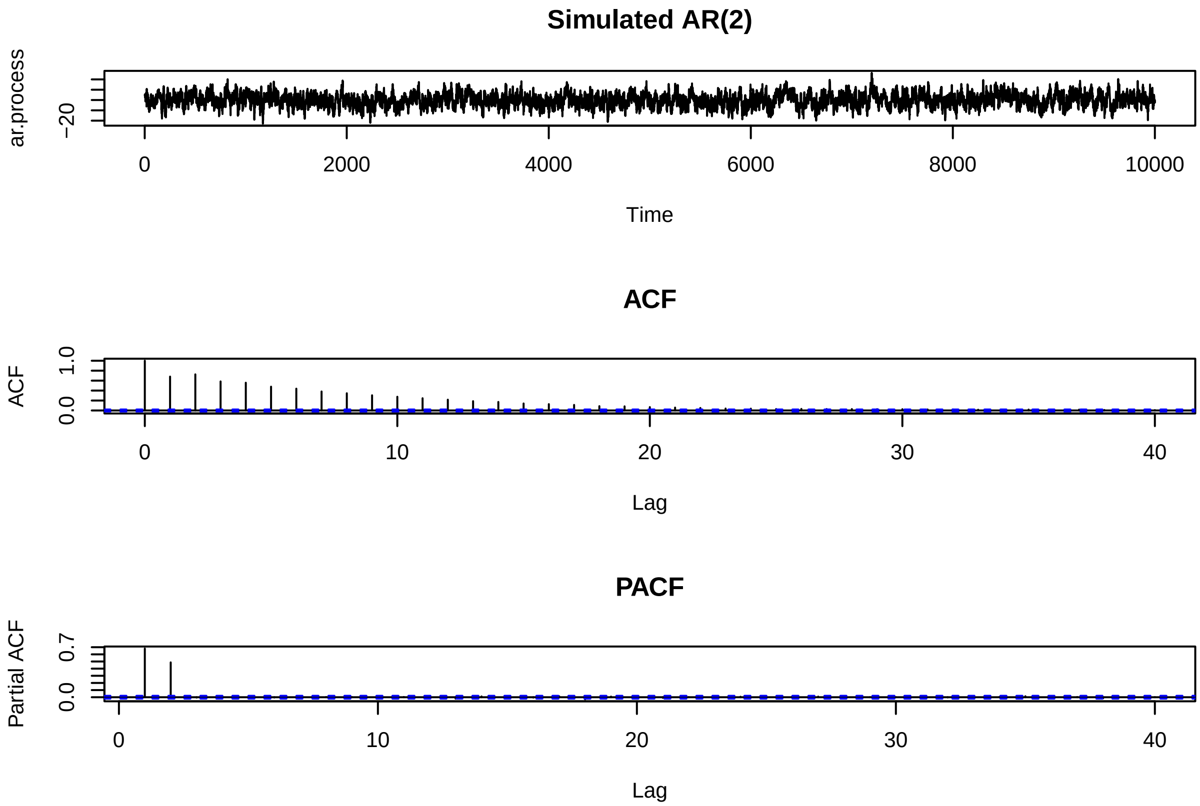
```r
# variance estimation
c0=acf(ar.process, type='covariance', plot=F)$acf[1]
var.hat=c0*(1-sum(phi.hat*r))
var.hat
```

```
## [1] 16.37169
```

```r
par(mfrow=c(3,1))
```




```r
# plot time series,acf,pacf
plot(ar.process, main='Simulated AR(2)')
acf(ar.process, main='ACF')
pacf(ar.process, main='PACF')
```




```r
#AR(3) simulation
sigma=4
phi=NULL
phi[1:3]=c(1/3,1/2,7/100)
n=100000
ar3.process=arima.sim(n,model=list(ar=c(1/3,1/2, 7/100)), sd=4)
r=NULL
r[1:3]=acf(ar3.process, plot=F)$acf[2:4]
r
```

```
## [1] 0.7862807 0.8189033 0.7383634
```

```r
R=matrix(1,3,3)
R[1,2]=r[1]
R[1,3]=r[2]
R[2,1]=r[1]
R[2,3]=r[1]
R[3,1]=r[2]
R[3,2]=r[1]
R
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7862807 0.8189033
## [2,] 0.7862807 1.0000000 0.7862807
## [3,] 0.8189033 0.7862807 1.0000000
```

```
# b-column vector on the right
b=matrix(0,3,1)# b- column vector with no entries
b[1,1]=r[1]
b[2,1]=r[2]
b[3,1]=r[3]
b
```

```
##           [,1]
## [1,] 0.7862807
## [2,] 0.8189033
## [3,] 0.7383634
```

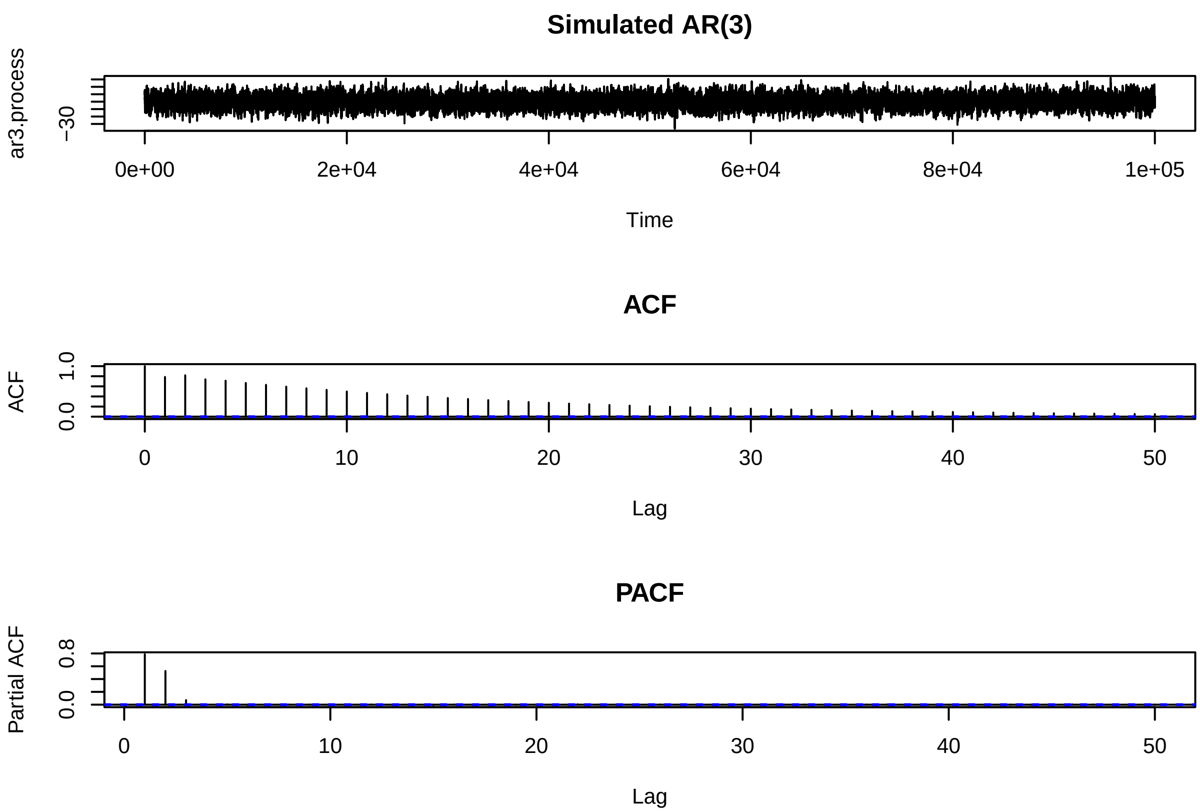
```
# solve Rx=b and find phi's
phi.hat=solve(R,b)
phi.hat
```

```
##            [,1]
## [1,] 0.33564293
## [2,] 0.49913030
## [3,] 0.07104774
```

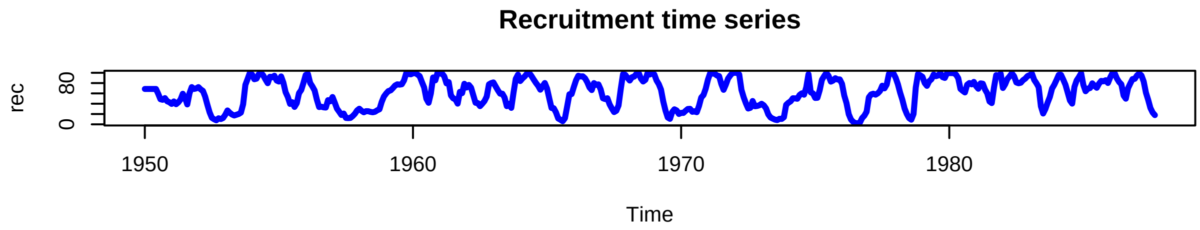
```
# sigma estimation
c0=acf(ar3.process, type='covariance', plot=F)$acf[1]
var.hat=c0*(1-sum(phi.hat*r))
var.hat
```

```
## [1] 15.93116
```

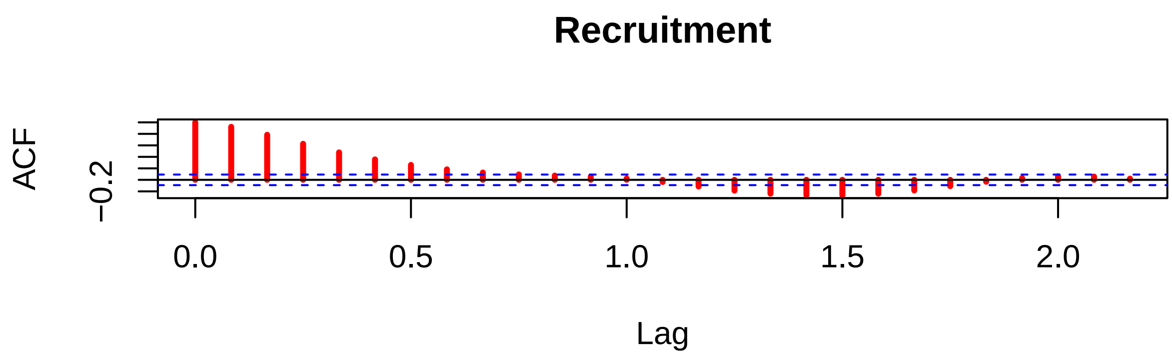
```
par(mfrow=c(3,1))
plot(ar3.process, main='Simulated AR(3)')
acf(ar3.process, main='ACF')
pacf(ar3.process, main='PACF')
```

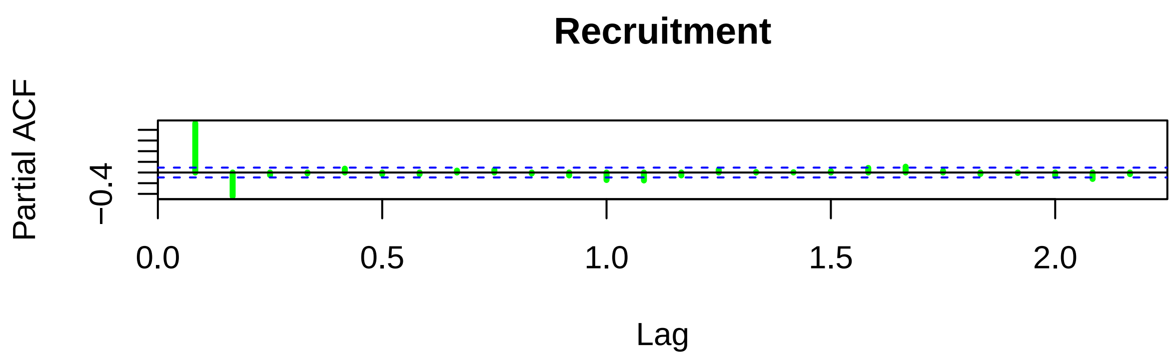



```
#Modeling recruitment time
library(astsa)
my.data=rec

# Plot rec
plot(rec, main='Recruitment time series', col='blue', lwd=3)
# subtract mean to get a time series with mean zero
ar.process=my.data-mean(my.data)

# ACF and PACF of the process
par(mfrow=c(2,1))
```




```
acf(ar.process, main='Recruitment', col='red', lwd=3)
pacf(ar.process, main='Recruitment', col='green', lwd=3)
```






```
# order
p=2

# sample autocorreleation function r
r=NULL
r[1:p]=acf(ar.process, plot=F)$acf[2:(p+1)]
cat('r=',r,'\n')
```

```
## r= 0.9218042 0.7829182
```

```
# matrix R
R=matrix(1,p,p) # matrix of dimension 2 by 2, with entries all 1's.

# define non-diagonal entires of R
for(i in 1:p){
    for(j in 1:p){
        if(i!=j)
            R[i,j]=r[abs(i-j)]
    }
}
R
```

```
##           [,1]      [,2]
## [1,] 1.0000000 0.9218042
## [2,] 0.9218042 1.0000000
```

```r
# b-column vector on the right
b=NULL
b=matrix(r,p,1)# b- column vector with no entries
b
```

```
##           [,1]
## [1,] 0.9218042
## [2,] 0.7829182
```

```r
# solve(R,b) solves Rx=b, and gives x=R^(-1)b vector
phi.hat=NULL
phi.hat=solve(R,b)[,1]
phi.hat
```

```
## [1]  1.3315874 -0.4445447
```

```r
#variance estimation using Yule-Walker Estimator
c0=acf(ar.process, type='covariance', plot=F)$acf[1]
c0
```

```
## [1] 780.991
```

```r
var.hat=c0*(1-sum(phi.hat*r))
var.hat
```

```
## [1] 94.17131
```

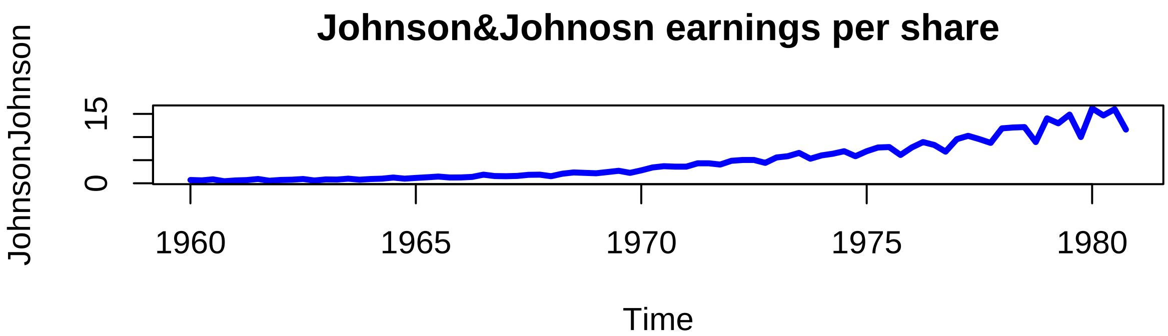
```r
# constant term in the model
phi0.hat=mean(my.data)*(1-sum(phi.hat))
phi0.hat
```

```
## [1] 7.033036
```

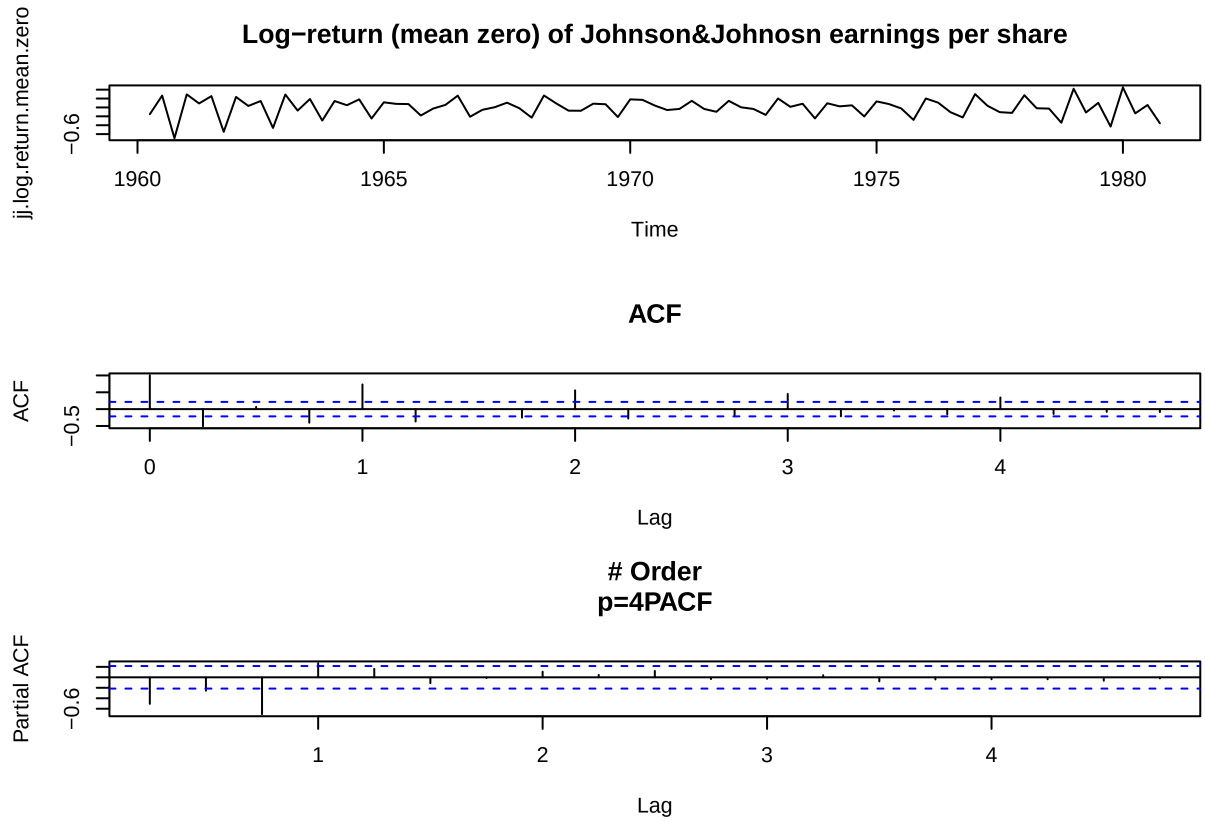
```r
cat("Constant:", phi0.hat," Coeffcinets:", phi.hat, " and Variance:", var.hat, '\n')
```

```
## Constant: 7.033036  Coeffcinets: 1.331587 -0.4445447  and Variance: 94.17131
```

```r
# Johnson Johnson model fitting

# Time plot for Johnson&Johnson
plot(JohnsonJohnson, main='Johnson&Johnosn earnings per share', col='blue', lwd=3)
# log-return of Johnson&Johnson
jj.log.return=diff(log(JohnsonJohnson))
jj.log.return.mean.zero=jj.log.return-mean(jj.log.return)
# Plots for log-returns
par(mfrow=c(3,1))
```




```
plot(jj.log.return.mean.zero, main='Log-return (mean zero) of Johnson&Johnosn earnings per share')
acf(jj.log.return.mean.zero, main='ACF')
pacf(jj.log.return.mean.zero, main='# Order
p=4PACF')
```




```
# sample autocorreleation function r
r=NULL
r[1:p]=acf(jj.log.return.mean.zero, plot=F)$acf[2:(p+1)]
r
```

```
## [1] -0.50681760  0.06710084
```

```
# matrix R
R=matrix(1,p,p) # matrix of dimension 4 by 4, with entries all 1's.

# define non-diagonal entires of R
for(i in 1:p){
    for(j in 1:p){
        if(i!=j)
            R[i,j]=r[abs(i-j)]
    }
}
R
```

```
##            [,1]       [,2]
## [1,]  1.0000000 -0.5068176
## [2,] -0.5068176  1.0000000
```

```
# b-column vector on the right
b=matrix(r,p,1)# b- column vector with no entries
b
```

```
##              [,1]
## [1,] -0.50681760
## [2,]  0.06710084
```

```
phi.hat=solve(R,b)[,1]
phi.hat
```

```
## [1] -0.6362359 -0.2553547
```

```
# Variance estimation using Yule-Walker Estimator
c0=acf(jj.log.return.mean.zero, type='covariance', plot=F)$acf[1]
c0
```

```
## [1] 0.04365692
```

```
var.hat=c0*(1-sum(phi.hat*r))
var.hat
```

```
## [1] 0.03032754
```

```
# Constant term in the model
phi0.hat=mean(jj.log.return)*(1-sum(phi.hat))
phi0.hat
```

```
## [1] 0.06368409
```

```
cat("Constant:", phi0.hat," Coeffcinets:", phi.hat, " and Variance:", var.hat, '\n')
```

```
## Constant: 0.06368409  Coeffcinets: -0.6362359 -0.2553547  and Variance: 0.03032754
```