Advanced Time Series Analysis: ARIMA Modeling, Smoothing Techniques, and Spectral Density Comparison

Seoyeon Park

```r
# Load the libraries you might need
options(tinytex.verbose = TRUE)

library(astsa)
library(forecast)
```

ARIMA Model Fitting and Forecasting for Time Seriesfor Economic Turnover in Switzerland:

STEP 1- Model 1: Fit a model using `auto.arima` to the data X_t. Set `lambda` = "`auto`" to apply Box-Cox transformation automatically. we call this model **Model 1**.

```r
# 1. Read the data. define it as a time series. (Switzerland)
data = read.csv( "switzerland.csv")
data$Year <- as.numeric(substr(data$TIME, 1, 4))
data$Month <- as.numeric(substr(data$TIME, 6, 7))
X_t <- ts(data$Switzerland, start=data$Year[1], frequency=12)
#X_t #2000 Jan-Feb to 2019 Jan-Feb

# 2. Fit Model 1
Model1 <- auto.arima(X_t, lambda="auto")
summary(Model1)
```

```
## Series: X_t
## ARIMA(1,1,3)(0,1,1)[12]
## Box Cox transformation: lambda= 0.1132106
##
## Coefficients:
##           ar1     ma1      ma2     ma3     sma1
##       -0.9257  0.1552  -0.6390  0.1677  -0.3571
## s.e.   0.0422  0.0766   0.0607  0.0792   0.0703
##
## sigma^2 = 0.0006466:  log likelihood = 512.7
## AIC=-1013.41   AICc=-1013.02   BIC=-992.86
##
## Training set error measures:
##                       ME    RMSE      MAE         MPE     MAPE      MASE
## Training set -0.02813678 1.33436 1.025651 -0.04418564 1.132448 0.5814235
##                     ACF1
## Training set -0.02272477
```

- **Model 1** is fitted to **ARIMA(1,1,3)(0,1,1)[12]** including both non-seasonal and seasonal components.

  - (1+0.9257B)(1-B)(1-B^{12})X_t = (1-0.3571B^{12})(1+0.1552B-0.6390B^2+0.1677B^3)Z_t
    * For the non-seasonal component, the model is **ARMA(1,3)** with **differencing 1**.
    * For the seasonal component, the model has an MA part order of 1 and **differencing with one season which is 12**.

- **Box-Cox Transformation**: The lambda value is 0.1132106 applied to stabilize the variance
- **Error Measures**:
  - The various error measures indicate the fit of the model, with lower values generally indicating a better fit. The **AIC, AICc, and BIC** values help in model comparison, with lower values indicating a better model.
- **Training Errors**:
  - Training errors are provided which help to understand if the model works well in the training data.

STEP 2- Model 2: Fit a model using `auto.arima` to the data X_t. Set `lambda` = "`auto`" to apply Box-Cox transformation automatically and set `stationary` = `TRUE`. we call this model **Model 2**.

```
# 3. Fit Model 2
Model2 <- auto.arima(X_t, lambda = "auto", stationary = TRUE)
summary(Model2)
```

```
## Series: X_t
## ARIMA(0,0,1) with non-zero mean
## Box Cox transformation: lambda= 0.1132106
##
## Coefficients:
##          ma1    mean
##       0.8855  5.8670
## s.e.  0.0304  0.0193
##
## sigma^2 = 0.02552:  log likelihood = 99.89
## AIC=-193.78   AICc=-193.68   BIC=-183.34
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE     MASE
## Training set 0.3743156 8.882414 7.217688 -0.4961775 7.948855 4.091578
##                      ACF1
## Training set -0.03163915
```

- **Model 2** is fitted to **ARIMA(0,0,1)** with non-zero mean. X_t = 5.8670 + (1-0.8855B)Z_t
  - The model does not apply differencing (d=0, D=0) and assumes the data is stationary.
  - The model includes an MA part order of 1 (q=1) and does not include any seasonal components (P=Q=S=0).
- The lambda value is 0.1132 applied to stabilize the variance
- Similarly error measures and training errors are provided which is helpfult to evaluate and compare the models

Comparison of ARIMA Models

Analyzed the differences between two fitted ARIMA models by comparing the orders of p, d, q, P, D, Q and S . This comparison provided insights into the non-seasonal and seasonal components of the time series and helped determine the most suitable model for economic turnover data in Switzerland.

- **Model1 : ARIMA(1,1,3)(0,1,1)[12]**
  - Hence, p=1, d=1, q=3, and, P=0,D=1, Q=1, S=12
- **Model2: ARIMA(0,0,1) with non-zero mean**
  - Hence, p=0, d=0, q=1, and, P=D=Q=S=0

<Differences between two fitted model>

- **Differencing**:
  - **Model 1**: Applies both non-seasonal differencing (d=1) and seasonal differencing (D=1).
  - **Model 2**: Assumes the data is stationary and does not apply any differencing (d=0, D=0)
- **Seasonality**:
  - **Model 1**: Accounts for seasonality with a seasonal period of 12 months (S=12) and includes seasonal moving average terms (Q=1).
  - **Model 2**: Does not account for seasonality (P=Q=S=0).
- **Model Components**:
  - **Model 1**: Combines non-seasonal ARIMA(1,1,3) with seasonal ARIMA(0,1,1)[12] components.
  - **Model 2**: Fits a simple ARIMA(0,0,1) model with a non-zero mean, capturing only the short-term dependencies without differencing or seasonality.

Model Comparison and Selection:
Compared Model 1 and Model 2 using AIC and BIC criteria to evaluate model fit and complexity. Based on this comparison, selected the preferred model for forecasting Switzerland's economic turnover, justifying the choice based on lower error metrics and overall model performance.

Table 1: **Model 1** has significantly lower AIC, AICc, and BIC values compared to **Model 2**.

	Model1	Model2
AIC	-1013.41	-193.78
AICc	-1013.02	-193.68
BIC	-992.86	-183.34

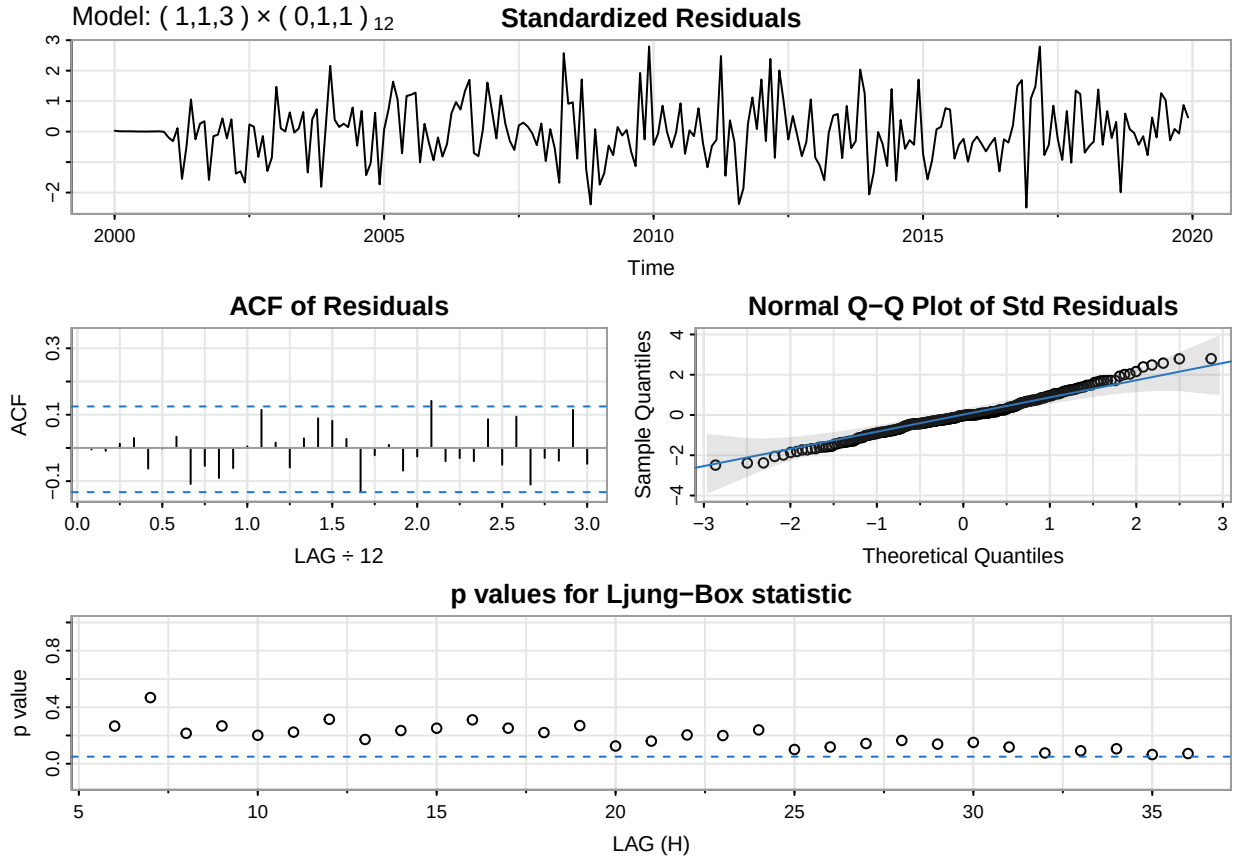
- This indicates that Model 1 provides a better fit to the data while accounting for the complexity of the model.
- Therefore, based on these criteria, **Model 1** is preferred over **Model 2**.

Residual Diagnostic Checking:
Performed diagnostic checks on the residuals of the fitted ARIMA models, using plots to confirm the choice of the preferred model. Evaluated key factors such as the autocorrelation of residuals and normality to ensure the model's validity, mentioning at least two points of confirmation.

```r
# Diagnostic Checking

# Fit Model 1 using sarima
sarima_model1 <- sarima(X_t, p=1, d=1, q=3, P=0, D=1, Q=1, S=12)
```

```
## initial  value 0.666294
## iter   2 value 0.394177
## iter   3 value 0.345981
## iter   4 value 0.342390
## iter   5 value 0.341134
## iter   6 value 0.341098
## iter   7 value 0.341096
## iter   8 value 0.341092
## iter   9 value 0.341081
## iter  10 value 0.341056
## iter  11 value 0.341009
## iter  12 value 0.340949
## iter  13 value 0.340939
## iter  14 value 0.340912
## iter  15 value 0.340909
## iter  16 value 0.340908
## iter  17 value 0.340908
## iter  18 value 0.340907
## iter  19 value 0.340905
## iter  20 value 0.340901
## iter  21 value 0.340897
## iter  22 value 0.340894
## iter  23 value 0.340894
## iter  24 value 0.340894
## iter  25 value 0.340894
## iter  25 value 0.340894
## iter  25 value 0.340894
## final  value 0.340894
## converged
## initial  value 0.343448
## iter   2 value 0.343403
## iter   3 value 0.343390
## iter   4 value 0.343379
## iter   5 value 0.343377
## iter   6 value 0.343375
## iter   7 value 0.343353
## iter   8 value 0.343328
## iter   9 value 0.343302
## iter  10 value 0.343291
## iter  11 value 0.343289
## iter  12 value 0.343289
## iter  12 value 0.343289
## final  value 0.343289
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##      Estimate     SE t.value p.value
```

```
## ar1   -0.2224 0.5426 -0.4098  0.6823
## ma1   -0.5868 0.5374 -1.0919  0.2761
## ma2   -0.0964 0.4404 -0.2188  0.8270
## ma3    0.1040 0.0935  1.1119  0.2674
## sma1  -0.3499 0.0676 -5.1752  0.0000
##
## sigma^2 estimated as 1.9665 on 222 degrees of freedom
##
## AIC = 3.577319  AICc = 3.578515  BIC = 3.667846
##
```




```r
# Print summary of Model 1
summary(sarima_model1)
```

```
##                   Length Class  Mode
## fit               14     Arima  list
## degrees_of_freedom  1     -none- numeric
## ttable             20     -none- numeric
## ICs                 3     -none- numeric
```

Standardized Residuals

- **Plot**: The standardized residuals are centered around zero within -3 and 3 with no clear patterns or trends.

- **Interpretation**: This indicates that the ARIMA model has effectively captured the main structure of the data, leaving only random noise.

ACF of Residuals

- **Plot**: The autocorrelation function (ACF) plot shows that most of the autocorrelations are within the significance bounds (blue dashed lines), with no significant spikes.
- **Interpretation**: This indicates minimal remaining autocorrelation, suggesting that the model has adequately captured the autocorrelation structure of the original series.

Normal Q-Q Plot of Residuals

- **Plot**: The Q-Q plot shows that the points lie approximately along the 45-degree reference line, with some minor deviations at the tails.
- **Interpretation**: This suggests that the residuals are approximately normally distributed.

Ljung-Box Test

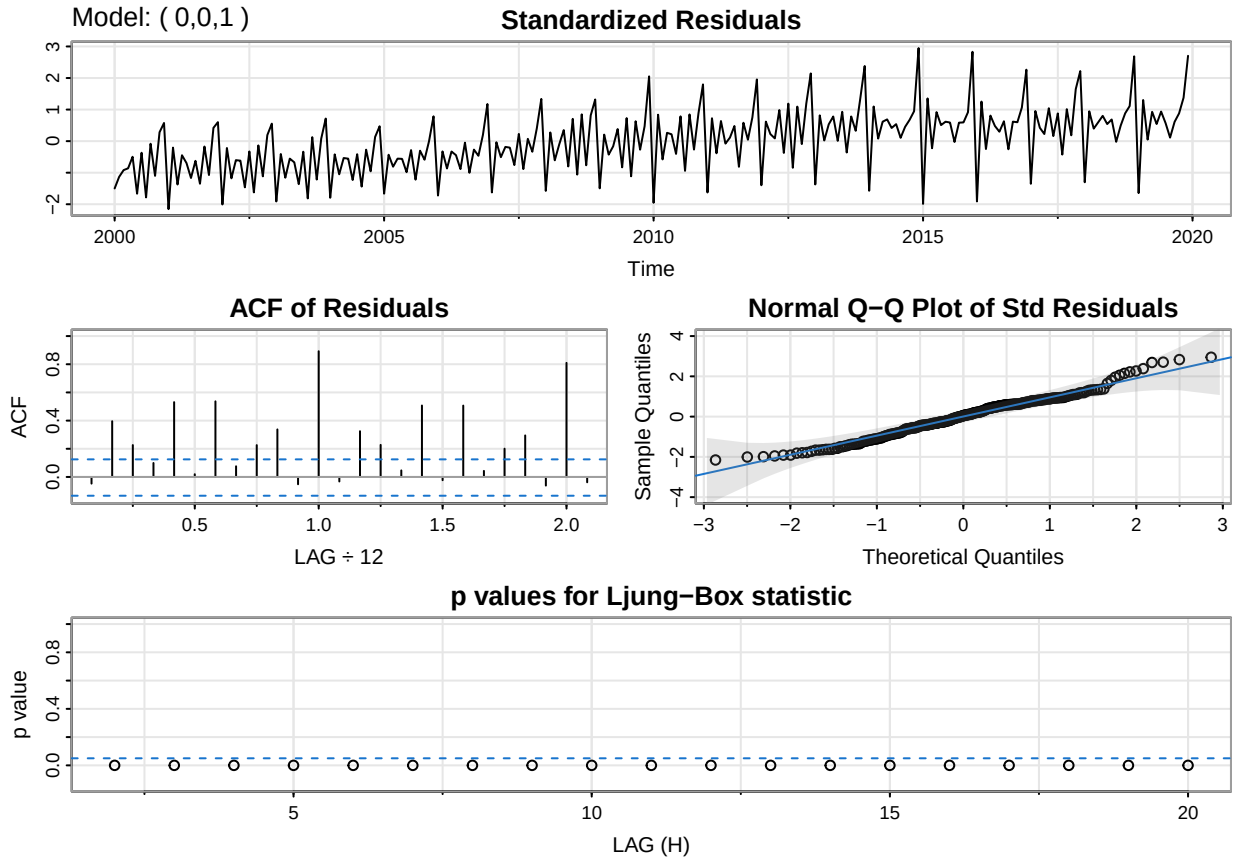
- **Plot**: The p-values for the Ljung-Box test are mostly above the significance level (0.05) for most lags.
- **Interpretation**: This indicates that the null hypothesis of no autocorrelation cannot be rejected at most lags, confirming that the residuals are white noise.

Therefore, Model1 appears to fit the data well.

```
sarima(X_t, p=0, d=0, q=1)
```

```
## initial  value 2.449183
## iter   2 value 2.223525
## iter   3 value 2.213465
## iter   4 value 2.208791
## iter   5 value 2.208151
## iter   6 value 2.208102
## iter   7 value 2.208091
## iter   8 value 2.208089
## iter   9 value 2.208089
## iter  10 value 2.208089
## iter  10 value 2.208089
## iter  10 value 2.208089
## final  value 2.208089
## converged
## initial  value 2.209478
## iter   2 value 2.209462
## iter   3 value 2.209453
## iter   4 value 2.209448
## iter   5 value 2.209445
## iter   6 value 2.209445
## iter   6 value 2.209445
## final  value 2.209445
## converged
```

```
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ma1     0.8507 0.0386 22.0518       0
## xmean  90.5869 1.0834 83.6145       0
##
## sigma^2 estimated as 82.56051 on 238 degrees of freedom
##
## AIC = 7.281767  AICc = 7.281978  BIC = 7.325275
##
```



ACF of Residuals

- **Plot**: The autocorrelation function (ACF) plot shows that several autocorrelations exceed the significance bounds (blue dashed lines), particularly at higher lags.
- **Interpretation**: This indicates remaining autocorrelation in the residuals, suggesting that Model 2 has not fully captured the autocorrelation structure of the original series.

Ljung-Box Test

- **Plot**: The p-values for the Ljung-Box test are mostly below the significance level (0.05) for many lags.
- **Interpretation**: This indicates that the null hypothesis of no autocorrelation is rejected at several lags, confirming that the residuals are not white noise.

Therefore, Model2 does not fit the data well.

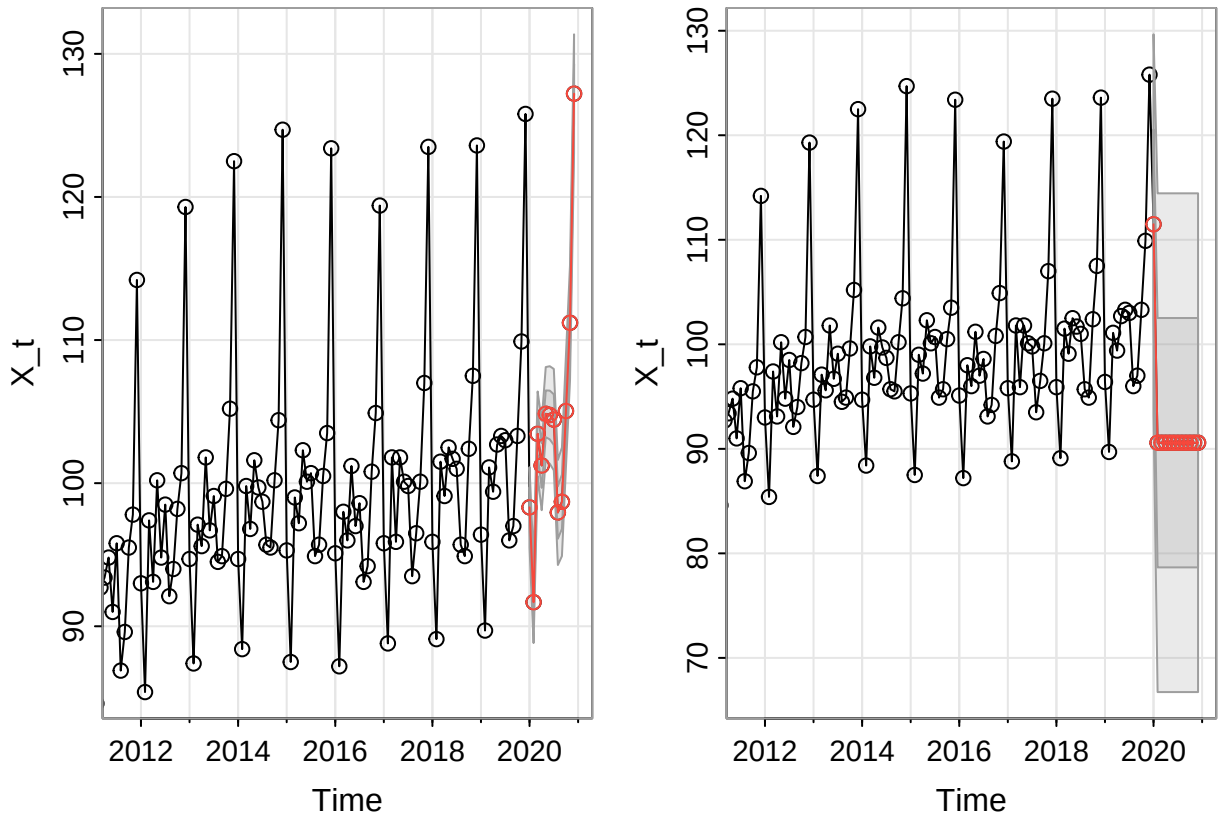
In conclusion, based on the diagnostic checks, **Model 1 (ARIMA(1,1,3)(0,1,1)[12])** is preferred over **Model 2 (ARIMA(0,0,1) with Non-Zero Mean)**.

12-Month Forecasting Using SARIMA:
Performed a 12-month forecast using SARIMA models for Switzerland's economic turnover. Compared the forecast accuracy of two different models to determine which one provided a better prediction, analyzing the results and justifying the choice based on forecasting performance.

```
# Forecast the next 12 months using sarima.for
par(mfrow=c(1, 2))
sarima.for(X_t, n.ahead=12, p=1, d=1, q=3, P=0, D=1, Q=1, S=12) #model1
```

```
## $pred
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2020  98.30956  91.68066 103.44926 101.23422 104.84611 104.76420 104.43869
##            Aug       Sep       Oct       Nov       Dec
## 2020  97.95116  98.69768 105.04097 111.20678 127.21805
##
## $se
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2020 1.402319 1.427609 1.478546 1.562249 1.633702 1.703830 1.770825 1.835453
##           Sep      Oct      Nov      Dec
## 2020 1.897864 1.958292 2.016909 2.073870
```

```
sarima.for(X_t, n.ahead=12, p=0, d=0, q=1) #model2
```



```
## $pred
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 2020 111.48584  90.58691  90.58691  90.58691  90.58691  90.58691  90.58691
##             Aug       Sep       Oct       Nov       Dec
## 2020  90.58691  90.58691  90.58691  90.58691  90.58691
##
## $se
##             Jan       Feb       Mar       Apr       May       Jun       Jul
## 2020  9.086282 11.929125 11.929125 11.929125 11.929125 11.929125 11.929125
##             Aug       Sep       Oct       Nov       Dec
## 2020 11.929125 11.929125 11.929125 11.929125 11.929125
```

Model 1: ARIMA(1,1,3)(0,1,1)[12]

- **Forecast Plot**:
  - The forecasted values (in red) follow the observed values closely, maintaining the trend and seasonality.
  - Since both models aim to capture the same level of confidence, the model with the narrower confidence interval is preferred. It provides the same level of confidence but in a smaller range, indicating more accurate and reliable predictions.
  - The confidence intervals (shaded area) are reasonably narrow compared to Model2 indicating the better prediction.

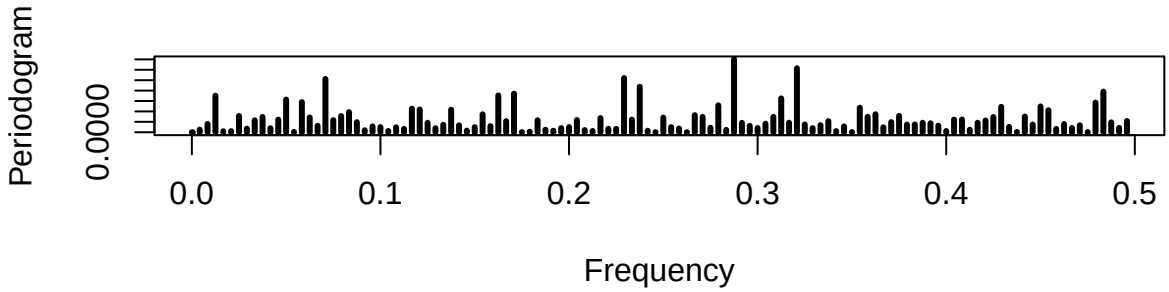
Model 2: ARIMA(0,0,1) with Non-Zero Mean

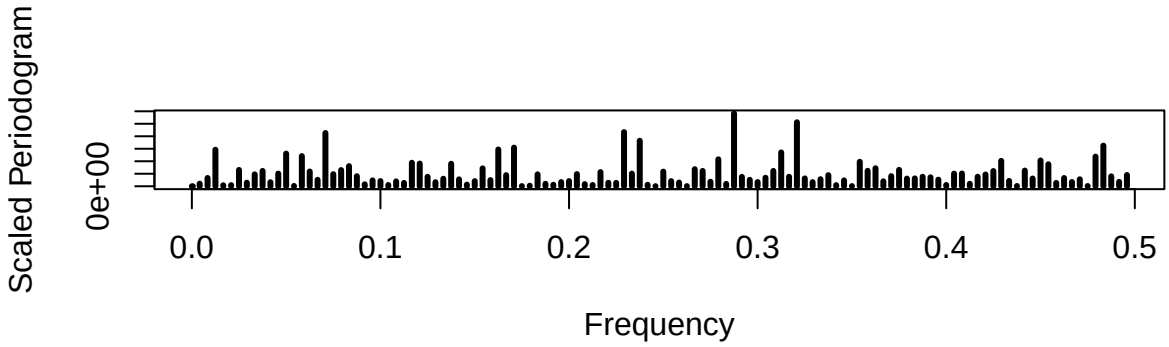
- **Forecast Plot**:
  - The forecasted values (in red) show a more pronounced deviation from the observed values.
  - The confidence intervals (shaded area) are wider compared to Model1

Therefore, **Model 1** is preferred for forecasting the next 12 months as it provides more accurate and confident forecasts compared to **Model 2**.

Residual Analysis and Periodogram Plotting:
Extracted the residuals from the preferred ARIMA model and plotted the periodogram to identify key frequencies. Analyzed the maximum values in the periodogram along with their corresponding frequencies to evaluate the residual behavior.

```r
# Periodogram

# Extract residuals from the preferred model (Model 1)
residuals_model1 <- residuals(Model1)

# Compute and plot the periodogram of the residuals
n <- length(residuals_model1)
Per <- Mod(fft(residuals_model1 - mean(residuals_model1)))^2 / n
sPer <- 4 / n * Per  # Scaled periodogram
Freq <- (1:n - 1) / n  # Fourier or fundamental frequencies

# Plot the periodogram
par(mfrow = c(2, 1))
plot(Freq[1:(n/2)], Per[1:(n/2)], type = 'h', lwd = 3, ylab = "Periodogram", xlab = "Frequency")
plot(Freq[1:(n/2)], sPer[1:(n/2)], type = 'h', lwd = 3, ylab = "Scaled Periodogram", xlab = "Frequency")
```






```r
# Identify the maximum value(s) and corresponding frequency(ies)
max_value <- max(sPer[1:(n/2)])
max_freq <- Freq[which.max(sPer[1:(n/2)])]

# Print the maximum value and corresponding frequency
cat("Maximum value:", max_value, "\n")
```

```
## Maximum value: 5.834582e-05
```

```r
cat("Corresponding frequency:", max_freq, "\n")
```

```
## Corresponding frequency: 0.2875
```

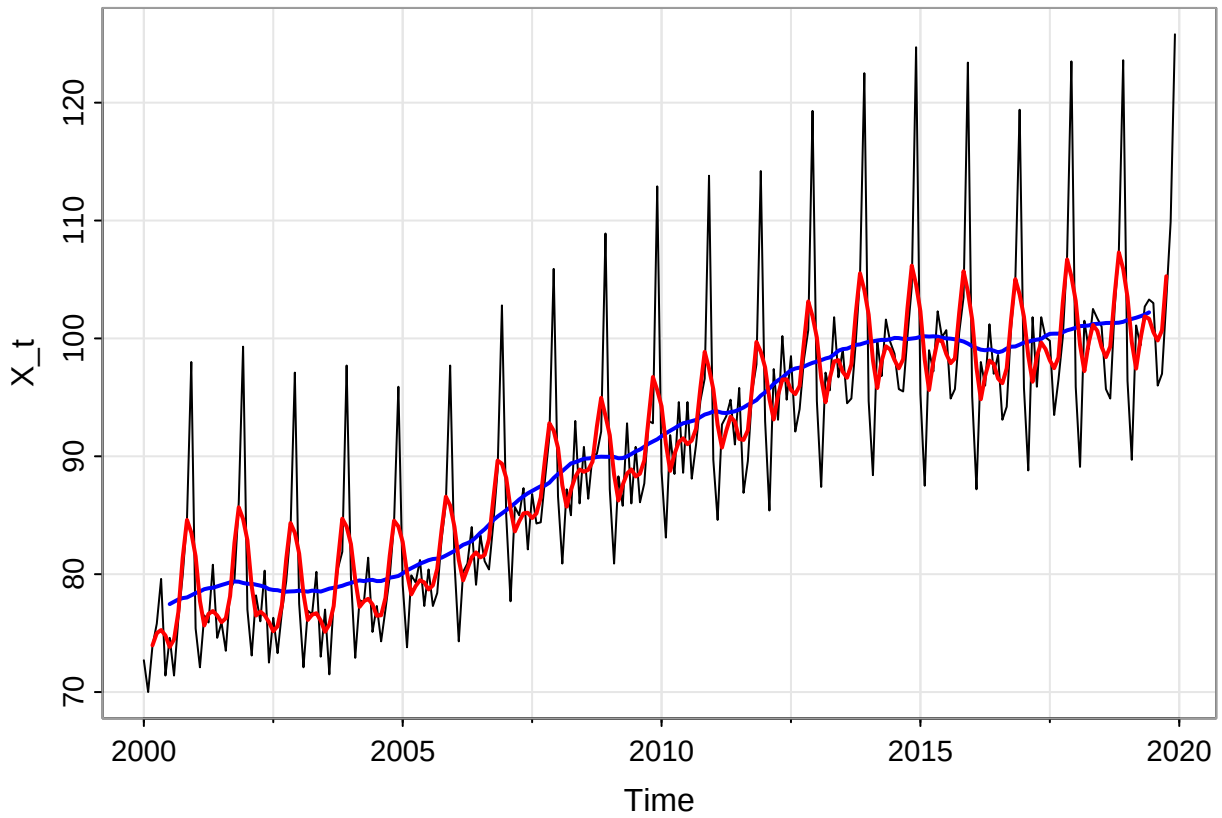
- The periodogram of the residuals from Model 1 shows a notable peak at a frequency of 0.2875 with a maximum value of approximately 5.834582e-05.
- This suggests that there is some remaining periodic behavior in the residuals that the model did not capture.
- Despite this, the model is preferred overall based on its lower AIC and BIC values, and the diagnostic checks indicating a good fit to the main structure of the data. Further refinement may be needed to fully address this periodic component.

Smoothing Time Series Data using different methods

Smoothing Time Series with Symmetric Moving Averages:
Applied symmetric moving average filters with window sizes k=2 and k=6 to smooth the time series data.

Evaluated which window size provided a smoother trend and determined which filter best captured the seasonal behavior of the time series.

```r
# Load the data related to your assigned country
data = read.csv( "switzerland.csv")
data$Year <- as.numeric(substr(data$TIME, 1, 4))
data$Month <- as.numeric(substr(data$TIME, 6, 7))
X_t <- ts(data$Switzerland, start=data$Year[1], frequency=12)

# symmetric moving average filters

#k=6
wgts_k6 = c(.5, rep(1,11), .5)/12
ma_6 = filter(X_t, sides=2, filter=wgts_k6)

#k=2
wgts_k2 = c(.5, rep(1,3), .5)/4
ma_2 = filter(X_t, sides=2, filter=wgts_k2)

#plot
tsplot(X_t, col="black") #plot original data X_t
lines(ma_6, lwd=2, col="blue") #plot MA with k=6
lines(ma_2, lwd=2, col="red") #plot MA with k=2
```



```
#par(fig = c(.75, 1, .75, 1), new = TRUE) # the insert
#nwgts = c(rep(0,20), wgts_k6, rep(0,20))
#plot(nwgts, type="l", ylim = c(-.02,.1), xaxt='n', yaxt='n', ann=FALSE)
```

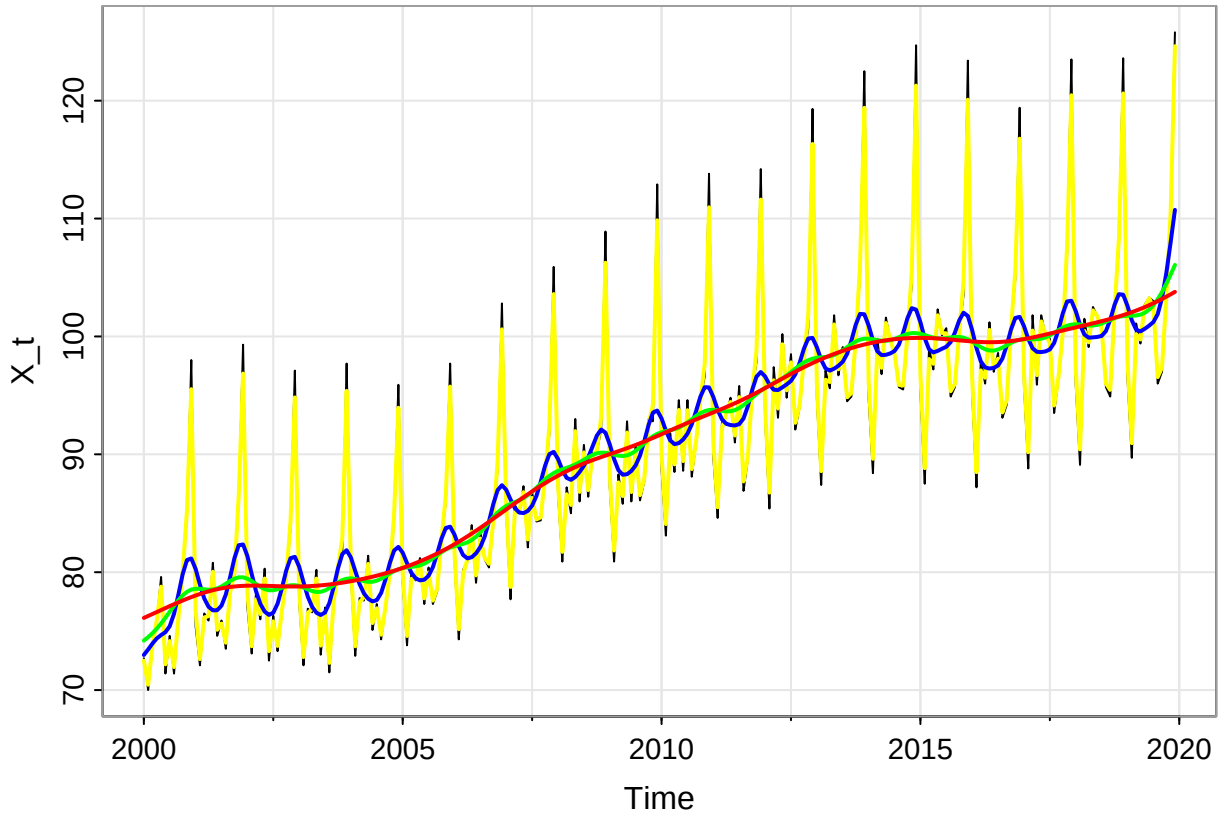
- **For $k = 2$:** - The symmetric moving average filter with $k = 2$ provides a less smooth trend as it captures more short-term fluctuations. - It better captures the seasonal behavior of the time series because the smaller window size allows for more detailed tracking of the data variations.
- **For $k = 6$:**
  - The symmetric moving average filter with $k = 6$ provides a smoother trend by reducing short-term fluctuations and highlighting the long-term trend.
  - It captures less of the seasonal behavior compared to $k = 2$ due to the broader window size which averages out more of the short-term variations.

Therefore, the $k = 2$ filter captures the seasonal behavior better, while the $k = 6$ filter provides a smoother long-term trend.

Kernel Smoothing for Time Series:

Applied Kernel Smoothing using a normal kernel with bandwidths `bandwidth` $= 0.1, 0.5, 1, 2$ to smooth the time series data. Analyzed which bandwidth provided a better smoother for the trend and identified the bandwidth that best captured the seasonal behavior of the time series.

```
# Load the data related to your assigned country
data = read.csv( "switzerland.csv")
data$Year <- as.numeric(substr(data$TIME, 1, 4))
data$Month <- as.numeric(substr(data$TIME, 6, 7))
X_t <- ts(data$Switzerland, start=data$Year[1], frequency=12)

# Kernel Smoothing with normal

# Define bandwidths
bandwidths <- c(0.1, 0.5, 1, 2)

# Plot the original time series
tsplot(X_t, col="black")

# Apply kernel smoothing for different bandwidths and plot
colors <- c("yellow", "blue", "green", "red")
for (i in 1:length(bandwidths)) {
  smoothed <- ksmooth(time(X_t), X_t, "normal", bandwidth=bandwidths[i])
  lines(smoothed, lwd=2, col=colors[i])
}
```



Bandwidth	0.1	0.5	1	2
	Yellow	Blue	Green	Red

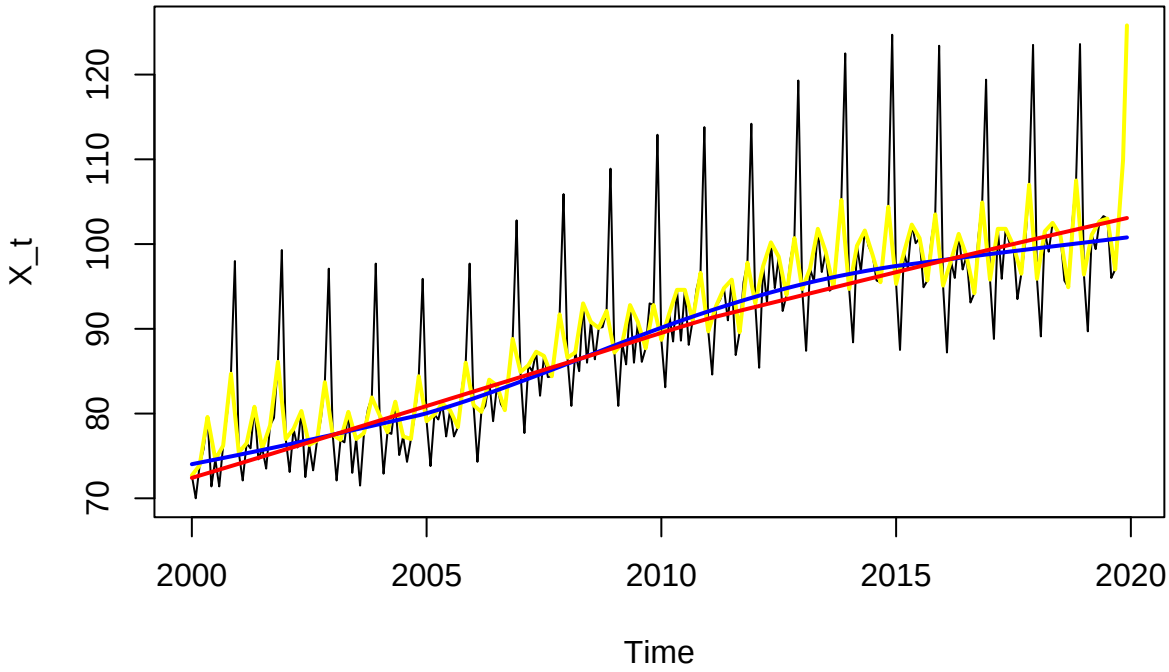
- As you can see above, the yellow line hides most of the black plot, indicating that it captures the seasonal behavior of the time series very well. The blue line also captures the seasonality effectively, followed by the green and red lines. This shows that as the bandwidth decreases, it better captures the seasonality due to considering a narrower time period.
- Conversely, the red line is the smoothest plot as it considers the longest period at once, followed by the green, blue, and yellow lines. Therefore, as the bandwidth increases, the plot becomes smoother, providing a clearer long-term trend but less detail in seasonal variations.

Lowess Smoothing for Time Series:

Applied Lowess smoothing with span values of $\texttt{f} = 0.01, 0.5, 0.9$ to the time series data. Evaluated which span provided a better smoother for the trend and identified the span that best captured the seasonal behavior of the time series.

```
# Load the data related to your assigned country
data = read.csv( "switzerland.csv")
data$Year <- as.numeric(substr(data$TIME, 1, 4))
data$Month <- as.numeric(substr(data$TIME, 6, 7))
X_t <- ts(data$Switzerland, start=data$Year[1], frequency=12)

# Lowess
plot(X_t)
```

```
lines(lowess(X_t, f=0.01), lwd=2, col="yellow")
lines(lowess(X_t, f=0.5), lwd=2, col="blue")
lines(lowess(X_t, f=0.9), lwd=2, col="red")
```

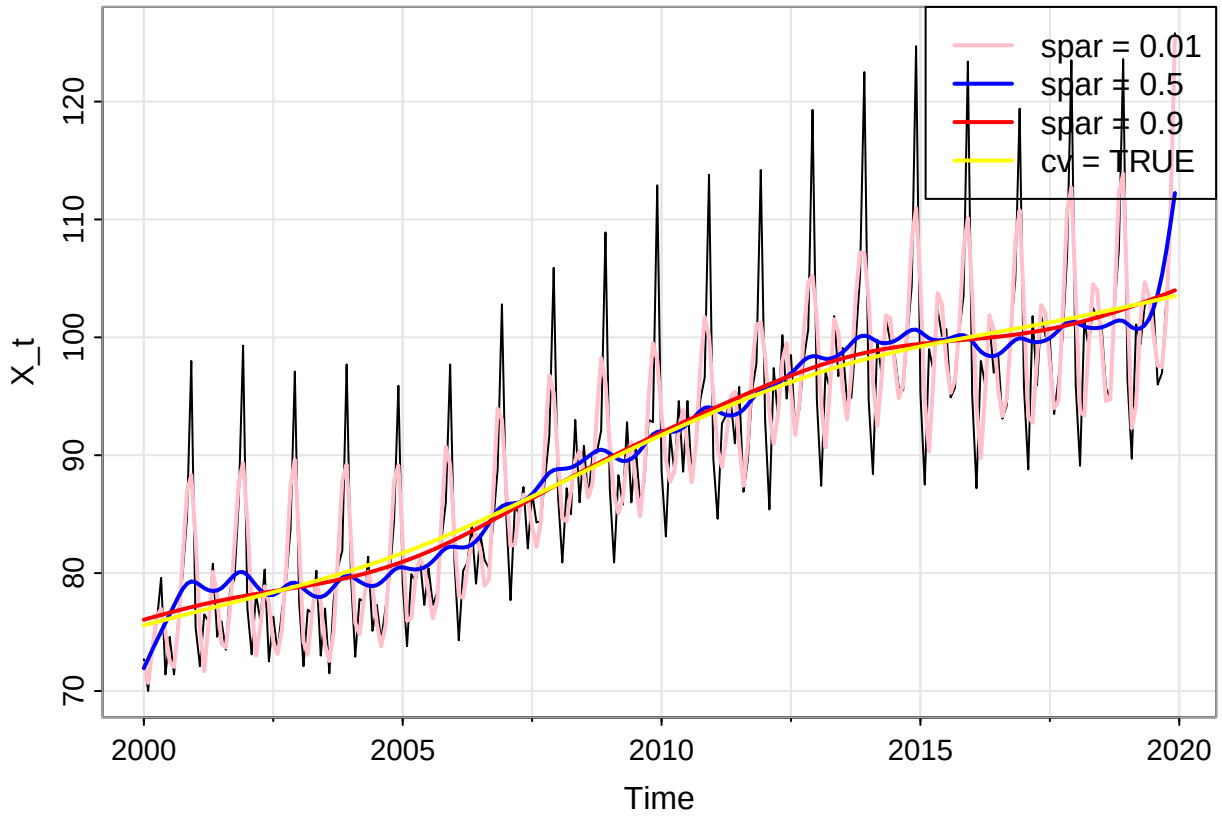


- As shown above, the yellow line (span = 0.01) closely follows the black line (original plot), indicating that it captures the seasonal behavior well but is less smooth compared to the blue line (span = 0.5) and the red line (span = 0.9).
- The blue and red lines are almost straight, which shows a very smooth trend. The blue line, however, has a slight up-and-down trend, similar to a sine function, capturing more seasonal behavior than the red line. Therefore, the blue line captures the seasonal behavior better than the red line, while the red line is smoother than the blue line.
- In conclusion, as the span f increases, the line becomes smoother in the order of 0.9 (red), 0.5 (blue), and 0.01 (yellow). Conversely, as the span f decreases, the line captures the seasonal behavior better in the order of 0.01 (yellow), 0.5 (blue), and 0.9 (red).

Smoothing Splines for Time Series:

Applied smoothing splines with parameter values `spar` $= 0.01, 0.5, 0.9$, and used cross-validation (cv = TRUE) to optimize the smoothing. Plotted the time series along with the fitted smoothers, and determined which smoothing parameter provided a better fit for the trend and which best captured the seasonal behavior of the time series.

```r
# Load the data related to your assigned country
data = read.csv( "switzerland.csv")
data$Year <- as.numeric(substr(data$TIME, 1, 4))
data$Month <- as.numeric(substr(data$TIME, 6, 7))
X_t <- ts(data$Switzerland, start=data$Year[1], frequency=12)

tsplot(X_t)

lines(smooth.spline(time(X_t), X_t, spar = 0.01), col = "pink", lwd = 2)  # spar = 0.01
lines(smooth.spline(time(X_t), X_t, spar = 0.5), col = "blue", lwd = 2)    # spar = 0.5
lines(smooth.spline(time(X_t), X_t, spar = 0.9), col = "red", lwd = 2)     # spar = 0.9
lines(smooth.spline(time(X_t), X_t, cv = TRUE), col = "yellow", lwd = 2)    # cv = TRUE


legend("topright", legend = c("spar = 0.01", "spar = 0.5", "spar = 0.9", "cv = TRUE"),
       col = c("pink", "blue", "red", "yellow"), lwd = 2)
```



- **Pink line (spar = 0.01)**: This line follows the data very closely, capturing the seasonal behavior effectively. It is the least smooth among the splines due to the low `spar` value, indicating a high level of detail and less smoothing.
- **Blue line (spar = 0.5)**: This line still captures some of the seasonal behavior but is smoother than the pink line. It provides a balance between capturing seasonal patterns and overall trend.
- **Red line (spar = 0.9)**: This line is very smooth and captures the overall trend well but misses finer seasonal details. The high `spar` value indicates a greater emphasis on smoothing, reducing the noise

and short-term fluctuations.

- **Yellow line (cv = TRUE)**: This line is obtained using cross-validation to automatically determine the optimal `spar` value. It provides a balanced fit, smoother than the pink line but less smooth than the red line. This approach finds an optimal balance between capturing the seasonal behavior and the overall trend.
- In short, as the `spar` value **decreases**, the smoother captures more detailed seasonality but becomes less smooth overall. Conversely, as the `spar` value **increases**, the smoother becomes smoother and better at capturing the overall trend, but it captures less of the detailed seasonality and short-term variations.

Comparison of Theoretical Spectral Density and Periodogram in Simulated Data

In applications, we will often observe series containing a signal that has been delayed by some unknown time D, i.e.,

$$X_t = Y_t + AY_{t-D} + V_t,$$

where Y_t and V_t are stationary and independent with zero means and spectral densities $f_Y(\omega)$ and $f_V(\omega)$, respectively. The delayed signal is multiplied by some unknown constant A.

$$f_X(\omega) = [1 + A^2 + 2A\cos(2\pi\omega D)]f_Y(\omega) + f_V(\omega).$$

```
set.seed(8219)
```

Step 1- Simulate Y_t: Generate $n = 1000$ observations from an AR(1) time series with $\phi = 0.8$ and $Z_t \sim N(0, 2)$. Note that 2 is the standard deviation of the normal distribution.

```
Y_t <- arima.sim(n = 1000, list(ar = 0.8), sd = 2)
```

Step 2- Simulate V_t: Generate $n = 1000$ observations from normal noise $N(0, 1)$ independent of Y_t.

```
V_t <- rnorm(n=1000, mean = 0, sd = 1)
```

Step 3- Choose D and A: Set the delay to $D = 4$ and $A = 0.6$.

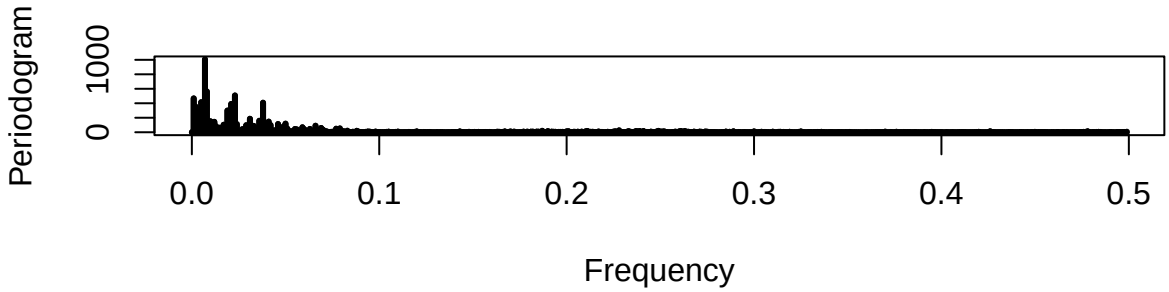
```
D <- 4
A <- 0.6
```

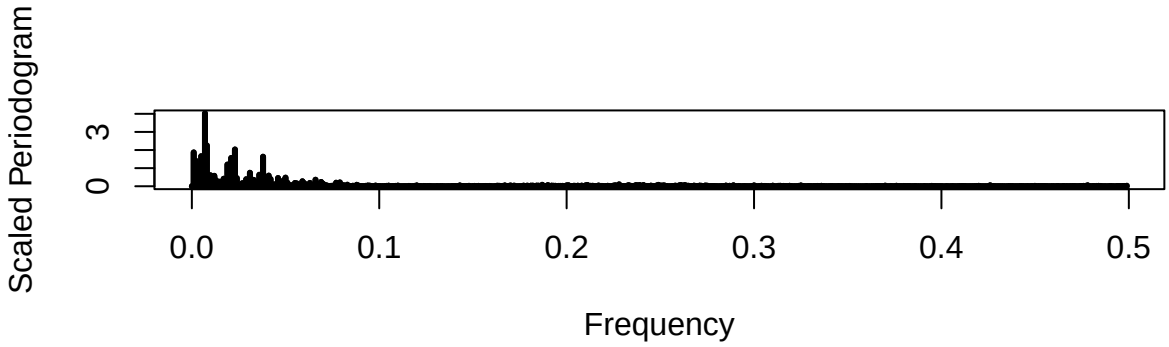
Step 4- Construct X_t: Use the provided equation to create the time series X_t.

```
X_t <- Y_t + A * c(rep(0, D), Y_t[1:(length(Y_t) - D)]) + V_t
```

Step 5- Periodogram: Compute the periodogram of X_t and plot it.

```
n<-1000
Per    = Mod(fft(X_t-mean(X_t)))^2/n  # The periodogram
sPer = 4/n * Per # The scaled periodogram
Freq  = (1:n -1)/n     # Fourier or fundamental frequencies.

par(mfrow = c(2,1))
plot(Freq[1:n/2], Per[1:n/2], type='h', lwd=3, ylab="Periodogram", xlab="Frequency")
#abline(h=sigma2, col = 2)
plot(Freq[1:n/2], sPer[1:n/2], type='h', lwd=3, ylab="Scaled Periodogram", xlab="Frequency")
```

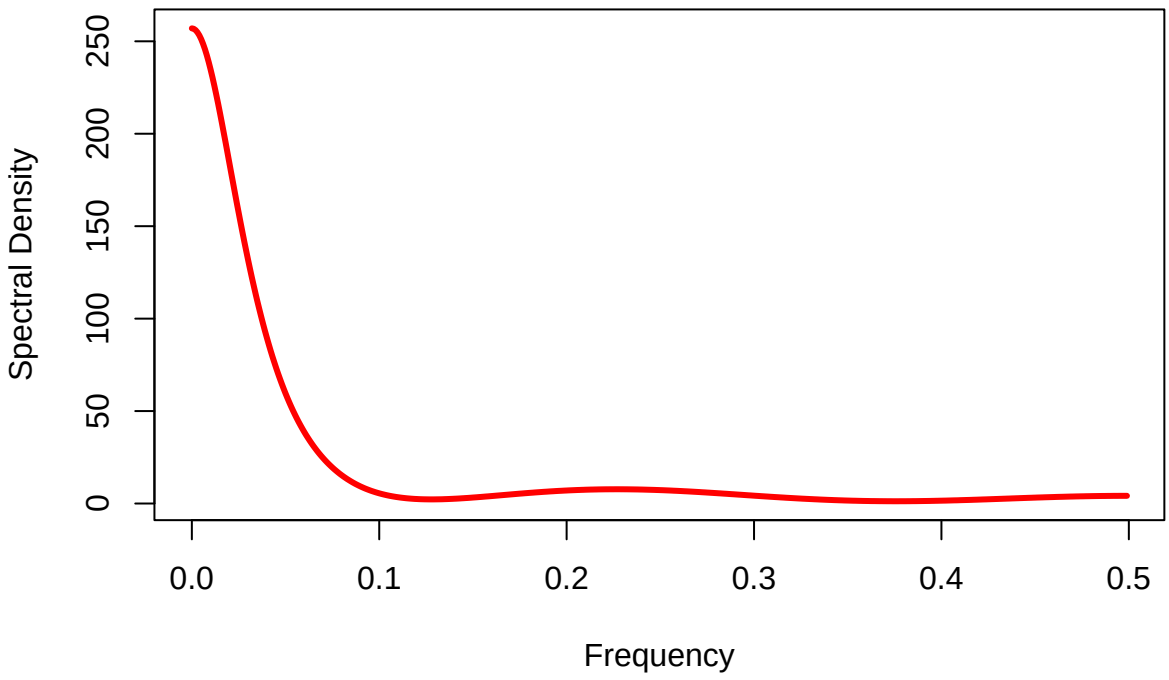




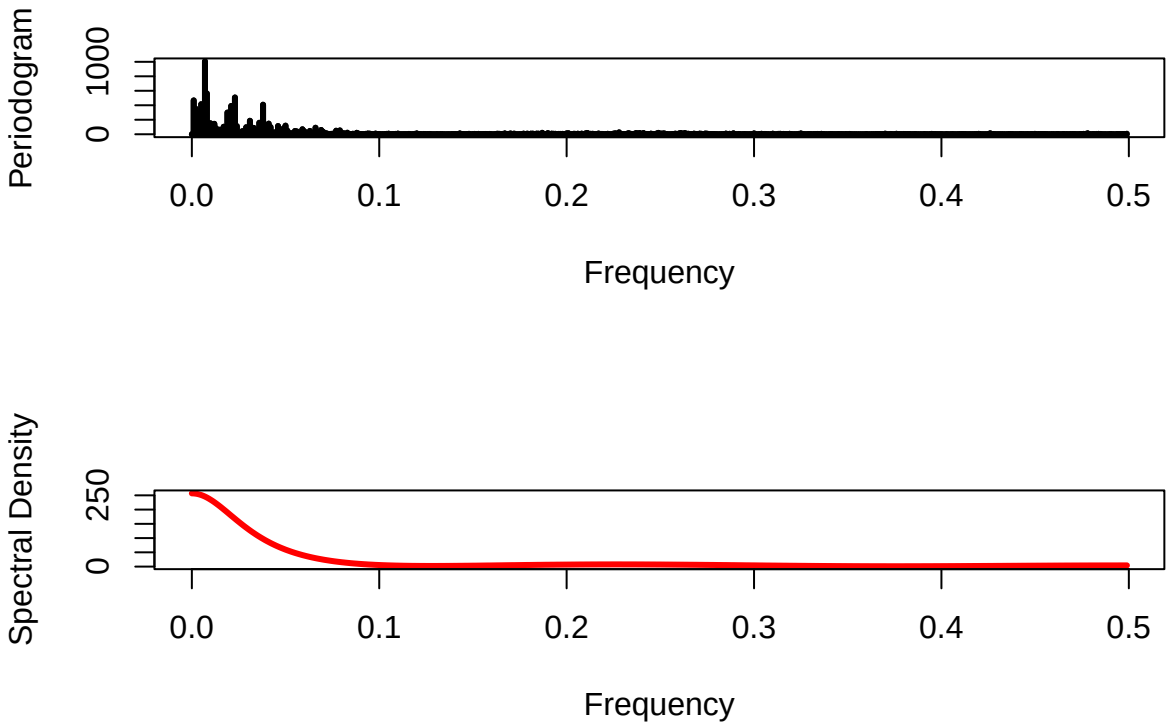

```
#arma.spec(var.noise = sigma2, n.freq = n/2, main="White Noise", col=4)
```

Step 6- Theoretical spectral density and its plot: Calculate and plot the theoretical spectral density $f_X(\omega)$.

```
f_Y <- function(omega) {
  (4 / (1.64 - 1.6 * cos(2*pi*omega)))
}

f_V <- function(omega) {
  rep(1, length(omega))
}

f_X <- function(omega) {
  (1 + 0.6^2 + 2 * 0.6 * cos(2 * pi * omega * 4)) * f_Y(omega) + f_V(omega)
}

omega <- Freq[1:(n/2)]
spec_X <- f_X(omega)

# Plot the theoretical spectral density
plot(omega, spec_X, type = 'l', lwd = 3, col = 'red', ylab = "Spectral Density", xlab = "Frequency")
```




```
par(mfrow = c(2,1))
plot(Freq[1:n/2], Per[1:n/2], type='h', lwd=3, ylab="Periodogram", xlab="Frequency")
#plot(Freq[1:n/2], sPer[1:n/2], type='h', lwd=3, ylab="Scaled Periodogram", xlab="Frequency")
plot(omega, spec_X, type = 'l', lwd = 3, col = 'red', ylab = "Spectral Density", xlab = "Frequency")
```




```
#arma.spec(var.noise = sigma2, n.freq = n/2, main="White Noise", col=4)
```

it is sufficient to provide a brief description of the similarities and differences between the theoretical spectral density and the periodogram of the simulated data.

- **Overall Shape:** Both the theoretical spectral density and the periodogram have a similar overall shape, with predominant frequency at lower frequencies and densities decreasing to 0 as the frequency increases.
- **Smoothness:** The theoretical spectral density is smoother than the periodogram. The periodogram shows more variability because it is based on a single simulated dataset.
- **Amplitude Differences:** The amplitude of the peaks in the periodogram differs from the theoretical spectral density due to the finite sample size and randomness in the simulation.
- **Convergence with More Simulations:** With more simulations, the periodogram would become smoother and more similar to the theoretical spectral density.