Merical Methods in Finance

Homework 5

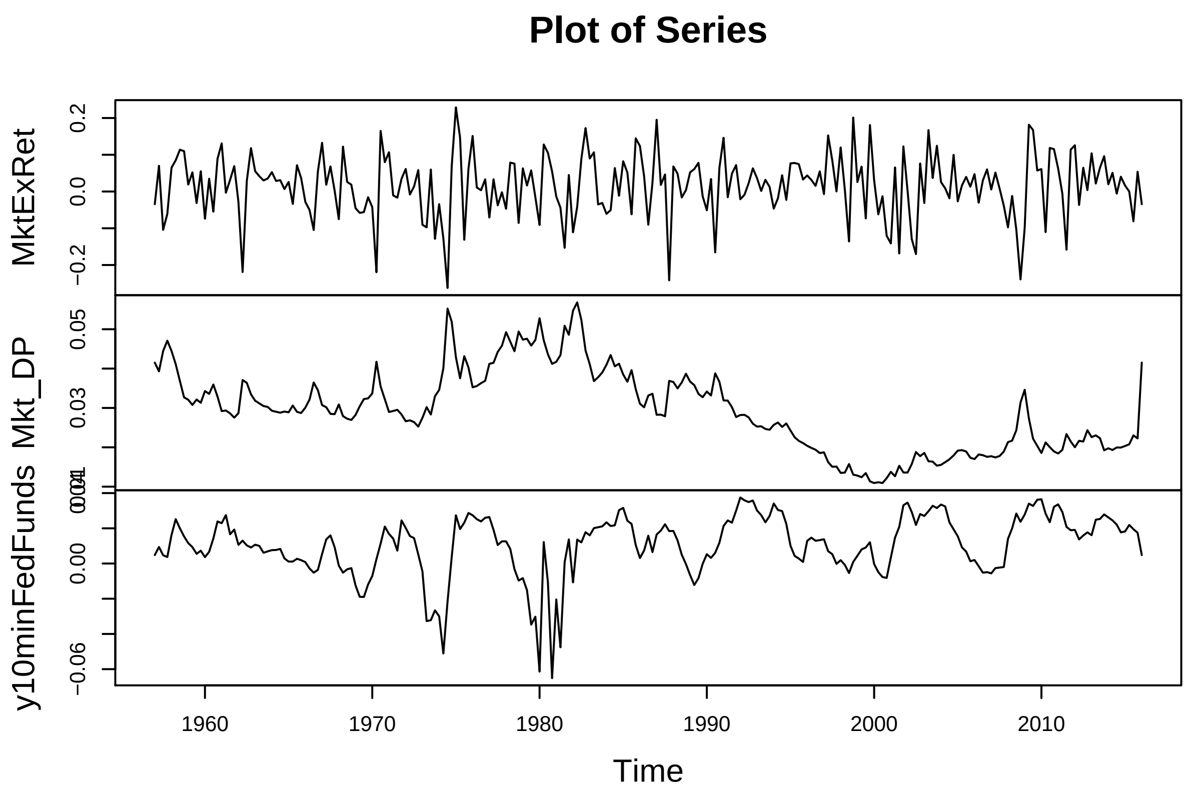
YiTao Hu, Jin (Jane) Huangfu, Charles Rambo, Junyu (Kevin) Wu

2/7/2020

Problem 1

Part 1

```
data = read.csv("MktRet_DP_TermSpread.csv")

data_2 = ts(data[, c(2,3, 4)], start = 1957, end = 2016, frequency = 4)

plot(data_2, main = "Plot of Series")
```



By using the `mean`, `sd`, and `cor` functions of R, we were able to find the mean, standard devation, and first order autocorrelations. If we assume that these are sttionary $AR(1)$ models, then the half life obtained using the formual

$$-\frac{\log(2)}{log(|\phi_1|)} = -\frac{\log(2)}{log(|\rho_1|)},$$

where ρ_1 denotes the first order autocorrelation. The results and code are shown below.

	Mean	Standard Devation	First order autocorrelation	Half life
$MktExRet$	0.1605535	0.08434985	0.06969561	0.2602277
Mkt_DP	0.02936459	0.01033875	0.96190641	17.8470814
$y10minFedFunds$	0.01062797	0.01702303	0.80393301	3.1760873

```
info = data.frame(matrix(0, nrow = 3, ncol = 4),
                  row.names = c("MktExRet", "Mkt_DP", "y10minFedFunds"))
colnames(info) = c("Mean", "Standard Devation", "First order autocorrelation",
                   "Half life")
info[, 1] = apply(data, 2, mean)[2:4]
info[, 2] = apply(data, 2, sd)[2:4]


acfs = c()
i = 1
for(name in colnames(data)){

  acfs[i] = cor(data[[name]], back(data[[name]], 1), use = "pairwise.complete.obs")
  i = i + 1
}

info[, 3]= acfs[2:4]
info[, 4] = -log(2)/log(acfs[2:4])
```

Part 2

Suppose

$$Y_t = \begin{bmatrix} MktExRet_t \\ Mkt\_DP_t \\ y10minFedFunds_t \end{bmatrix} \quad \text{and} \quad Y_t = \Phi_0 + \Phi_1 Y_{t-1} + \epsilon_t.$$

Using R, we were able to find Φ_0 and Φ_1. We introduced the matrix Φ within our code. Its definition in terms of Φ_0 and Φ_1 is

$$\Phi := [\Phi_0 \quad \Phi_1].$$

Its entries are shown below.

```
data["MktExRet_lag"] = back(data[["MktExRet"]])
data["Mkt_DP_lag"] = back(data[["Mkt_DP"]])
data["y10minFedFunds_lag"] = back(data[["y10minFedFunds"]])
data = na.omit(data)

model_1 = lm(MktExRet ~ MktExRet_lag + Mkt_DP_lag + y10minFedFunds_lag, data = data)
model_2 = lm(Mkt_DP ~ MktExRet_lag + Mkt_DP_lag + y10minFedFunds_lag, data = data)
model_3 = lm(y10minFedFunds ~ MktExRet_lag + Mkt_DP_lag + y10minFedFunds_lag, data = data)

Phi = rbind(model_1$coef, model_2$coef, model_3$coef)
Phi
```

```
##        (Intercept) MktExRet_lag Mkt_DP_lag y10minFedFunds_lag
## [1,] -0.038274656  0.048519121 1.45092241         1.04958093
## [2,]  0.002121376 -0.002282204 0.94027718        -0.03882632
## [3,]  0.001872253 -0.014836149 0.01047743         0.82164129
```

Our estimates and residuals are

$$\hat{Y}_t := \Phi_0 + \Phi_1 Y_{t-1} \quad \text{and} \quad e := Y_t - \hat{Y}_t,$$

where $\hat{Y}$ and e are $3 \times N$ matrices.

The White variance-covariance matrix of the parameters in the i-th row of Φ is

$$(Y_{t-1}Y_{t-1}')^{-1} Y_{t-1} \Lambda_i Y_{t-1}' (Y_{t-1}Y_{t-1}')^{-1},$$

where Λ_i is a diagonal matrix with the square of the i-th row of e on its main diagonal. The standard errors for the parameters are simply the square root of the results along the diagonal of the final product.

```
Lambda_1 = diag(model_1$residuals^2)
Lambda_2 = diag(model_2$residuals^2)
Lambda_3 = diag(model_3$residuals^2)

ones = rep(1, nrow(data))
X = rbind(ones, data$MktExRet_lag, data$Mkt_DP_lag , data$y10minFedFunds_lag)
Y = rbind(data$MktExRet, data$Mkt_DP, data$y10minFedFunds)

W_1 = solve(X %*% t(X)) %*% X %*% Lambda_1 %*% t(X) %*% solve(X %*% t(X))
W_2 = solve(X %*% t(X)) %*% X %*% Lambda_2 %*% t(X) %*% solve(X %*% t(X))
W_3 = solve(X %*% t(X)) %*% X %*% Lambda_3 %*% t(X) %*% solve(X %*% t(X))

SE = rbind(sqrt(diag(W_1)), sqrt(diag(W_2)), sqrt(diag(W_3)))
rownames(SE) = c("Phi_1j", "Phi_2j", "Phi_3j")
colnames(SE) = c("Phi_i1", "Phi_i2", "Phi_i3", "Phi_i4")
SE
```

```
##              Phi_i1      Phi_i2     Phi_i3     Phi_i4
## Phi_1j 0.0194643374 0.072541975 0.57421160 0.38948934
## Phi_2j 0.0005824776 0.002308165 0.01934000 0.01683741
## Phi_3j 0.0018677643 0.007773175 0.07347993 0.08633115
```

The R^2 values are

```
R_sqr = rbind(summary(model_1)$r.squared, summary(model_2)$r.squared,
              summary(model_3)$r.squared)
R_sqr
```

```
##            [,1]
## [1,] 0.0607173
## [2,] 0.9298475
## [3,] 0.6515704
```

Part 3

We used R to find the eigenvalues, which were positive real numbers. They were

```
Phi_0 = Phi[ , 1]
Phi_1 = Phi[ , 2:4]

eg = eigen(Phi_1)
eg$values
```

```
## [1] 0.94071593 0.79530199 0.07441967
```

Their moduli are simply their values so we can conclude that the $VAR(1)$ model is stationarry since all of the eigenvalues are less than one.

Part 4

Total Variance

Let

$$Y_t = \Phi_0 + \Phi_1 Y_{t-1} + \epsilon \quad \text{implies} \quad Var(Y_t) = \Phi_1 Var(Y_t)\Phi_1' + \Sigma,$$

where Σ denotes the variance-covariance matrix of ϵ. Call $Var(Y_t) = \Gamma_0$. If $\otimes$ denotes the Kronecker product, our work implies that

$$\text{vec}(\Gamma_0) = (\Phi_1 \otimes \Phi_1)\text{vec}(\Gamma_0) + \text{vec}(\Sigma).$$

Hence, we have

$$\text{vec}(\Gamma_0) = (I - \Phi_1 \otimes \Phi_1)^{-1}\text{vec}(\Sigma).$$

We can then convert Γ_0 back into a matrix using the `matrix` function.

```
e = rbind(model_1$residuals, model_2$residuals, model_3$residuals)
kron = kronecker(Phi_1, Phi_1)
I = diag(9)
vec_Gamma = solve(I - kron) %*% vec(e %*% t(e)/ncol(e))
Gamma = matrix(vec_Gamma, nrow = 3, ncol = 3)
Gamma
```

```
##               [,1]           [,2]           [,3]
## [1,]  0.0070971555 -0.00012457099  0.00031135366
## [2,] -0.0001245710  0.00010293491 -0.00004670494
## [3,]  0.0003113537 -0.00004670494  0.00028973273
```

In particular, we note that the square root of the first entry is 0.08424462, which is almost exactly the same as the standard devation of excess returns calculated in part 1.

Variance Implied by Model

Let us also find the variance explained by the model, i.e. the effect when we assume $\epsilon \equiv 0$. We have

$$Y_t = \Phi_0 + \Phi_1 Y_{t-1}.$$

This implies

$$Var(Y_t) = Var(\Phi_1 Y_{t-1}) = \Phi_1 Var(Y_{t-1})\Phi_1'.$$

We note that

$$Var(Y_{t-1}) = (Y_{t-1} - E[Y_{t-1}])(Y_{t-1} - E[Y_{t-1}])'.$$

```
X_mod = X[2:4, ] - apply(X[2:4, ], 1, mean)

implied_var = Phi_1 %*% (X_mod %*% t(X_mod)/ncol(X_mod)) %*% t(Phi_1)
implied_var
```

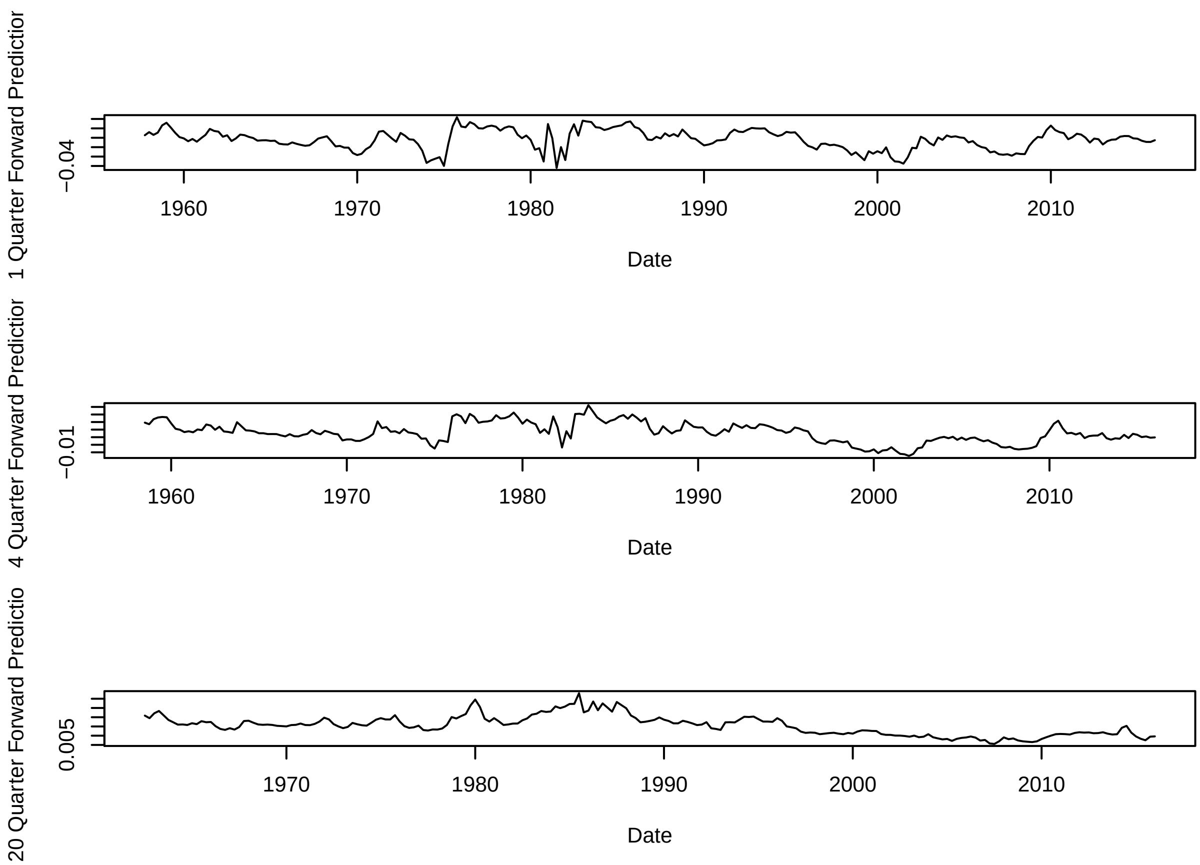
```
##               [,1]           [,2]           [,3]
## [1,] 0.00043133506  0.00008262172  0.00019961851
## [2,] 0.00008262172  0.00009880379 -0.00004305473
## [3,] 0.00019961851 -0.00004305473  0.00018871971
```

In particular, we note that the square root of the first entry is the volatility of expected returns. As a result, it is 0.020768608.

Part 5

To make computations easier, we converted $\Phi := [\Phi_0 \quad \Phi_1]$ into a square matrix by adding the first row of I_4 into the first row of Φ. We called the result Φ_{mod}.

```
Phi_mod = rbind(diag(ncol(X))[1, ], Phi)

pred_1 = (Phi_mod %*% X)[2, ]

pred_4 = ((Phi_mod %^% 4) %*% X)[2, ]

pred_20 = ((Phi_mod %^% 20) %*% X)[2, ]

n = nrow(data["Date"])
par(mfrow=c(3,1))
plot(data[2:n, "Date"], pred_1[1:(n - 1)], type = "l",
     ylab = "1 Quarter Forward Prediction", xlab = "Date")
plot(data[5:n, "Date"], pred_4[1:(n - 4)], type = "l",
     ylab = "4 Quarter Forward Prediction", xlab = "Date")
plot(data[21:n, "Date"], pred_20[1:(n -20)], type = "l",
     ylab = "20 Quarter Forward Prediction", xlab = "Date")
```



At a four quarter lag, the DP-ratio's effect on excess returns is 3.5857164 times the previous ratio, and the twenty quarter ahead forcast, it is sill 0.078479592. At four quarters the term-spread is 0.1986280 and at twenty quarters it is -0.07281989. Consider Φ^{20}_{mod}; none of its the other import coefficients are as large as the coefficients for these two:

$$\Phi^{20}_{mod} = \begin{bmatrix} 1.000000000 & 0.00000000000 & 0.000000000 & 0.00000000 \\ 0.009417606 & 0.00085155281 & 0.078479592 & -0.07281989 \\ 0.035079586 & 0.00016096191 & 0.007221075 & -0.01142492 \\ 0.006202652 & 0.00007229522 & 0.020392499 & -0.01040152 \end{bmatrix}.$$

Part 6

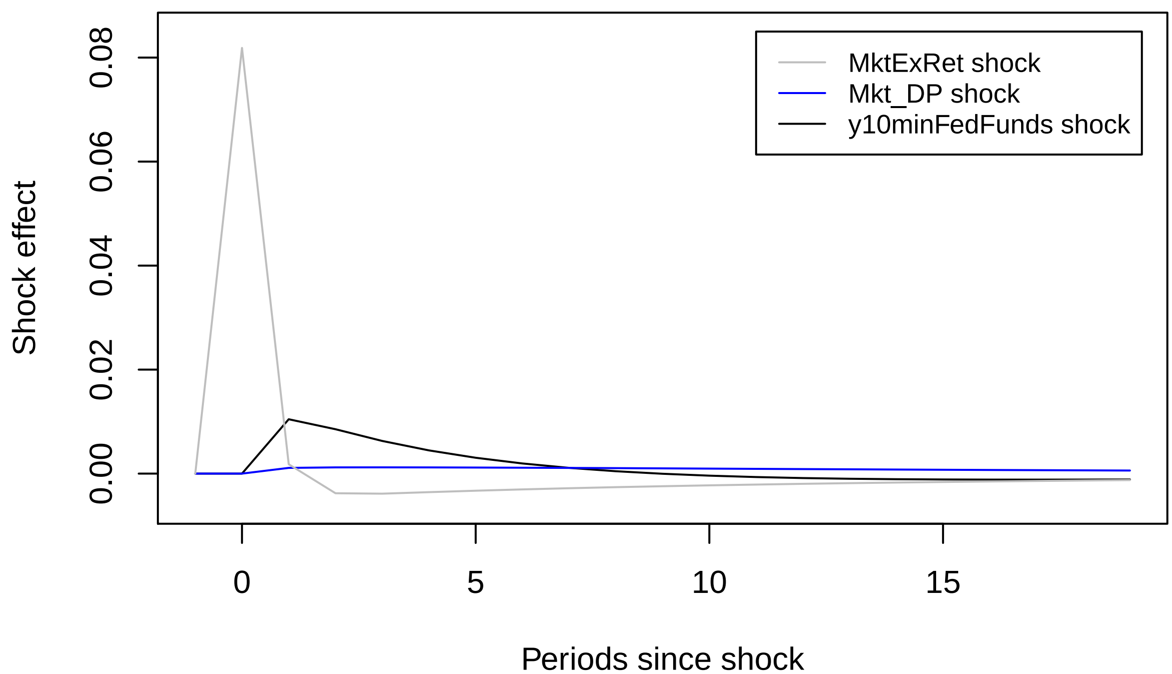
We first found the Cholesky decomposition of Σ, which we called Σ_C. We can use this to find the orthogonalized error variable ϵ^o. We have

$$\epsilon = \Sigma_C \epsilon^o \quad \text{implies} \quad \epsilon^o = \Sigma_C^{-1}\epsilon.$$

We then computed that standard devation accross the observations. We demeaned the $VAR(1)$ so there is no need for Φ_0.

```
Sigma = e %*% t(e)/ncol(e)
Sigma_C = t(chol(Sigma))
e_o = solve(Sigma_C) %*% e

shocks = apply(e_o, 1, sd)

mu = matrix(0, ncol = 1, nrow = 3)
#matrix(info[, "Mean"], nrow = nrow(Phi_1), ncol = 1)

shock_effects_Mkt = matrix(0, nrow = nrow(Phi_1), ncol = 21)
rownames(shock_effects_Mkt) = c("MktExRet", "Mkt_DP", "y10minFedFunds")

shock_effects_Mkt[, 1] = mu

shock_Mkt = matrix(c(shocks[1], 0, 0), nrow = 3, ncol = 1)

for(i in 1:20){

  shock_effects_Mkt[, i + 1]  = (Phi_1 %*% shock_effects_Mkt[, i]
                                 + (i == 1)*Sigma_C %*% shock_Mkt)

}

shock_effects_DP = matrix(0, nrow = nrow(Phi_1), ncol = 21)
rownames(shock_effects_DP) = c("MktExRet", "Mkt_DP", "y10minFedFunds")

shock_effects_DP[, 1] = mu
shock_DP = matrix(c(0, shocks[2], 0), nrow = 3, ncol = 1)

for(i in 1:20){

  shock_effects_DP[, i + 1]  = (Phi_1 %*% shock_effects_DP[, i]
                                + (i == 1)*Sigma_C %*% shock_DP)

}

shock_effects_spread = matrix(0, nrow = nrow(Phi_1), ncol = 21)
shock_effects_spread[ , 1] = mu
rownames(shock_effects_spread) = c("MktExRet", "Mkt_DP", "y10minFedFunds")

shock_effects_spread[, 1] =  mu
shock_spread = matrix(c(0, 0, shocks[3]), nrow = 3, ncol = 1)

for(i in 1:20){

  shock_effects_spread[, i + 1]  = (Phi_1 %*% shock_effects_spread[, i]
```

```
                                  + (i == 1)*Sigma_C %*% shock_spread)

}

plot(1:21 - 2, shock_effects_spread[1,], type = "l", xlab = "Periods since shock",
      col = "black",  ylim=c(-0.006,0.085), ylab = "Shock effect")
lines(1:21 - 2, shock_effects_DP[1,], type = "l", xlab = "Periods since shock",
      col = "blue")
lines(1:21 - 2, shock_effects_Mkt[1,], type = "l", col = "gray")

legend(11, 0.085, legend= c("MktExRet shock", "Mkt_DP shock", "y10minFedFunds shock"),
       col=c("gray", "blue", "black"), lty=1, cex=0.8)
```



Part 7

For part 7, we considered the out of sample predictive ability of our model for excess returns. We formulated a training and test set and used the training set to predict the first value in the test set. We then added the test observation to the training set and repeated the process.

```
n = round(0.8*nrow(data))
Train = data[1:n, ]
Test = data[(n + 1):nrow(data), ]

ones = rep(1, nrow(Test))
X_test = rbind(ones, Test$MktExRet_lag, Test$Mkt_DP_lag , Test$y10minFedFunds_lag)
Y_test = rbind(ones, Test$MktExRet, Test$Mkt_DP, Test$y10minFedFunds)
```

```r
errors = c()

for(i in 1:nrow(Test) - 1){

  model_1 = lm(MktExRet ~ MktExRet_lag + Mkt_DP_lag + y10minFedFunds_lag,
              data = data[1:(n + i - 1), ])

  Phi = matrix(model_1$coef, ncol = 4, nrow =1)

  errors[i] = Y_test[2, i] - Phi %*% X_test[, i]

    }
```

We found a substatiall diminished R^2 value:

```r
MSE_out = mean(errors^2)
var_total = mean(Y_test[2, ]^2)

R_sqrs_out = 1 - MSE_out/var_total
R_sqrs_out
```

```
## [1] 0.01854192
```