Problem Set 3, Econ 211C

Swati Sharma

Question 1

Consider the $ARMA(1,1)$ process

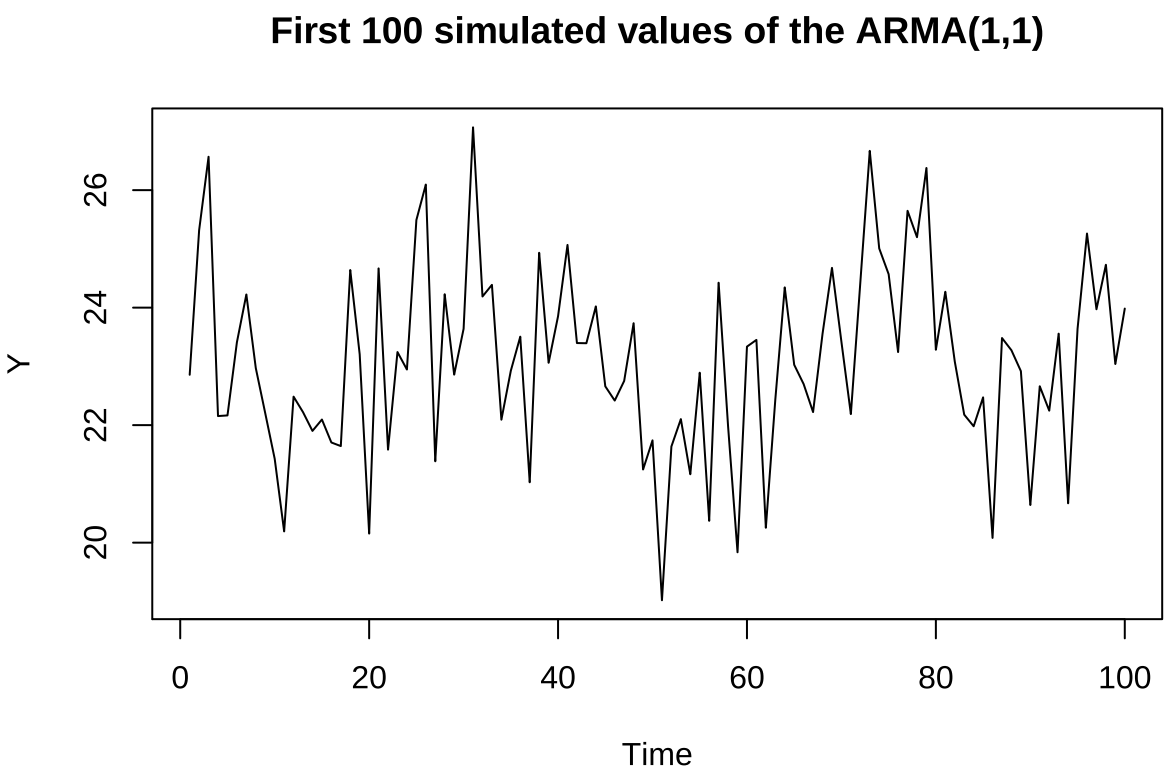
$$Y_t = 3.2 + 0.86Y_{t-1} + \varepsilon_t - 1.4\varepsilon_{t-1},\ \varepsilon_t \sim WN(0,1). \tag{1}$$

Simulate $n = 1010$ observations of this process.

Solution:

```r
n = 1010
cc = 3.2
phi = 0.86
theta = -1.4
eps = rnorm(n,0,1)
y = rep(NA,n)
y[1] = cc + phi*cc/(1-phi)
for(i in 2:n){
  y[i] = cc + phi*y[i-1] + eps[i] + theta*eps[i-1]
}
```

```r
plot.ts(y[1:100],  main = paste("First 100 simulated values of the ARMA(1,1)"), ylab = "Y")
```



a. (10 points)

Use the first 1000 observations from your simulation to estimate an $ARMA(1,1)$ model. Feel free to use the `arima` function. Report your parameter estimates with standard errors and the variance of the residuals. Use your the parameter estimates to compute and report forecasts for $s = 1, \ldots, 10$. Do not use any pre-packaged functions in `R` – compute and report the forecasts using basic mathematical operations.

Solution:

```
arma11Est = arima(y[1:1000],order=c(1,0,1))
cat("Parameter Estimates: \n")
```

```
## Parameter Estimates:
```

```
arma11Est$coef
```

```
##        ar1        ma1  intercept
##  0.8592349 -0.7232978 22.8489284
```

```
cat("\n", "Standard Errors: \n")
```

```
##
##  Standard Errors:
```

```
sqrt(diag(arma11Est$var.coef))
```

```
##        ar1        ma1  intercept
## 0.04276107 0.05724630 0.08637614
```

```
cat("\n", "Variance of Residuals:", var(arma11Est$residuals))
```

```
##
```

```
##  Variance of Residuals: 1.944992
```

```r
steps = 10
muHat = arma11Est$coef[3]
phiHat = arma11Est$coef[1]
thetaHat = arma11Est$coef[2]
yHat = rep(NA,steps)
yHat[1] =  muHat + phiHat*(y[1000]-muHat) + thetaHat*eps[1000]
for(s in 2:steps){
  yHat[s] = muHat + phiHat*(yHat[s-1]-muHat)
}
```

```r
cat("Forecast: \n", yHat)
```

```
## Forecast:
##  24.3121 24.10614 23.92917 23.77711 23.64645 23.53419 23.43773 23.35485 23.28363 23.22244
```

b. (10 points)

Estimate an $AR(1)$ model using the first 1000 observations from your simulation. Do not use the `arima` function or any other pre-packaged function. Instead, compute the esimates with matrix operations. Use the parameter estimates to compute and report forecasts for $s = 1, \ldots, 10$. Do not use any pre-packaged functions in `R` – compute and report the forecasts using basic mathematical operations.

Solution:

```r
Y = y[1:(n-10)] ## first 1000 simulated values of y
X = cbind(rep(1,n-10),y[2:(n-9)])  ## X matrix of 1000 ones and lagged values of y (t-1)
beta = solve(t(X)%*%X)%*%t(X)%*%Y  ## solving for coefficient
cat("Parameter Estimates: \n", beta)
```

```
## Parameter Estimates:
##  18.51125 0.1895875
```

```r
steps = 10
phiHatAR = beta[2]
muHatAR = beta[1]/(1-phiHatAR)
yHatAR = rep(NA,steps)
yHatAR[1] =  muHatAR + phiHatAR*(y[1000]-muHatAR)
yHatAR[2] =  muHatAR + phiHatAR*(yHat[1]-muHatAR)
for(s in 3:steps){
  yHatAR[s] = muHatAR + phiHatAR*(yHatAR[s-1]-muHatAR)
}
cat("Forecast: \n", yHat)
```

```
## Forecast:
##  24.3121 24.10614 23.92917 23.77711 23.64645 23.53419 23.43773 23.35485 23.28363 23.22244
```

c. (15 points)

Compute the theoretical MSE values for the forecasts associated with $s = 1, \ldots, 10$, using both the $ARMA(1,1)$ and $AR(1)$ parameter estimates.

Solution:

```
## calculating psi coefficients for MA infinite representation of ARMA(1,1)
ARMApsi = ARMAtoMA(arma11Est$coef[1], arma11Est$coef[2], 1)

ar1Est = arima(y[1:1000],order=c(1,0,0))

ARMAmse = matrix(NA,10,1)
ARmse = matrix(NA,10,1)

for (i in 1:10) {
  ## calculating theoretical MSE for ARMA(1,1)
  ARMAmse[i,] = var(arma11Est$residuals)*sum(ARMApsi^(2*i))
  ## calculating theoretical MSE for AR(1)
  ARmse[i,] = var(ar1Est$residuals)*sum(beta[2]^(2*i))
}
cat("Theoretical MSE for forecast using ARMA(1,1) \n", ARMAmse)
```

```
## Theoretical MSE for forecast using ARMA(1,1)
##  0.03594132 0.0006641563 1.227288e-05 2.267895e-07 4.190821e-09 7.74418e-11 1.43104e-12 2.644405e-14
```

```
cat("\nTheoretical MSE for forecast using AR(1) \n", ARmse)
```

```
##
## Theoretical MSE for forecast using AR(1)
##  0.07216893 0.002593999 9.323724e-05 3.351266e-06 1.20456e-07 4.329603e-09 1.556208e-10 5.593545e-12
```

d. (25 points)

Simulate 100 new datasets of size $n = 1010$. For each dataset, estimate the $ARMA(1,1)$ and $AR(1)$ models using the first 1000 observations and compute the forecasts for $s = 1, \ldots, 10$. Compute the forecast errors for each set of forecasts (using the true values that you simulated), and compute the sample MSEs of your forecasts.

Solution:

```
n = 1010
cc = 3.2
phi = 0.86
theta = -1.4

steps = 10

ARMAmse = matrix(NA, 100, 1)
ARmse = matrix(NA, 100, 1)

for(j in 1:100) {
  ## simulation
  eps = rnorm(n,0,1)
  y = rep(NA,n)
  y[1] = cc + phi*cc/(1-phi)
  for(i in 2:n){
    y[i] = cc + phi*y[i-1] + eps[i] + theta*eps[i-1]
  }
```

```r
  ## ARMA(1,1)
  arma11Est = arima(y[1:1000],order=c(1,0,1))

  muHat = arma11Est$coef[3]
  phiHat = arma11Est$coef[1]
  thetaHat = arma11Est$coef[2]

  yHat = rep(NA,steps)
  yHat[1] =  muHat + phiHat*(y[1000]-muHat) + thetaHat*eps[1000]
  for(s in 2:steps){
    yHat[s] = muHat + phiHat*(yHat[s-1]-muHat)
  }
  ARMAmse[j,1] = mean((yHat-y[1000:steps])^2)

  ## AR(1)
  Y = y[1:(n-10)] ## first 1000 simulated values of y
  X = cbind(rep(1,n-10),y[2:(n-9)])  ## X matrix of 1000 ones and lagged values of y (t-1)
  beta = solve(t(X)%*%X)%*%t(X)%*%Y  ## solving for coefficient

  phiHatAR = beta[2]
  muHatAR = beta[1]/(1-phiHatAR)
  yHatAR = rep(NA,steps)
  yHatAR[1] =  muHatAR + phiHatAR*(y[1000]-muHatAR)
  yHatAR[2] =  muHatAR + phiHatAR*(yHat[1]-muHatAR)
  for(s in 3:steps){
    yHatAR[s] = muHatAR + phiHatAR*(yHatAR[s-1]-muHatAR)
  }
  ARmse[j,1] = mean((yHatAR-y[1000:steps])^2)
}
cat("ARMA(1,1) MSE Values: \n", ARMAmse, "\nAR(1) MSE Values: \n", ARmse)
```

```
## ARMA(1,1) MSE Values:
##  2.422354 2.282692 2.014806 2.715537 2.068895 2.154255 2.219912 2.114286 2.586866 2.452097 2.121894 3
## AR(1) MSE Values:
##  2.161093 2.166365 1.960649 2.241176 2.040015 2.138115 2.164844 2.040633 2.178714 2.131216 1.94692 2
```

Question 2

Download daily 1-year and 10-year U.S. Treasury yield data for the period May 21, 2013 - May 20, 2018.

```r
library(Quandl)
library(vars)
treasury = Quandl("USTREASURY/YIELD",start_date="2013-05-21",end_date="2018-05-20",type="xts")
treasury <-  as.data.frame(treasury)
```

a. (10 points)

Estimate a bivariate VAR for the log interest rates. Report your parameter estimates and provide some interpretation.

Solution:

```r
treasury <- treasury[,c(4,9)]
treasury <- log(treasury)
varEst <- VAR(treasury, p=1)
coef(varEst)
```

```
## $X6.MO
##               Estimate  Std. Error      t value  Pr(>|t|)
## X6.MO.l1  0.9973860208 0.002416851 412.67999854 0.0000000
## X7.YR.l1 -0.0011855571 0.017571148  -0.06747181 0.9462169
## const    -0.0002875813 0.013507623  -0.02129029 0.9830175
##
## $X7.YR
##              Estimate   Std. Error     t value    Pr(>|t|)
## X6.MO.l1 0.0003745993 0.0005611567   0.6675484 0.504545392
## X7.YR.l1 0.9893130884 0.0040797586 242.4930446 0.000000000
## const    0.0085798248 0.0031362688   2.7356791 0.006313472
```

The estimated VAR suggests that the lagged (by a single time period) values of 1-year and 10-year treasury log interest rates increase the next period's 1-year log rate by 0.9989 and -0.0111 points respectively if they both equal 1. These increases are to a constant of 0.0106. However, the p-value on the lagged 10-year term is quite large.

They also increase the next period's 10-year log rate by -2.51e-05 and 9.93e-01 respectively if they are equal to 1. These increases are to a constant of 5.755e-03. The p-value on the lagged 1-year term is very large.

b. (15 points)

Compute and report the impulse response functions of the fitted VAR model.

Solution:

```r
#varFit <- fitted(varEst)
irf(varEst, n.ahead=15, ortho = FALSE, boot=FALSE)
```

```
##
## Impulse response coefficients
## $X6.MO
##           X6.MO        X7.YR
##  [1,] 1.0000000 0.0000000000
##  [2,] 0.9973860 0.0003745993
##  [3,] 0.9947784 0.0007442161
##  [4,] 0.9921772 0.0011089061
##  [5,] 0.9895824 0.0014687242
##  [6,] 0.9869939 0.0018237249
##  [7,] 0.9844117 0.0021739622
##  [8,] 0.9818359 0.0025194892
##  [9,] 0.9792664 0.0028603587
## [10,] 0.9767033 0.0031966229
## [11,] 0.9741464 0.0035283332
## [12,] 0.9715958 0.0038555408
## [13,] 0.9690515 0.0041782961
## [14,] 0.9665135 0.0044966491
## [15,] 0.9639817 0.0048106491
## [16,] 0.9614562 0.0051203450
```

```
##
## $X7.YR
##              X6.MO     X7.YR
##  [1,]  0.000000000 1.0000000
##  [2,] -0.001185557 0.9893131
##  [3,] -0.002355345 0.9787399
##  [4,] -0.003509541 0.9682794
##  [5,] -0.004648317 0.9579301
##  [6,] -0.005771848 0.9476911
##  [7,] -0.006880302 0.9375610
##  [8,] -0.007973849 0.9275388
##  [9,] -0.009052656 0.9176233
## [10,] -0.010116887 0.9078133
## [11,] -0.011166707 0.8981078
## [12,] -0.012202275 0.8885056
## [13,] -0.013223753 0.8790057
## [14,] -0.014231298 0.8696069
## [15,] -0.015225066 0.8603081
## [16,] -0.016205213 0.8511084
```

c. (15 points)

Use the fitted VAR model to compute and report 15-step forecasts for the two series.

Solution:

```r
predict(varEst, n.ahead=15)
```

```
## $X6.MO
##            fcst        lower     upper        CI
##  [1,] 0.7336392  0.527599124 0.9396793 0.2060401
##  [2,] 0.7301271  0.439127986 1.0211262 0.2909991
##  [3,] 0.7266276  0.370699172 1.0825560 0.3559284
##  [4,] 0.7231407  0.312692886 1.1335884 0.4104478
##  [5,] 0.7196663  0.261377399 1.1779551 0.4582889
##  [6,] 0.7162043  0.214836077 1.2175726 0.5013682
##  [7,] 0.7127547  0.171928699 1.2535808 0.5408260
##  [8,] 0.7093175  0.131911274 1.2867237 0.5774062
##  [9,] 0.7058925  0.094266261 1.3175187 0.6116262
## [10,] 0.7024796  0.058616290 1.3463430 0.6438633
## [11,] 0.6990789  0.024676047 1.3734818 0.6744029
## [12,] 0.6956902 -0.007776461 1.3991569 0.7034667
## [13,] 0.6923136 -0.038918045 1.4235452 0.7312316
## [14,] 0.6889488 -0.068892283 1.4467899 0.7578411
## [15,] 0.6855959 -0.097817689 1.4690095 0.7834136
##
## $X7.YR
##           fcst     lower    upper         CI
##  [1,] 1.102301 1.0544616 1.150140 0.04783944
##  [2,] 1.099375 1.0320740 1.166677 0.06730148
##  [3,] 1.096480 1.0144820 1.178478 0.08199784
##  [4,] 1.093614 0.9994222 1.187806 0.09419167
##  [5,] 1.090777 0.9860121 1.195542 0.10476514
```

```
##  [6,] 1.087970 0.9737961 1.202143 0.11417349
##  [7,] 1.085191 0.9625019 1.207880 0.12268886
##  [8,] 1.082440 0.9519505 1.212930 0.13048967
##  [9,] 1.079718 0.9420171 1.217418 0.13770064
## [10,] 1.077023 0.9326100 1.221436 0.14441318
## [11,] 1.074356 0.9236594 1.225053 0.15069674
## [12,] 1.071716 0.9151106 1.228322 0.15660563
## [13,] 1.069103 0.9069200 1.231287 0.16218332
## [14,] 1.066517 0.8990518 1.233982 0.16746530
## [15,] 1.063957 0.8914762 1.236438 0.17248101
```