Lab 4 - Learning Via Computer Simulation

Benjamin M. Taylor, Kevin Hayes

January 20, 2023

1 AR(1) Process

The following R code simulates and plots an AR(1) time series process of length $n = 1001$ having parameter $\phi = 0.7$ with $\sigma_e^2 = 1$, plots the empirical / estimated ACF (correlogram) and PACF (partial correlogram) and their theoretical counterparts. Write the code into an R script file and execute the code from there.

[ ]:
```
options(repr.plot.width=15, repr.plot.height=15) # makes plots bigger in the␣
 ↪webpage
```

[ ]:
```
alpha = 0.7
Yt = arima.sim(model = list(ar = alpha), n = 101)
```

[ ]:
```
AR1.acf = ARMAacf(ar = alpha, lag.max = 25)
AR1.pacf = ARMAacf(ar = alpha, lag.max = 25, pacf = TRUE)
names(AR1.pacf) = names(AR1.acf)[-1]

layout(mat= matrix(c(1,1,2:5), byrow=TRUE, ncol=2))

plot(Yt, main= "Simulated AR(1) Process")
acf(Yt, lag.max = 25, main= "Estimated ACF")
pacf(Yt, lag.max = 25, main= "Estimated PACF")
barplot(AR1.acf, col = "blue", main= "Theoretical ACF")
barplot(AR1.pacf, col = "red", main= "Theoretical PACF")
```

Execute the code 3 times to make sure the results are consistent.

- Repeat the previous exercise using the value $\phi = -0.7$.

- Investigate $\phi = 0.7$ and then $\phi = -0.7$ when $n = 101$.
- Investigate $\phi = 0.99$ and $n = 1001$.
- Investigate $\phi = -0.99$ and $n = 1001$.

2 AR(2) Process

The following R command simulates the AR(2) process $Y_t = 0.7Y_{t-1} - 0.4Y_{t-2} + e_t$ with $\sigma_e^2 = 1$. Plot the data and investigate the shape of the ACF and PACF. Repeat three times.

[ ]:
```
Yt = arima.sim(list(ar = c(0.7, -0.4)), n=1001)
```

[ ]:
```

```

Repeat the previous exercise using a shorter time series. Try the AR(2) model with $\phi_1 = 0.7$ and $\phi_2 = 0.4$.

What went wrong?

[ ]:
```

```

3 Stationarity of the AR(2) Process

The AR(2) model $(1-aB)(1-bB)Y_t = e_t$ expands to

$$Y_t = \overbrace{(a+b)}^{\phi_1} Y_{t-1} + \overbrace{(-ab)}^{\phi_2} Y_{t-2} + e_t.$$

For this model to be stationary we require that the roots (both real and complex) of the generating polynomial $\phi(B) = (1-aB)(1-bB) = 0$ lie outside the unit circle. That is, we require $|a| < 1$ and $|b| < 1$. Also, as there are only two roots, if complex roots arise then $b = \bar{a}$. We can investigate this algebraically elsewhere. Here can use computer simulation to map and colour the admissible regions in the plane (ϕ_1, ϕ_2).

Specifying admissible real roots is straightforward. Specifying admissible complex roots we need to write $a = x + iy$ and $b = \bar{a} = x - iy$, so that $\phi_1 = 2x$ and $\phi_2 = -(x^2 + y^2)$. The following R code will do this:

[ ]:
```
n = 5000
a = runif(n, min= -1, max = 1)
b = runif(n, min= -1, max = 1)
Phi.RootsReal = cbind( a+b , -a*b )
```

[ ]:
```
x = runif(n, min= -1, max = 1)
y = numeric()
for(i in 1:n){
    y[i] = runif(1, -1*sqrt(1-(x[i])^2), sqrt(1-(x[i])^2))
}

Phi.RootsComplex = cbind( 2*x , -(x^2 +y^2) )
```

[ ]:
```
plot(Phi.RootsReal, col="blue", xlab= expression(phi[1]),ylab=␣
 ↪expression(phi[2]))
points(Phi.RootsComplex, col="magenta")

curve(1-x, add = TRUE, lwd = 3)         # condition phi1 + phi2 < 1
curve(1+x, add = TRUE, lwd = 3)         # condition phi2 - phi1 < 1
```

```
curve(-x^2/4, add = TRUE, lwd = 3)     # condition (phi1)^2 + 4phi2 < 0
```

[ ]:
```
CheckComplexRoots = (Phi.RootsComplex[,1])^2 + 4* Phi.RootsComplex[,2]
all(CheckComplexRoots < 0)
```

4 ARMA(p,q) Process

Investigate the simulated models in the following R code:

[ ]:
```
Yt = arima.sim(n = 100, list(ar = c(0.9,-0.5), ma = c(-0.2, 0.25)), sd = sqrt(2))

# mildly long-tailed
Xt = arima.sim(n = 100, list(ar=c(0.9,-0.5), ma=c(-0.2,0.25)),
               sd=sqrt(2), rand.gen=function(n,...){return(sqrt(0.2)*rt(n,df=5))})

Zt = arima.sim(list(ar = 0.7, ma = -0.7), n = 500)
```

5 Yule-Walker Equations

Consider again the time series `lh` measuring the Luteinizing hormone level in blood samples at 10 mins intervals from a human female, 48 samples. The R command `ar(lh)` automatically selects and fits an AR(3) model to these data. Try it. The default fitting algorithm of `ar` is based on the Yule-Walker equations. The theoretical Yule-Walker equations for an AR(3) model can be written

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_1 & \gamma_2 \\ \gamma_1 & \gamma_0 & \gamma_1 \\ \gamma_2 & \gamma_1 & \gamma_0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}$$

with

$$\gamma_0 = \sum_{j=1}^{3} \phi_j \gamma_j$$

If we replace the theoretical covariances with their sample-statistics counterparts and solve, we should obtain estimate $\hat{\phi}_1$, $\hat{\phi}_2$, and $\hat{\phi}_3$ for the AR(3) model.

The following R code does this manually to show how the magic-box answers from `ar(lh)` are obtained.

[ ]:
```
ar(lh)
```

```
[ ]: lh.acf = acf(lh, lag.max=3)
     lh.acf
```

```
[ ]: names(lh.acf)
```

```
[ ]: gamma.vec = matrix(data = lh.acf$acf[-1], ncol=1)

     Gamma.mtx = diag(3)
     diag(Gamma.mtx) = lh.acf$acf[1]
     Gamma.mtx[abs(row(Gamma.mtx)-col(Gamma.mtx)) == 1] = lh.acf$acf[2]
     Gamma.mtx[abs(row(Gamma.mtx)-col(Gamma.mtx)) == 2] = lh.acf$acf[3]
```

```
[ ]: gamma.vec
```

```
[ ]: Gamma.mtx
```

```
[ ]: solve(Gamma.mtx, gamma.vec)
```

Finally, have a look at the sunspot data...

```
[ ]: (sunspot.ar <- ar(sunspot.year))
     sunspot.lead = predict(sunspot.ar, n.ahead = 25)

     ts.plot(sunspot.year, sunspot.lead$pred, col=c(5,2))
```

6 ARIMA(p,d,q) Models

The R code below simulates the time series model:

$$
\begin{aligned}
(1-1.7B+0.7B^2)Y_t &= e_t, \\
(1-0.7B)(1-B)Y_t &= e_t, \\
(1-B)Y_t &= 0.7(1-B)Y_{t-1}+e_t, \\
(Y_t-Y_{t-1}) &= 0.7(Y_{t-1}-Y_{t-2})+e_t, \\
Y_t &= 1.7Y_{t-1}-0.7Y_{t-2}+e_t
\end{aligned}
$$

Note that the coefficients of Y on the LHS sum $1.7-0.7=1$. This is a quick check to spot that $B=1$ is a root of the AR characteristic polynomial $\phi(B)=0$.

```
[ ]: ts.sim = arima.sim(list(order = c(1,1,0), ar = 0.7), n = 500)
     d = diff(ts.sim)

     ts.plot(ts.sim)
```

[ ]:
```
ts.plot(d)
```

[ ]:
```
par(mfrow=c(2,2))

acf(ts.sim)
pacf(ts.sim)
acf(d)
pacf(d)
```

Note: The easiest way to get the roots of the polynomial (1-1.7B+ 0.7B^2)=0 is the R command

[ ]:
```
polyroot(z = c(0.7,-1.7,1) )
```

7 Simulating Time Series in R

Use `arima.sim()` to simulate from the model

$$Y_t = 2.7Y_{t-1} - 2.4Y_{t-2} + 0.7Y_{t-3} + e_t - 0.5e_{t-1}.$$

You should produce time series plots, ACF plots, PACF plots as necessary. It is a good idea to use the command `set.seed()` so that your simulation results can be reproduced. **Use your own student ID number as the seed number.** Compare the ACF (and PACF) plots to their theoretical counterparts.

Upload a "boilerplate" plot of the time series, the ACF and the PACF.