ORIE5550_Homework5_Markdown

Luis Alonso Cendra Villalobos (lc2234)

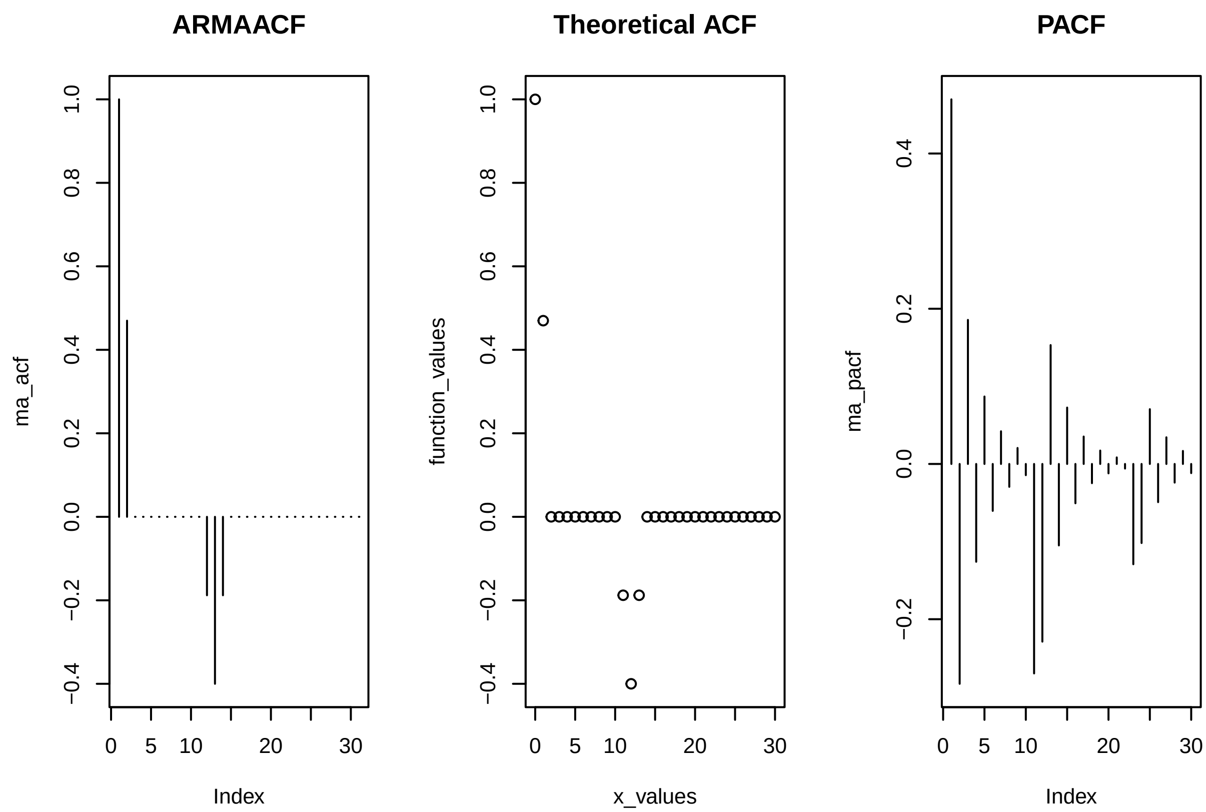
2024-03-07

Problem 1

Consider the following seasonal MA model with zero mean and period $s = 12$ given by $X_t = (I + \theta_1 B)(I + \Theta_1 B^{12})Z_t = (I + \theta_1 B + \Theta_1 B^{12} + \theta_1\Theta_1 B^{13})Z_t$ with $Z_t \sim WN(0, \sigma_Z^2)$. This model is denoted SARIMA(0,0,1) × (0,0,1)[12]. Do the following for this model.

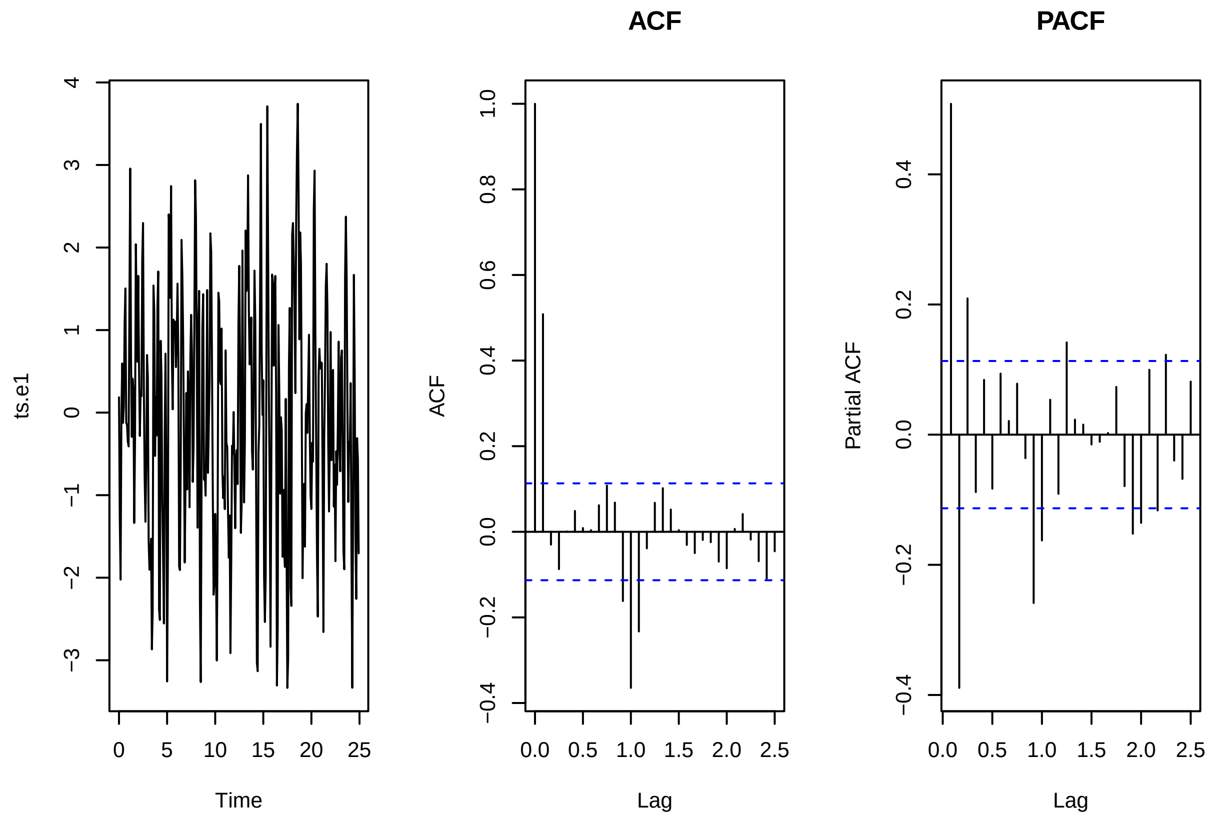
###(a) Compute theoretically the ACF for this model; Then, fix $\theta_1 = 0.7$ and $\Theta_1 = -0.5$. Use ARMAacf to check if your theoretical ACF is identical to the result from the function for lags $h = 1, ..., 30$

```r
#theoretical ACF calculation
custom_function <- function(x) {
  if (x == 0) {
    return(1)
  } else if (x == 1) {
    return(0.4698)
  } else if (x == 11 | x == 13) {
    return(-0.1879)
  } else if (x == 12) {
    return(-0.4)
  } else {
    return(0)
  }
}
x_values <- 0:30
function_values <- sapply(x_values, custom_function)

# ACF and PACF generated by R's ARMAacf
ma_acf <- ARMAacf(ma = c(0.7,0,0,0,0,0,0,0,0,0,0,-0.5,-0.5*0.7),lag.max=30,pacf=F)
ma_pacf <- ARMAacf(ma = c(0.7,0,0,0,0,0,0,0,0,0,0,-0.5,-0.5*0.7),lag.max=30,pacf=T)

par(mfrow = c(1, 3))
plot(ma_acf, type='h', main = "ARMAACF")
plot(x_values, function_values, main = "Theoretical ACF")
plot(ma_pacf, type='h', main = "PACF")
```



(b) Use set.seed(99) to generate time series data of length 300 from this model assuming that $\theta_1 = 0.7$, $\Theta_1 = -0.5$, and {Zt} is IID standard normal; Produce a time plot and sample ACF and PACF plots

```
# SARIMA$(1,0,0)\times (1,0,0)_{12}$
set.seed(99)
ts.e1 <- sarima.sim(ma=0.7, sma= -0.5, S=12,n=300)
par(mfrow = c(1, 3))
plot.ts(ts.e1)
acf(ts.e1,lag.max = 30, main= "ACF")
pacf(ts.e1,lag.max = 30, main= "PACF")
```



(c) For the generated data in (b), fit a SARIMA(0,0,1)×(0,0,1)12 model by using the function Arima from the R package forecast; Write down the exact model that was fitted.

```
sarima_fit <- arima(ts.e1, order = c(0, 0, 1),
                    seasonal = list(order = c(0, 0, 1), period = 12),
                    include.mean = FALSE)
sarima_fit
```

```
##
## Call:
## arima(x = ts.e1, order = c(0, 0, 1), seasonal = list(order = c(0, 0, 1), period = 12),
##     include.mean = FALSE)
##
## Coefficients:
##          ma1     sma1
##       0.7461  -0.5557
## s.e.  0.0371   0.0560
##
## sigma^2 estimated as 1.03:  log likelihood = -432.72,  aic = 871.44
```

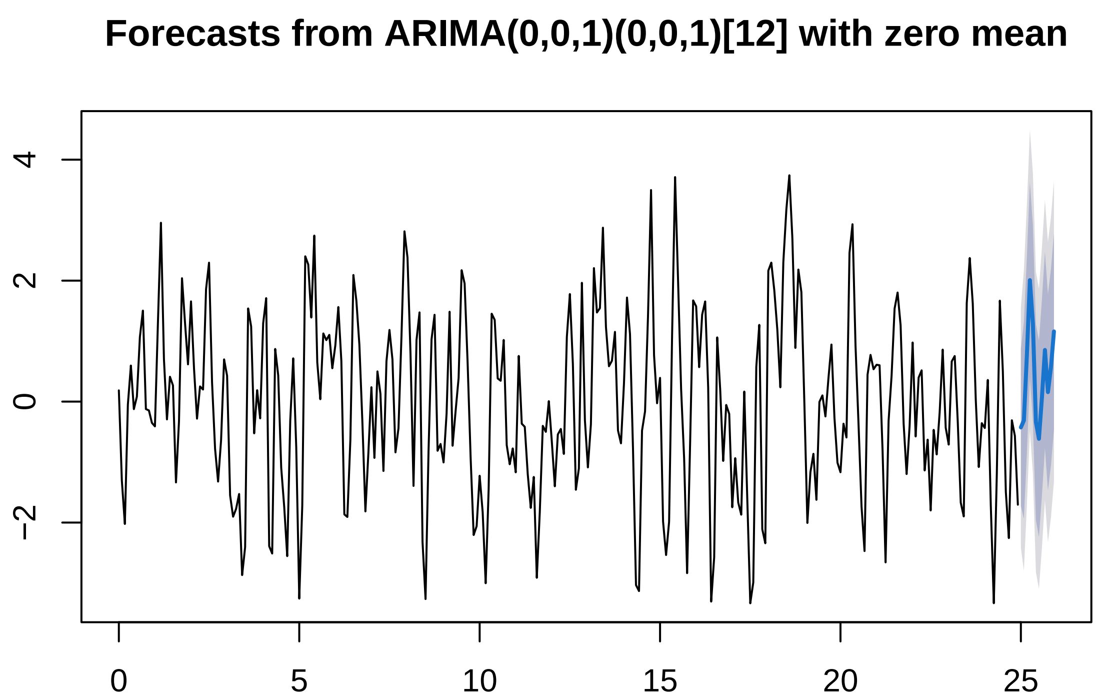
```
theta_ma1 = sarima_fit$coef[1]
theta_sma1 = sarima_fit$coef[2]
```

The exact model is:

$$X_t = Z_t + 0.75Z_{t-1} + -0.56Z_{t-12} + (0.75) * (-0.56)Z_{t-13} \text{ with } \{Z_t : t \in \mathbb{Z}\} \sim WN(0, \sigma_z^2)$$

(d) For the generated data, produce 12-steps-ahead forecasts by using the R function forecast and the model fitted in part (c); Include the usual plot with forecasts.

```
h <- 12
ts.e1.forecast <- forecast(sarima_fit, h)
plot(ts.e1.forecast)
```

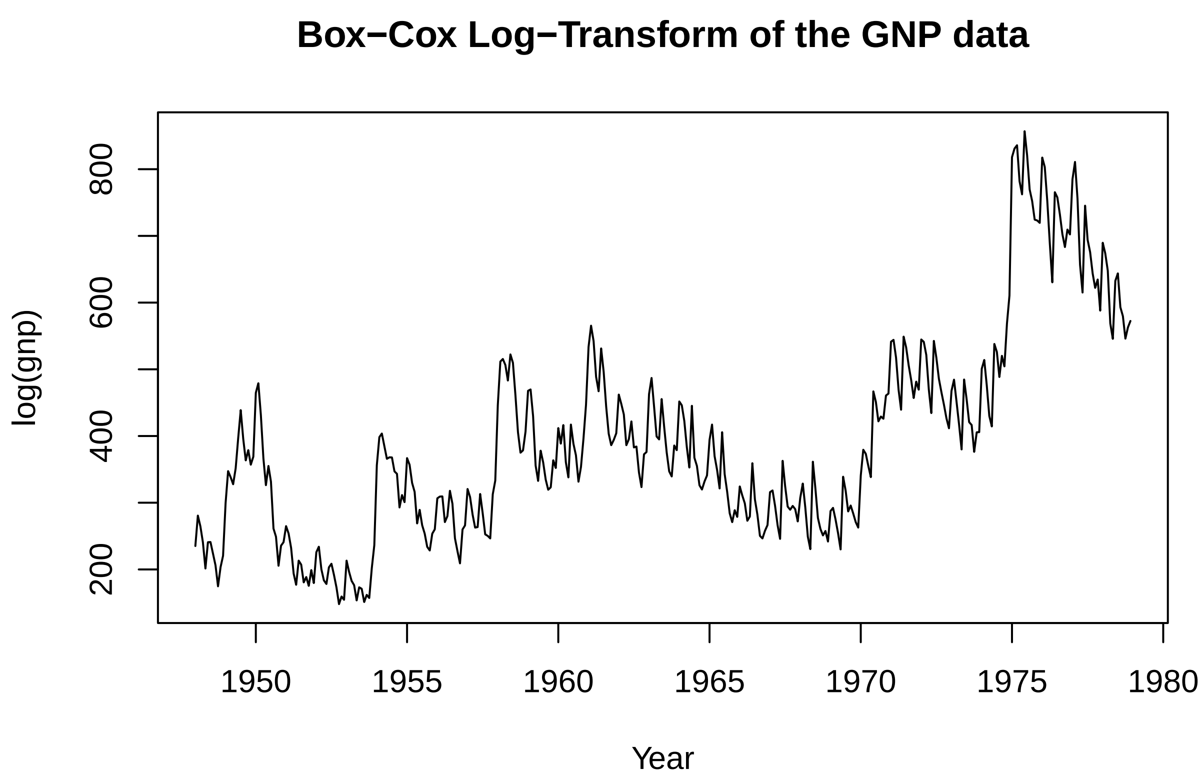


Problem 2.

Consider the monthly U.S. unemployment series unemp in the R package astsa. Leave out the last 12 observations and denote these samples as test data.

(a) Take the logarithm of the series remaining unemp and call the result series training data. By referring to Problem 2 in Homework 2, fit a quadratic polynomial trend and a seasonal component to the training data using regression; Consider the residual series from the regression and use the function auto.arima to fit the "best" SARIMA model for it; Use this SARIMA model to forecast the residual series 12 steps ahead; Finally, combined with the regression, translate this forecast into the 12-step-ahead predictions of the "original scale"; On the time plot of unemp (the whole series), overlap original scale of fitted values and predictions with different colors.

```
unemp_data <- ts(unemp, start = c(1948, 1), end = c(1978, 12) , frequency = 12)
plot(unemp_data, type="l", main="Box-Cox Log-Transform of the GNP data",
ylab = "log(gnp)", xlab = "Year")
```




```
# Leave out the last 12 observations as test data
training_data <- head(unemp_data,-12)
training_data = log(training_data)
test_data <- tail(unemp_data, 12)
tt <- seq(1, length(training_data), by=1)
tt2 <- tt^2
```

```r
y <- training_data

fitModel <- lm( y ~ tt + tt2 +
              cos(2*tt*pi/12) + sin(2*tt*pi/12) +
              cos(2*tt*pi/6) + sin(2*tt*pi/6) +
              cos(2*tt*pi/4) + sin(2*tt*pi/4))

summary(fitModel)
```

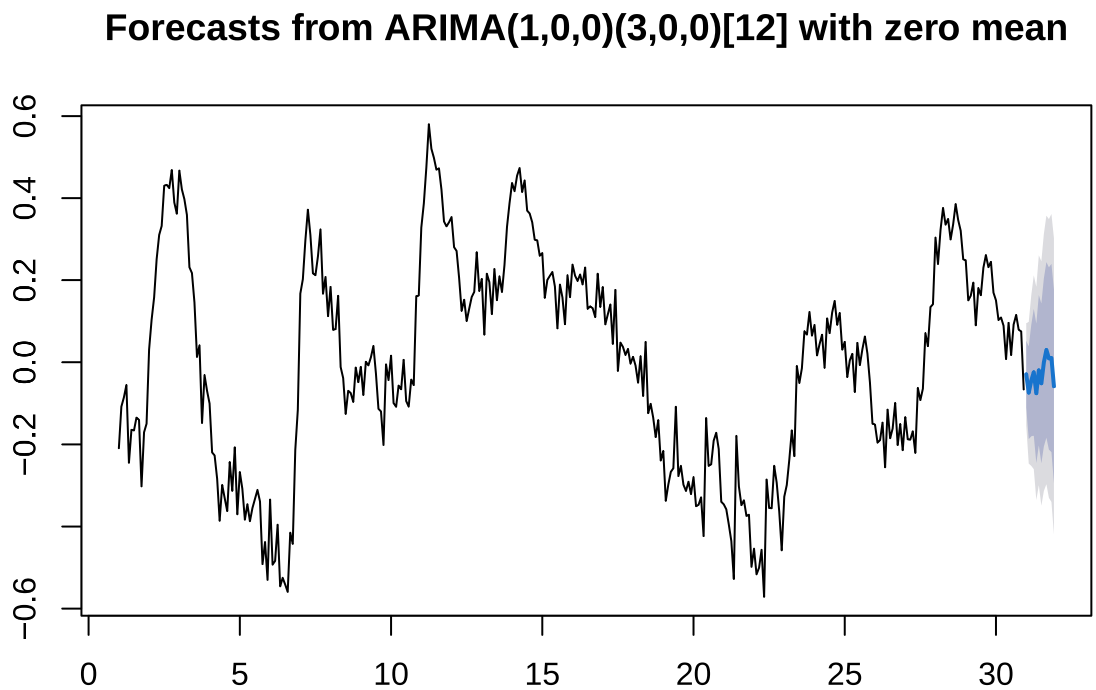
```
##
## Call:
## lm(formula = y ~ tt + tt2 + cos(2 * tt * pi/12) + sin(2 * tt *
##     pi/12) + cos(2 * tt * pi/6) + sin(2 * tt * pi/6) + cos(2 *
##     tt * pi/4) + sin(2 * tt * pi/4))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.57133 -0.19716  0.01718  0.19083  0.58010
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5.581e+00  4.135e-02 134.969  < 2e-16 ***
## tt                   2.361e-05  5.289e-04   0.045   0.9644
## tt2                  6.870e-06  1.419e-06   4.842 1.93e-06 ***
## cos(2 * tt * pi/12) -1.354e-02  1.938e-02  -0.698   0.4853
## sin(2 * tt * pi/12)  7.932e-02  1.939e-02   4.091 5.32e-05 ***
## cos(2 * tt * pi/6)   1.523e-02  1.938e-02   0.786   0.4326
## sin(2 * tt * pi/6)   8.274e-02  1.938e-02   4.269 2.53e-05 ***
## cos(2 * tt * pi/4)  -3.839e-02  1.938e-02  -1.981   0.0484 *
## sin(2 * tt * pi/4)  -1.859e-02  1.938e-02  -0.959   0.3381
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.26 on 351 degrees of freedom
## Multiple R-squared:  0.544,  Adjusted R-squared:  0.5336
## F-statistic: 52.34 on 8 and 351 DF,  p-value: < 2.2e-16
```

```r
residuals <- ts(fitModel$residuals, frequency=12)

auto_sarima <- auto.arima(residuals, max.p = 5, max.q = 5, max.d = 2,
                        start.p = 0, start.q = 0,
                        max.P = 5, max.Q = 5, max.D = 2,
                        start.P = 0, start.Q = 0,
                        allowdrift = FALSE, ic = "aic")

h <- 12
residuals.forecast <- forecast(auto_sarima, h)
summary(residuals.forecast)
```

```
##
## Forecast method: ARIMA(1,0,0)(3,0,0)[12] with zero mean
##
```

```
## Model Information:
## Series: residuals
## ARIMA(1,0,0)(3,0,0)[12] with zero mean
##
## Coefficients:
##          ar1    sar1    sar2    sar3
##       0.9661  0.2840  0.2435  0.2221
## s.e.  0.0125  0.0525  0.0553  0.0554
##
## sigma^2 = 0.004015:  log likelihood = 478.4
## AIC=-946.8   AICc=-946.63   BIC=-927.37
##
## Error measures:
##                        ME       RMSE        MAE      MPE    MAPE      MASE
## Training set 3.795917e-05 0.06301439 0.04917142 15.39465 73.7514 0.2404558
##                   ACF1
## Training set 0.1027848
##
## Forecasts:
##        Point Forecast      Lo 80      Hi 80      Lo 95      Hi 95
## Jan 31  -0.0290396232 -0.1102482 0.05216898 -0.1532375 0.09515822
## Feb 31  -0.0740212308 -0.1869395 0.03889701 -0.2467148 0.09867232
## Mar 31  -0.0444051415 -0.1804063 0.09159603 -0.2524010 0.16359071
## Apr 31  -0.0240658173 -0.1785310 0.13039940 -0.2603000 0.21216834
## May 31  -0.0756425066 -0.2455412 0.09425615 -0.3354801 0.18419507
## Jun 31  -0.0192455525 -0.2023801 0.16388898 -0.2993257 0.26083455
## Jul 31  -0.0514230371 -0.2461018 0.14325570 -0.3491585 0.24631240
## Aug 31  -0.0001681622 -0.2050362 0.20469992 -0.3134869 0.31315053
## Sep 31   0.0302986297 -0.1836429 0.24424011 -0.2968966 0.35749389
## Oct 31   0.0094126968 -0.2126637 0.23148913 -0.3302239 0.34904930
## Nov 31   0.0105698183 -0.2188397 0.23997936 -0.3402818 0.36142144
## Dec 31  -0.0586142157 -0.2946631 0.17743466 -0.4196198 0.30239139
```

```r
plot(residuals.forecast)
```




```
forecasted_residuals<-matrix(0,nrow=1,ncol=12)
for (i in 1:12) {
  forecasted_residuals[1,i]<-as.numeric(residuals.forecast$mean)[i]
}

estimated_coeffs<-matrix(0,nrow=1,ncol=9)
for (i in 1:9) {
  estimated_coeffs[1,i]<-fitModel$coefficients[i]
}

time_matrix<-matrix(0,nrow=12,ncol=9)
time_matrix[,1]<- 1
time_matrix[,2]<- 361:372
time_matrix[,3]<- time_matrix[,2]^2
time_matrix[,4]<- cos(2 * time_matrix[,2] * pi / 12)
time_matrix[,5]<- sin(2 * time_matrix[,2] * pi / 12)
time_matrix[,6]<- cos(2 * time_matrix[,2] * pi / 6)
time_matrix[,7]<- sin(2 * time_matrix[,2] * pi / 6)
time_matrix[,8]<- cos(2 * time_matrix[,2] * pi / 4)
time_matrix[,9]<- sin(2 * time_matrix[,2] * pi / 4)
time_matrix<-t(time_matrix)

trendAndseasonal_forecast = estimated_coeffs %*% time_matrix
ts.forecast.pred<-trendAndseasonal_forecast + forecasted_residuals
exp(ts.forecast.pred)
```

```
##           [,1]     [,2]    [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
## [1,] 695.1989 720.4996 687.389 622.2749 600.5097 704.5926 687.6466 654.1976
##           [,9]    [,10]    [,11]   [,12]
## [1,] 627.7188 615.9107 631.9977 629.525
```

```
msfe_2a <-mean((exp(ts.forecast.pred)-as.numeric(test_data))^2)
msfe_2a
```

```
## [1] 2973.572
```

```
##############
##Fitted values


estimated_coeffs<-matrix(0,nrow=1,ncol=9)
for (i in 1:9) {
  estimated_coeffs[1,i]<-fitModel$coefficients[i]
}

time_matrix<-matrix(0,nrow=360,ncol=9)
time_matrix[,1]<- 1
time_matrix[,2]<- 1:360
time_matrix[,3]<- time_matrix[,2]^2
time_matrix[,4]<- cos(2 * time_matrix[,2] * pi / 12)
time_matrix[,5]<- sin(2 * time_matrix[,2] * pi / 12)
time_matrix[,6]<- cos(2 * time_matrix[,2] * pi / 6)
time_matrix[,7]<- sin(2 * time_matrix[,2] * pi / 6)
time_matrix[,8]<- cos(2 * time_matrix[,2] * pi / 4)
time_matrix[,9]<- sin(2 * time_matrix[,2] * pi / 4)
time_matrix<-t(time_matrix)

fitted_values = estimated_coeffs %*% time_matrix


#Plotting

library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

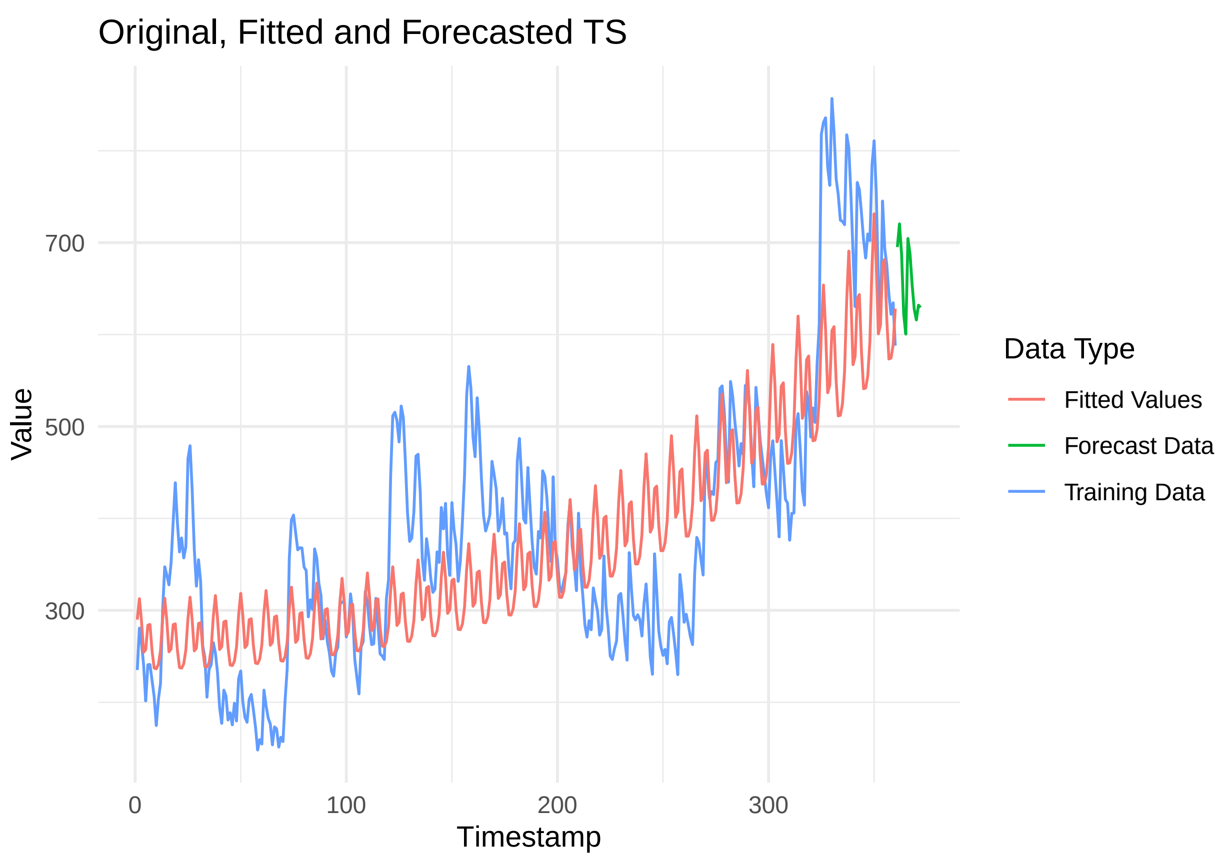
```
df1 <- data.frame(train_compare = exp(training_data), timestamp = seq(1,360,1))
df2 <- data.frame(forecast_data = as.numeric(exp(ts.forecast.pred)), timestamp = seq(361,372,1))
df3 <- data.frame(fitted_compare = as.numeric(exp(estimated_coeffs %*% time_matrix)), timestamp = seq(1
merged_df <- merge(df1, df2, by = "timestamp", all = TRUE)
merged_df <- merge(merged_df, df3, by = "timestamp", all = TRUE)

ggplot(merged_df, aes(x = timestamp)) +
```

```r
  geom_line(aes(y = train_compare, color = "Training Data")) +
  geom_line(aes(y = forecast_data, color = "Forecast Data")) +
  geom_line(aes(y = fitted_compare, color = "Fitted Values")) +
  labs(x = "Timestamp", y = "Value", color = "Data Type") +
  ggtitle("Original, Fitted and Forecasted TS") +
  theme_minimal()
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 360 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



The original scale forecasts are: 695.1988914, 720.4995621, 687.38901, 622.2748613, 600.5097362, 704.5926135, 687.6465817, 654.1976348, 627.7188076, 615.9107471, 631.9977453, 629.5250317

(b) For the same training data, fit the "best" SARIMA model (allowing for the differencing and seasonal differencing); Use this SARIMA model to forecast 12 steps ahead; Translate this forecast into the 12-step-ahead predictions of the "original scale"; On the time plot of unemp, overlap original scale of fitted values and predictions with different colors.

```r
sarima_fit_2b <- auto.arima(training_data, max.p = 5, max.q = 5, max.d = 2,
                        start.p = 0, start.q = 0,
                        max.P = 5, max.Q = 5, max.D = 2,
                        start.P = 0, start.Q = 0,
                        allowdrift = FALSE, ic = "aic")

summary(sarima_fit_2b)
```

```
## Series: training_data
## ARIMA(2,0,2)(5,1,1)[12]
##
## Coefficients:
##          ar1      ar2      ma1     ma2     sar1     sar2     sar3     sar4
##       1.8136  -0.8297  -0.8006  0.1326  -0.9878  -0.6648  -0.4387  -0.2019
## s.e.  0.0670   0.0648   0.0842  0.0586   0.6669   0.4424   0.3013   0.1988
##          sar5    sma1
##       -0.0705  0.3292
## s.e.   0.0790  0.6674
##
## sigma^2 = 0.003691:  log likelihood = 481.08
## AIC=-940.16   AICc=-939.37   BIC=-897.79
##
## Training set error measures:
##                       ME       RMSE        MAE        MPE      MAPE    MASE
## Training set 0.00424329 0.05886964 0.04492058 0.06614545 0.7719482 0.22612
##                       ACF1
## Training set -0.001189095
```

```r
h <- 12
unemp.forecast <- forecast(sarima_fit_2b, h)
unemp.forecast$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1978 6.531323 6.531456 6.460234 6.344919 6.275515 6.468446 6.421205 6.371383
##           Sep      Oct      Nov      Dec
## 1978 6.351978 6.315707 6.357131 6.343060
```

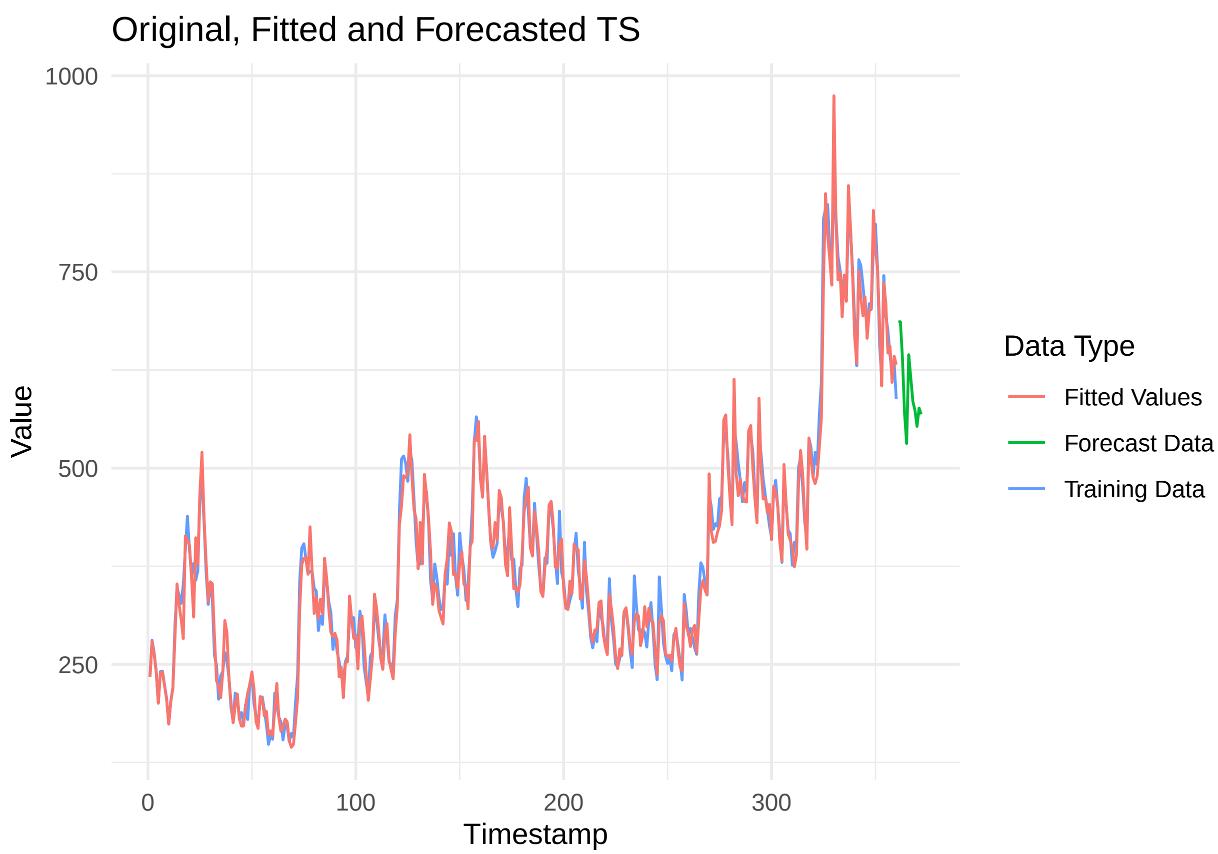
```r
original_series_forecast = exp(unemp.forecast$mean)
msfe_2b<-mean((original_series_forecast - as.numeric(test_data))^2)
msfe_2b
```

```
## [1] 149.4465
```

```r
#Plotting

library(ggplot2)
```

```r
library(tidyr)

df1 <- data.frame(train_compare = exp(training_data), timestamp = seq(1,360,1))
df2 <- data.frame(forecast_data = original_series_forecast, timestamp = seq(361,372,1))
df3 <- data.frame(fitted_compare = exp(sarima_fit_2b$fitted), timestamp = seq(1,360,1))
merged_df <- merge(df1, df2, by = "timestamp", all = TRUE)
merged_df <- merge(merged_df, df3, by = "timestamp", all = TRUE)

ggplot(merged_df, aes(x = timestamp)) +
  geom_line(aes(y = train_compare, color = "Training Data")) +
  geom_line(aes(y = forecast_data, color = "Forecast Data")) +
  geom_line(aes(y = fitted_compare, color = "Fitted Values")) +
  labs(x = "Timestamp", y = "Value", color = "Data Type") +
  ggtitle("Original, Fitted and Forecasted TS") +
  theme_minimal()
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).

## Warning: Removed 360 rows containing missing values or values outside the scale range
## (`geom_line()`).

## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



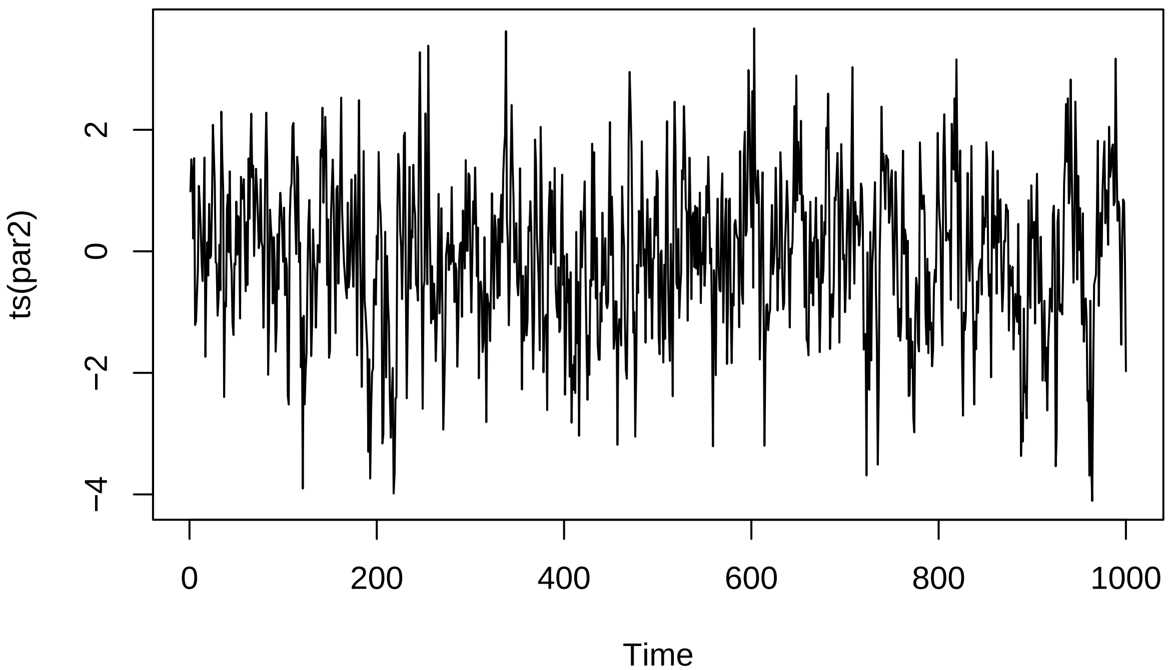
(c) Compare the predicted values in (a) and (b) with test data by computing the mean squared prediction errors; Which of the approaches, (a) or (b), gives a smaller prediction error?

Approach (b) has a lower mean squared forecast error of 149.4465474 than approach (a) with a MSFE of 2973.5715053. We can see from the fitted values compassion to the original series that the approach (a) fits the trend reasonably but misses some of the seasonality and spikes from the unemployment data, which are better approximated by the SARIMA model.

Problem 3

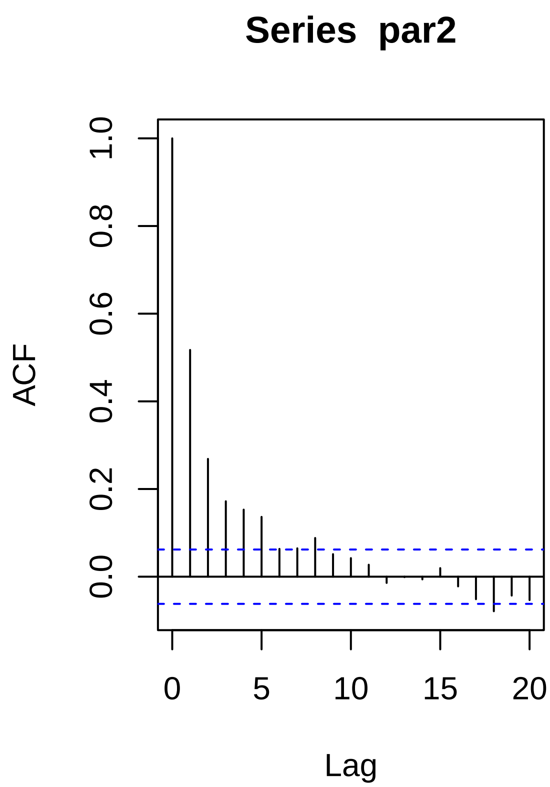
(a) Write down the exact form of the PAR(2) model that the series is generated from. That is, specify the parameters $\phi_{j,k}$, _{Z,k}^2 and m_k for $j = 1, 2$ and $k = 1, ..., 24$

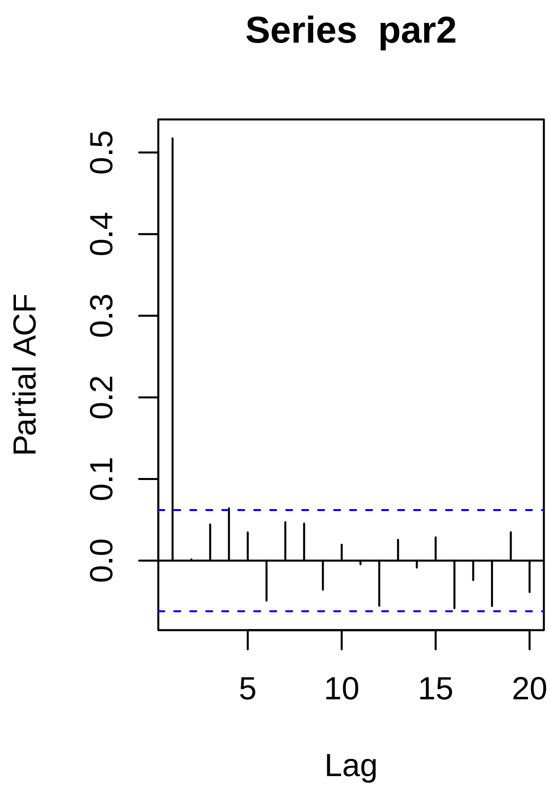
```
set.seed(1)
s <- 24
TT <- 1000
p <- 2
a <- matrix(0,s,p)
a[1,1] <- 0.5
a[2,2] <- 0.4
phia <- ab2phth(a)
phi0 <- phia$phi
phi0 <- as.matrix(phi0)
del0 <- matrix(1,s,1)
PAR2 <- makepar(TT,phi0,del0)
par2 <- PAR2$y
plot(ts(par2))
```



(b) Produce the sample ACF and PACF plots of the generated series. Treating the series as zero mean stationary, which zero mean stationary AR model is suggested by the auto.arima function?

```
par(mfrow=c(1,2))
acf(par2,lag.max=20)
pacf(par2,lag.max=20)
```






```r
AR_model <- auto.arima(par2, max.p = 5, max.q = 5, max.d = 2,
                        start.p = 0, start.q = 0,
                        max.P = 5, max.Q = 5, max.D = 2,
                        start.P = 0, start.Q = 0,
                        allowmean = FALSE, stationary = "TRUE",
                        ic = "aic")

summary(AR_model)
```

```
## Series: par2
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.5208
## s.e.  0.0270
##
## sigma^2 = 1.211:  log likelihood = -1514.15
## AIC=3032.31   AICc=3032.32   BIC=3042.12
##
## Training set error measures:
##                       ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -0.04550431 1.099722 0.8697959 98.97653 213.6743 0.8597051
##                      ACF1
## Training set -0.002929964
```

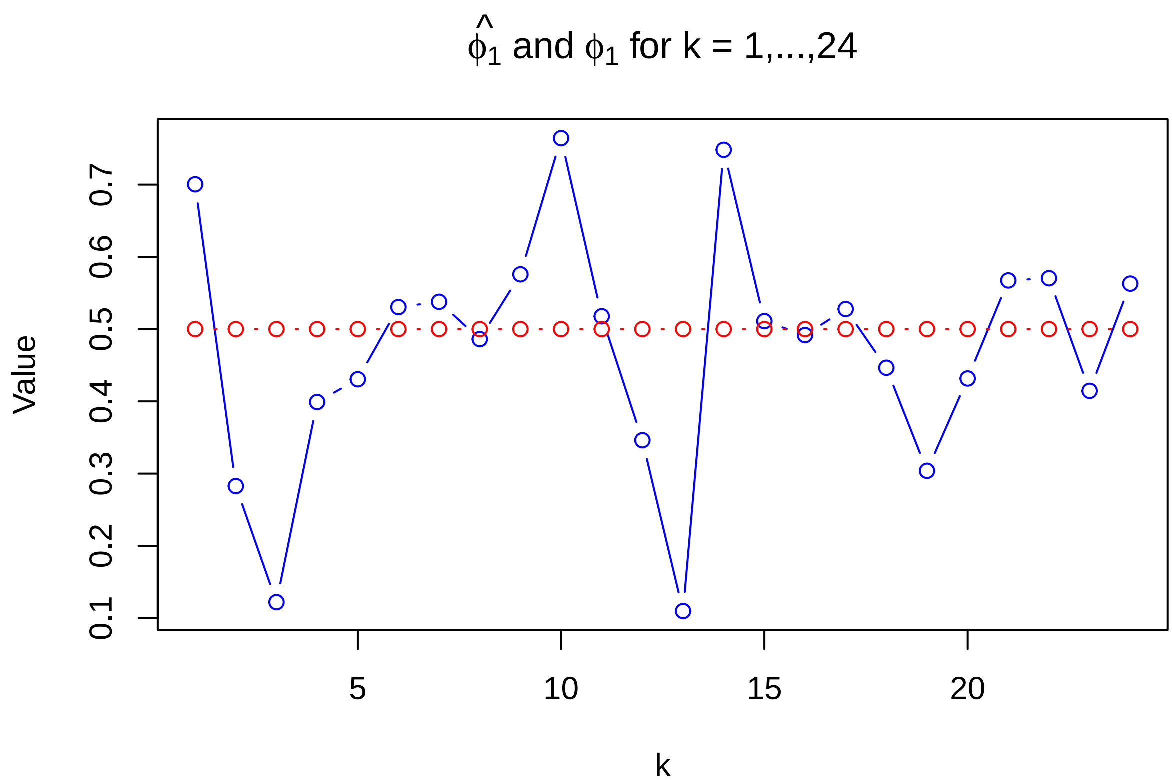
```r
phi_1 = AR_model$coef[1]
```

The model suggested is an AR(1):

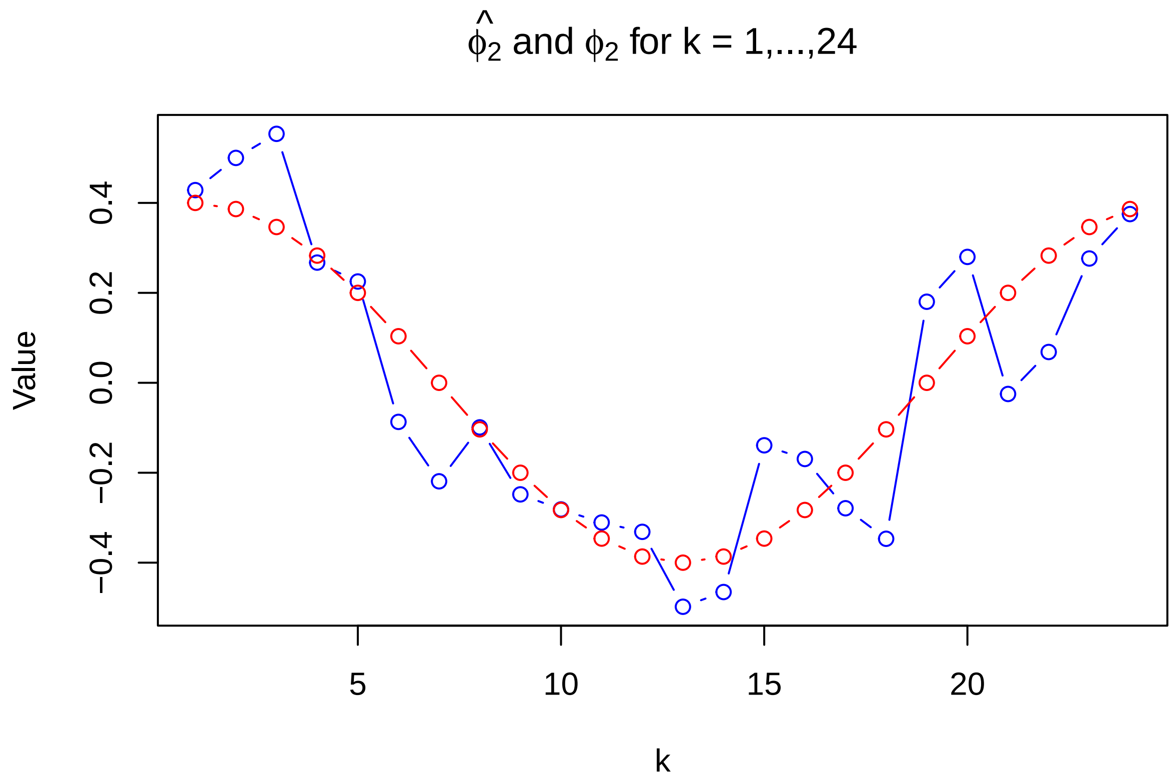
$$X_t = 0.5208X_{t-1} + Z_t \text{ with } \{Z_t : t \in \mathbb{Z}\} \sim WN(0, \sigma_z^2)$$

(c) Use the command perYW(par,24,2,NaN) to fit the PAR(2) model for the generated series; Produce two plots: One plot showing $\widehat{\phi_{1,k}}$ and $\phi_{1,k}$ for $k = 1, \ldots, 24$ and the other showing $\widehat{\phi_{2,k}}$ and $\phi_{2,k}$

```r
out.par<-perARMA::perYW(par2,24,2,NaN)

plot.new()

plot(out.par$phi[,1], type = 'b', col = 'blue', xlab = 'k', ylab = 'Value', main = expression(hat(phi[1
lines(phi0[,1], col = 'red', type = 'b')
```




```r
plot(out.par$phi[,2], type = 'b', col = 'blue', xlab = 'k', ylab = 'Value', main = expression(hat(phi[2
lines(phi0[,2], col = 'red', type = 'b')
```



Problem 4.

Consider bank calls from the R package fpp3 in Problem 4 (b) in Homework 2. We will use the aggregated time series as in the previous homework. The code hw5student.R for producing the series can be found in Homework folder on Canvas. Do the following.

```
rm(list=ls())
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.3.2

## -- Attaching packages ---------------------------------------------- fpp3 0.5 --

## v tibble      3.2.1     v tsibbledata 0.4.1
## v dplyr       1.1.4     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v tsibble     1.1.4     v fabletools  0.3.4

## Warning: package 'tibble' was built under R version 4.3.2

## Warning: package 'dplyr' was built under R version 4.3.2

## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## Warning: package 'tsibble' was built under R version 4.3.2
```

```
## Warning: package 'tsibbledata' was built under R version 4.3.2
```

```
## Warning: package 'feasts' was built under R version 4.3.2
```

```
## Warning: package 'fabletools' was built under R version 4.3.2
```

```
## Warning: package 'fable' was built under R version 4.3.2
```

```
## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```r
library(perARMA)
library(partsm)

data(bank_calls)

# bank_calls$DateTime[1:24]
# bank_calls$DateTime[((7-1)*24+1):(7*24)]
# bank_calls$DateTime[c(7*24+1,7*24+2,7*24+3)]

TT <- length(bank_calls$Calls)
floor(TT/169)
```

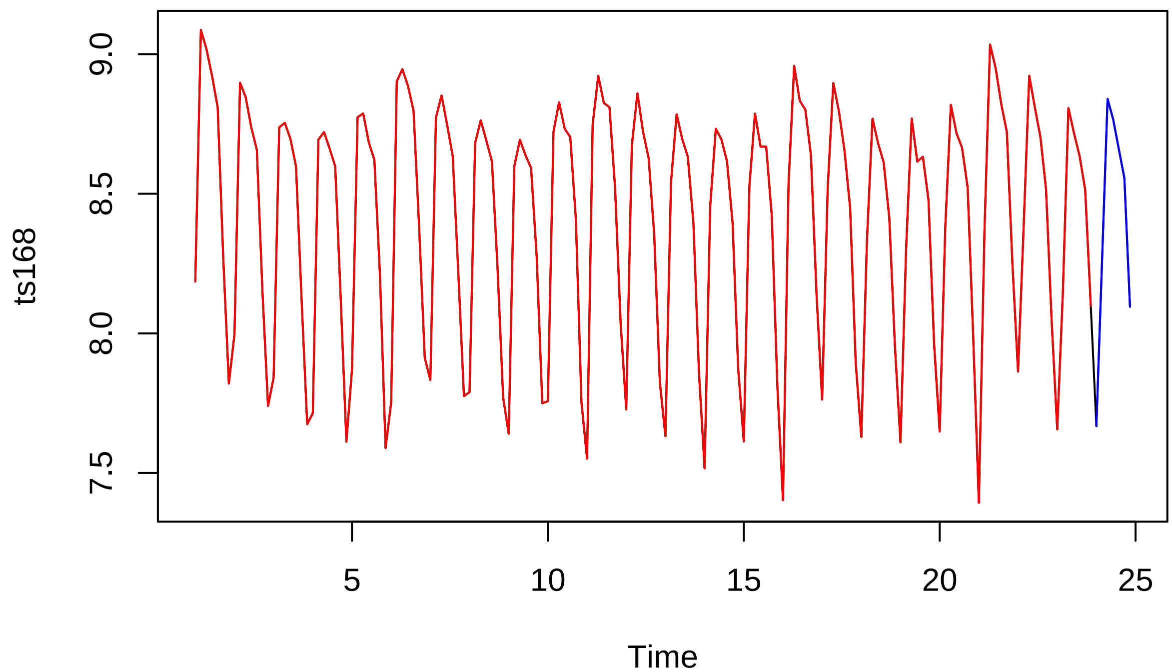
```
## [1] 164
```

```r
floor(TT/169)*169 - TT
```

```
## [1] 0
```

```r
ts <- vector("numeric",169)
for (k in 1:169){

  if (k %% 7 == 0){
    ts[k] <- sum(bank_calls$Calls[((k-1)*24+1):(k*24+1)])
  }else{
    ts[k] <- sum(bank_calls$Calls[((k-1)*24+1):(k*24)])
  }
}

# 168 = 24*7
ts168 <- log(ts[1:168])
ts168 <- ts(ts168,start=c(1,1),end=c(24,7),frequency=7)
plot.ts(ts168)
```

```
# Hold out the last 7 observations
ts168_train <- ts(ts168[-c(162:168)],start=c(1,1),end=c(23,7),frequency=7)
ts168_test <- ts(ts168[c(162:168)],start=c(24,1),end=c(24,7),frequency=7)

plot.ts(ts168)
lines(ts168_train,col="red")
lines(ts168_test,col="blue")
```




```
ts168_test_demean <- ts168_test - mean(ts168_test)
```

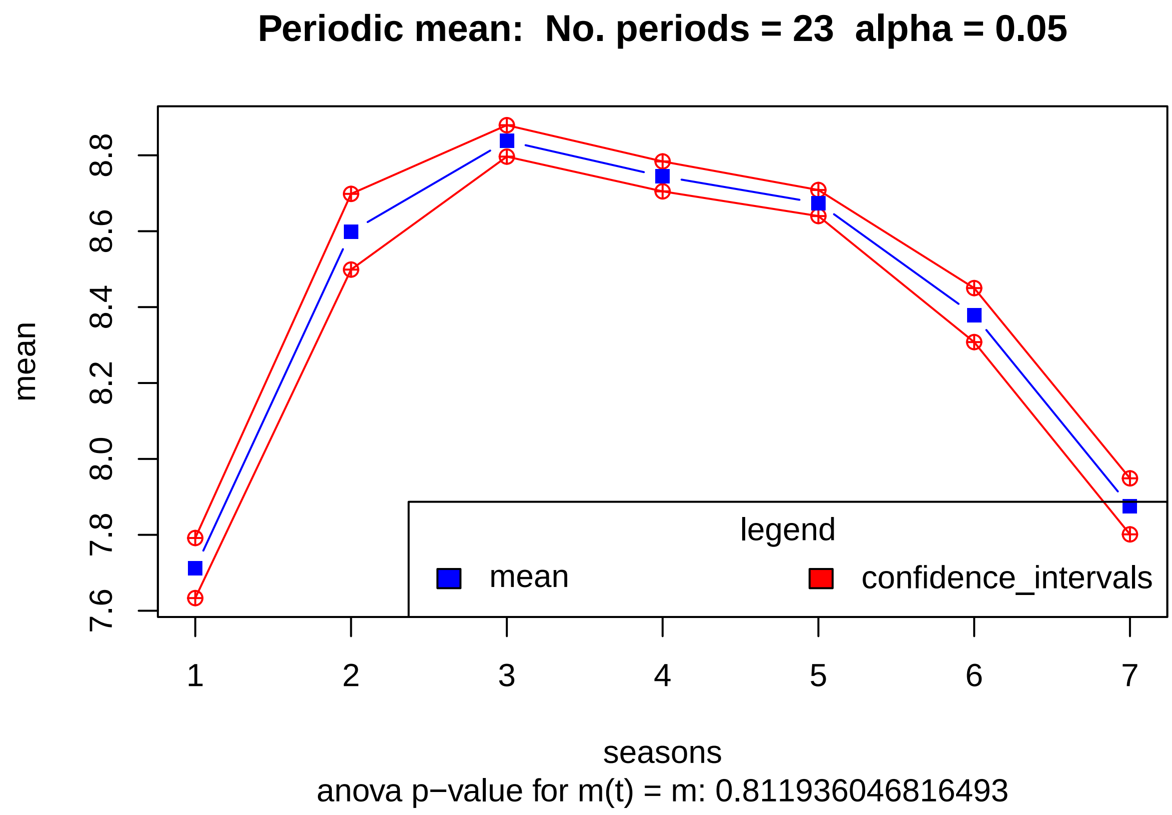
(a) For ts168 train, calculate and remove weekly means; Then calculate and remove seasonal/periodic means; Denote the series that the weekly means and seasonal/periodic means are removed by ts168_train3. Produce a time plot and sample ACF and PACF plots of the series.

```
library(fpp3)

#Weekly means
weekly_mean<-c(mean(ts168_train[1:35]),diff(cumsum(ts168_train)[seq(35,140,35)]/35),mean(ts168_train[14

#Demean the series
ts168_train2 <- ts168_train - head(rep(weekly_mean,each=35),161)
```

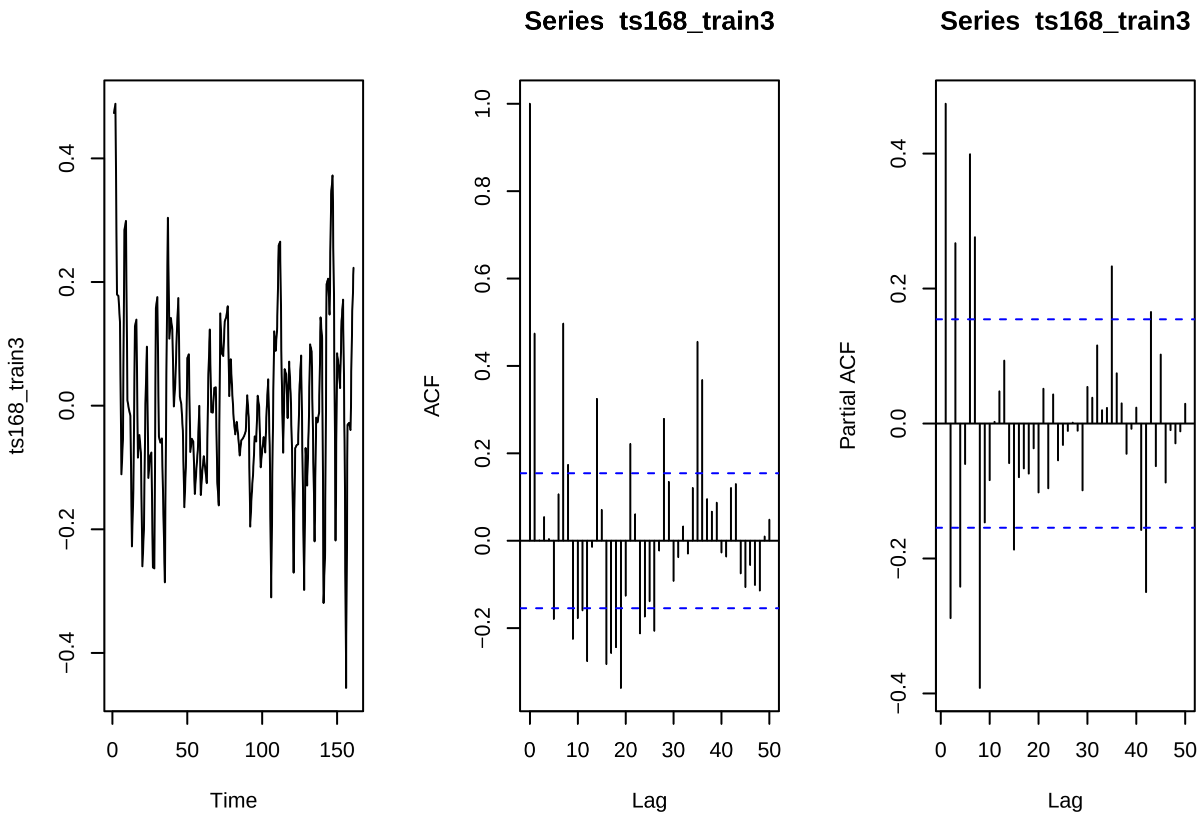
```r
#calculating and removing seasonal/periodic means
ts168_train<-as.numeric(ts168_train)
ts168_train_perm <-permest(t(ts168_train),T_t=7,alpha=0.05,missval=NaN,datastr='ts168_train')
```

```
## found  23  periods of length  7  with remainder of  0
```

```
## 2323232323232300000007.712519775811318.598584818172078.837874066809758.744379440269878.674223404899
```




```r
ts168_train3<-ts168_train-rep(ts168_train_perm$pmean,23) #161/7 = 23


par(mfrow=c(1,3))
plot.ts(ts168_train3)
acf(ts168_train3,lag.max=50)
pacf(ts168_train3,lag.max=50)
```



(b) For ts168_train3. fit PAR(p) with season s = 7 by using fit.ar.par. The lag of the AR coefficients can be chosen by aic or bic; Also fit SARMA(p,0)×(P,0)[7] by using auto.arima with proper input;

```r
library(partsm)

detcomp<-list(regular=c(0,0,0),seasonal=c(1,0),regvar=0)

#lag section:
aic<-bic<-Fnextp<-Fpval<-rep(NA,10)
for(p in 1:10){

 lmpar<-fit.ar.par(wts=ts168_train3,type="PAR",detcomp=detcomp,p=p)
 aic[p]<-AIC(lmpar@lm.par,k=2)
 bic[p]<-BIC(lmpar@lm.par)

 #H_0:phi_{p+1,k}=0,k=1,...,s
 Fout<-Fnextp.test(wts=ts168_train3,detcomp=detcomp,p=p,type="PAR")
 Fnextp[p] <-Fout@Fstat
 Fpval[p]<-Fout@pval
}
which.min(aic)
```

```
## [1] 8
```

```r
which.min(bic)
```

```
## [1] 8
```

```r
p_par <- 8 # selected lag order
out.par<-fit.ar.par(wts=ts168_train3,type="PAR",detcomp=detcomp,p=p_par)
summary(out.par)
```

```
## ----
##   PAR model of order 8 .
##
##   y_t = alpha_{1,s}*y_{t-1} + alpha_{2,s}*y_{t-2} + ... + alpha_{p,s}*y_{t-p} + coeffs*detcomp + eps
## ----
##   Autoregressive coefficients.
##
##             s=1
## alpha_1s  0.75
## alpha_2s -0.23
## alpha_3s  0.07
## alpha_4s -0.05
## alpha_5s -0.09
## alpha_6s -0.07
## alpha_7s  0.71
## alpha_8s -0.52
##
##
## Call:
## lm(formula = MLag[, 1] ~ 0 + Yperlag + MDT)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.290271 -0.042360 -0.002879  0.036854  0.237587
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## Yperlag1  0.748786   0.071371  10.491  < 2e-16 ***
## Yperlag2 -0.228781   0.078840  -2.902  0.00429 **
## Yperlag3  0.070143   0.081076   0.865  0.38840
## Yperlag4 -0.049536   0.081451  -0.608  0.54404
## Yperlag5 -0.093583   0.083372  -1.122  0.26353
## Yperlag6 -0.074931   0.088264  -0.849  0.39732
## Yperlag7  0.707532   0.084881   8.336 5.60e-14 ***
## Yperlag8 -0.518645   0.072336  -7.170 3.62e-11 ***
## MDT      -0.005070   0.006751  -0.751  0.45384
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08261 on 144 degrees of freedom
##   (8 observations deleted due to missingness)
## Multiple R-squared:  0.6571, Adjusted R-squared:  0.6356
## F-statistic: 30.66 on 9 and 144 DF,  p-value: < 2.2e-16
```

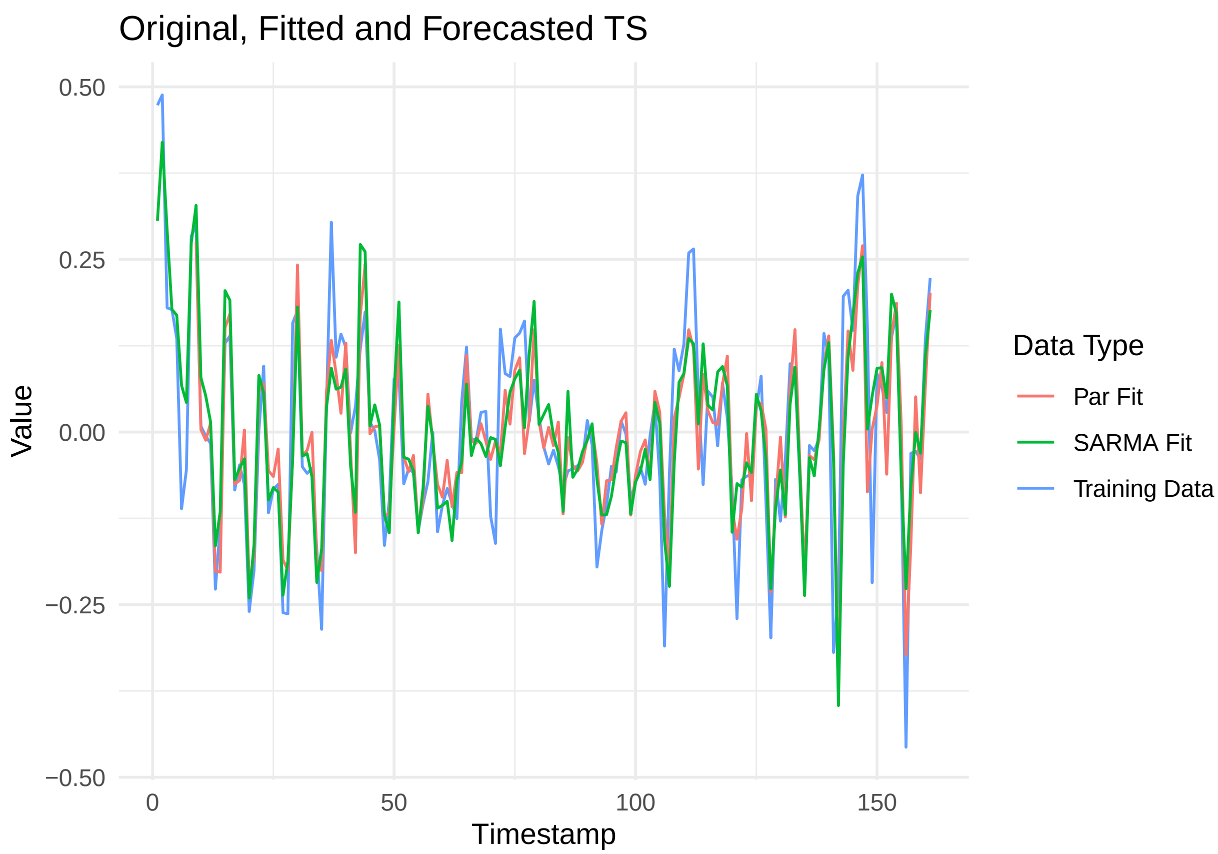
```r
# SARMA Model
# seasonality 7
ts168_train3 <- ts(ts168_train3, frequency=7)
# only allow p and P lags
SARMA_q4 <- auto.arima(ts168_train3, max.p = 10, max.q = 0, max.d = 0,
                        start.p = 0, start.q = 0,
                        max.P = 10, max.Q = 0, max.D = 0,
                        start.P = 0, start.Q = 0,
                        allowdrift = FALSE, ic = "aic")
summary(SARMA_q4)
```

```
## Series: ts168_train3
## ARIMA(1,0,0)(2,0,0)[7] with zero mean
##
## Coefficients:
##          ar1    sar1    sar2
##       0.7191  0.5671  0.3037
## s.e.  0.0556  0.0743  0.0786
##
## sigma^2 = 0.007292:  log likelihood = 164.29
## AIC=-320.57   AICc=-320.32   BIC=-308.25
##
## Training set error measures:
##                        ME       RMSE        MAE       MPE     MAPE      MASE
## Training set -0.003603748 0.08459609 0.06026708 -62.83334 170.4757 0.5781909
##                    ACF1
## Training set 0.05358782
```

(c) Draw a time series plot of ts168 train3 and overlap the fitted values of PAR and SAR models with different colors; Compute the mean squared error of the residuals from both models.

```r
par.mod<-slot(out.par,"lm.par")

df1 <- data.frame(train_compare = ts168_train3, timestamp = seq(1,161,1))
df2 <- data.frame(par_fitted = par.mod$fitted.values, timestamp = seq(9,161,1))
df3 <- data.frame(sarma_fitted = SARMA_q4$fitted, timestamp = seq(1,161,1))
merged_df <- merge(df1, df2, by = "timestamp", all = TRUE)
merged_df <- merge(merged_df, df3, by = "timestamp", all = TRUE)

ggplot(merged_df, aes(x = timestamp)) +
  geom_line(aes(y = train_compare, color = "Training Data")) +
  geom_line(aes(y = par_fitted, color = "Par Fit")) +
  geom_line(aes(y = sarma_fitted, color = "SARMA Fit")) +
  labs(x = "Timestamp", y = "Value", color = "Data Type") +
  ggtitle("Original, Fitted and Forecasted TS") +
  theme_minimal()
```

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```r
# Mean squared Residuals

residuals_par <- par.mod$residuals
residuals_mse_par <- mean((residuals_par)^2)
residuals_mse_par
```

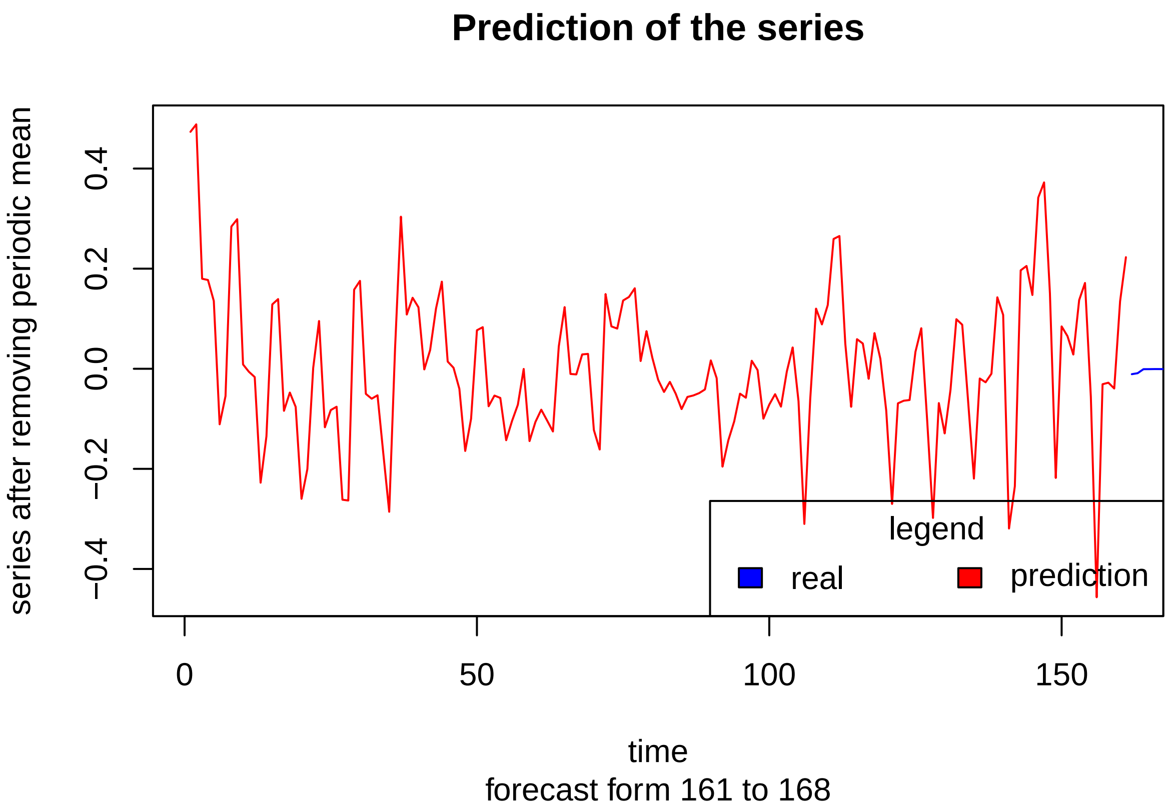
```
## [1] 0.006423559
```

```r
residuals_sarma <- SARMA_q4$residuals
residuals_mse_sarma <- mean((residuals_sarma)^2)
residuals_mse_sarma
```

```
## [1] 0.007156499
```

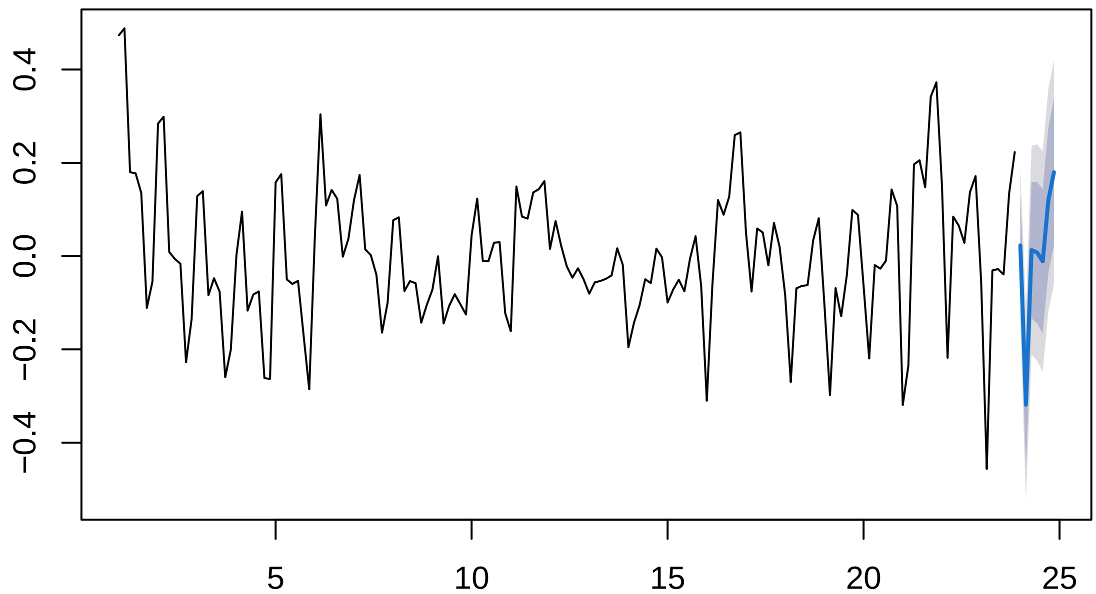
(d) Produce 7-step-ahead forecasts by using predictperYW for the PAR model and forecast for the SAR model; For the series ts168 test demean given in the code h5p4student.R, compute mean squared forecast errors with seasonal/periodic mean + forecast from the PAR model and seasonal/periodic mean + SAR model; Conclude which time series model, SAR or PAR, has the smaller forecast error.

```r
par.pred<-predictperYW(ts168_train_perm$xd,7,1,NaN,(161+7))$new
```




```r
h <- 7
SARMA_q4.forecast <- forecast(SARMA_q4, h)
plot(SARMA_q4.forecast)
```

Forecasts from ARIMA(1,0,0)(2,0,0)[7] with zero mean




```r
#par_demean_forecast = (ts168_train_perm$pmean + as.numeric(par.pred))
MSFE_par_demean = mean((as.numeric(par.pred) - ts168_test_demean)^2)
MSFE_par_demean
```

```
## [1] 0.1505339
```

```r
#sarma_demean_forecast = (ts168_train_perm$pmean + as.numeric(SARMA_q4.forecast$mean))
MSFE_sarma_demean = mean((as.numeric(SARMA_q4.forecast$mean) - ts168_test_demean)^2)
MSFE_sarma_demean
```

```
## [1] 0.1743746
```

Conclusion:

PAR model lower mean squared forecast error of 0.1505339 than the SARMA model with a MSFE of 0.1743746. In question 4.c., we saw how the mean squared residuals of the PAR model are lower 0.0064236 than those of the SARMA model 0.0071565. In the plot, we can observe how both models track the series well, but the PAR model out-performs SARMA in periods of high volatility (spikes).