Lab4

Cuong Ly

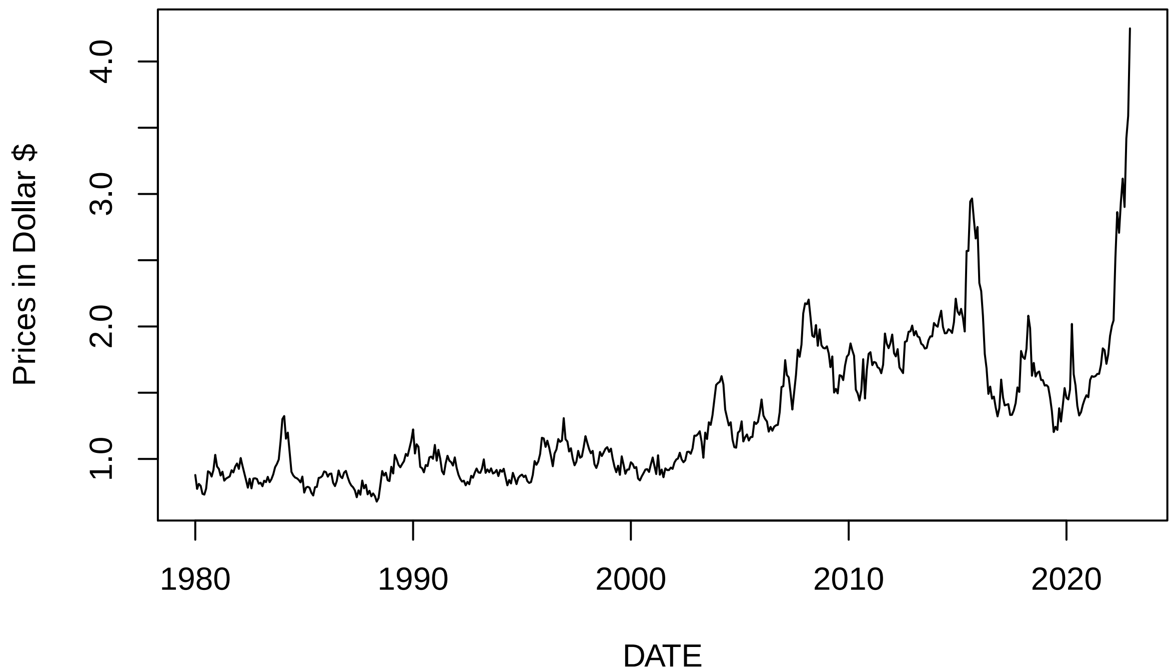
2023-02-09

Problem 1

```
# Import the data
egg <- read.csv('/Users/taikhanghao/Desktop/spring 23/time series/eggs.csv')
```

```
# Convert to time series object
egg$DATE <- as.Date(egg$DATE,format = "%Y-%m-%d")
egg_ts <- ts(egg$APU0000708111,start = 1980,frequency = 12)
```

```
plot(egg,type = 'l',ylab = "Prices in Dollar $")
```

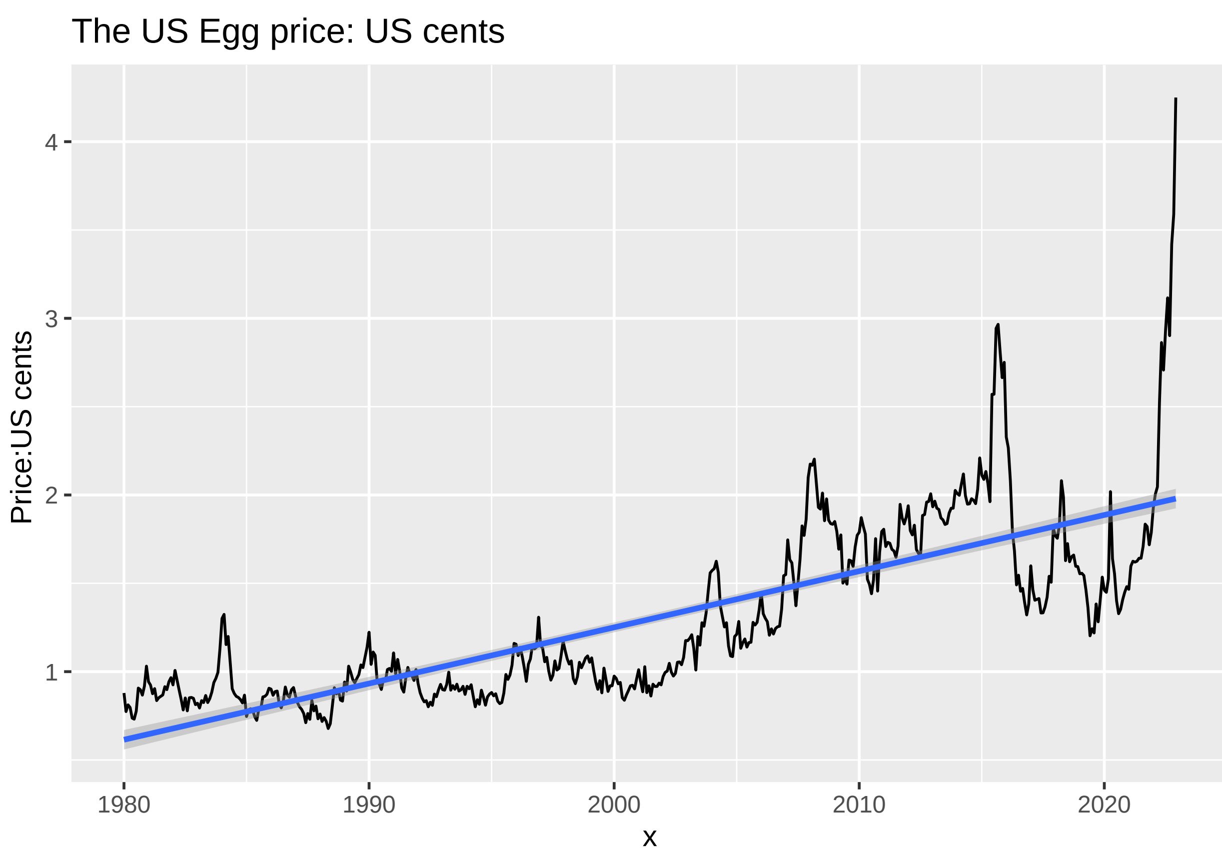



```
fit = lm(egg_ts~time(egg_ts), na.action=NULL)
# regress chicken on time
#time creates the vector of times at which a time series was sampled.

summary(fit)
```

```
##
## Call:
## lm(formula = egg_ts ~ time(egg_ts), na.action = NULL)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6652 -0.2122 -0.0248  0.1568  2.2705
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -62.356502   2.269852  -27.47   <2e-16 ***
## time(egg_ts)   0.031804   0.001134   28.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3198 on 514 degrees of freedom
## Multiple R-squared:  0.6047, Adjusted R-squared:  0.604
## F-statistic: 786.4 on 1 and 514 DF,  p-value: < 2.2e-16
```

```
y=egg_ts
x=time(egg_ts)
DD<-data.frame(x,y)
ggp <- ggplot(DD, aes(x, y)) +
  geom_line()

ggp +
  stat_smooth(method = "lm",
              formula = y ~ x,
              geom = "smooth") +ggtitle("The US Egg price: US cents ")+ylab("Price:US cents ")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

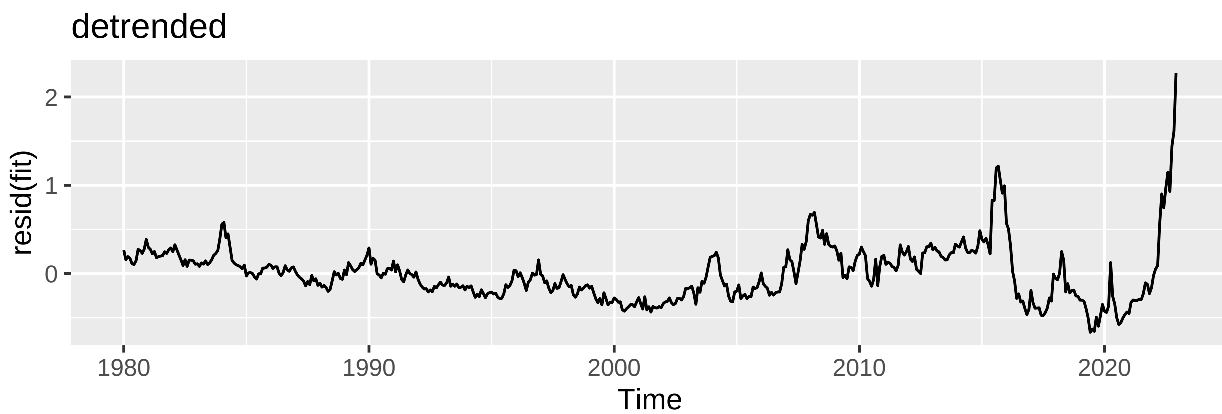


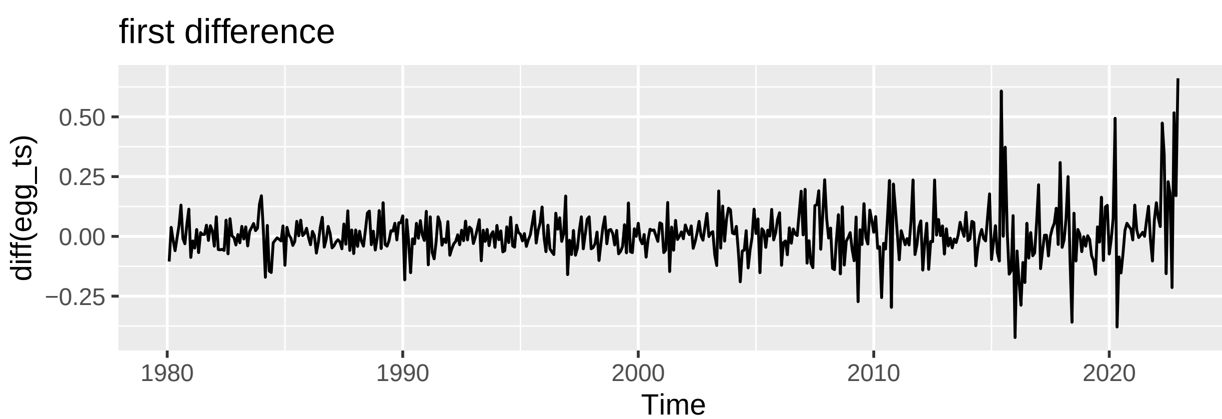

```
##
```

Egg price stays relatively constant from 1980 to 2010. It rose sharply and decline dramastically in 2016 and 2017 due to the widespread avian influenza epidemic. Once again, egg price skyrockted in the beginning of 2023 due to the outbreak of influenza.

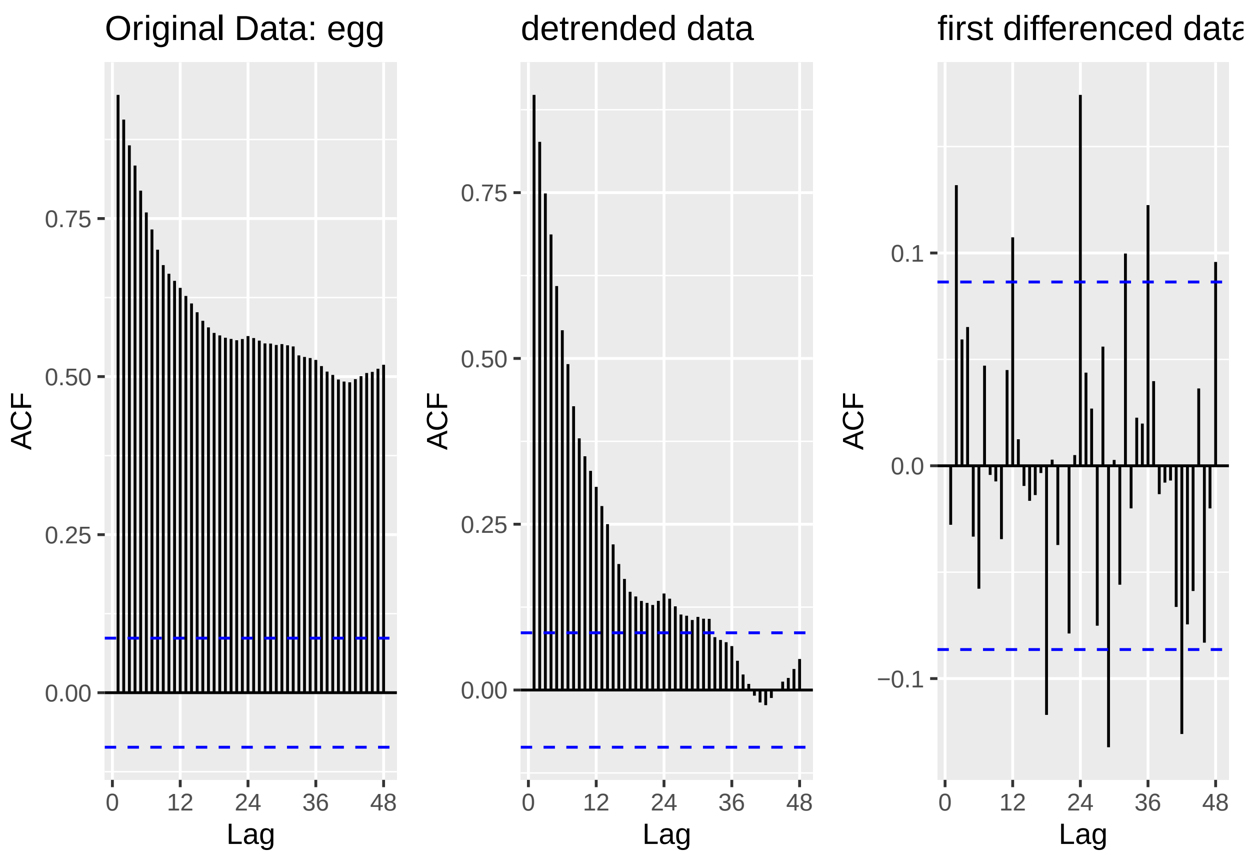
Detrend

```r
require(gridExtra)

plot1<-autoplot(resid(fit), main="detrended")
plot2<-autoplot(diff(egg_ts), main="first difference")

grid.arrange(plot1, plot2,nrow=2)
```





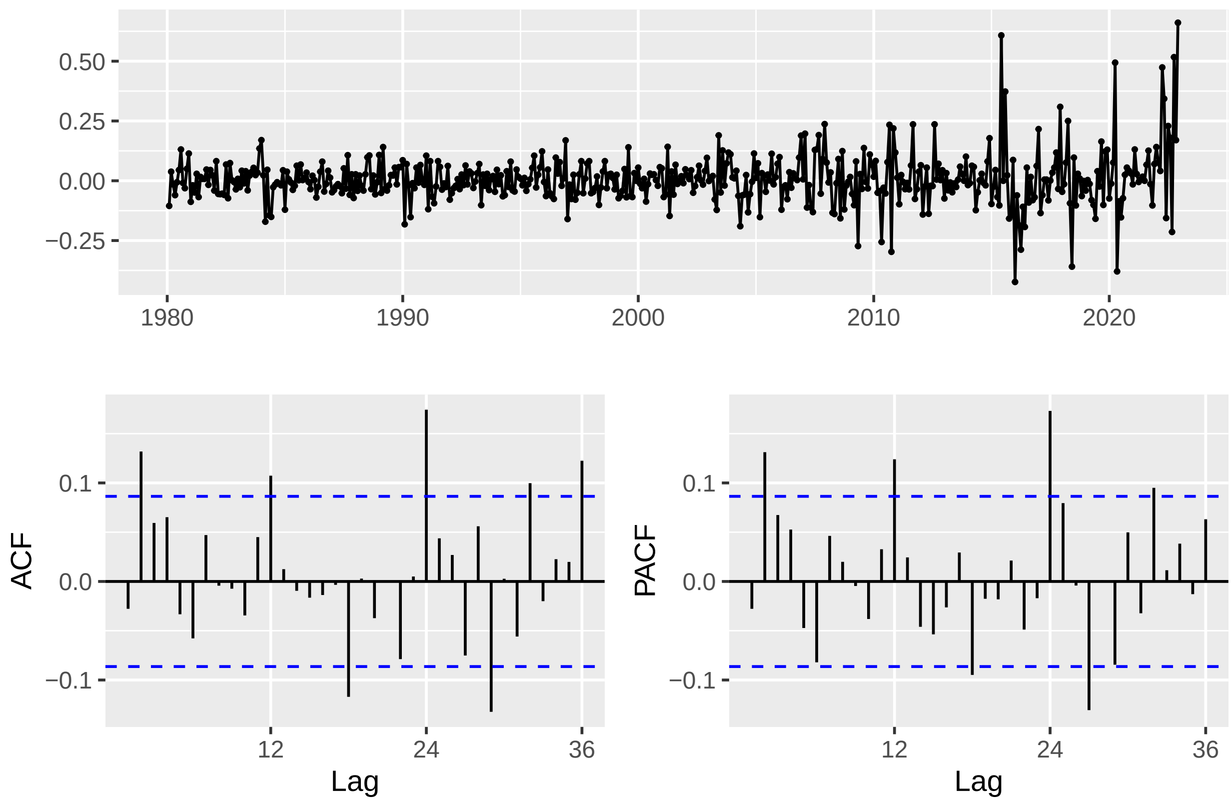

```r
#par(mfrow=c(3,1)) # plot ACFs

plot1 <- ggAcf(egg_ts, 48, main="Original Data: egg")
plot2 <- ggAcf(resid(fit), 48, main="detrended data")
plot3 <- ggAcf(diff(egg_ts), 48, main="first differenced data")

grid.arrange(plot1, plot2, plot3,ncol=3)
```



The detrend method does not work well for this egg price. Since in the ACF plot, the significant does not drop after 1,2, indicating the series is still not stationary. In contrast, difference method makes the series significant at lag 2

Differencing

```
egg_ts %>% diff() %>% ggtsdisplay()
```



Problem 2

Part a

```r
# AR(3)
plot1 <- autoplot(arima.sim(list(order=c(3,0,0), ar = c(.2,-.5,.3)), n=100), ylab="x",main=(expression(
```

```
## Warning in is.na(main): is.na() applied to non-(list or vector) of type
## 'expression'

## Warning in is.na(main): is.na() applied to non-(list or vector) of type
## 'expression'
```

```r
# MA(1)
plot2 <- autoplot(arima.sim(list(order=c(0,0,1), ma=.9), n=100), ylab="x",main=(expression(MA(1)~~~phi=
```

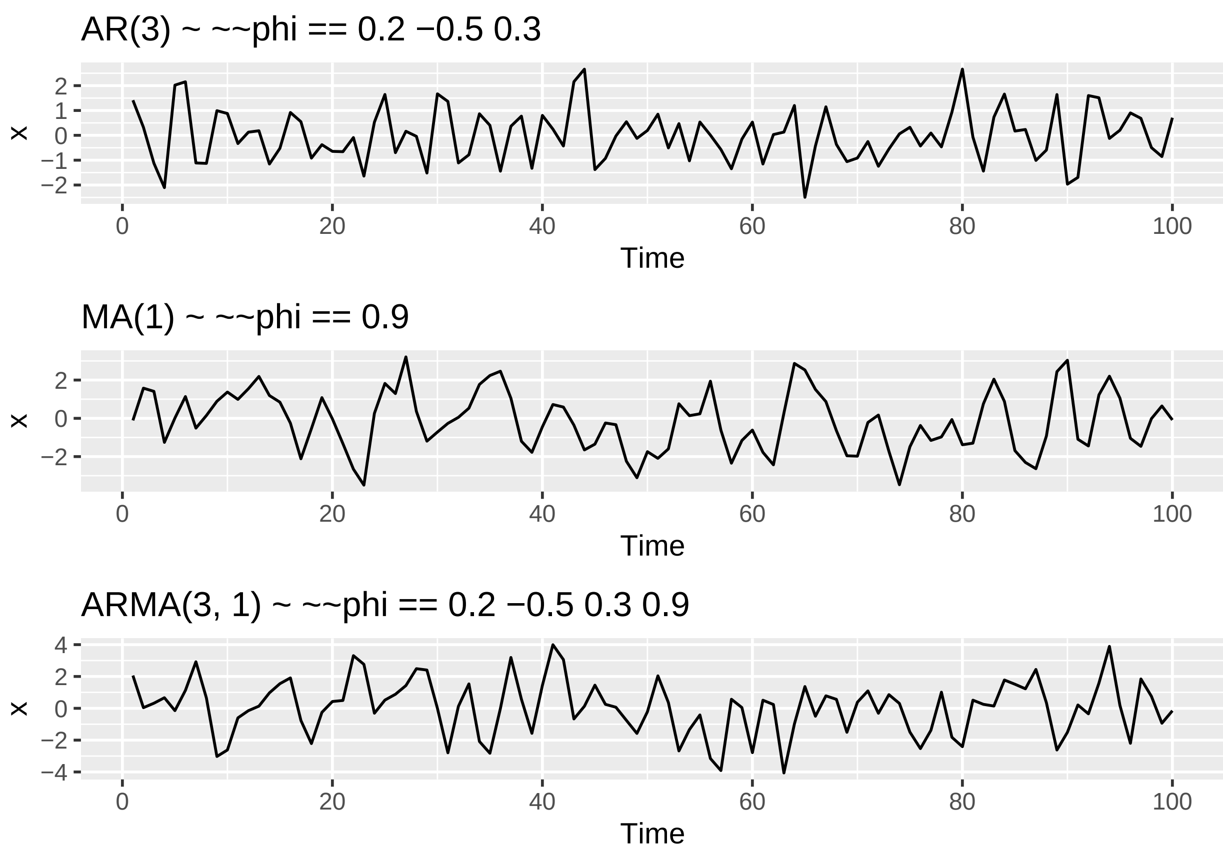
```
## Warning in is.na(main): is.na() applied to non-(list or vector) of type
```

```
## 'expression'

## Warning in is.na(main): is.na() applied to non-(list or vector) of type
## 'expression'
```

```
# ARMA(3,1)
plot3 <- autoplot(arima.sim(list(order=c(3,0,1), ar=c(.2,-.5,.3), ma=c(.9)), n=100), ylab="x",main=(exp
```

```
## Warning in is.na(main): is.na() applied to non-(list or vector) of type
## 'expression'

## Warning in is.na(main): is.na() applied to non-(list or vector) of type
## 'expression'
```

```
grid.arrange(plot1, plot2,plot3,nrow=3)
```



Plot ACF and PACF plot for AR(3)

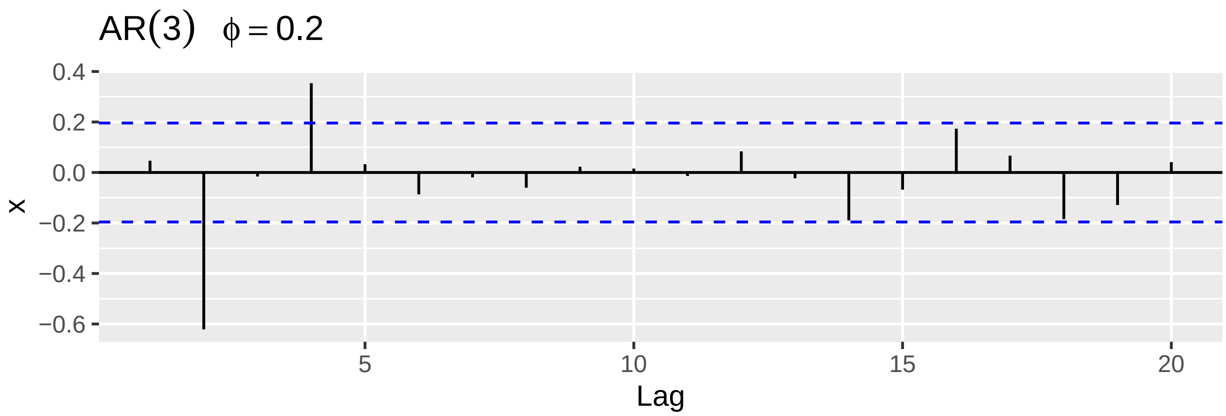
```
plot1<-ggAcf(arima.sim(list(order=c(3,0,0), ar = c(.2,-.5,.3)), n=100), ylab="x",
     main=(expression(AR(3)~~~phi==0.2,-0.5,0.3)))
```

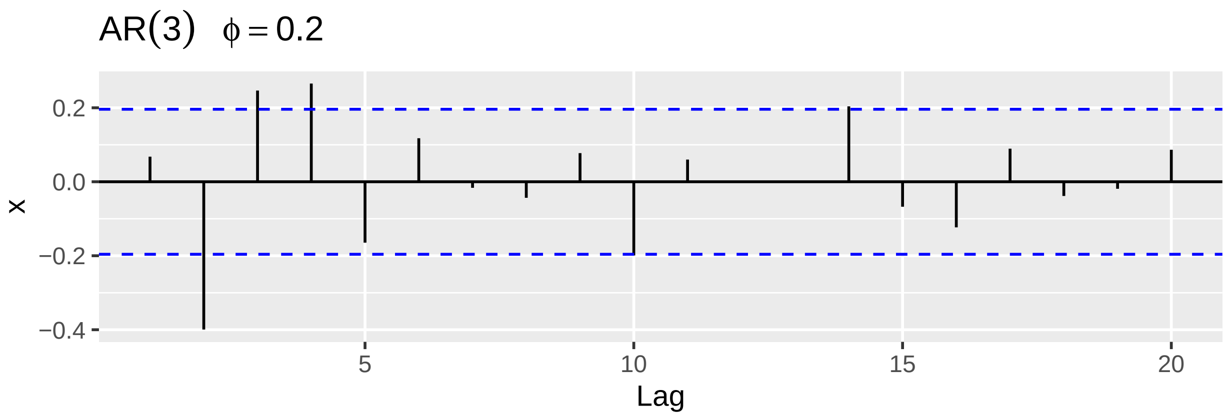
```
## Warning: Ignoring unknown parameters: ylab, main
```

```
plot2<- ggPacf(arima.sim(list(order=c(3,0,0), ar = c(.2,-.5,.3)), n=100), ylab="x",
     main=(expression(AR(3)~~~phi==0.2,-0.5,0.3)))
```

```
## Warning: Ignoring unknown parameters: ylab, main
```

```
grid.arrange(plot1, plot2,nrow=2)
```





For AR model, we look at PACF graph. At 3, the graph is significant. After 3, it is not significant anymore. Thus, the graph confirms it is AR(3)

Plot ACF and PACF plot for MA(1)

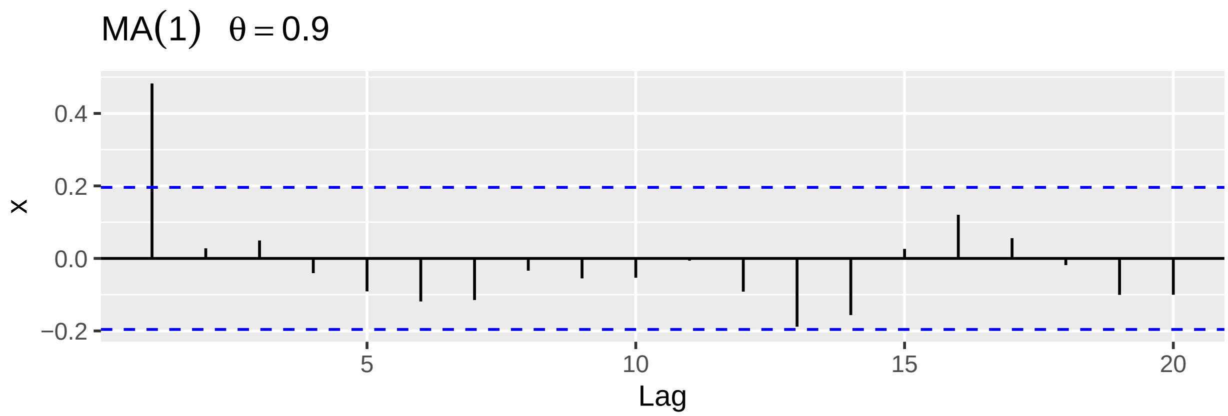
```
p1<-ggAcf(arima.sim(list(order=c(0,0,1), ma=.9), n=100), ylab="x",
     main=(expression(MA(1)~~~theta==0.9)))
```

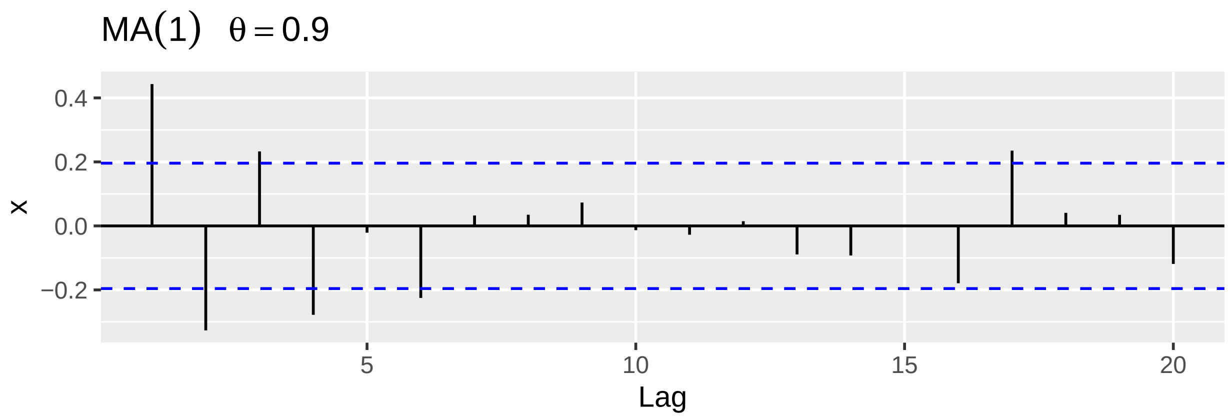
```
## Warning: Ignoring unknown parameters: ylab, main
```

```
p2<-ggPacf(arima.sim(list(order=c(0,0,1), ma=.9), n=100), ylab="x",
    main=(expression(MA(1)~~~theta==0.9)))
```

```
## Warning: Ignoring unknown parameters: ylab, main
```

```
grid.arrange(p1, p2,nrow=2)
```

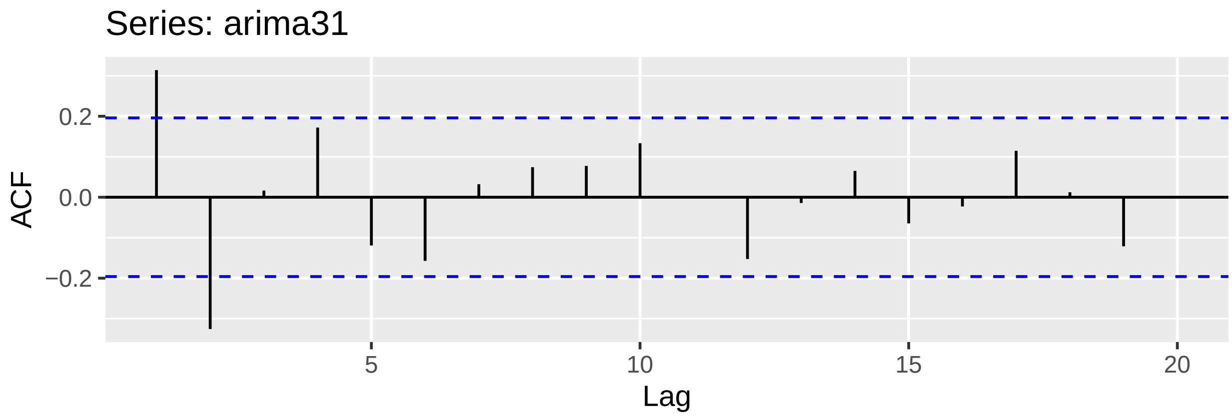


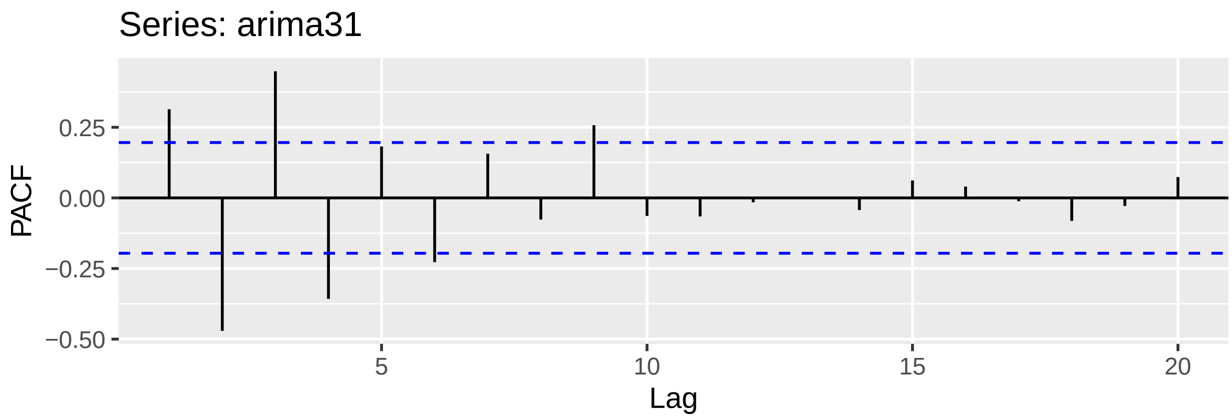


For MA model, we look at ACF graph. At 1, the graph is significant. After 1, it is not significant anymore. Thus, the graph confirms it is MA(1)

Plot ACF and PACF plot for ARMA(3,1)

```
set.seed((150))
arima31 =arima.sim(list(order=c(3,0,1), ar=c(.2,-.5,.3), ma=c(.9)), n=100)

p1<-ggAcf(arima31)
p2<-ggPacf(arima31)

grid.arrange(p1, p2,nrow=2)
```





For MA model, we look at ACF graph. At 1 and 2, the graph is significant. After 2, it is not significant anymore. For AR model, we look at PACF graph. At 1,2,3,4, the graph is significant. After 4, it is not significant anymore. Thus, we have 8 combinations, ARMA(1,1), ARMA(1,2), ARMA(2,1), ARMA(2,2), ARMA(3,1), ARMA(3,2), ARMA(4,1), ARMA(4,2)

Part b

```
#arma(1,1)
arma11 =arima.sim(list(order=c(1,0,1), ar=.2, ma=.3), n=10000)
fit11 <- Arima(arma11, order=c(1, 0, 1),include.drift = TRUE)
summary(fit11)
```

```
## Series: arma11
## ARIMA(1,0,1) with drift
##
## Coefficients:
##          ar1     ma1  intercept  drift
##       0.1814  0.3238     0.0002      0
## s.e.  0.0201  0.0191     0.0323      0
##
## sigma^2 = 1:  log likelihood = -14188
## AIC=28385.99   AICc=28386   BIC=28422.05
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE    MAPE      MASE
## Training set 2.62619e-05 0.9998483 0.7972874 97.33315 297.935 0.8356279
```

```
##                        ACF1
## Training set 0.0008800439
```

```r
#arma(1,2)
arma12 =arima.sim(list(order=c(1,0,2), ar=.2, ma=c(.3,.5)), n=10000)
fit12 <- Arima(arma12, order=c(1, 0, 2),include.drift = TRUE)
summary(fit12)
```

```
## Series: arma12
## ARIMA(1,0,2) with drift
##
## Coefficients:
##          ar1     ma1     ma2  intercept  drift
##       0.1986  0.3206  0.4981     0.0042      0
## s.e.  0.0179  0.0157  0.0101     0.0452      0
##
## sigma^2 = 0.9938:  log likelihood = -14156.33
## AIC=28324.66   AICc=28324.67   BIC=28367.92
##
## Training set error measures:
##                          ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.0003116795 0.9966591 0.7905306 89.57167 359.5804 0.8166846
##                        ACF1
## Training set -0.002167262
```

```r
#arma(2,1)
arma21 =arima.sim(list(order=c(2,0,1), ar=c(.2,.5), ma=c(.3)), n=10000)
fit21 <- Arima(arma21, order=c(2, 0, 1),include.drift = TRUE)
summary(fit21)
```

```
## Series: arma21
## ARIMA(2,0,1) with drift
##
## Coefficients:
##          ar1     ar2     ma1  intercept  drift
##       0.1927  0.5067  0.3084     0.0608      0
## s.e.  0.0237  0.0162  0.0268     0.0879      0
##
## sigma^2 = 1.022:  log likelihood = -14297.6
## AIC=28607.2   AICc=28607.21   BIC=28650.46
##
## Training set error measures:
##                          ME     RMSE       MAE      MPE   MAPE      MASE
## Training set -0.0001438591 1.010836 0.8090509 87.14591 335.95 0.8679814
##                       ACF1
## Training set 0.001587458
```

```r
#arma(2,2)
arma22 =arima.sim(list(order=c(2,0,2), ar=c(.2,.5), ma=c(.3,.5)), n=10000)
fit22 <- Arima(arma22, order=c(2, 0, 2),include.drift = TRUE)
summary(fit22)
```

```
## Series: arma22
## ARIMA(2,0,2) with drift
##
## Coefficients:
##          ar1     ar2     ma1     ma2  intercept  drift
##       0.2115  0.4953  0.2868  0.4960     0.1530      0
## s.e.  0.0120  0.0119  0.0117  0.0104     0.1216      0
##
## sigma^2 = 1.002:  log likelihood = -14196.99
## AIC=28407.98   AICc=28407.99   BIC=28458.45
##
## Training set error measures:
##                         ME     RMSE       MAE     MPE     MAPE      MASE
## Training set -4.93938e-05 1.000647 0.7989058 219.967 448.0443 0.6863059
##                     ACF1
## Training set 0.002658372
```

```
#arma(3,1)
arma31 =arima.sim(list(order=c(3,0,1), ar=c(.2,-.5,.3), ma=c(.3)), n=10000)
fit31 <- Arima(arma31, order=c(3, 0, 1),include.drift = TRUE)
summary(fit31)
```

```
## Series: arma31
## ARIMA(3,0,1) with drift
##
## Coefficients:
##          ar1      ar2     ar3     ma1  intercept  drift
##       0.1723  -0.4868  0.2795  0.3288    -0.0240      0
## s.e.  0.0209   0.0099  0.0141  0.0211     0.0258      0
##
## sigma^2 = 1.011:  log likelihood = -14240.19
## AIC=28494.38   AICc=28494.39   BIC=28544.85
##
## Training set error measures:
##                          ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -8.935454e-07 1.005038 0.8018486 72.56644 272.4254 0.6217425
##                      ACF1
## Training set 0.0003251052
```

```
#arma(3,2)
arma32 =arima.sim(list(order=c(3,0,2), ar=c(.2,-.5,.3), ma=c(.3,.6)), n=10000)
fit32 <- Arima(arma32, order=c(3, 0, 2),include.drift = TRUE)
summary(fit32)
```

```
## Series: arma32
## ARIMA(3,0,2) with drift
##
## Coefficients:
##          ar1      ar2     ar3     ma1     ma2  intercept  drift
##       0.0689  -0.2335  0.1874  0.4281  0.3880    -0.0281      0
## s.e.  0.3416   0.1617  0.0806  0.3418  0.1847     0.0371      0
##
## sigma^2 = 0.9966:  log likelihood = -14168.77
```

```
## AIC=28353.55   AICc=28353.56   BIC=28411.23
##
## Training set error measures:
##                         ME     RMSE       MAE     MPE     MAPE     MASE
## Training set -5.211586e-05 0.997928 0.7960254 62.4937 242.7571 0.856858
##                      ACF1
## Training set 0.0005221293
```

```r
#arma(4,1)
arma41 =arima.sim(list(order=c(4,0,1), ar=c(.2,-.5,.3,-.7), ma=c(.3)), n=10000)
fit41 <- Arima(arma41, order=c(4, 0, 1),include.drift = TRUE)
summary(fit41)
```

```
## Series: arma41
## ARIMA(4,0,1) with drift
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ma1  intercept  drift
##       0.2021  -0.5016  0.3038  -0.6931  0.3010     0.0005      0
## s.e.  0.0094   0.0070  0.0071   0.0077  0.0127     0.0152      0
##
## sigma^2 = 0.9672:  log likelihood = -14020.81
## AIC=28057.63   AICc=28057.64   BIC=28115.31
##
## Training set error measures:
##                         ME      RMSE       MAE     MPE     MAPE      MASE
## Training set -3.418912e-06 0.9831121 0.7861035 82.4428 283.1184 0.4724694
##                      ACF1
## Training set -0.001738375
```

```r
#arma(4,2)
arma42 =arima.sim(list(order=c(4,0,2), ar=c(.2,-.5,.3,-.7), ma=c(.3,.6)), n=10000)
fit42 <- Arima(arma42, order=c(4, 0, 2),include.drift = TRUE)
summary(fit42)
```

```
## Series: arma42
## ARIMA(4,0,2) with drift
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ma1     ma2  intercept  drift
##       0.1863  -0.4978  0.3107  -0.6911  0.3052  0.6043    -0.0445      0
## s.e.  0.0086   0.0080  0.0069   0.0074  0.0101  0.0101     0.0225      0
##
## sigma^2 = 0.9959:  log likelihood = -14166.56
## AIC=28351.11   AICc=28351.13   BIC=28416
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 5.479827e-05 0.9975421 0.7985749 152.5398 379.4287 0.5602724
##                    ACF1
## Training set 0.00803583
```

ARMA(1,1) has the lowest AIC and BIC. Thus, it is the best model

Part c x_t = 0.2x_(t-1) + w_t + 0.3w(t-1)

Part d

```r
# ARMA(1,1)
set.seed(151)
xt11=arima.sim(list(order=c(1,0,1), ar=.2,ma=.3),n=1000)
auto.arima(xt11)
```

```
## Series: xt11
## ARIMA(0,0,2) with zero mean
##
## Coefficients:
##          ma1     ma2
##       0.5341  0.1141
## s.e.  0.0312  0.0328
##
## sigma^2 = 0.9764:  log likelihood = -1406.13
## AIC=2818.26   AICc=2818.28   BIC=2832.98
```

```r
# ARMA(1,2)
set.seed(151)
xt12=arima.sim(list(order=c(1,0,2), ar=.2,ma=c(.3,.5)),n=1000)
auto.arima(xt12)
```

```
## Series: xt12
## ARIMA(1,0,2) with zero mean
##
## Coefficients:
##          ar1     ma1     ma2
##       0.1817  0.3541  0.5143
## s.e.  0.0556  0.0486  0.0302
##
## sigma^2 = 0.9753:  log likelihood = -1405.36
## AIC=2818.73   AICc=2818.77   BIC=2838.36
```

```r
# ARMA(2,1)
set.seed(151)
xt21=arima.sim(list(order=c(2,0,1), ar=c(.2,.5), ma=c(.3)),n=1000)
auto.arima(xt21)
```

```
## Series: xt21
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##          ar1     ar2     ma1
##       0.2231  0.4892  0.3059
## s.e.  0.0772  0.0547  0.0856
##
## sigma^2 = 0.9916:  log likelihood = -1413.67
## AIC=2835.35   AICc=2835.39   BIC=2854.98
```

```r
# ARMA(2,2)
set.seed(151)
xt22=arima.sim(list(order=c(2,0,2), ar=c(.2,.5), ma=c(.3,.5)),n=1000)
auto.arima(xt22)
```

```
## Series: xt22
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1     ar2     ma1     ma2
##       0.2030  0.5016  0.3385  0.5041
## s.e.  0.0375  0.0364  0.0371  0.0312
##
## sigma^2 = 0.9897:  log likelihood = -1412.92
## AIC=2835.84   AICc=2835.9   BIC=2860.37
```

```r
# ARMA(3,1)
set.seed(151)
xt31=arima.sim(list(order=c(3,0,1), ar=c(.2,-.5,.3), ma=c(.3)),n=1000)
auto.arima(xt31)
```

```
## Series: xt31
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2     ar3     ma1
##       0.2476  -0.5200  0.3131  0.2835
## s.e.  0.0675   0.0321  0.0453  0.0703
##
## sigma^2 = 0.9917:  log likelihood = -1413.34
## AIC=2836.68   AICc=2836.74   BIC=2861.22
```

```r
# ARMA(3,2)
set.seed(151)
xt32=arima.sim(list(order=c(3,0,2), ar=c(.2,-.5,.3), ma=c(.3,.6)),n=1000)
auto.arima(xt32)
```

```
## Series: xt32
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ma1
##       1.0846  -0.3745  0.0768  -0.5547
## s.e.  0.2566   0.1379  0.0322   0.2560
##
## sigma^2 = 0.9938:  log likelihood = -1413.98
## AIC=2837.96   AICc=2838.02   BIC=2862.5
```

```r
# ARMA(4,1)
set.seed(151)
xt41=arima.sim(list(order=c(4,0,1), ar=c(.2,-.5,.3,-.7), ma=c(.3)),n=1000)
auto.arima(xt41)
```

```
## Series: xt41
## ARIMA(4,0,1) with zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4     ma1
##       0.1804  -0.5025  0.2701  -0.6838  0.3285
## s.e.  0.0304   0.0229  0.0232   0.0243  0.0407
##
## sigma^2 = 0.9737:  log likelihood = -1404.78
## AIC=2821.56   AICc=2821.65   BIC=2851.01
```

```r
# ARMA(4,2)
set.seed(151)
xt42=arima.sim(list(order=c(4,0,2), ar=c(.2,-.5,.3,-.7), ma=c(.3,.6)),n=1000)
auto.arima(xt42)
```

```
## Series: xt42
## ARIMA(1,0,5) with zero mean
##
## Coefficients:
##           ar1     ma1     ma2     ma3      ma4      ma5
##       -0.2439  0.7380  0.2116  0.1028  -0.4952  -0.4706
## s.e.   0.0745  0.0678  0.0426  0.0341   0.0324   0.0286
##
## sigma^2 = 1.188:  log likelihood = -1503.08
## AIC=3020.16   AICc=3020.27   BIC=3054.51
```

Using auto.arima(), ARMA(1,1) still has the lowest AIC and BIC. This finding matches the result in part b

Part e The best model is ARMA(1,1). Lag 1 is significant and has a strong correlation with the current state. The noise at lag 1 is also significant. This differs from the original model- ARMA(3,3)