Time Series Analysis - Lab 02 (Group 7)

Anubhav Dikshit (anudi287) and Maximilian Pfundstein (maxpf364)

2019-09-19

Contents

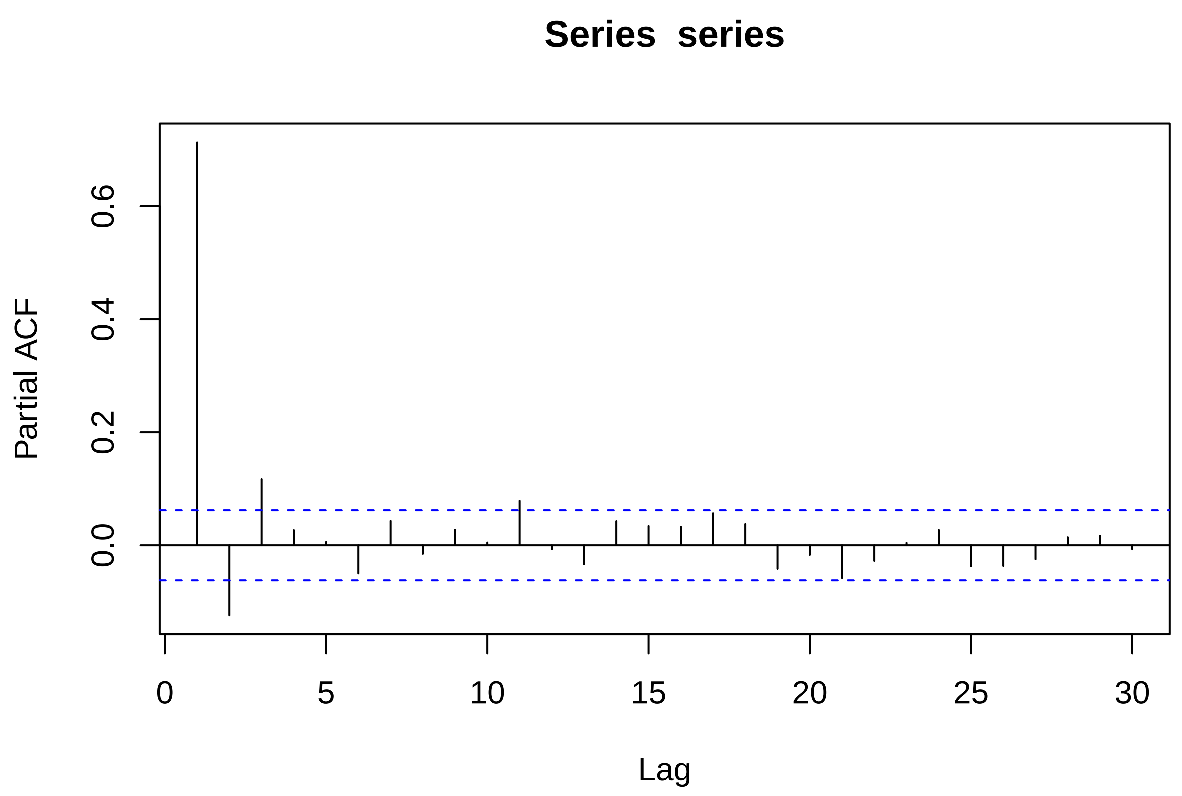
1 Assignment 1: Computations with simulated data

1.1 Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation

Task: Generate 1000 observations from AR(3) process with $\phi_1 = 0.8, \phi_2 = -0.2, \phi_3 = 0.1$. Use these data and the definition of PACF to compute ϕ_{33} from the sample, i.e. write your own code that performs linear regressions on necessarily lagged variables and then computes an appropriate correlation. Compare the result with the output of function `pacf()` and with the theoretical value of ϕ_{33}.

Answer: First the sampling and looking at the built-in PACF.

```r
model = list(ar = c(0.8, -0.2, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 1000)
pacf(series)
print(pacf(series))
```



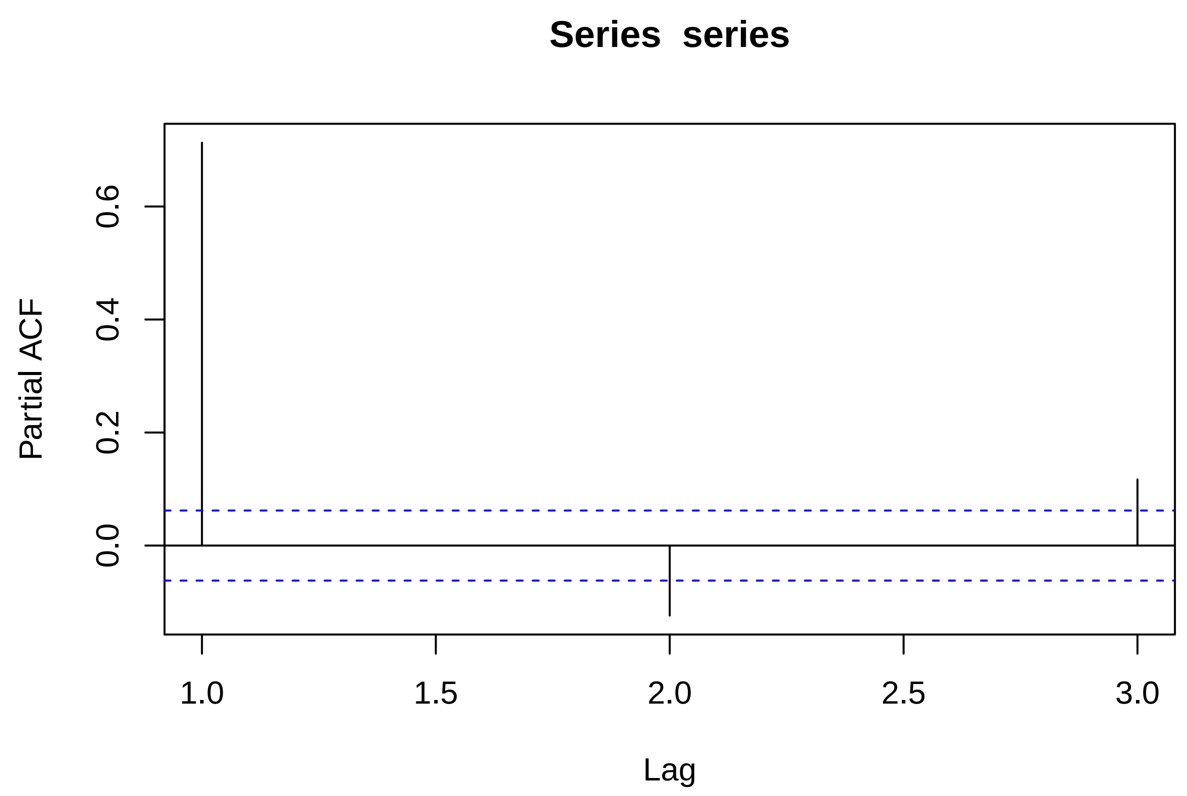
```
##
## Partial autocorrelations of series 'series', by lag
##
##      1      2      3      4      5      6      7      8      9     10
##  0.713 -0.124  0.117  0.027  0.006 -0.050  0.043 -0.015  0.027  0.005
##     11     12     13     14     15     16     17     18     19     20
##  0.079 -0.007 -0.033  0.043  0.034  0.033  0.057  0.038 -0.042 -0.017
##     21     22     23     24     25     26     27     28     29     30
## -0.058 -0.027  0.004  0.027 -0.037 -0.036 -0.025  0.014  0.017 -0.007
```

Now we do it on our own.

```
pacf_ar = function(series., lag.max = 30) {

  covariances = vector(length=lag.max)
  series = as.vector(series)

  for (lag in 1:lag.max) {

    # Create a dataframe with the lagged variables and omit NAs
    df = data.frame(y = series)
    df_colnames = c("y")

    if (lag == 1) {
      df = na.omit(cbind(df, lag(series, lag)))
      covariances[1] = cor(df[,1], df[,2])
      next
    }

    for (t in 1:(lag-1)) {
      df_colnames = c(df_colnames, paste("t_", t, sep=""))
      df = cbind(df, lag(series, t))
    }

    #df = na.omit(df)
    df = df[(1+lag):nrow(df),]
    colnames(df) = df_colnames

    # Second df
    df2 = data.frame(y = series)
    df2_colnames = c("y")

    for (t in 1:(lag-1)) {
      df2_colnames = c(df2_colnames, paste("t+", t, sep=""))
      df2 = cbind(df2, lead(series, t))
    }

    #df2 = na.omit(df2)
    df2 = df2[1:(nrow(df2)-lag),]
    colnames(df2) = df2_colnames

    # Performing LinReg
    x_t_dash  = lm(y ~ ., df)$residual
    x_t_dash_dash  = lm(y ~ ., df2)$residual

    covariances[lag] = cor(x_t_dash, x_t_dash_dash)
  }
  return(covariances)
}

pacf_ar(series, 3)
```

```
## [1]  0.7139526 -0.1250914  0.1146076
```

```
print(pacf(series, lag.max = 3))
```



```
##
## Partial autocorrelations of series 'series', by lag
##
##      1      2      3
##  0.713 -0.124  0.117
```

```
ARMAacf(model$ar, lag.max = 3, pacf = TRUE)
```

```
## [1]  0.7027027 -0.1212121  0.1000000
```

1.2 Methods of Moments, Conditional Least Squares and Maximum Likelihood

Task: Simulate an AR(2) series with $\phi_1 = 0.8, \phi_2 = 0.1$ and $n = 100$. Compute the estimated parameters and their standand errors by using three methods: method of moments (Yule-Walker equations), conditional least squares and maximum likelihood (ML) and compare their results to the true values. Which method does seem to give the best result? Does theoretical value for ϕ_2 fall within confidence interval for ML estimate?

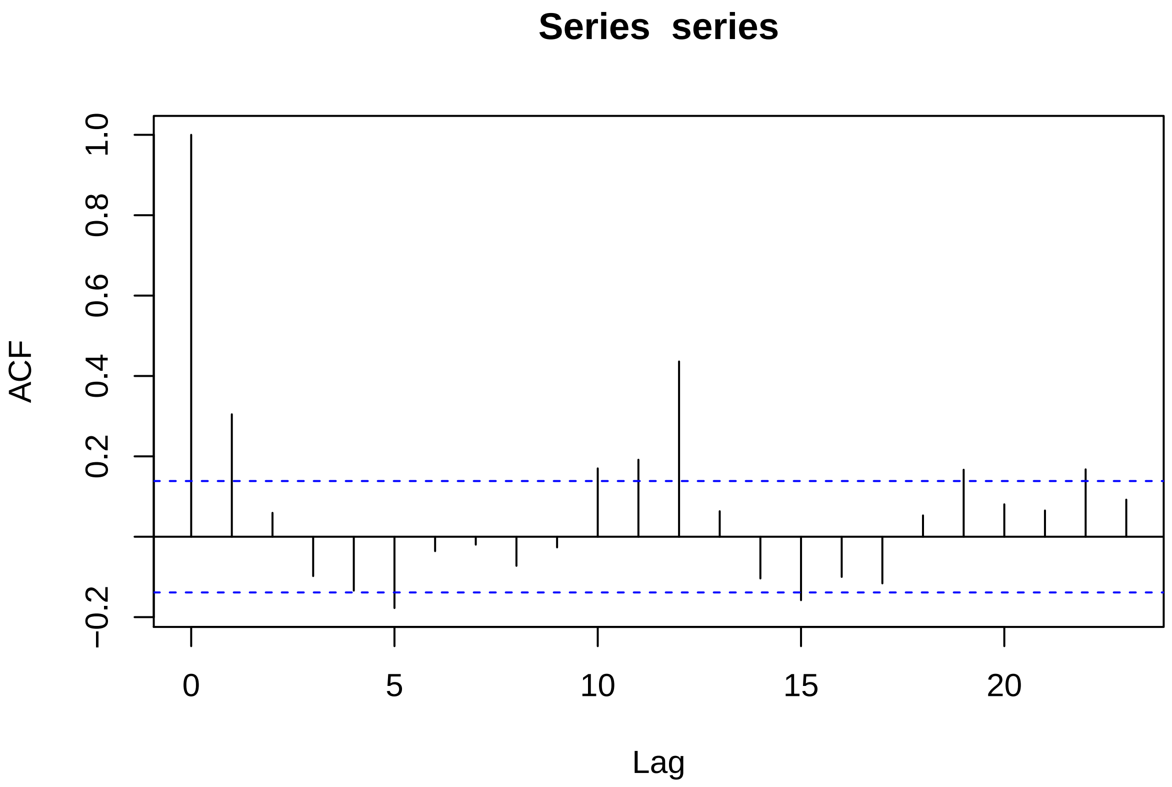
1.3 Sample and Theoretical ACF and PACF

Task: Generate 200 observations of a seasonal ARIMA$(0,0,1) \times (0,0,1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using `arima.sim()`. Plot sample ACF and PACF and also theoretical ACF and PACF. Which patterns can you see at the theoretical ACF and PACF? Are they repeated at the sample ACF and PACF?

Answer: TODO. Plotting of theoretical values and comparison.

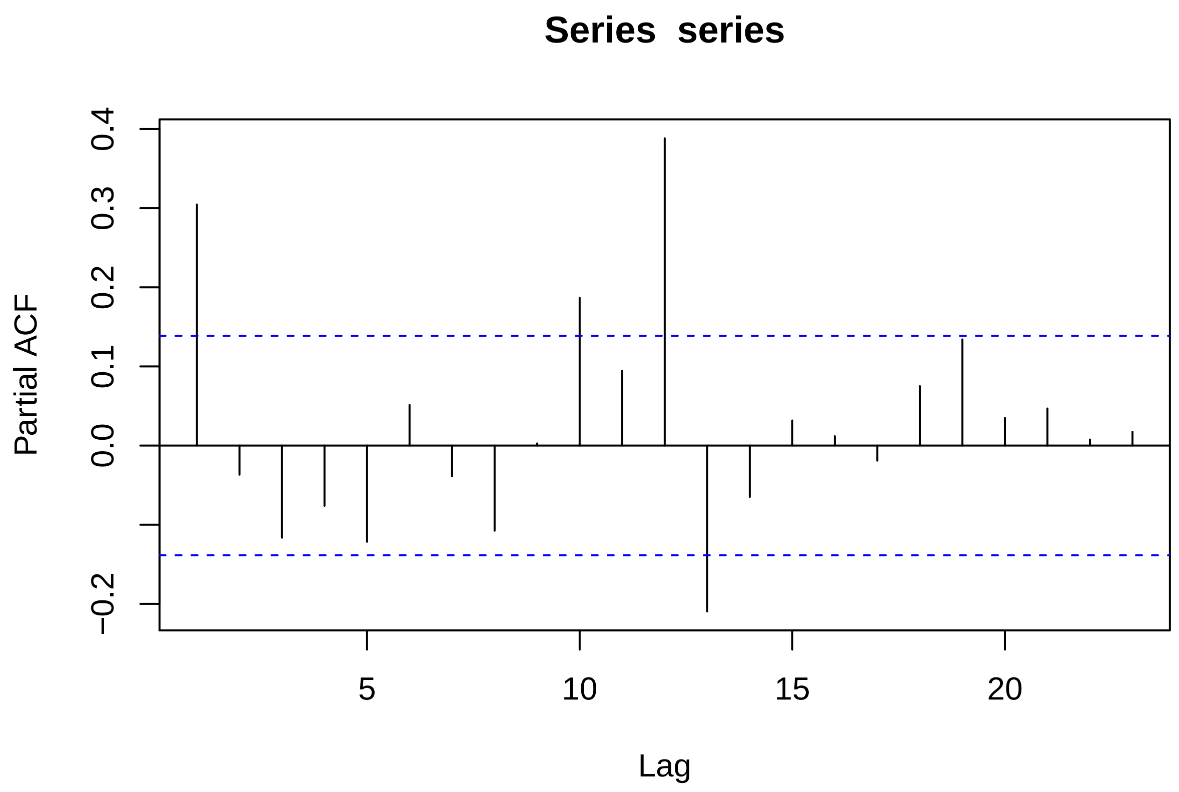
```
theta = 0.3
Theta = 0.6

model = list(ma = c(theta, rep(0, 10), Theta, theta*Theta))
```

```r
series = arima.sim(model, n=200)

print(acf(series))
```



```
##
## Autocorrelations of series 'series', by lag
##
##      0      1      2      3      4      5      6      7      8      9
##  1.000  0.305  0.059 -0.098 -0.134 -0.177 -0.036 -0.020 -0.072 -0.027
##     10     11     12     13     14     15     16     17     18     19
##  0.170  0.192  0.436  0.064 -0.104 -0.158 -0.100 -0.116  0.053  0.167
##     20     21     22     23
##  0.081  0.065  0.168  0.092
```

```r
print(pacf(series))
```



```
##
## Partial autocorrelations of series 'series', by lag
##
##      1      2      3      4      5      6      7      8      9     10
##  0.305 -0.037 -0.116 -0.076 -0.122  0.051 -0.039 -0.108  0.003  0.187
##     11     12     13     14     15     16     17     18     19     20
##  0.094  0.388 -0.210 -0.065  0.032  0.012 -0.019  0.075  0.134  0.035
##     21     22     23
##  0.047  0.008  0.017
```

```
ARMAacf(model$ma, lag.max = 3, pacf = TRUE)
```

```
## [1]  1.013001 11.238756 -1.083580
```

```
ARMAacf(model$ma, lag.max = 3, pacf = FALSE)
```

```
##         0         1         2         3
## 1.0000000 1.0130014 0.7320337 0.2386327
```

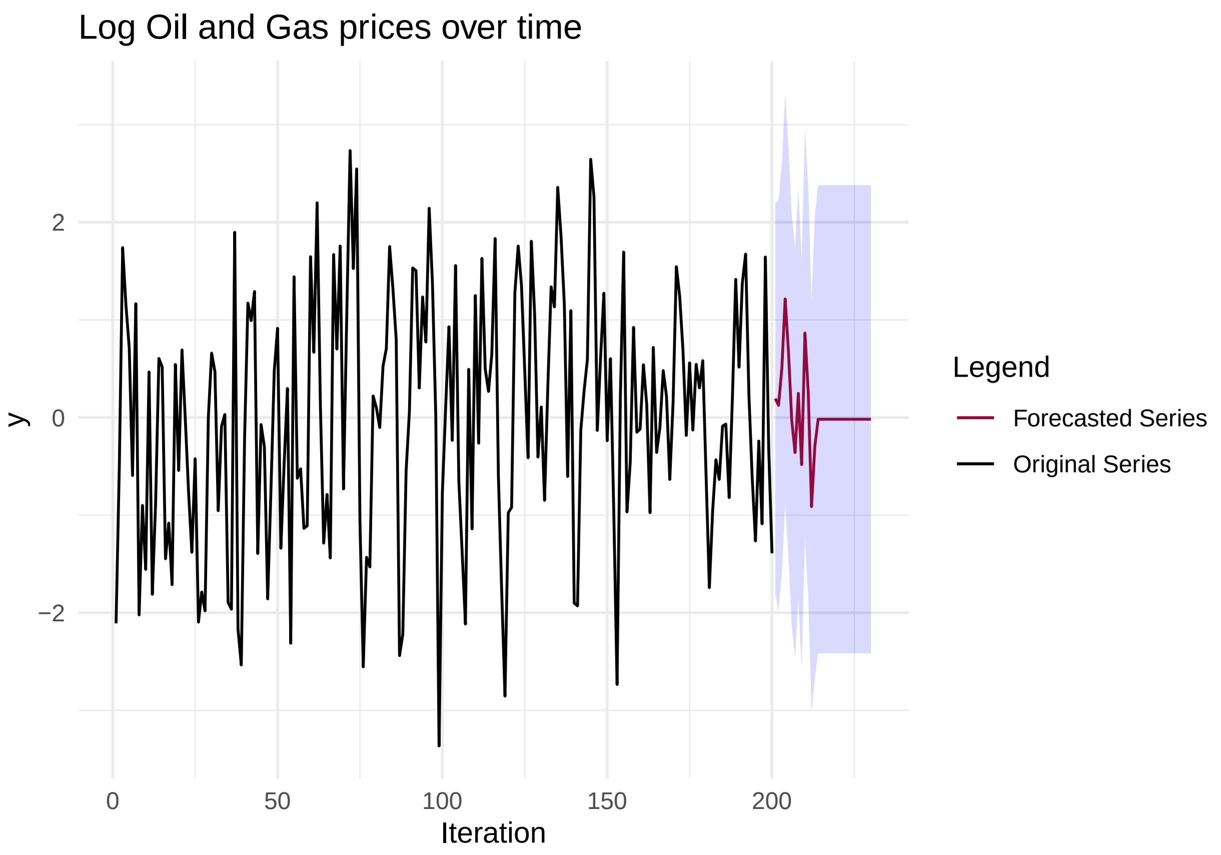
```
#acf(series)
#pacf(series)
```

1.4 Forecast and Predition

Task: Generate 200 observations of a seasonal ARIMA$(0,0,1) \times (0,0,1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using `arima.sim()`. Fit ARIMA$(0,0,1) \times (0,0,1)_{12}$ model to the data, compute forecasts and a prediction band 30 points ahead and plot the original data and the forecast with the prediction band. Fit the same data with function `gausspr()` from package `kernlab` (use default settings). Plot the original data and predicted data from $t = 1$ to $t = 230$. Compare the two plots and make conclusions.

```r
fitted_model = arima(series,
                     order = c(0, 0, 1),
                     seasonal = list(order= c(0, 0, 1), period = 12))

prediction = predict(fitted_model, n.ahead = 30)

df = data.frame(time = c(1:length(series)),
                data = series)

df2 = data.frame(time = c((length(series)+1):
                            (length(prediction$pred)+length(series))),
                 forecast = prediction$pred,
                 upper_boundary = prediction$pred + 1.96*prediction$se,
                 lower_boundary = prediction$pred - 1.96*prediction$se)

ggplot() +
  geom_ribbon(aes(x = df2$time,
                  ymin=df2$lower_boundary,
                  ymax=df2$upper_boundary),
                  fill = "#0000ff", alpha = 0.15) +
  geom_line(aes(x = df$time, y = df$data, colour = "Original Series")) +
  geom_line(aes(x = df2$time, y = df2$forecast, colour = "Forecasted Series")) +
  labs(title = "Log Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#900C3F", "#000000")) +
  theme_minimal()
```

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



1.5 Prediction Band

Task: Generate 50 observations from ARMA(1, 1) process with $\phi = 0.7$, $\theta = 0.50$. Use first 40 values to fit an ARMA(1,1) model with $\mu = 0$. Plot the data, the 95% prediction band and plot also the true 10 values that you initially dropped. How many of them are outside the prediction band? How can this be interpreted?

2 Assignment 2: ACF and PACF diagnostics

2.1 ARIMA Model Suggestion

Task: For data series `chicken` in package `astsa` (denote it by `x_t`) plot 4 following graphs up to 40 lags: ACF(x_t), PACF(x_t), ACF(∇x_t), PACF(∇x_t) (group them in one graph). Which ARIMA(p, d, q) or ARIMA$(p, d, q) \times (P, D, Q)_s$ models can be suggested based on this information only? Motivate your choice.

2.2 More Datasets

Task: Repeat step 1 for the following datasets: `so2`, `EQcount`, `HCT` in package `astsa`.

3 Assignment 3: ARIMA modeling cycle

In this assignment, you are assumed to apply a complete ARIMA modeling cycle starting from visualization and detrending and ending up with a forecasting.

3.1 Finding a Suitable ARIMA Model (oil)

Task: Find a suitable ARIMA(p, d, q) model for the data set `oil` present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, unit root test, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model. Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

3.2 Finding a Suitable ARIMA Model (unemp)

Task: Find a suitable ARIMA$(p, d, q) \times (P, D, Q)_s$ model for the data set unemp present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model (write in the backshift operator notation without expanding the brackets). Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

4 Source Code

```r
library(dplyr)
library(ggplot2)
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)

model = list(ar = c(0.8, -0.2, 0.1), ma = c())
set.seed(12345)
series = arima.sim(model = model, n = 1000)
pacf(series)
print(pacf(series))


pacf_ar = function(series., lag.max = 30) {

  covariances = vector(length=lag.max)
  series = as.vector(series)

  for (lag in 1:lag.max) {

    # Create a dataframe with the lagged variables and omit NAs
    df = data.frame(y = series)
    df_colnames = c("y")

    if (lag == 1) {
      df = na.omit(cbind(df, lag(series, lag)))
      covariances[1] = cor(df[,1], df[,2])
      next
    }

    for (t in 1:(lag-1)) {
      df_colnames = c(df_colnames, paste("t_", t, sep=""))
      df = cbind(df, lag(series, t))
    }

    #df = na.omit(df)
    df = df[(1+lag):nrow(df),]
    colnames(df) = df_colnames

    # Second df
    df2 = data.frame(y = series)
    df2_colnames = c("y")

    for (t in 1:(lag-1)) {
      df2_colnames = c(df2_colnames, paste("t+", t, sep=""))
      df2 = cbind(df2, lead(series, t))
    }

    #df2 = na.omit(df2)
    df2 = df2[1:(nrow(df2)-lag),]
    colnames(df2) = df2_colnames

    # Performing LinReg
    x_t_dash  = lm(y ~ ., df)$residual
    x_t_dash_dash  = lm(y ~ ., df2)$residual

    covariances[lag] = cor(x_t_dash, x_t_dash_dash)
  }
  return(covariances)
}

pacf_ar(series, 3)
print(pacf(series, lag.max = 3))
ARMAacf(model$ar, lag.max = 3, pacf = TRUE)


theta = 0.3
Theta = 0.6

model = list(ma = c(theta, rep(0, 10), Theta, theta*Theta))

series = arima.sim(model, n=200)

print(acf(series))
print(pacf(series))
ARMAacf(model$ma, lag.max = 3, pacf = TRUE)
ARMAacf(model$ma, lag.max = 3, pacf = FALSE)

#acf(series)
#pacf(series)
```

```r
fitted_model = arima(series,
                     order = c(0, 0, 1),
                     seasonal = list(order= c(0, 0, 1), period = 12))

prediction = predict(fitted_model, n.ahead = 30)

df = data.frame(time = c(1:length(series)),
                data = series)

df2 = data.frame(time = c((length(series)+1):
                            (length(prediction$pred)+length(series))),
                 forecast = prediction$pred,
                 upper_boundary = prediction$pred + 1.96*prediction$se,
                 lower_boundary = prediction$pred - 1.96*prediction$se)

ggplot() +
  geom_ribbon(aes(x = df2$time,
                  ymin=df2$lower_boundary,
                  ymax=df2$upper_boundary),
                  fill = "#0000ff", alpha = 0.15) +
  geom_line(aes(x = df$time, y = df$data, colour = "Original Series")) +
  geom_line(aes(x = df2$time, y = df2$forecast, colour = "Forecasted Series")) +
  labs(title = "Log Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#900C3F", "#000000")) +
  theme_minimal()
```