R Notebook

Code

Author: Chih-Hsuan (Carolyn) Kao;

Email: chkao831@stanford.edu (mailto:chkao831@stanford.edu);

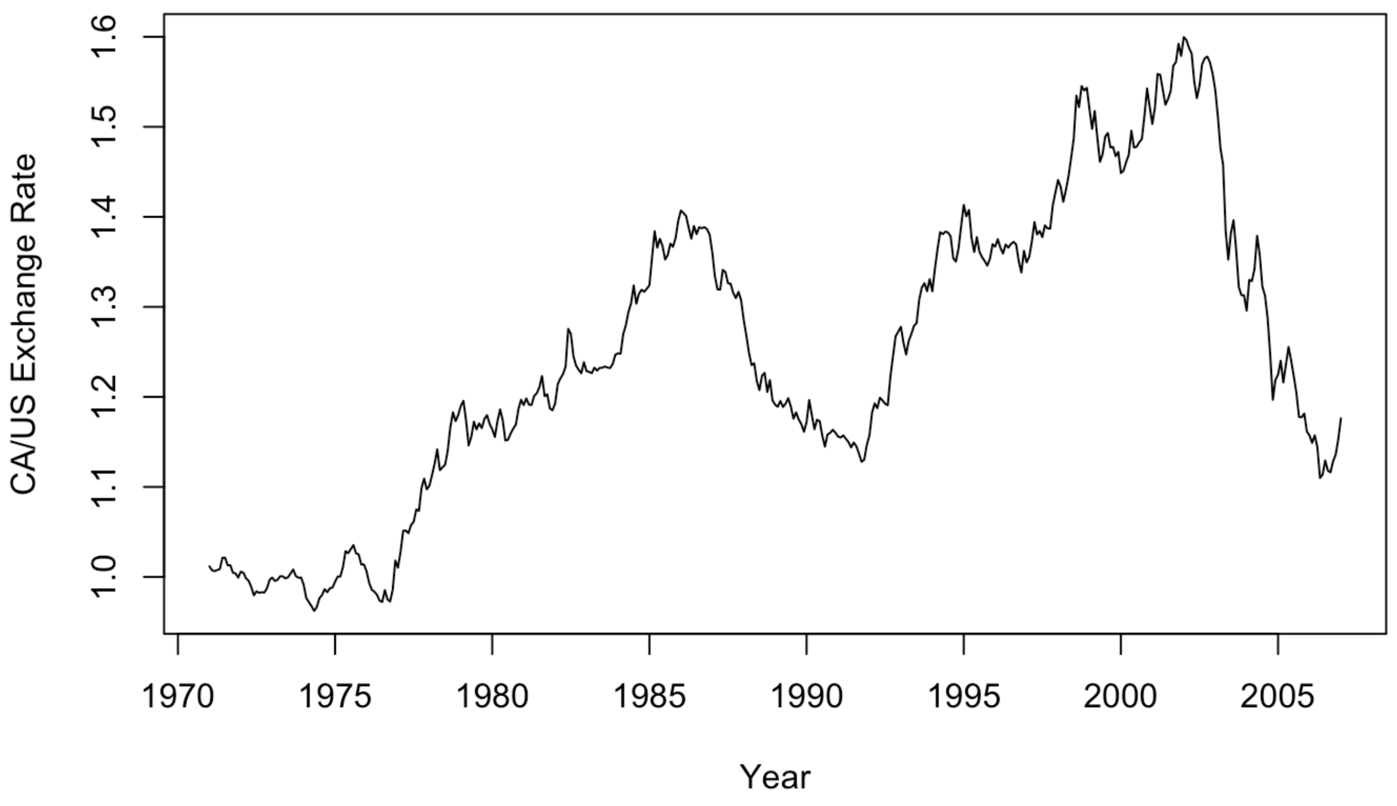
Date: Nov 22nd, 2019;

Title: STATS 240 HW3 Autumn19-20;

Problem 5.8 part (a) Plot the time series and its ACF

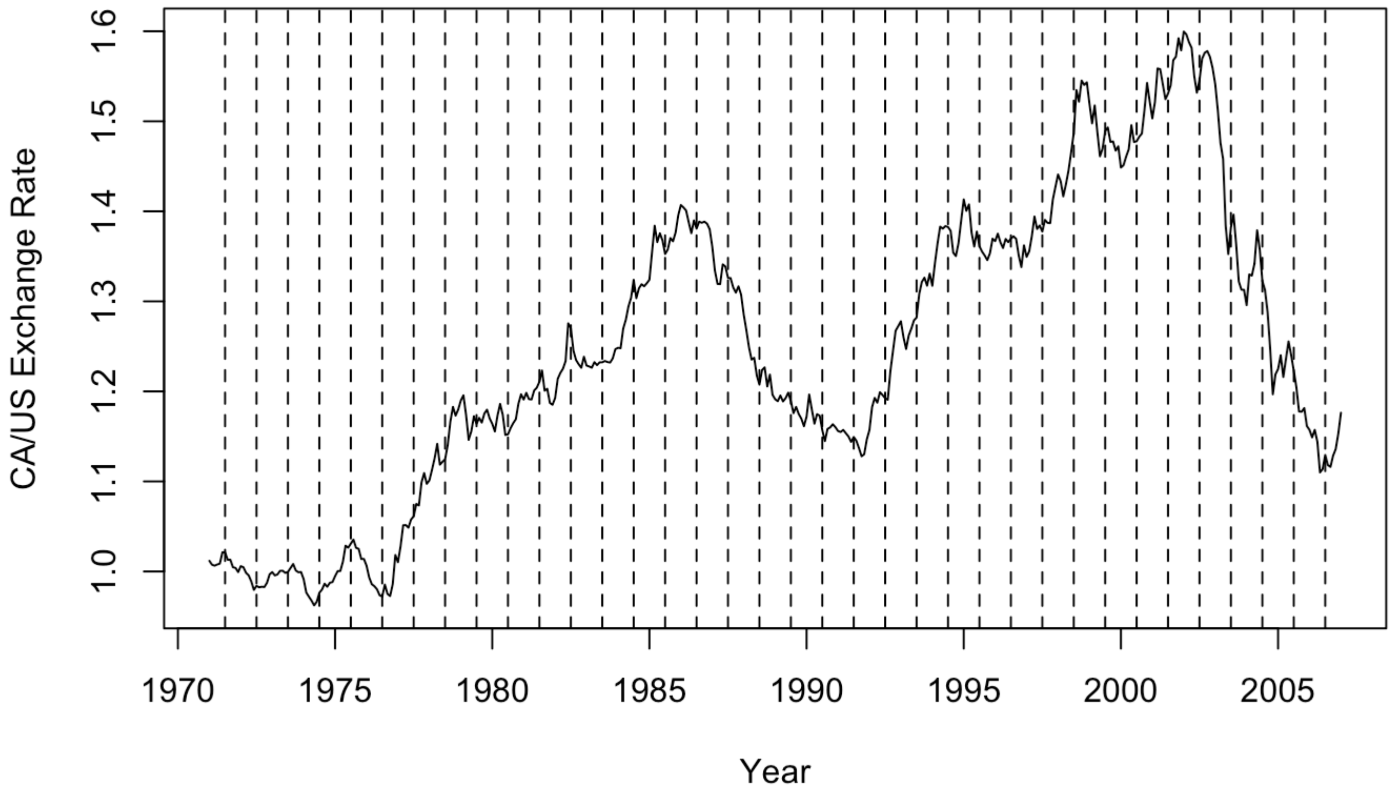
Hide

```
table = read.table("m_caus_ex.txt",
                   skip = 2,
                   sep = "",
                   header = FALSE,
                   fill = FALSE,
                   strip.white = TRUE)
data = ts(table["V2"],start = 1971, end = 2007, frequency = 12)
plot(data,xlab = "Year", ylab= "CA/US Exchange Rate")
```



Hide

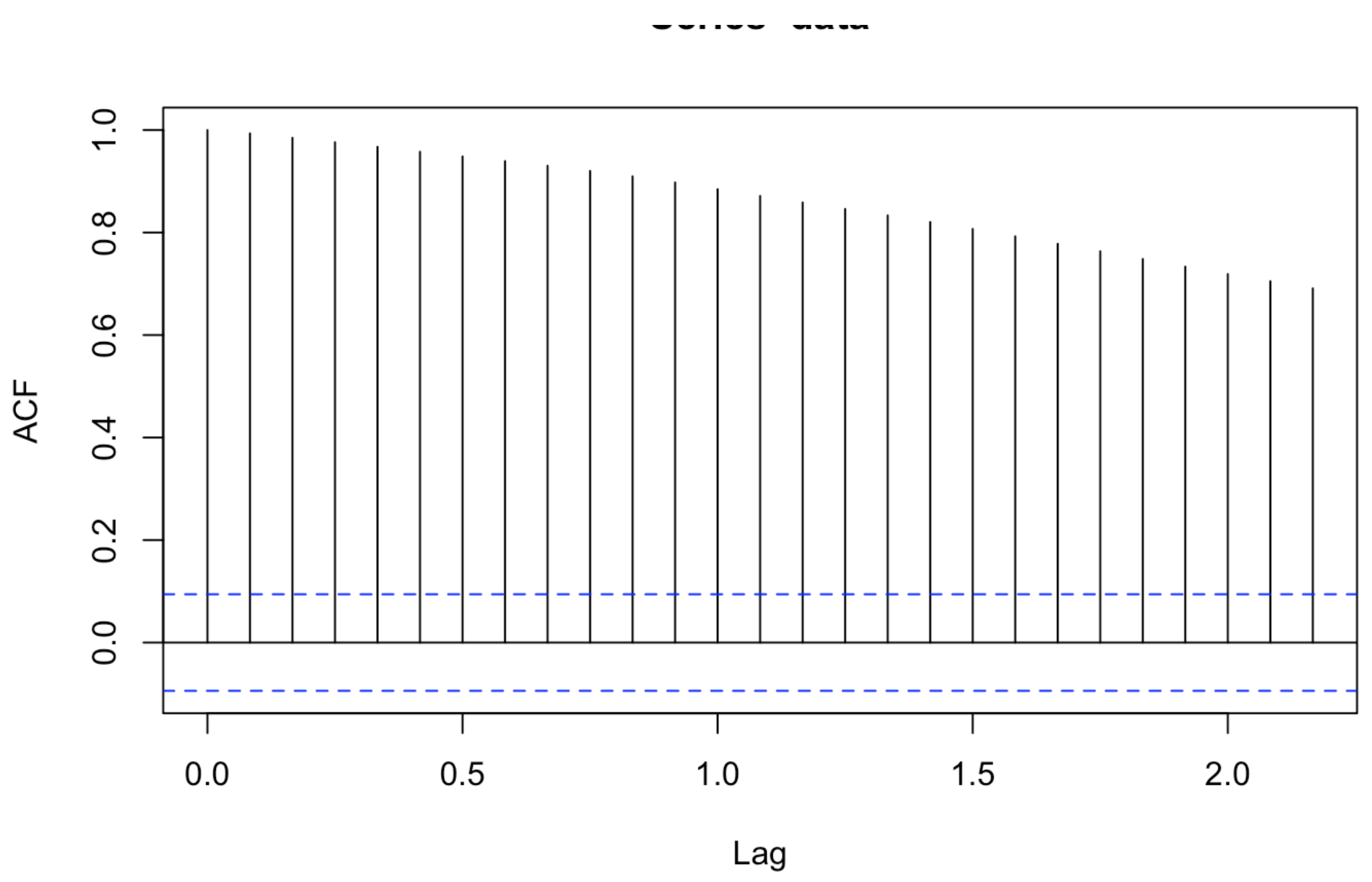
```
plot(data,xlab = "Year", ylab= "CA/US Exchange Rate")
abline(v=seq(1971+6/12,2007,1),lty=2)
```



The time series plot of the original Canada/US exchange rate are presented above. From the time series plot, I don't see apparent seasonality judging from the fact that the patterns do not repeat with a fixed period of time. Indeed, when we decompose the data into monthly analysis, there're generally some trends within each window; but looking back at the big picture, there's no pattern within a fixed seasonality period. Hence, I would say there's no seasonal effects for the original data.

Hide

```
acf(data)
```



Hide

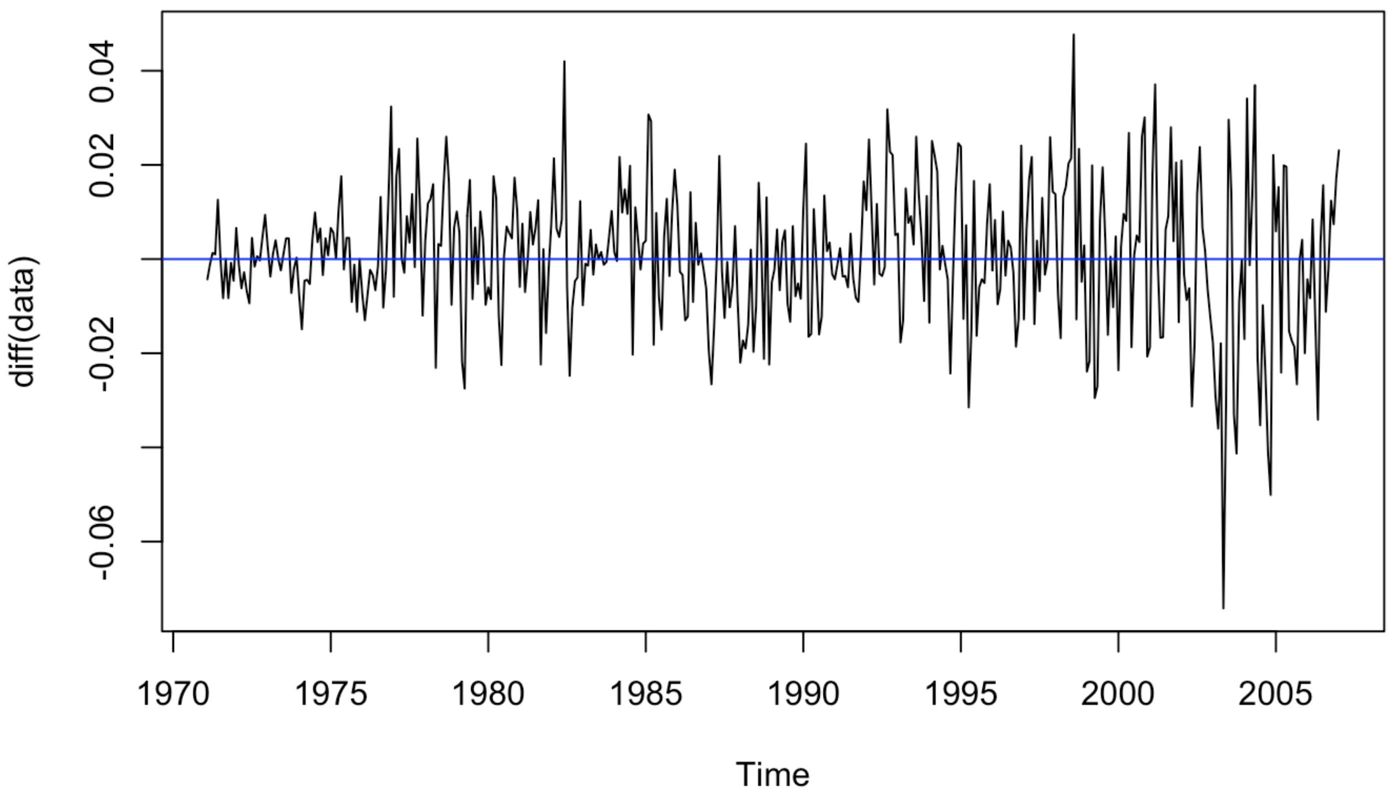
```
library(tseries)
adf.test(data, alternative="stationary")
```

```


    Augmented Dickey-Fuller Test

data:  data
Dickey-Fuller = -0.82146, Lag order = 7, p-value = 0.9594
alternative hypothesis: stationary
```

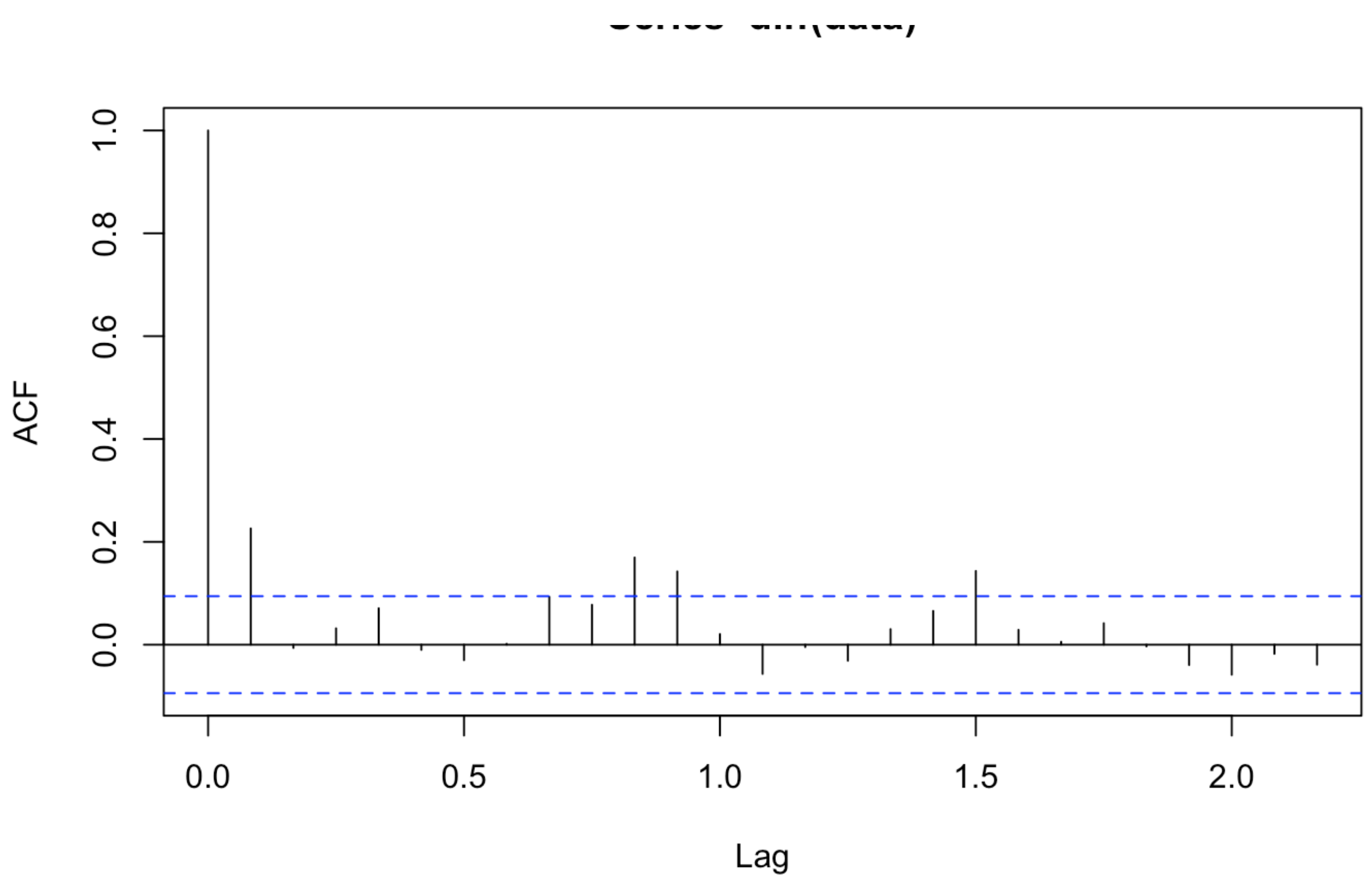
ACF is an auto-correlation function which gives us values of auto-correlation of the series with its lagged values. The slow decay of the autocorrelation function suggests the data follow a long-memory process. That is, the future values of the series are heavily affected by its past values. The duration of shocks is relatively persistent and influence the data several observations ahead. However, by my previous analysis, the heavy correlation with previous observations does not necessarily suggest there is seasonality pattern. In addition, I also perform an adf test above. The Augmented Dicky-Fuller (adf) test examines the null hypothesis that the data has a unit root. The p value is very big, suggesting we fail to reject the null hypothesis that the series has unit root. Having unit root generally implies “random walk with drift” and that a systematic pattern is unpredictable. To sum up, I would suppose that the mean and other indicators will change over time and that the process as a whole is not stationary. Hence, it might further suggest that first differences may be needed to render the data stationary.

Hide

```
plot(diff(data))
abline(h=0,col="blue")
```



```
acf(diff(data))
```

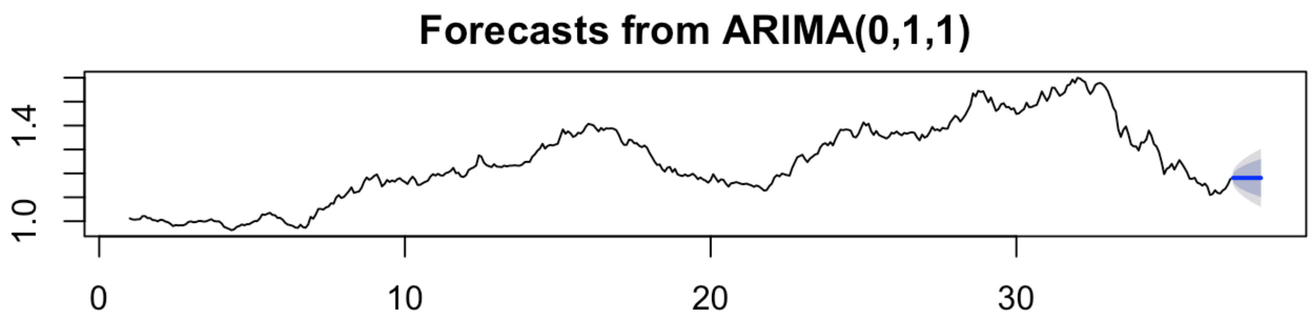


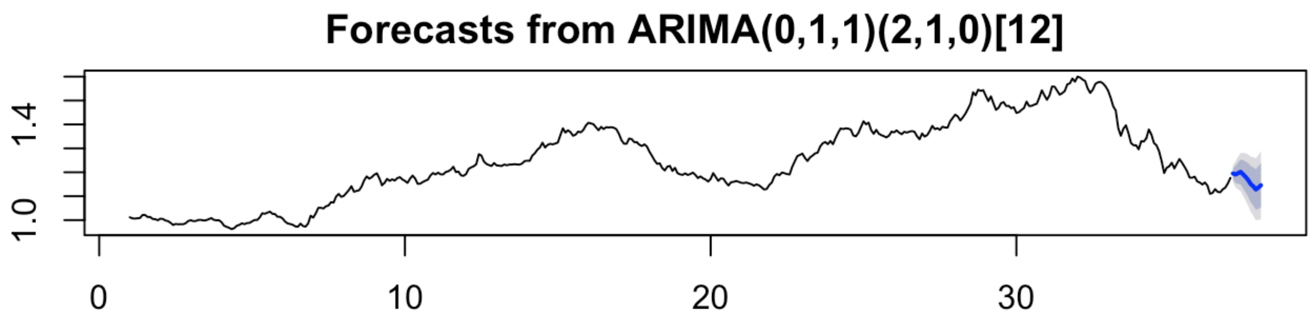
Now I difference the data to get the differenced series, hoping to yield a somewhat stationary result. From the time series plot, although we can have some explosive pattern as we approach the very end, we can draw a flat line through the graph when diff=0. From the acf plot, I would say although the decay becomes very fast compared to the original undifferenced data, it does not decay fast enough–the first few lags still exceed the confidence band. Thus, I suppose the differenced series become approximately weakly stationary. By the definition of unit-root stationary in Section 5.2.4 in textbook, along with my previous analysis, I suggest that there is no apparent seasonal effect in the original series, and that we fail to reject that there is unit-root nonstationary patterns in the series.

Problem 5.8 part (b) Build time series to forecast the next 6 months from Dec 2006

Hide

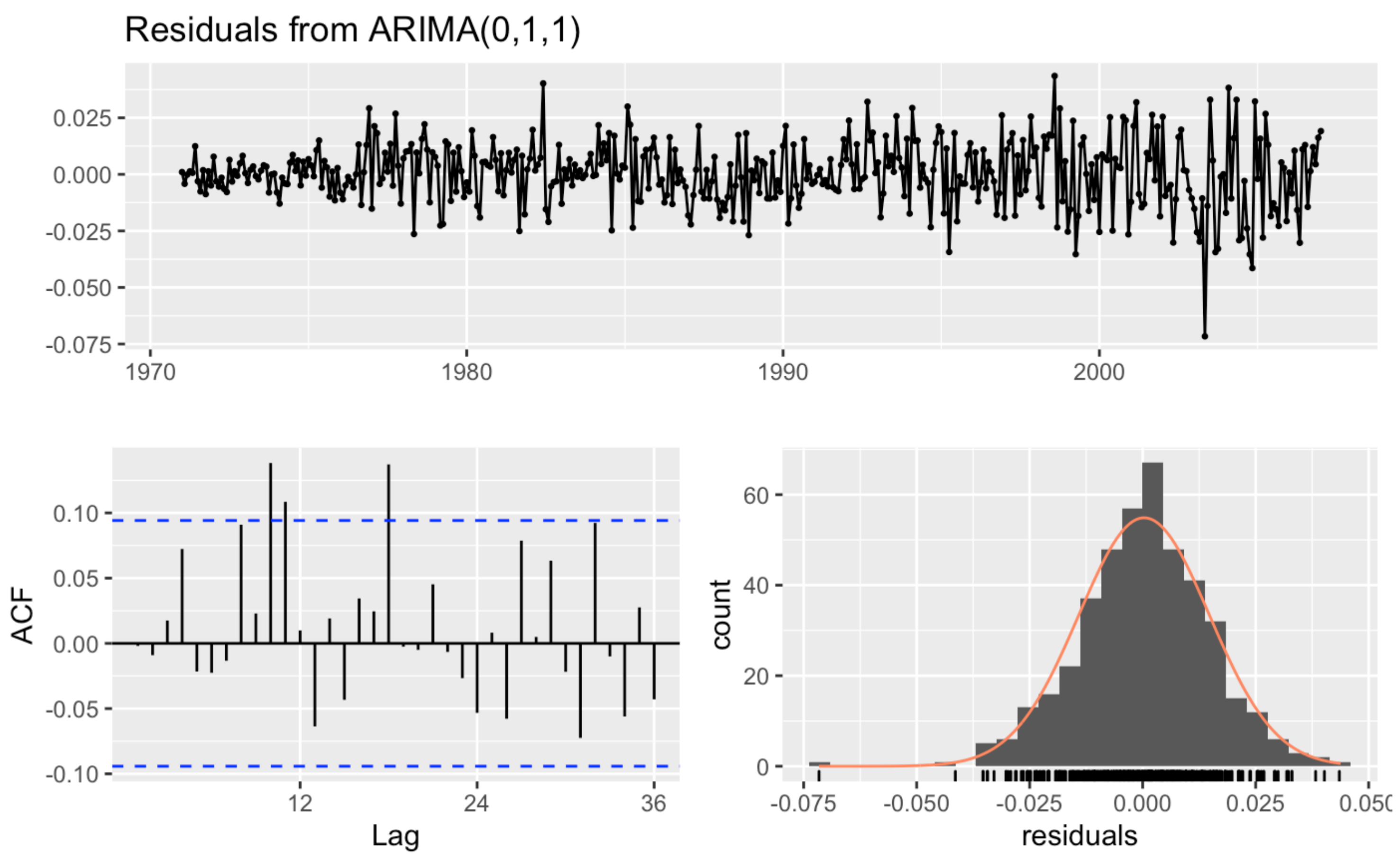
```
# predictive accuracy
library(forecast)

par(mfrow=c(2,1))
plot(forecast(auto.arima(ts(data,frequency=12),D=0),h=12)) #without seasonality--cons
istent with my part (a) analysis
plot(forecast(auto.arima(ts(data,frequency=12),D=1),h=12)) #with seasonality
```





Hide

```
checkresiduals(auto.arima(data),test=FALSE)
```

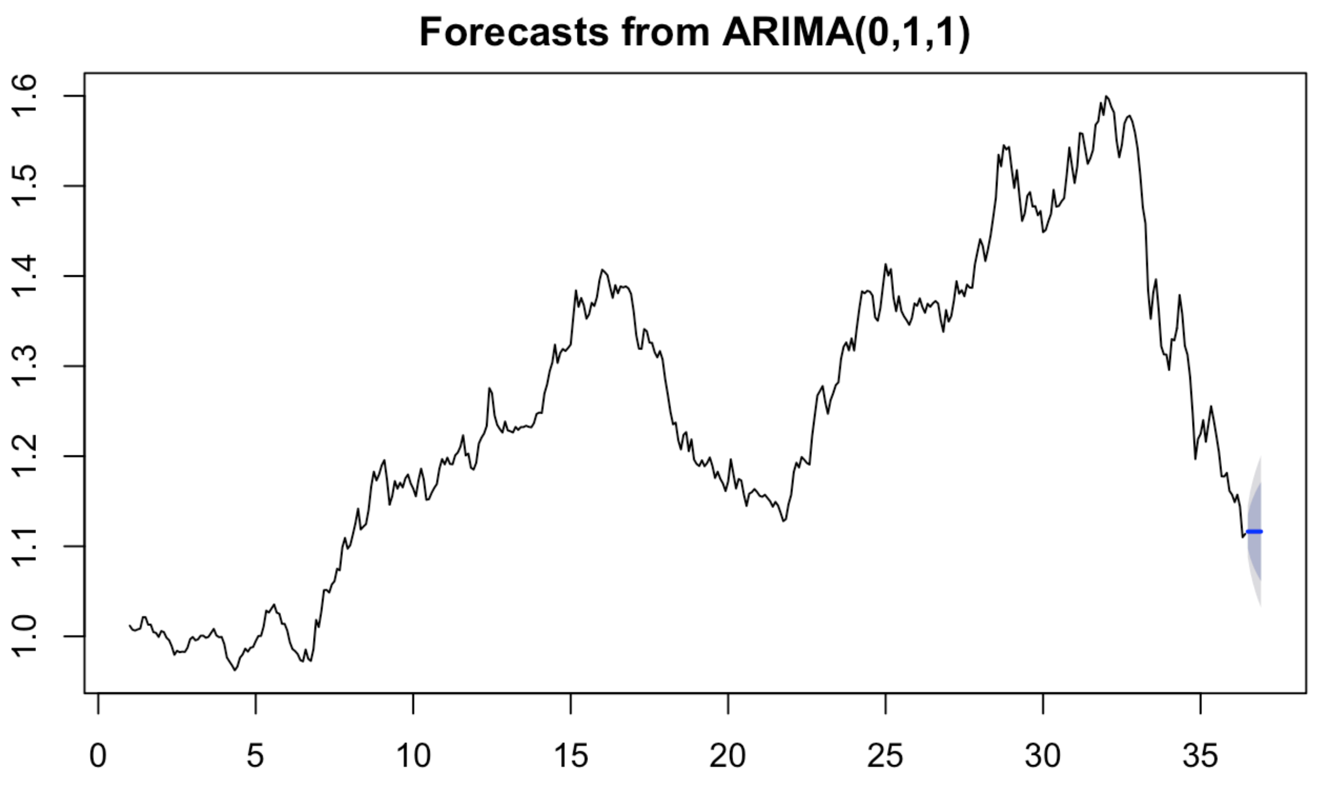


I used ARIMA(p,d,q) to select my p,d,q parameters to perform forecasting. I chose it mainly because in theory, ARIMA models are the most general class of models in order to forecast a time series which can be made to be somehow stationary by differencing as I've previously shown. The ARIMA model may choose to correct autocorrelated errors in a random walk model by suggesting a smoothing model. Since I observe that the time series is nonstationary, that is, it exhibits some noisy fluctuations, the random walk model does not perform as well as a moving average of past values. Rather than taking the most recent observation as the forecast of the next observation, it would sometimes be better to use an average of the last few observations in order to filter out the noise and more accurately estimate the local mean. From the forecast, we can see that ARIMA modeling yields ARIMA(p,d,q) = (0,1,1) in this case, so it actually suggests an exponential smoothing approach in the forecast as I mentioned. As we check residuals of the forecasted series, admittedly the acf still exceeds beyond the confidence band several times, and the residual fluctuates greatly especially in latter years. Hence, although it's the most optimal forecasting model I would choose, the performance is far from perfect. I've respectively plotted the forecast of ARIMA(0,1,1) without (above) and with (below) seasonality. Without forcing seasonality, the forecast of the next 6 months is approximately a flat line, which is not really pragmatic in my opinion. By forcing seasonality, even though it does not align with my analysis, the forecast becomes more reasonable to me. But on the basis of my analysis, I would stick to the case where seasonality is not enforced.

c. Compute 6-month ahead based on model and compare with actual rate

Hide

```
sub = window(data, start=c(1971,1), end=c(2006,6), frequency=12)
subdata = ts(sub)

#WITHOUT seasonality
plot(forecast(auto.arima(ts(subdata,frequency=12),D=0),h=6))
```



Hide

```
forecast(auto.arima(ts(subdata,frequency=12),D=0),h=6) #forecasted
```

	Point Forecast <dbl>	Lo 80 <dbl>	Hi 80 <dbl>	Lo 95 <dbl>	Hi 95 <dbl>
Jul 36	1.116323	1.097696	1.134950	1.087835	1.144810
Aug 36	1.116323	1.086661	1.145985	1.070959	1.161687
Sep 36	1.116323	1.078737	1.153909	1.058840	1.173806
Oct 36	1.116323	1.072214	1.160432	1.048864	1.183782
Nov 36	1.116323	1.066538	1.166107	1.040184	1.192462
Dec 36	1.116323	1.061447	1.171199	1.032397	1.200249

6 rows

Hide

```
truevec = c(table[433, "V2"],
            table[434, "V2"],
            table[435, "V2"],
            table[436, "V2"],
            table[437, "V2"],
            table[438, "V2"])

estivec = as.numeric(forecast(auto.arima(ts(subdata,frequency=12),D=0),h=6)$mean) #tr
ue

abs(truevec - estivec) #is the difference between forecasted and actual rates
```

```
[1] 0.05997704 0.05467704 0.05187704 0.01867704 0.02122296 0.05122296
```

The difference between actual exchange rates and my forecasted rates is printed to the console result above.

Problem 6.3 part(a) Compute the sample mean, variance, skewness, excess kurtosis and Ljung-Box statistics up to lag 10

Hide

```
library(e1071)

par(mfrow=c(1,1))
d = read.table("w_logret_3stocks.txt",
               sep="\t",
               header = TRUE,
               fill=FALSE,
               strip.white=TRUE)
citi = c(d[,3])
citi_mean <- mean(citi)
citi_var <- var(citi)
print(paste0("citi mean is ", citi_mean))
```

```
[1] "citi mean is 0.00138091963293051"
```

Hide

```
print(paste0("citi var is ", citi_var))
```

```
[1] "citi var is 0.000371858910305793"
```

Hide

```
print(paste0("citi skewness is ", skewness(citi)))
```

```
[1] "citi skewness is -0.367064991366745"
```

Hide

```
print(paste0("citi kurtosis is ", kurtosis(citi)))
```

```
[1] "citi kurtosis is 5.57716293756215"
```

Hide

```
Box.test(citi, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```


    Box-Ljung test

data:  citi
X-squared = 20.388, df = 10, p-value = 0.02579
```

Hide

```
PFE = c(d[,4])
PFE_mean = mean(PFE)
PFE_var = var(PFE)
print(paste0("PFE mean is ", PFE_mean))
```

```
[1] "PFE mean is 0.00125838770468278"
```

Hide

```
print(paste0("PFE var is ", PFE_var))
```

```
[1] "PFE var is 0.000277445670722418"
```

Hide

```
print(paste0("PFE skewness is ", skewness(PFE)))
```

```
[1] "PFE skewness is -0.260852478468434"
```

Hide

```
print(paste0("PFE kurtosis is ", kurtosis(PFE)))
```

```
[1] "PFE kurtosis is 1.53244315564333"
```

Hide

```
Box.test(PFE, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```


	Box-Ljung test

data:  PFE
X-squared = 14.773, df = 10, p-value = 0.1406
```

Hide

```
GM = c(d[,4])
GM_mean = mean(GM)
GM_var = var(GM)
print(paste0("GM mean is ", GM_mean))
```

```
[1] "GM mean is 0.00125838770468278"
```

Hide

```
print(paste0("GM var is ", GM_var))
```

```
[1] "GM var is 0.000277445670722418"
```

Hide

```
print(paste0("GM skewness is ", skewness(GM)))
```

```
[1] "GM skewness is -0.260852478468434"
```

Hide

```
print(paste0("GM kurtosis is ", kurtosis(GM)))
```

```
[1] "GM kurtosis is 1.53244315564333"
```

Hide

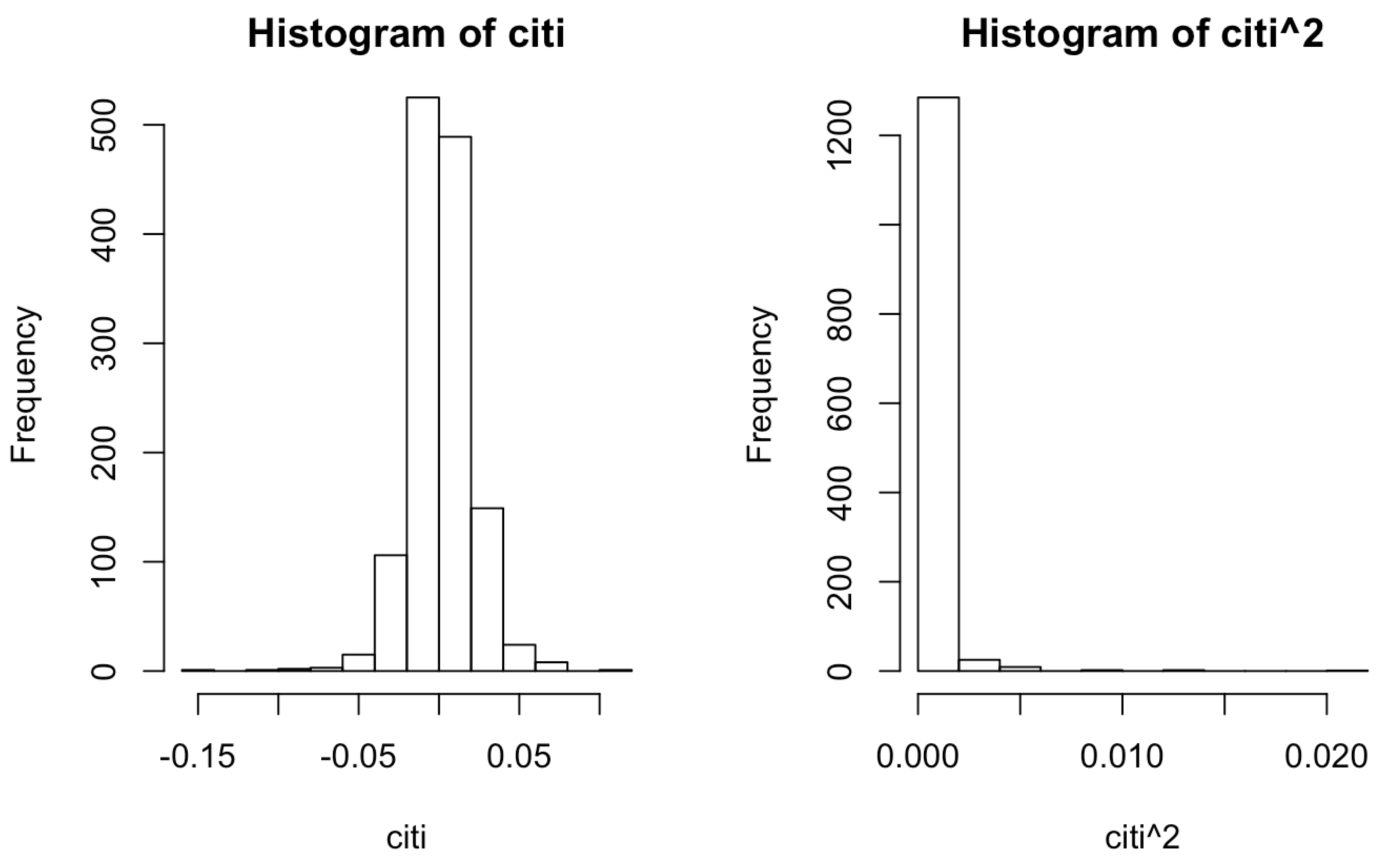
```
Box.test(GM, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```

    Box-Ljung test

data:  GM
X-squared = 14.773, df = 10, p-value = 0.1406
```

Problem 6.3 part(b) Plot histograms of returns and squared returns

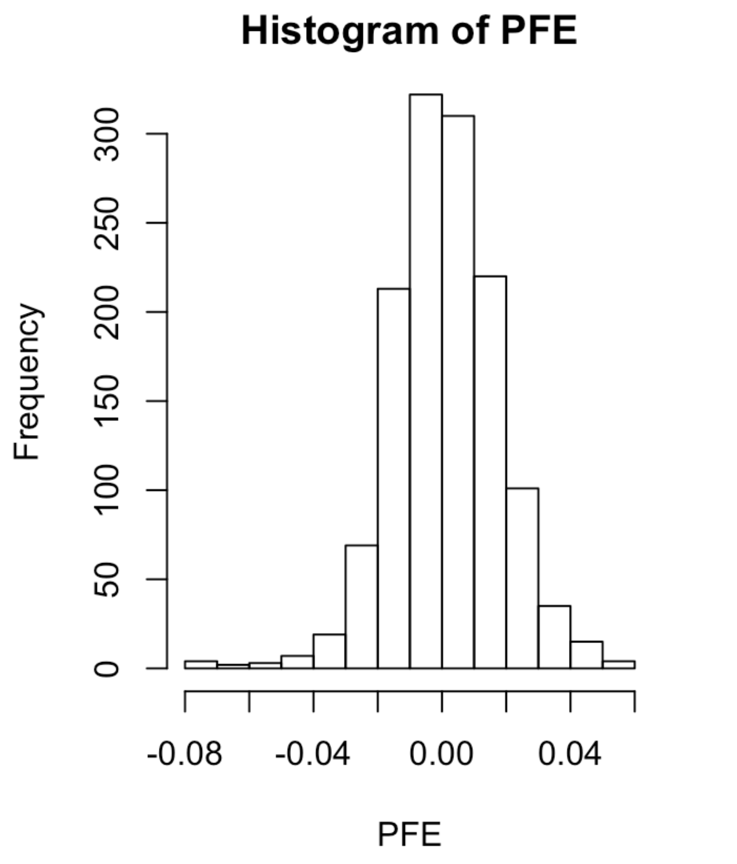
Hide

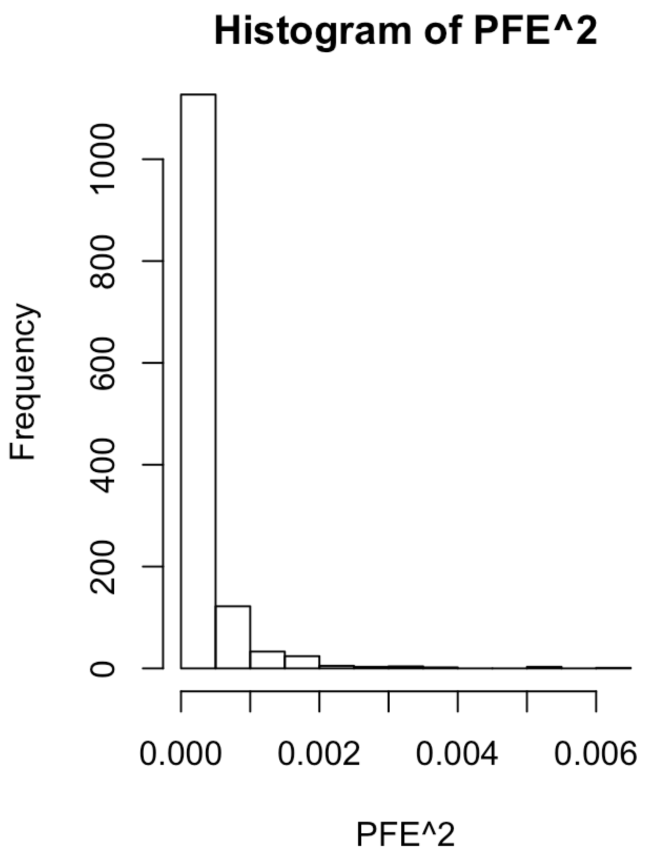
```
par(mfrow=c(1,2))
hist(citi)
hist(citi^2)
```



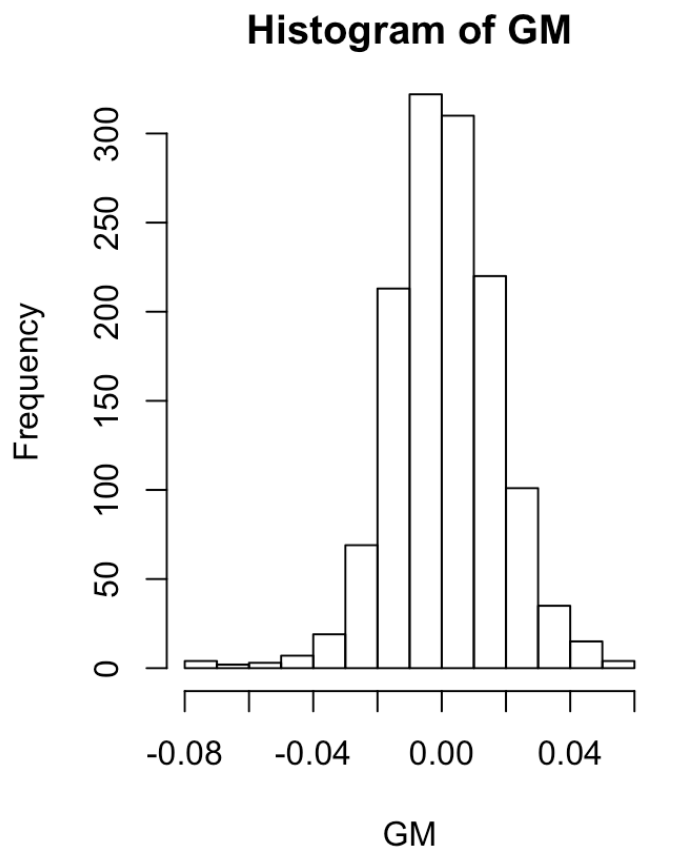
Hide

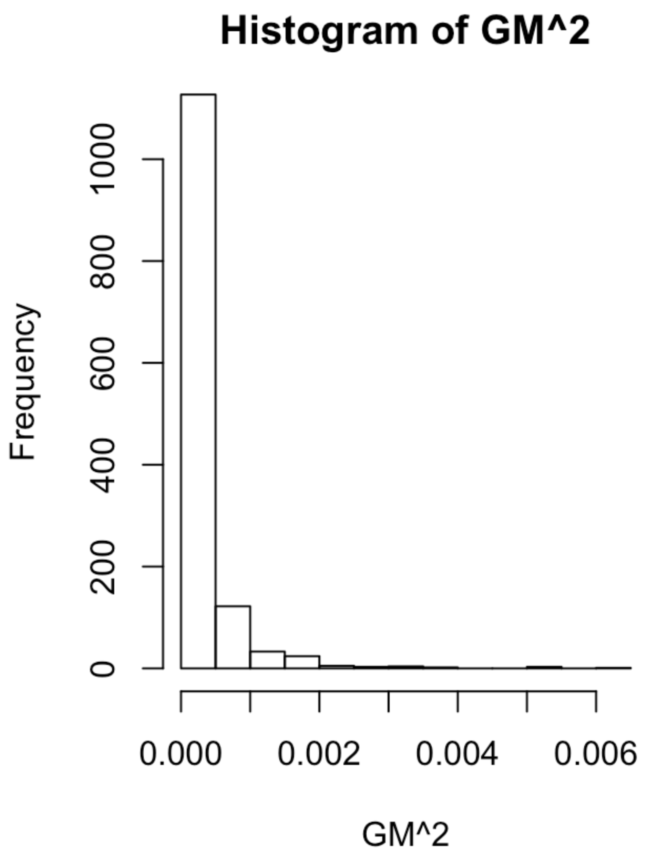
```
par(mfrow=c(1,2))
hist(PFE)
hist(PFE^2)
```





```
par(mfrow=c(1,2))
hist(GM)
hist(GM^2)
```





Problem 6.3 part(c) Perform the Jarque-Bera Test of the H0 = Data have the skewness and kurtosis matching a normal distribution. If fail to reject, we fail to reject normality of data; if reject, assert no normality.

Hide

```
library(tseries)

jarque.bera.test(citi)
```

```

    Jarque Bera Test

data:  citi
X-squared = 1753.7, df = 2, p-value < 2.2e-16
```

Hide

```
jarque.bera.test(PFE)
```

```

    Jarque Bera Test

data:  PFE
X-squared = 145.76, df = 2, p-value < 2.2e-16
```

Hide

```
jarque.bera.test(GM)
```

```
	Jarque Bera Test

data:  GM
X-squared = 145.76, df = 2, p-value < 2.2e-16
```

Hence, judging from the extremely small p-value for each of three stocks' weekly log returns, we reject the null hypothesis that the data matches a normal distribution using skewness and kurtosis coefficients. That is, the datas are not normal.

Problem 6.3 part(d) Plot ACF of return and squared return

Hide

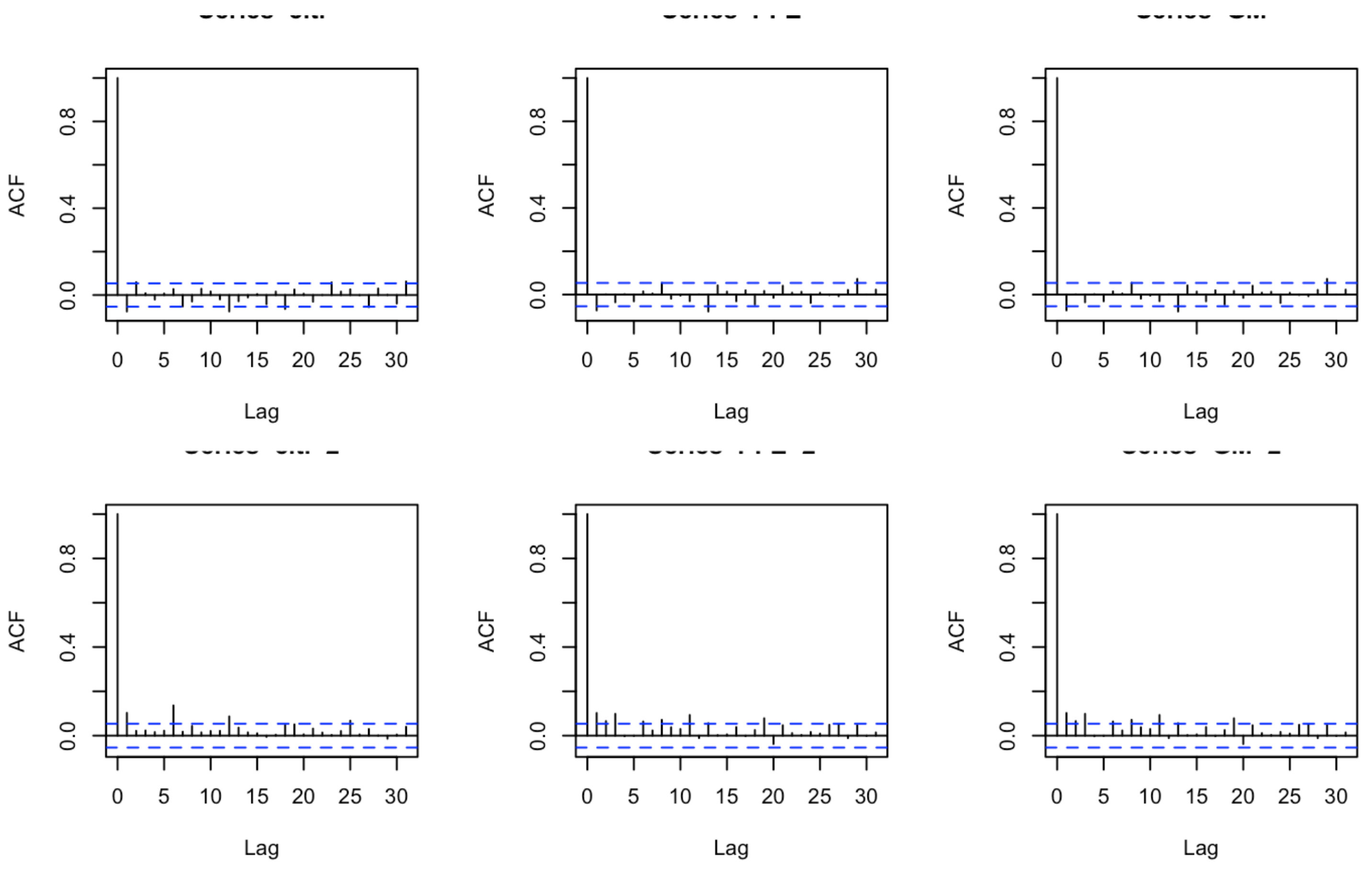
```


par(mfrow=c(2,3))
acf(citi)
acf(PFE)
```

Hide

```
acf(GM)
acf(citi^2)
```

Hide

```
acf(PFE^2)
acf(GM^2)
```



The upper panel shows the ACF of the unsquared three return series. Looking solely at these three stocks in the market, I would suggest the efficient-market hypothesis (EMH) does hold, but not perfectly. This is because for several lags, the acf value still lies outside the confidence bands. If EMH perfectly held, correlations would all have stayed around 0 within the ACF band.

The lower panel shows the ACF of the squared three return series. Because we assume it’s a mean zero quantity, the variance is equal to the expected value of squared data. This suggests we see longer memory of volatility in each of the three assets.

Problem 6.5 Build GARCH(1,1) model with standardized student-t

Hide

```
ibmdat = read.table("ibm_w_logret.txt",
                    sep="\t",
                    header = TRUE,
                    fill=FALSE,
                    strip.white=TRUE)[2]

library(rugarch)

spec_ibm <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
mean.model = list(armaOrder = c(0, 0), include.mean = TRUE, archm = FALSE,
archpow = 1, arfima = FALSE, external.regressors = NULL, archex = FALSE),
distribution.model = "std", start.pars = list(), fixed.pars = list())

ugarchfit(data=ibmdat, spec=spec_ibm)
```

```


*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model  : ARFIMA(0,0,0)
Distribution    : std

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000794    0.000248   3.2027 0.001362
omega   0.000003    0.000003   1.2199 0.222516
alpha1  0.060614    0.016449   3.6850 0.000229
beta1   0.923977    0.020854  44.3072 0.000000
shape   6.632252    0.888179   7.4672 0.000000

Robust Standard Errors:
        Estimate  Std. Error  t value Pr(>|t|)
mu      0.000794    0.000260   3.0490 0.002296
omega   0.000003    0.000010   0.3409 0.733181
alpha1  0.060614    0.048707   1.2445 0.213331
beta1   0.923977    0.067395  13.7098 0.000000
shape   6.632252    1.174328   5.6477 0.000000

LogLikelihood : 6865.311

Information Criteria
------------------------------------

Akaike       -5.7844
```

```
Bayes        -5.7722
Shibata      -5.7844
Hannan-Quinn -5.7800

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                        statistic p-value
Lag[1]                      1.331 0.24861
Lag[2*(p+q)+(p+q)-1][2]     2.801 0.15892
Lag[4*(p+q)+(p+q)-1][5]     6.361 0.07388
d.o.f=0
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic p-value
Lag[1]                     0.6280  0.4281
Lag[2*(p+q)+(p+q)-1][5]    0.9107  0.8796
Lag[4*(p+q)+(p+q)-1][9]    1.3122  0.9695
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]    0.3998 0.500 2.000  0.5272
ARCH Lag[5]    0.4850 1.440 1.667  0.8880
ARCH Lag[7]    0.8453 2.315 1.543  0.9374

Nyblom stability test
------------------------------------
Joint Statistic:  19.8192
Individual Statistics:
mu     0.04514
omega  3.35659
alpha1 0.27305
beta1  0.36140
shape  0.04802

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         1.28 1.47 1.88
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
```

	t-value <dbl>	prob <dbl>	sig <chr>
Sign Bias	0.9789353	0.32771201	

Negative Sign Bias	2.4767257	0.01332877	**
Positive Sign Bias	1.3606724	0.17374678	
Joint Effect	10.1236194	0.01754403	**

4 rows

```
Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     11.64       0.9003
2    30     21.22       0.8510
3    40     48.64       0.1385
4    50     57.05       0.2008


Elapsed time : 0.255156
```

The shape parameter captures the unknown degrees of freedom parameter v. From the optimal parameter “shape” under the GARCH Model Fit, the estimate shape is 6.632252 with corresponding standard error of 0.888179.