AUTOCOVARIANCE ESTIMATION FOR TIME SERIES DATA

When represented with a real time series data set $\boldsymbol{x}_{1:n} = (x_1, \ldots, x_n)$ it is of interest to attempt to learn which model might be appropriate for modelling the variation. We focus on stationary models – this assumes that appropriate pre-processing in the form of

- de-meaning (eg trend removal)
- removal of seasonal (periodic) components
- differencing or seasonal differencing

has been carried out prior to the analysis, so that we may assume that the data can be represented by a stationary, zero-mean process.

Autocovariance Estimation: A common first task is to identify the autocovariance (and autocorrelation) structure. As we have seen previously, the recommended moment-based (or model-free) estimate of the autocovariance at lag h is

$$\widehat{\gamma}_X(h) = \frac{1}{n}\sum_{t=1}^{n-h}(x_{t+h} - \overline{x})(x_t - \overline{x}) \qquad h = 0, 1, \ldots, h_{\max}$$

where $h_{\max}$ is chosen such that the corresponding estimator has reasonable properties; in practice, we cannot choose $h_{\max}$ too large as this reduces the number of terms in the sum by too large an amount, which renders the estimator highly variable. By standard statistical theory, we have that as $n \longrightarrow \infty$ that for the estimator

$$\sqrt{n}(\widehat{\gamma}_X(h) - \gamma_X(h)) \xrightarrow{d} Normal(0, v_X(h)) \qquad h \in \mathbb{Z}$$

for some *asymptotic variance* $v_X(h)$. However, note that for a finite data set the corresponding large sample approximation

$$\widehat{\gamma}_X(h) \approx Normal(\gamma_X(h), \widehat{v}_X(h)/n)$$

may not hold if h is too large.

EXAMPLE: To illustrate the challenges, consider the ARMA$(2,3)$ model

$$X_t = 0.5X_{t-1} + 0.2X_{t-2} + Z_t + Z_{t-1} + 2Z_{t-2} - Z_{t-3}$$

```
phi.vec<-c(0.5,0.2);theta.vec<-c(1,2,-1);sigmaZ<-1
true.var<-sigmaZ^2*(1+sum(ARMAtoMA(ar=phi.vec,ma=theta.vec,lag.max=2000)^2))
true.var

+ [1] 14.55128

true.acvs<-true.var*ARMAacf(ar=phi.vec,ma=theta.vec,lag.max=20)
true.acvs[1:11]

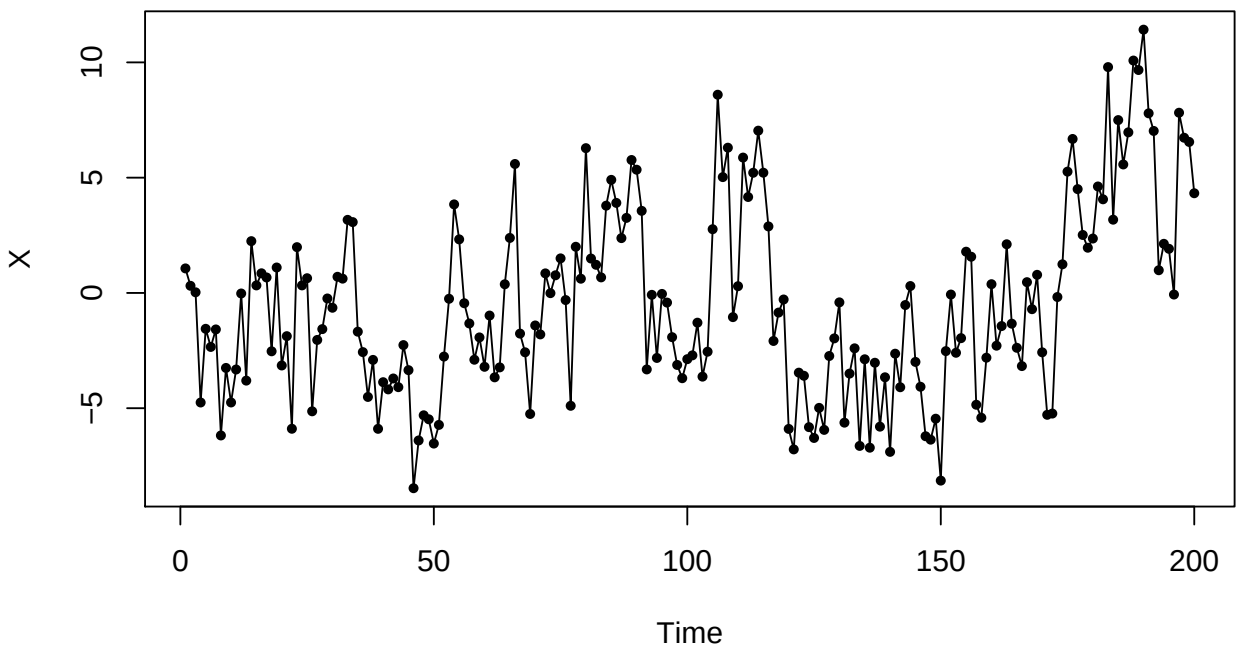
+          0          1          2          3          4          5
+ 14.5512821 10.4070513  8.6137821  5.3883013  4.4169071  3.2861138
+          6          7          8          9         10
+  2.5264383  1.9204419  1.4655086  1.1168427  0.8515231

true.acf<-true.acvs/true.var
true.acf[1:11]

+          0          1          2          3          4          5
+ 1.00000000 0.71519824 0.59196035 0.37029736 0.30354075 0.22582985
+          6          7          8          9         10
+ 0.17362307 0.13197751 0.10071337 0.07675218 0.05851877
```

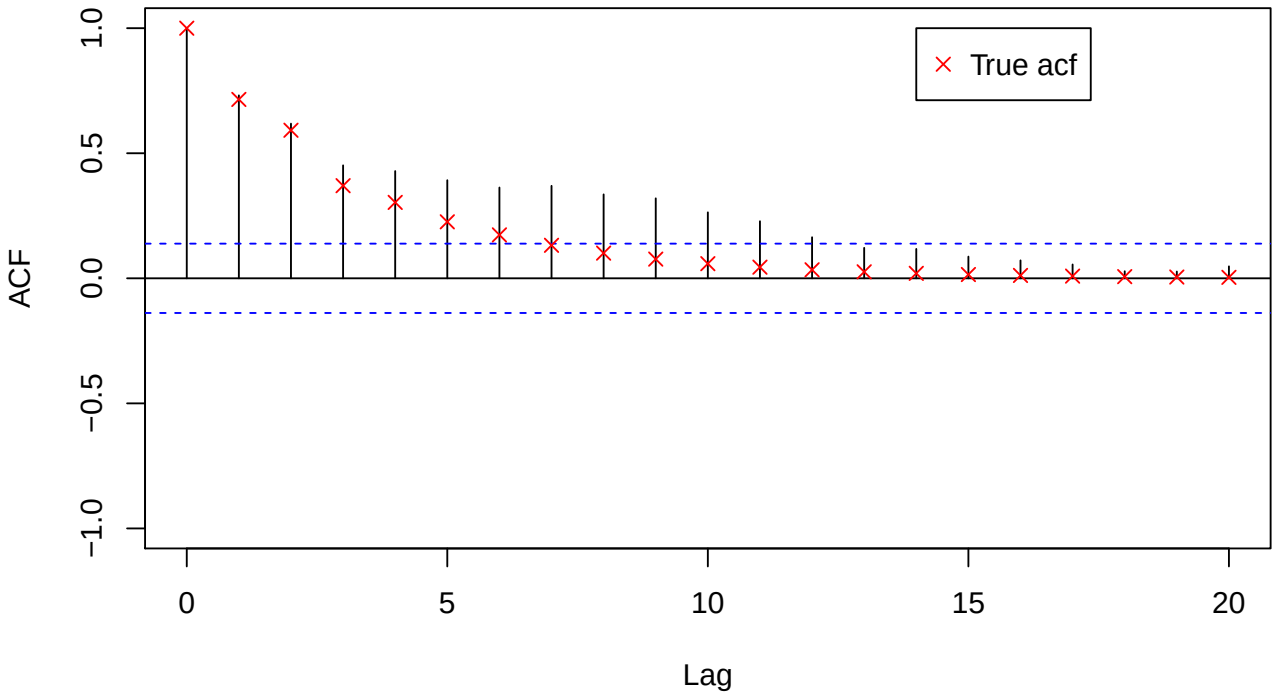
An example realization from this process (assuming a Gaussian white noise process with variance $\sigma_Z^2 = 1$) can be seen below

```r
set.seed(237);n<-200
X<-arima.sim(n,model=list(ar=phi.vec,ma=theta.vec))
par(mar=c(4,4,1,2));plot(X,type='l',xlab='Time');points(X,pch=19,cex=0.6)
```



The `acf` function plots the sample acf:

```r
par(mar=c(4,4,1,1))
acf(X,main='ACF for sample data',ylim=range(-1,1),lag.max=20)
points(0:20,true.acf,pch=4,col='red')
legend(14,1,c('True acf'),pch=4,col='red')
```



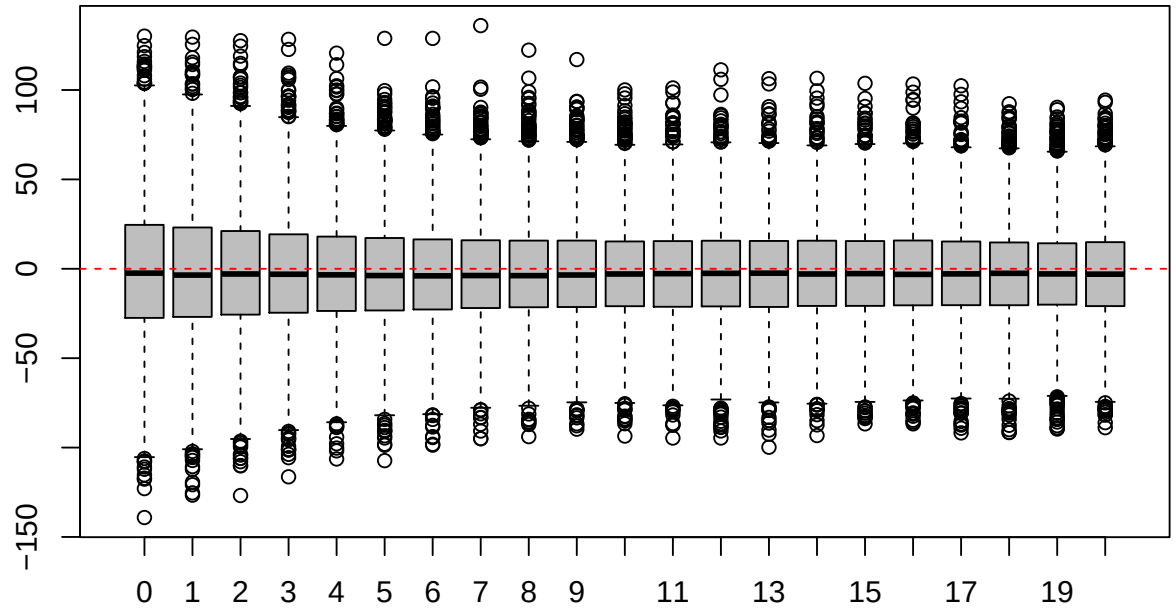
The dotted blue horizontal lines are at $\pm 1.96/\sqrt{n}$; they represent confidence bounds for the correlation assuming a null model that the process is a white noise process.

We can examine the statistical performance of the moment-based estimator in a simulation (or Monte Carlo) study. We generate 5000 independent instances of the series, estimate the correlation for each, and then summarize the results in a boxplot of the 'standardized' quantity

$$\sqrt{n}(\widehat{\gamma}_X(h) - \gamma_X(h))$$

We first do this for a very large sample of $n = 2000$:

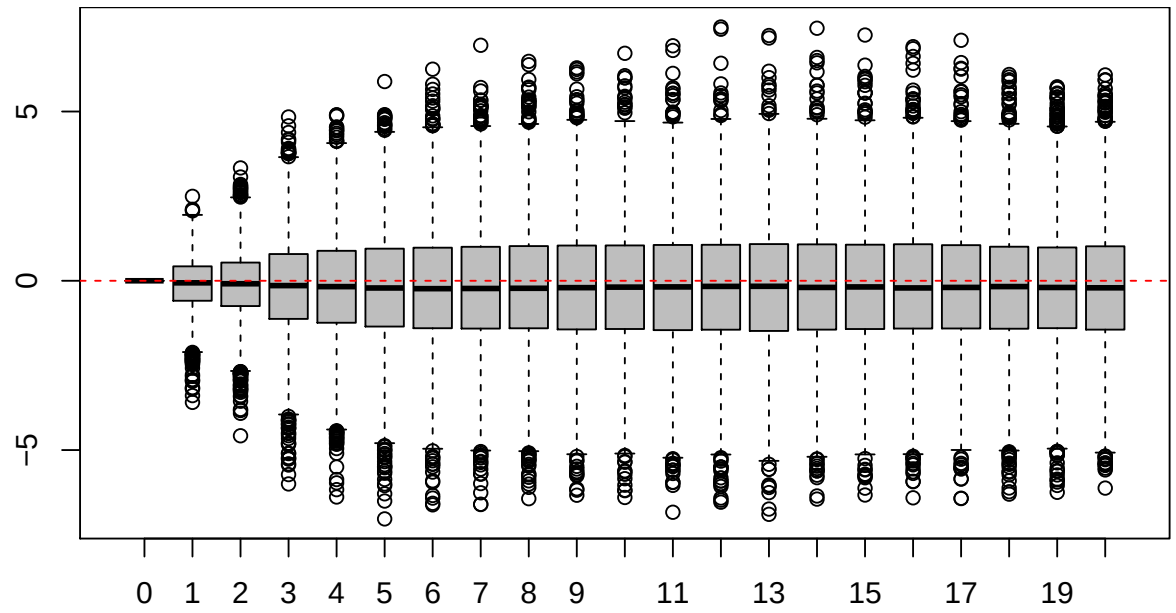
```
set.seed(237);n<-2000;nits<-5000;lmax<-20
gamma.hat<-rho.hat<-matrix(0,nrow=nits,ncol=lmax+1)
for(iter in 1:nits){
    X<-arima.sim(n,model=list(ar=phi.vec,ma=theta.vec))
    gamma.hat[iter,]<-acf(X,lag.max=lmax,plot=F,type='covariance')$acf
    rho.hat[iter,]<-gamma.hat[iter,]/gamma.hat[iter,1]
}
std.gamma.hat<-t(apply(gamma.hat,1,function(x){return(sqrt(n)*(x-true.acvs))})) #Standardize the estimates
std.rho.hat<-t(apply(rho.hat,1,function(x){return(sqrt(n)*(x-true.acf))}))      #Standardize the estimates
par(mar=c(4,4,0,2))
boxplot(std.gamma.hat,col='gray');abline(h=0,lty=2,col='red')
```




```
apply(std.gamma.hat,2,var)      #Compute the sampling variance of the estimator

+         0         1         2         3         4         5         6
+ 1441.9625 1342.8572 1184.7133 1054.5859  957.9531  891.2277  847.4864
+         7         8         9        10        11        12        13
+  814.8819  788.2386  767.5939  740.1154  731.4159  729.7457  717.0265
+        14        15        16        17        18        19        20
+  720.0722  715.2548  722.0339  708.5593  705.5363  697.3967  706.0338

boxplot(std.rho.hat,col='gray'); abline(h=0,lty=2,col='red')
```




```
apply(std.rho.hat,2,mean)      #Compute the expected value of the estimator

+           0           1           2           3           4           5
+  0.00000000 -0.09347263 -0.11627094 -0.17323685 -0.18298412 -0.20922968
+           6           7           8           9          10          11
+ -0.20791171 -0.20475852 -0.20219742 -0.19471397 -0.19510075 -0.19260982
+          12          13          14          15          16          17
+ -0.18848932 -0.19304002 -0.18889217 -0.18568950 -0.17706114 -0.17735521
+          18          19          20
+ -0.17636447 -0.19363959 -0.19289744

apply(std.rho.hat,2,var)      #Compute the sampling variance of the estimator

+         0         1         2         3         4         5         6
+ 0.0000000 0.5982961 0.9889849 2.1434132 2.4803109 2.8831536 3.1147014
+         7         8         9        10        11        12        13
+ 3.2614892 3.3336571 3.3764547 3.3425419 3.3694611 3.3986956 3.3625568
+        14        15        16        17        18        19        20
+ 3.3903029 3.3709057 3.4038404 3.3363790 3.3202073 3.2814180 3.3210410
```

The Monte Carlo study suggests that the standardized bias defined as

$$\sqrt{n}\mathbb{E}[(\widehat{\gamma}_X(h) - \gamma_X(h))]$$

is relatively low. It can be shown that the distribution of the vector of estimators of the first $h \ll n$ acf values, $\boldsymbol{\rho}_X(1:h) = (\rho_X(1), \ldots, \rho_X(h))^\top$ is approximately multivariate Normal, that is, as $n \longrightarrow \infty$, we have that

$$\sqrt{n}(\widehat{\rho}_X(1:h) - \rho_X(1:h)) \xrightarrow{d} Normal_h(\mathbf{0}, \mathbf{W})$$

where $\mathbf{W}$ is an $h \times h$ matrix with (i,j)th element

$$w_{ij} = \sum_{k=1}^{\infty}\{\rho_X(k+i) + \rho_X(k-i) - 2\rho_X(i)\rho_X(k)\}\{\rho_X(k+j) + \rho_X(k-j) - 2\rho_X(j)\rho_X(k)\};$$

this is known as **Bartlett's Formula**. Thus the diagonal elements are

$$w_{ii} = \sum_{k=1}^{\infty}\{\rho_X(k+i) + \rho_X(k-i) - 2\rho_X(i)\rho_X(k)\}^2 \qquad i = 1, \ldots, h.$$

The matrix $\mathbf{W}$ may be estimated by using the sample acf values.

```
hmax<-20
W<-matrix(0,hmax,hmax)
lmax<-2000
true.acf<-ARMAacf(ar=phi.vec,ma=theta.vec,lag.max=lmax+hmax)
for(i in 1:hmax){
    i1<-c(1:lmax)+i
    i2<-c(1:lmax)-i
    ti<-true.acf[i1+1]+true.acf[abs(i2)+1]-2*true.acf[i+1]*true.acf[1:lmax+1]
    for(j in 1:hmax){
        j1<-c(1:lmax)+j
        j2<-c(1:lmax)-j
        tj<-true.acf[j1+1]+true.acf[abs(j2)+1]-2*true.acf[j+1]*true.acf[1:lmax+1]
        W[i,j]<-sum(ti*tj)
    }
}
diag(W)

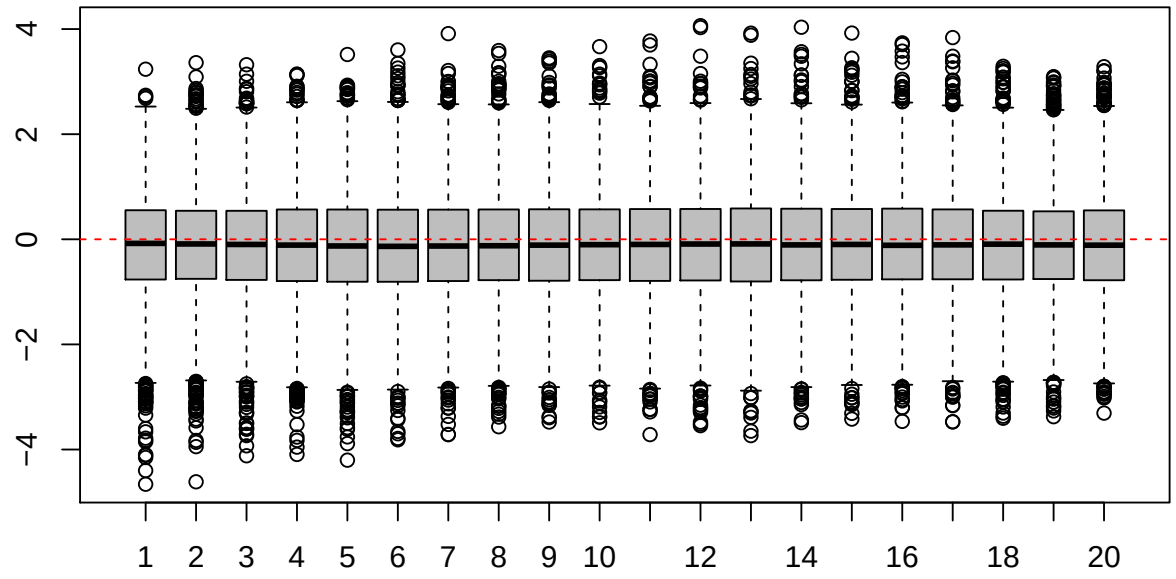
+  [1] 0.5938315 0.9842955 2.1225279 2.4342811 2.8009380 3.0096060 3.1601168
+  [8] 3.2562773 3.3192583 3.3594983 3.3851113 3.4012652 3.4113963 3.4177147
+ [15] 3.4216373 3.4240624 3.4255565 3.4264739 3.4270357 3.4273787
```

Therefore, according to this theory, for the lag-h acf, we have

$$\sqrt{n}\frac{(\widehat{\rho}_X(h)-\rho_X(h))}{\sqrt{w_{hh}}} \xrightarrow{d} Normal(0,1)$$

which we can assess in our simulation:

```
w.vec.root<-sqrt(diag(W))
std.rho.hat.final<-t(apply(std.rho.hat[,-1],1,function(x){return(x/w.vec.root)}))
par(mar=c(4,4,1,2))
boxplot(std.rho.hat.final,col='gray');abline(h=0,lty=2,col='red')
```



```
apply(std.rho.hat.final,2,var)      #Compute the sampling variance of the estimator

+         1         2         3         4         5         6         7
+ 1.0075183 1.0047643 1.0098398 1.0189090 1.0293529 1.0349200 1.0320787
+         8         9        10        11        12        13        14
+ 1.0237633 1.0172317 0.9949527 0.9953768 0.9992445 0.9856834 0.9919795
+        15        16        17        18        19        20
+ 0.9851733 0.9940941 0.9739670 0.9689866 0.9575091 0.9689740
```

This plot and collection of numerical values demonstrate that the asymptotic theory is holding for this sample size ($n = 2000$) as the sampling variances after standardization are all near one. In a single data set analysis, in a real application, we would have to estimate the $\mathbf{W}$ matrix also.

```
set.seed(237);n<-2000;lmax<-20
gamma.hat.1<-rho.hat.1<-rep(0,lmax+1)
X<-arima.sim(n,model=list(ar=phi.vec,ma=theta.vec))
hmax<-20
W.hat<-matrix(0,hmax,hmax)
lmax<-30
est.acf<-c(acf(X,plot=F,lag.max=lmax)$acf,rep(0,1000))
est.acf[1:30]

+  [1]  1.000000000  0.710444156  0.556062507  0.316896491  0.234319090
+  [6]  0.164010608  0.122338408  0.094165646  0.077996659  0.071580833
+ [11]  0.059461321  0.040922433  0.020469451  0.012050337  0.006490687
+ [16] -0.007407064 -0.004212646  0.008460077  0.025981739  0.043916123
+ [21]  0.061338853  0.064659587  0.064610289  0.045925726  0.035277242
+ [26]  0.028902529  0.010400002  0.009159115  0.004996961  0.022006456

for(i in 1:hmax){
    i1<-c(1:lmax)+i
    i2<-c(1:lmax)-i
    ti<-est.acf[i1+1]+est.acf[abs(i2)+1]-2*est.acf[i+1]*est.acf[1:lmax+1]
    for(j in 1:hmax){
        j1<-c(1:lmax)+j
        j2<-c(1:lmax)-j
        tj<-est.acf[j1+1]+est.acf[abs(j2)+1]-2*est.acf[j+1]*est.acf[1:lmax+1]
        W.hat[i,j]<-sum(ti*tj)
    }
}
diag(W.hat)

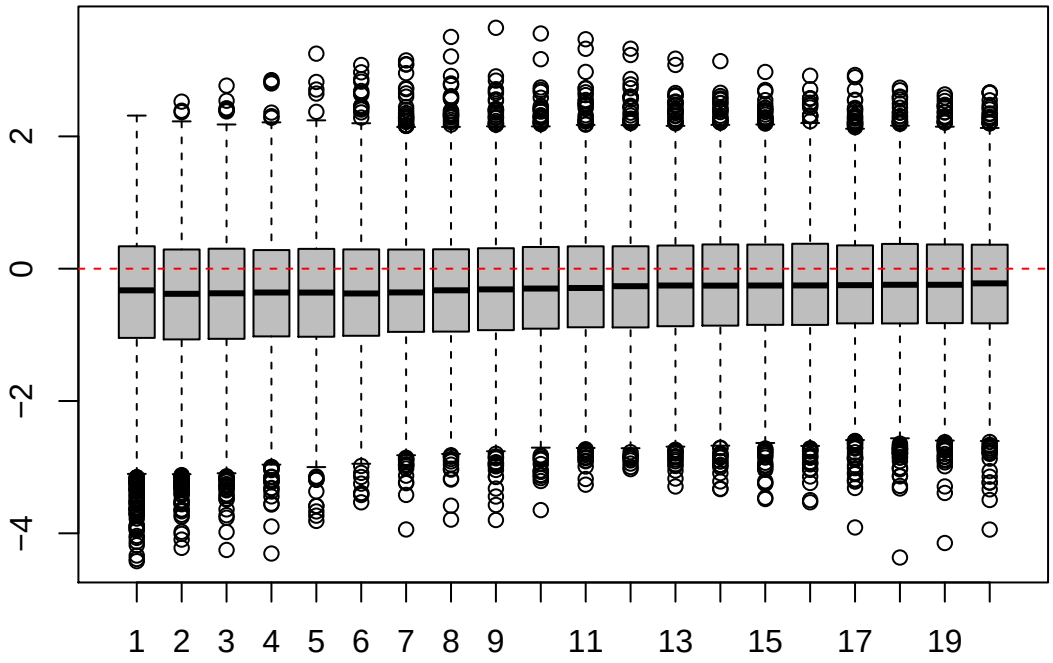
+  [1] 0.5330057 1.0155490 2.1797750 2.6163818 2.9206283 2.9981462 3.0287580
+  [8] 3.1032805 3.2671741 3.4501474 3.5110242 3.4466401 3.3428891 3.2956818
+ [15] 3.2843142 3.1978907 3.1331970 3.0879543 3.0531690 3.0226475
```

In this calculation, we have assumed that the acf is zero beyond lag 30; this is because the Bartlett Formula uses acf values at all positive lags, but the estimation becomes more variable quite quickly.

The estimation of the autocovariances is, however, much more challenging when the sample size is smaller. Consider the following simulation study when $n = 200$;

```
true.acvs<-true.var*ARMAacf(ar=phi.vec,ma=theta.vec,lag.max=20)
true.acf<-true.acvs/true.var
set.seed(237);n<-200;nits<-5000;lmax<-20
gamma.hat<-rho.hat<-matrix(0,nrow=nits,ncol=lmax+1)
for(iter in 1:nits){
    X<-arima.sim(n,model=list(ar=phi.vec,ma=theta.vec))
    gamma.hat[iter,]<-acf(X,lag.max=lmax,plot=F,type='covariance')$acf
    rho.hat[iter,]<-gamma.hat[iter,]/gamma.hat[iter,1]
}
std.gamma.hat<-t(apply(gamma.hat,1,function(x){return(sqrt(n)*(x-true.acvs))})) #Standardize the estimates
```

```
std.rho.hat<-t(apply(rho.hat,1,function(x){return(sqrt(n)*(x-true.acf))}))      #Standardize the estimates
w.vec.root<-sqrt(diag(W))
std.rho.hat.final<-t(apply(std.rho.hat[,-1],1,function(x){return(x/w.vec.root)}))
par(mar=c(4,4,1,2));boxplot(std.rho.hat.final,col='gray');abline(h=0,lty=2,col='red')
```




```
apply(std.rho.hat.final,2,mean)      #Expected value of the estimator (should be zero)

+          1          2          3          4          5          6
+ -0.3929837 -0.4174066 -0.3887334 -0.3805820 -0.3683742 -0.3568771
+          7          8          9         10         11         12
+ -0.3344497 -0.3206494 -0.3008357 -0.2918715 -0.2790247 -0.2683705
+         13         14         15         16         17         18
+ -0.2579897 -0.2495725 -0.2462800 -0.2422623 -0.2413727 -0.2352355
+         19         20
+ -0.2345425 -0.2303971

apply(std.rho.hat.final,2,var)      #Sampling variance of the estimator (should be one)

+         1         2         3         4         5         6         7
+ 1.0906607 1.0282749 0.9748589 0.9424465 0.9130587 0.8852233 0.8581751
+         8         9        10        11        12        13        14
+ 0.8452364 0.8470401 0.8410680 0.8346256 0.8410732 0.8328174 0.8258737
+        15        16        17        18        19        20
+ 0.8065561 0.7986100 0.7900063 0.7874685 0.7841517 0.7749167
```

It is clear from these figures that there is considerable **bias** in the estimation.

EXAMPLE: Consider the AR(2) model

$$X_t = 0.2X_{t-1} + 0.2X_{t-2} + Z_t.$$

For this model, the polynomial $\Phi(\mathrm{B})$ is

$$\Phi(\mathrm{B}) = 1 - 0.2\mathrm{B} - 0.2\mathrm{B}^2 = (1 - \alpha_1\mathrm{B})(1 - \alpha_2\mathrm{B})$$

where $\alpha_1 + \alpha_2 = 0.2$, $\alpha_1\alpha_2 = -0.2$. Note that

$$\Phi(z) = 1 - \phi_1 z - \phi_2 z^2 = 0 \qquad \Longrightarrow \qquad \frac{1}{\alpha_1\alpha_2} - \frac{\alpha_1 + \alpha_2}{\alpha_1\alpha_2} z - z^2 = 0$$

provided $|\alpha_1|, |\alpha_2| > 0$, and hence

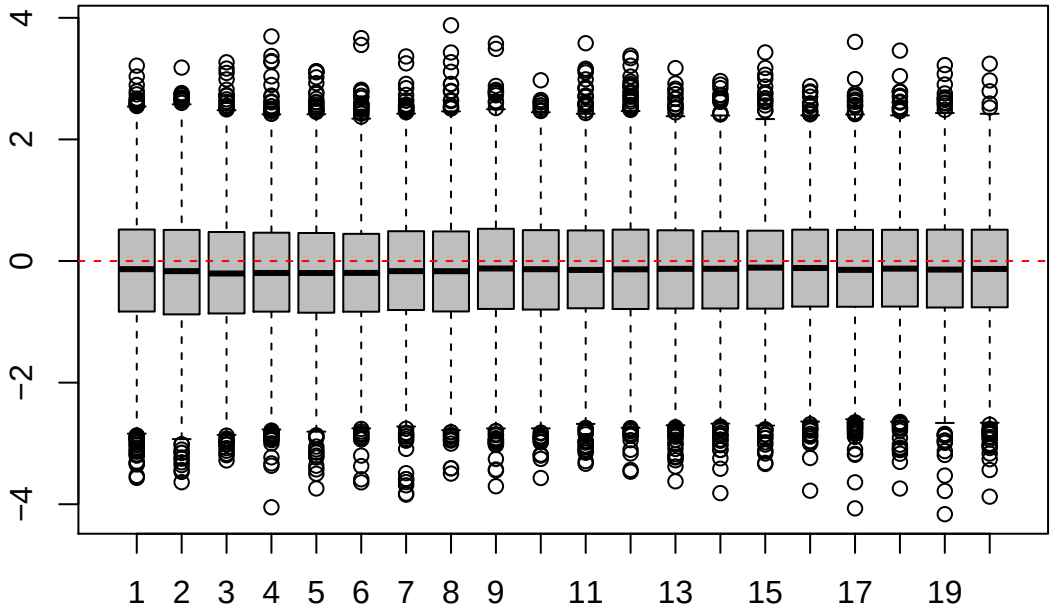
$$\left(z - \frac{1}{\alpha_1}\right)\left(z - \frac{1}{\alpha_2}\right) = 0.$$

This we can find the reciprocal roots using `polyroot` in `R`, as follows

```
alpha<-1/polyroot(c(1,-0.2,-0.2))
```

From this we deduce that α_1 and α_2 are $\alpha_1 = 0.5583 + 0i$, $\alpha_2 = -0.3583 + 0i$ and the model is stationary. Consider the following simulation study when $n = 200$;

```
phi.vec<-c(0.2,0.2);sigmaZ<-1
true.var<-sigmaZ^2*(1+sum(ARMAtoMA(ar=phi.vec,lag.max=2000)^2))
true.acvs<-true.var*ARMAacf(ar=phi.vec,lag.max=20)
true.acf<-true.acvs/true.var
set.seed(237);n<-200;nits<-5000;lmax<-20
gamma.hat<-rho.hat<-matrix(0,nrow=nits,ncol=lmax+1)
for(iter in 1:nits){
    X<-arima.sim(n,model=list(ar=phi.vec))
    gamma.hat[iter,]<-acf(X,lag.max=lmax,plot=F,type='covariance')$acf
    rho.hat[iter,]<-gamma.hat[iter,]/gamma.hat[iter,1]
}
std.gamma.hat<-t(apply(gamma.hat,1,function(x){return(sqrt(n)*(x-true.acvs))})) #Standardize the estimates
std.rho.hat<-t(apply(rho.hat,1,function(x){return(sqrt(n)*(x-true.acf))}))      #Standardize the estimates
hmax<-20;W<-matrix(0,hmax,hmax);lmax<-2000
true.acf<-ARMAacf(ar=phi.vec,lag.max=lmax+hmax)
for(i in 1:hmax){
    i1<-c(1:lmax)+i
    i2<-c(1:lmax)-i
    ti<-true.acf[i1+1]+true.acf[abs(i2)+1]-2*true.acf[i+1]*true.acf[1:lmax+1]
    for(j in 1:hmax){
        j1<-c(1:lmax)+j
        j2<-c(1:lmax)-j
        tj<-true.acf[j1+1]+true.acf[abs(j2)+1]-2*true.acf[j+1]*true.acf[1:lmax+1]
        W[i,j]<-sum(ti*tj)
    }
}
w.vec.root<-sqrt(diag(W))
std.rho.hat.final<-t(apply(std.rho.hat[,-1],1,function(x){return(x/w.vec.root)}))
par(mar=c(4,4,1,2))
boxplot(std.rho.hat.final,col='gray');abline(h=0,lty=2,col='red')
```




```
apply(std.rho.hat.final,2,mean)      #Expected value of the estimator (should be zero)

+           1           2           3           4           5           6
+ -0.1645267 -0.1855416 -0.1867278 -0.1763829 -0.1877539 -0.1797615
+           7           8           9          10          11          12
+ -0.1609284 -0.1653697 -0.1261525 -0.1427719 -0.1406311 -0.1435451
+          13          14          15          16          17          18
+ -0.1392815 -0.1379694 -0.1322735 -0.1283154 -0.1282011 -0.1296573
+          19          20
+ -0.1380905 -0.1289109

apply(std.rho.hat.final,2,var)      #Sampling variance of the estimator (should be one)

+         1         2         3         4         5         6         7
+ 0.9956602 0.9971202 0.9692941 0.9482326 0.9529278 0.9170073 0.9335741
+         8         9        10        11        12        13        14
+ 0.9096266 0.9328950 0.9020289 0.9225235 0.9409703 0.9057281 0.9148750
+        15        16        17        18        19        20
+ 0.8843549 0.8685277 0.8678458 0.8692688 0.8656248 0.8678129
```

Here there is less bias in the estimation, and the asymptotic approximation is somewhat better.

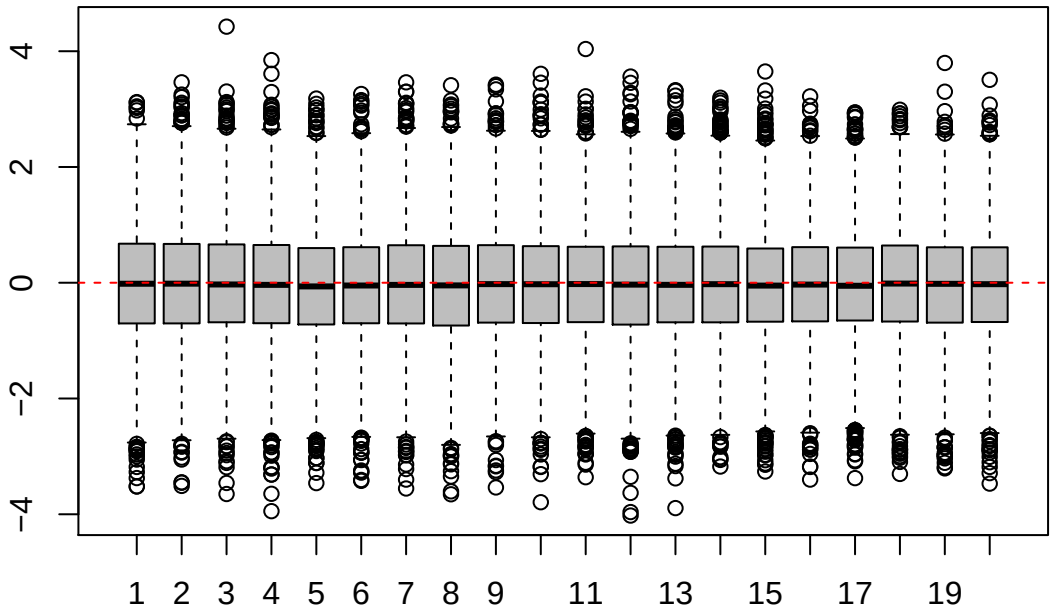
EXAMPLE: Consider the MA(3) model

$$X_t = Z_t + 2Z_{t-1} + 4Z_{t-2} - 3Z_{t-3}$$

Here the autocorrelation is zero for lags $h > 3$. Consider the following simulation study when $n = 200$;

```
theta.vec<-c(2,4,-3);sigmaZ<-1
true.var<-sigmaZ^2*(1+sum(ARMAtoMA(ma=theta.vec,lag.max=2000)^2))
true.acvs<-true.var*ARMAacf(ma=theta.vec,lag.max=20)
true.acf<-true.acvs/true.var
set.seed(237);n<-200;nits<-5000;lmax<-20
gamma.hat<-rho.hat<-matrix(0,nrow=nits,ncol=lmax+1)
for(iter in 1:nits){
    X<-arima.sim(n,model=list(ma=theta.vec))
```

```
    gamma.hat[iter,]<-acf(X,lag.max=lmax,plot=F,type='covariance')$acf
    rho.hat[iter,]<-gamma.hat[iter,]/gamma.hat[iter,1]
}
std.gamma.hat<-t(apply(gamma.hat,1,function(x){return(sqrt(n)*(x-true.acvs))})) #Standardize the estimates
std.rho.hat<-t(apply(rho.hat,1,function(x){return(sqrt(n)*(x-true.acf))}))      #Standardize the estimates
hmax<-20;W<-matrix(0,hmax,hmax)
lmax<-2000
true.acf<-ARMAacf(ma=theta.vec,lag.max=lmax+hmax)
for(i in 1:hmax){
    i1<-c(1:lmax)+i
    i2<-c(1:lmax)-i
    ti<-true.acf[i1+1]+true.acf[abs(i2)+1]-2*true.acf[i+1]*true.acf[1:lmax+1]
    for(j in 1:hmax){
        j1<-c(1:lmax)+j
        j2<-c(1:lmax)-j
        tj<-true.acf[j1+1]+true.acf[abs(j2)+1]-2*true.acf[j+1]*true.acf[1:lmax+1]
        W[i,j]<-sum(ti*tj)
    }
}
w.vec.root<-sqrt(diag(W))
std.rho.hat.final<-t(apply(std.rho.hat[,-1],1,function(x){return(x/w.vec.root)}))
par(mar=c(4,4,1,2));boxplot(std.rho.hat.final,col='gray');abline(h=0,lty=2,col='red')
```




```
apply(std.rho.hat.final,2,mean)      #Expected value of the estimator (should be zero)

+           1           2           3           4           5           6
+ -0.02626454 -0.01001287 -0.01118111 -0.02331945 -0.06496739 -0.05187266
+           7           8           9          10          11          12
+ -0.02776586 -0.04710220 -0.01735125 -0.03645971 -0.03030430 -0.03974097
+          13          14          15          16          17          18
+ -0.03693738 -0.02407596 -0.04421563 -0.02840304 -0.03730426 -0.01528664
+          19          20
+ -0.03585876 -0.03030525

apply(std.rho.hat.final,2,var)      #Sampling variance of the estimator (should be one)
```

```
+         1         2         3         4         5         6         7
+ 1.0126741 0.9941071 0.9970929 1.0098269 0.9628032 0.9507922 0.9629617
+         8         9        10        11        12        13        14
+ 0.9754340 0.9622944 0.9323666 0.9242198 0.9852227 0.9452925 0.9464319
+        15        16        17        18        19        20
+ 0.9110482 0.9100629 0.9238433 0.9182118 0.9007260 0.8873947
```

Here there almost no in the bias in the estimation of the acf, and the asymptotic approximation is somewhat better, although the asymptotic variance seems to be too small as h increases.