Forecasting Champagne Sales

PSTAT 174 Final Project

Chloe Dinh-Luong

2023-12-08

Abstract

This report presents a comprehensive analysis and forecasting of champagne sales using time series techniques. The initial steps involve data preprocessing, transforming the original sales data into a stationary and invertible form. Two candidate models, Model (A): $\text{SARIMA}(0,1,2)(0,1,1)_{12}$ and Model (B): $\text{SARIMA}(0,1,1)(0,1,1)_{12}$, were considered, with Model (B) selected as the preferred choice based on the Akaike Information Criterion corrected (AICc). Diagnostic checks were conducted to validate the selected model, confirming its stationarity, invertibility, and adherence to key assumptions. Despite non-Gaussian residuals, various diagnostic tests ensured the model's reliability for forecasting purposes. The forecasting results for the next 12 time points, along with confidence intervals, were visualized and analyzed, demonstrating the model's effectiveness in predicting future champagne sales. This forecasting model provides valuable insights for businesses and decision-makers in understanding and anticipating sales patterns.

Introduction

The dataset for this project consists of champagne sales data spanning from 1964 to 1972, focusing on two variables: month and sales. The "month" variable represents the time dimension in a time series format, and "sales" represents the quantity of champagne sold during each corresponding month. This dataset aims to capture the sales patterns of champagne over the specified time period.

The choice of using a champagne sales dataset from 1964 to 1972 offers an interesting and potentially insightful avenue for time series analysis. Understanding and predicting sales patterns in the champagne industry can have practical implications for businesses involved in production, distribution, and marketing of champagne.

The primary objective of this project is to address the challenge of forecasting future champagne sales. To achieve this, time series analysis techniques will be applied to model the historical sales data. Specifically, we will utilize SARIMA (Seasonal AutoRegressive Integrated Moving Average) models to account for seasonality, trends, and autocorrelation present in the time series data. The developed forecasting model aims to equip businesses in the champagne industry with a reliable tool for predicting sales volumes. Such predictions are crucial for effective business planning, optimizing inventory management, and facilitating strategic decision-making in a dynamic market. Through this analysis, we aim to contribute valuable insights to the champagne industry, enabling stakeholders to navigate market trends and plan for future demand.

The results of our time series analysis on champagne sales data reveal positive outcomes in the successful development of a forecasting model, specifically $\text{SARIMA}(0,1,1)(0,1,1)_{12}$. This model, despite showing minor deviations in residuals' normality, demonstrated effectiveness through diagnostic checks and accurate predictions for future sales volumes. The forecasting tool, designed to aid businesses in the champagne industry, offers actionable insights for inventory management and strategic decision-making. The conclusions underscore the suitability of the SARIMA model, emphasizing its potential to guide business strategies. The analysis utilized the champagne_sales.csv dataset, and the implementation was carried out using R.

Analysis

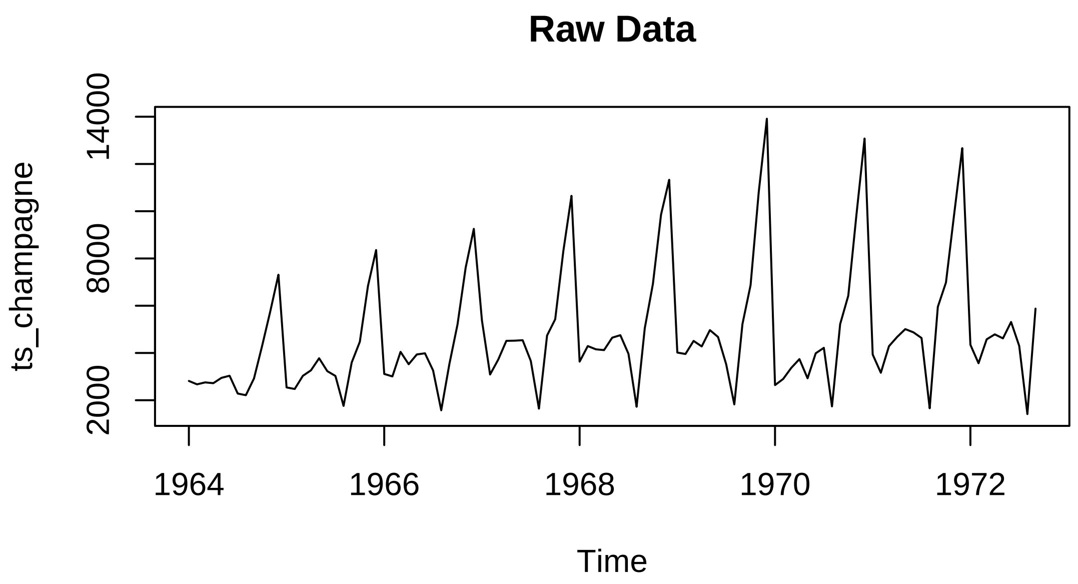
We start by reading in the champagne sales data and displaying the initial rows for a quick overview of its structure and content.

```r
champagne.csv <- read.table("champagne_sales.csv", sep = ",", header=F, skip=1)
head(champagne.csv)
```

```
##        V1   V2
## 1 1964-01 2815
## 2 1964-02 2672
## 3 1964-03 2755
## 4 1964-04 2721
## 5 1964-05 2946
## 6 1964-06 3036
```

Next, we extract the 'sales' variable from the dataset, convert it into a time series using ts(), and create a time series plot (ts.plot()).

```r
champagne = champagne.csv[,2]
fig.dim = c(4, 2)
ts_champagne = ts(champagne, start = c(1964,1), frequency = 12)
ts.plot(ts_champagne, main = "Raw Data")
```



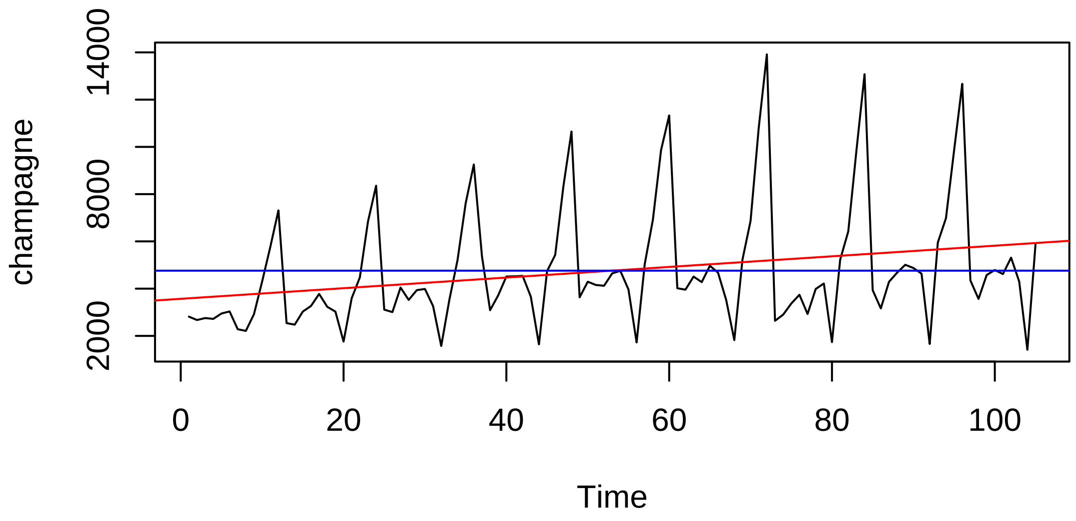
Then, we plot the time series of champagne sales using plot.ts() to visualize the champagne sales time series, inspect trends, and patterns. The length of the time series is calculated, and a linear regression model is fitted, representing sales as a function of time. The red regression line is added to the plot, aiding trend interpretation. The mean of champagne sales is calculated, and a blue horizontal line is included in the plot to signify the average sales level.

```r
plot.ts(champagne)

nt = length(champagne)
fit <- lm(champagne ~ as.numeric(1:nt)); abline(fit, col="red")
mean(champagne)
```

```
## [1] 4761.152
```

```r
abline(h=mean(champagne), col="blue")
```



Training and Test Sets

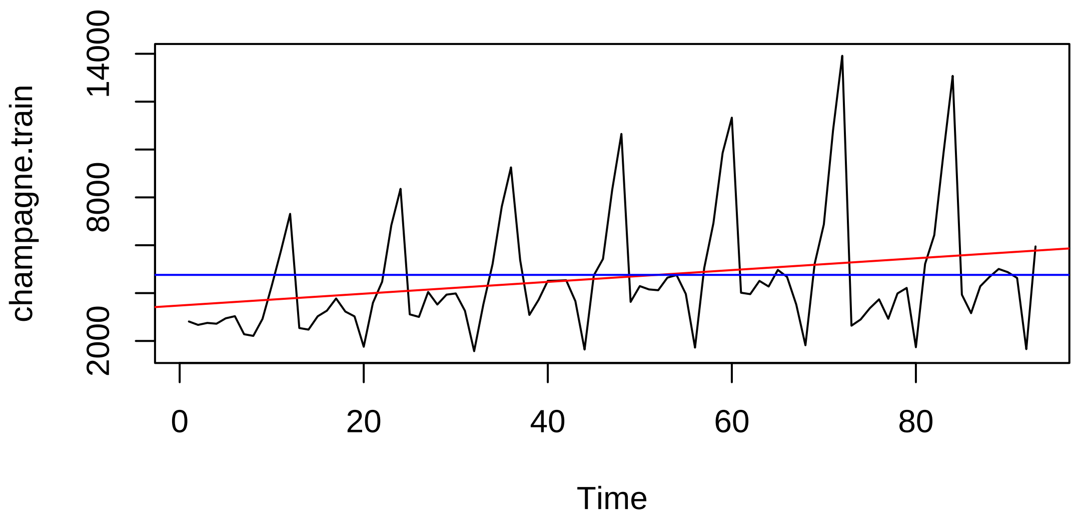
For evaluating our forecasting model, we split the time series data into training and test sets, as there is no new data for validation. The training period, represented by champagne.train, includes the first 93 observations, while the test period, champagne.test, consists of the subsequent 12 observations.

```r
champagne.train = champagne[c(1: 93)]
champagne.test = champagne[c(94:105)]
```

Plot Time Series (Training Set)

We start by visualizing the time series of the training set using plot.ts(). We fit a linear regression model to the training set and add the regression line to the plot in red. This step provides a closer look at the trends and patterns within the training data.

```r
plot.ts(champagne.train)
fit <- lm(champagne.train ~ as.numeric(1:length(champagne.train))); abline(fit, col="red")
abline(h=mean(champagne), col="blue")
```

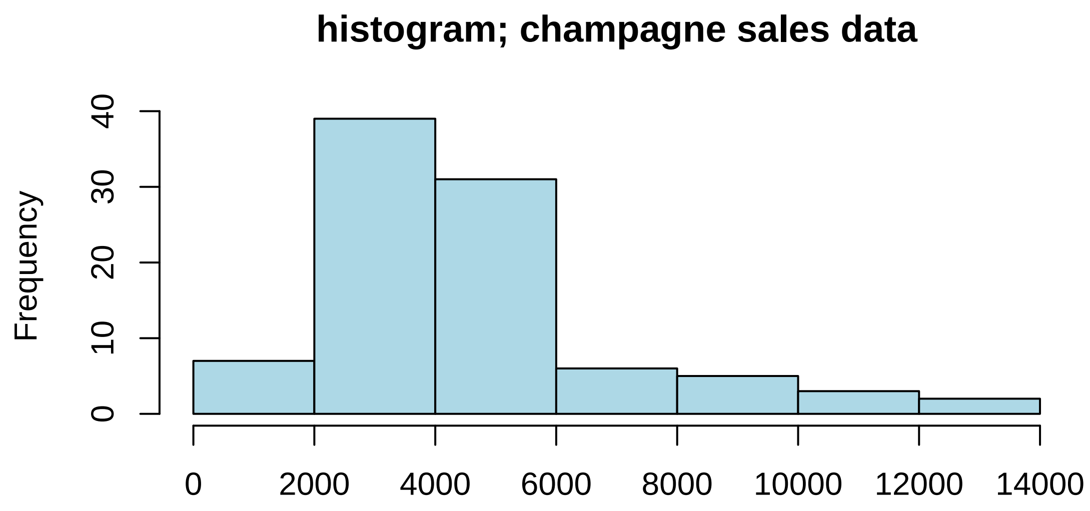


The regression line offers insights into the overall directional movement of the data. Here, the positive slope indicates a positive linear trend, suggesting a consistent upward pattern as the independent variable increases. This highlights the need to stabilize variance. Additionally, the chart reveals a recurring pattern at regular intervals, indicating seasonality. Sharp changes may signify external influences or outliers with potential significant impacts on the overall trend.

- Confirm Non-Stationarity

To confirm non-stationarity of the original data, we create a histogram using hist(). The histogram illustrates the frequency distribution of sales values, allowing us to observe any apparent patterns or anomalies.

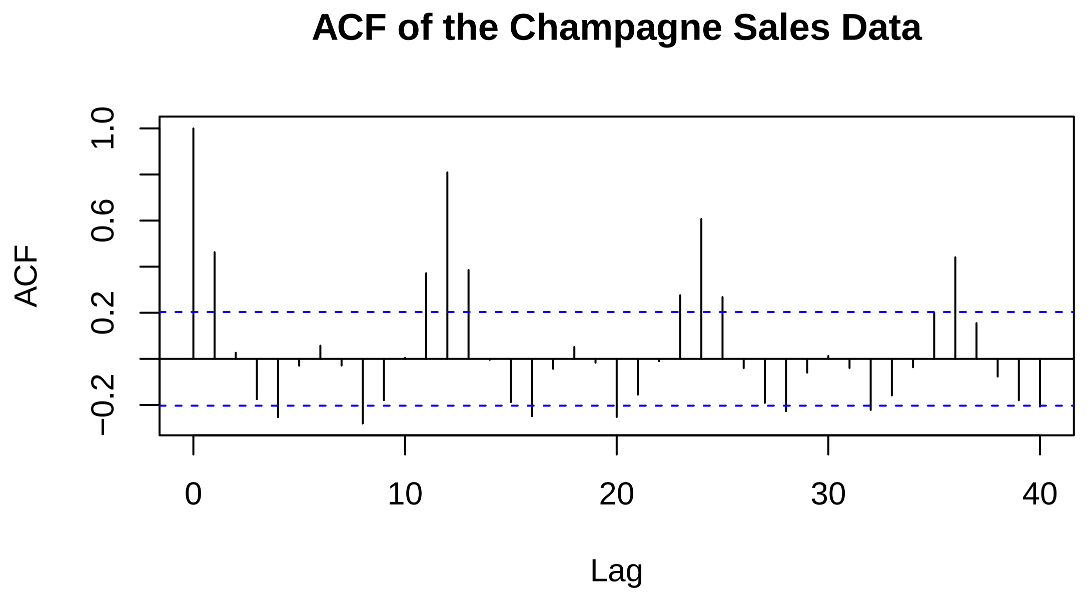
```
hist(champagne.train, col="light blue", xlab="", main="histogram; champagne sales data")
```



The right skewness observed in the histogram suggests that the data may not be normally distributed, and this non-normality could contribute to the variance instability.

We verify non-stationarity of the original data by inspecting the autocorrelation function (ACF) of the training set using acf(). This analysis reveals correlations between each observation and its lagged values. It helps identify potential seasonality or temporal patterns within the data.

```r
acf(champagne.train,lag.max=40, main="ACF of the Champagne Sales Data")
```



From the graph, we can see that the ACF is periodic, indicating the presence of autocorrelation at regular intervals, further supporting the evidence of seasonality in the data.

Our next steps involve stabilizing variance through necessary transformations and addressing seasonality and trend through differencing.

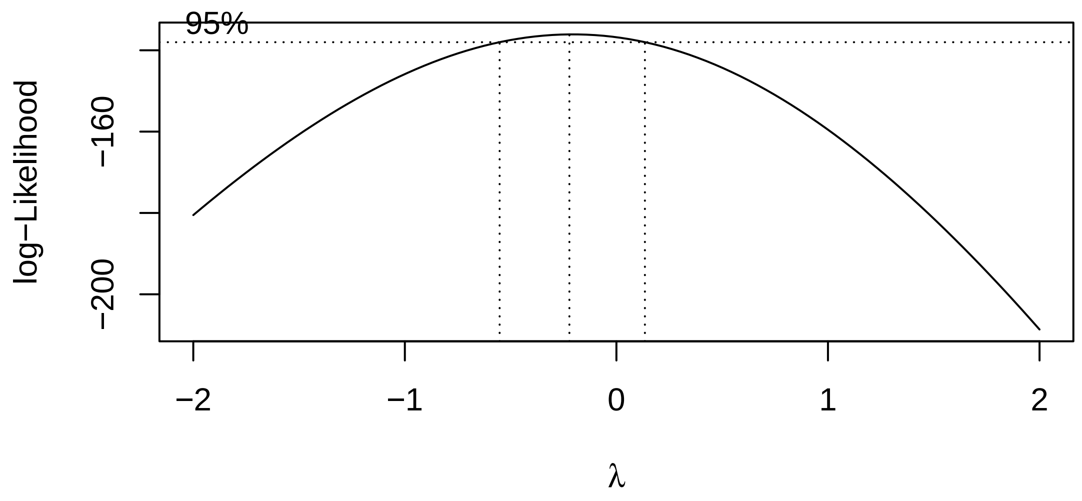
Transformations

We begin by exploring different transformations to achieve stationarity.

- Box-Cox Transformation

We perform the Box-Cox transformation on the training set using boxcox(). The resulting lambda value is stored in bcTransform which can be used for stabilizing variance through the Box-Cox transformation.

```
#install.packages("MASS")
library(MASS)
bcTransform <- boxcox(champagne.train~ as.numeric(1:length(champagne.train)))
```



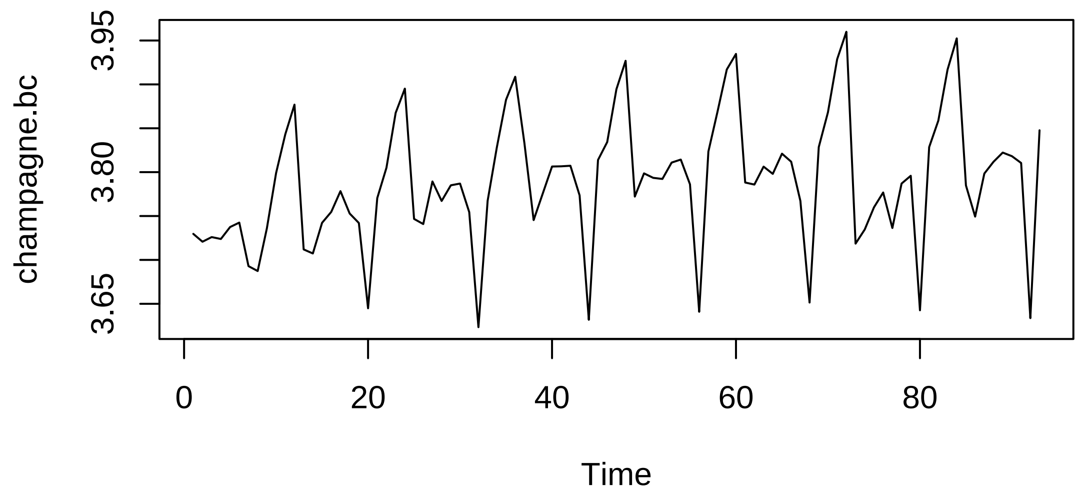
```
bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
```

```
## [1] -0.2222222
```

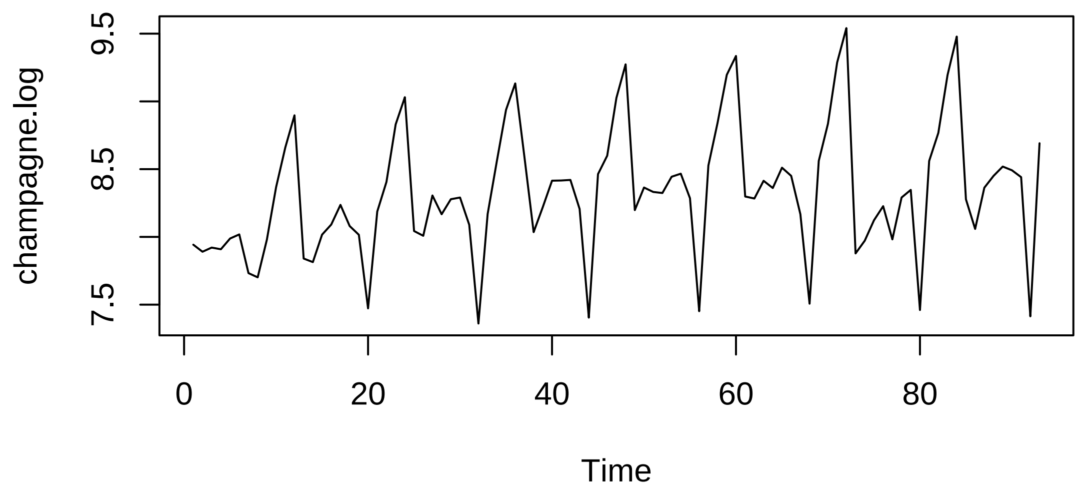
The bcTransform command yields a value of -0.22, closely approaching 0 and remaining within the confidence interval. For that reason, we opt to use $\lambda = 0$, corresponding to the natural logarithm.

The subsequent graphs depict the plot of U_t transformed using the λ provided by Box-Cox and U_t after the log transformation.

```
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
champagne.bc = (1/lambda)*(champagne.train^lambda-1)
champagne.log <- log(champagne.train)
plot.ts(champagne.bc)
```



```r
plot.ts(champagne.log)
```

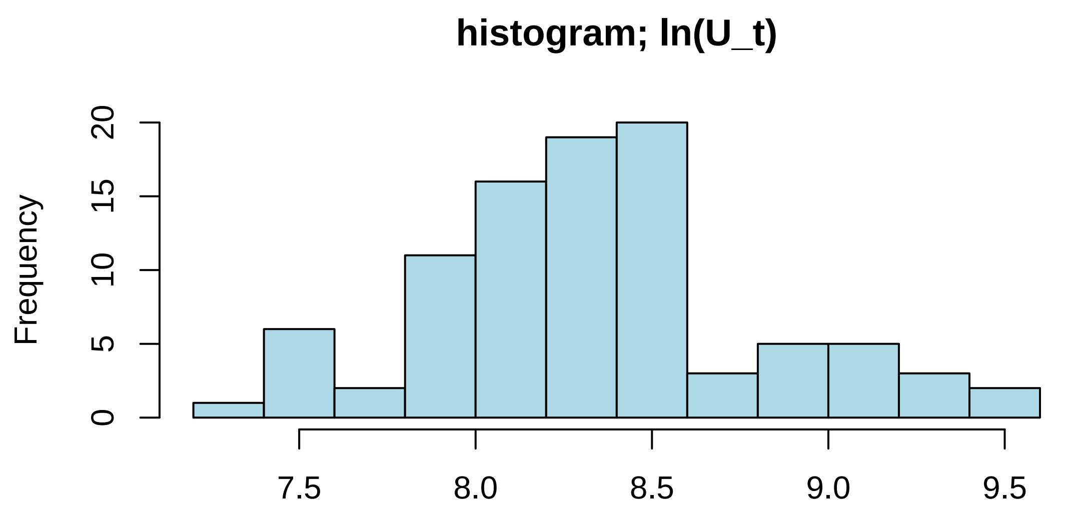


Upon examination, there appears to be minimal difference between the log transform and the Box-Cox transform. Therefore, we opt for the log transformation due to its simplicity and practicality.

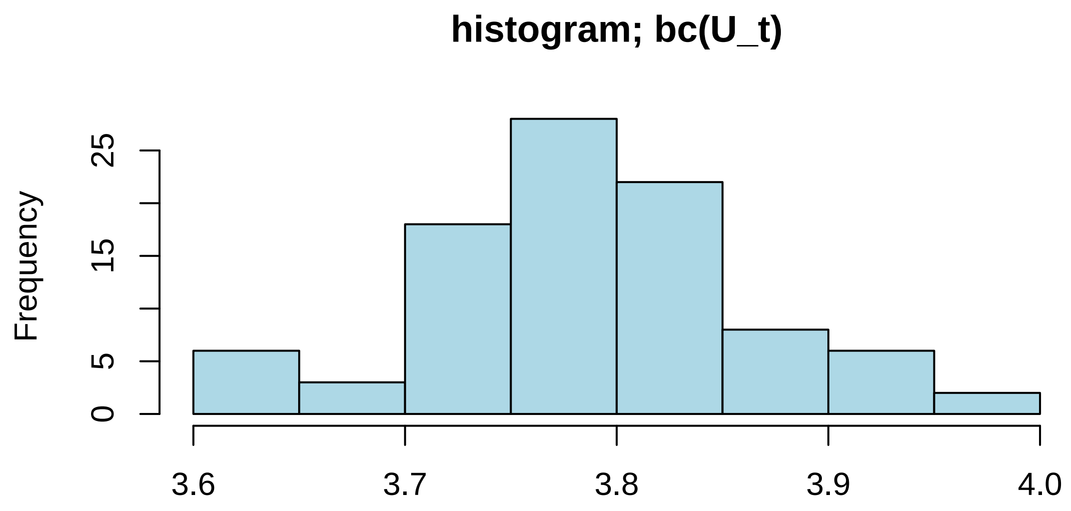
- Compare Histograms of U_t and ln(U_t)

Before we move on, let's take a look at the histograms. The subsequent graphs depict the histogram of Box-Cox(U_t) and histogram of ln(U_t).

```r
hist(champagne.log, col="light blue", xlab="", main="histogram; ln(U_t)")
```



```r
hist(champagne.bc, col="light blue", xlab="", main="histogram; bc(U_t)")
```

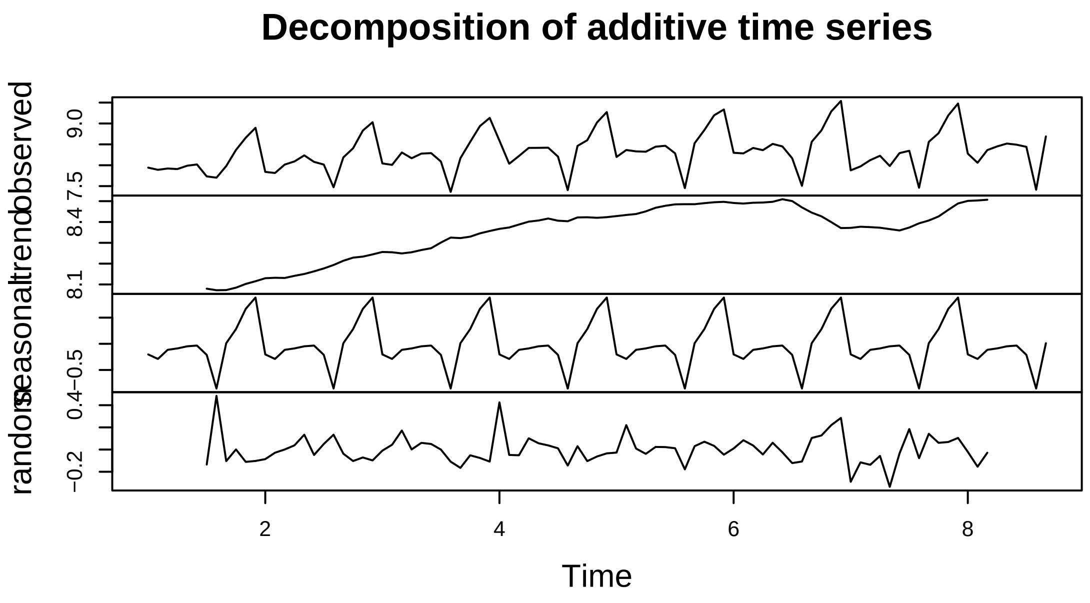


Both histograms seem to exhibit a generally normal distribution.

- Decomposition of ln(U_t)

Following the transformations, we conduct a time series decomposition to assess the components. The decomposition allows us to visualize and understand the individual components within the time series.

```r
library(ggplot2)
library(ggfortify)
#install.packages("ggplot2")
#install.packages("ggfortify")
y <- ts(as.ts(champagne.log), frequency = 12)
decomp <- decompose(y)
plot(decomp)
```



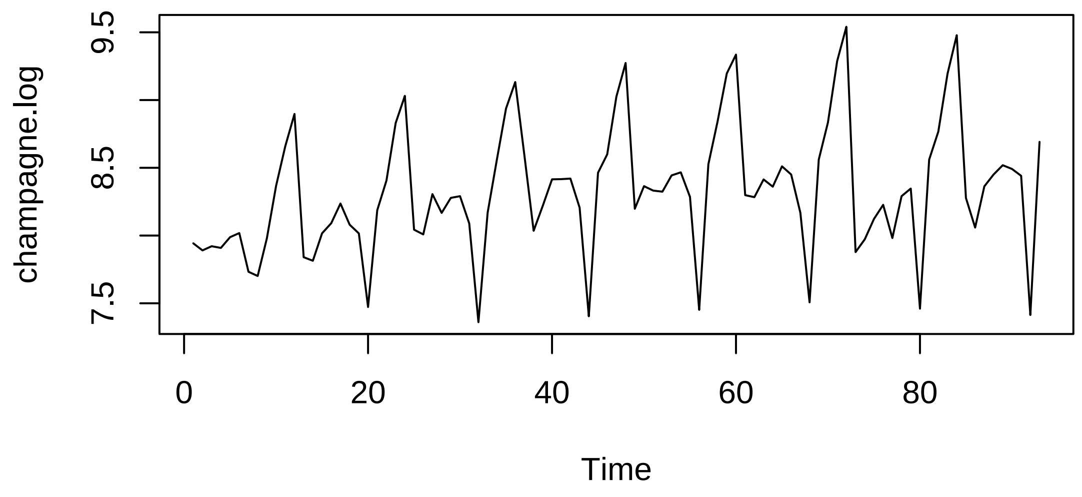
From the decomposition chart, we can assume a non-Gaussian distribution. The effects on non-Gaussian distribution include potential challenges in accurately capturing the statistical properties of the data and may impact the performance of certain modeling techniques that assume Gaussian distribution. Nevertheless, we proceed to remove seasonality.

- Differencing ln(U_t)

During differencing to eliminate seasonality trend, we carefully monitor the variance.

We start by examining our original data (champagne.log).

```r
plot.ts(champagne.log)
```



```r
var(champagne.log)
```

```
## [1] 0.2219931
```

The original data exhibits seasonality and a trend, with a variance of 0.2219931.

The first difference is taken at a lag of 12 to account for the observed seasonality.

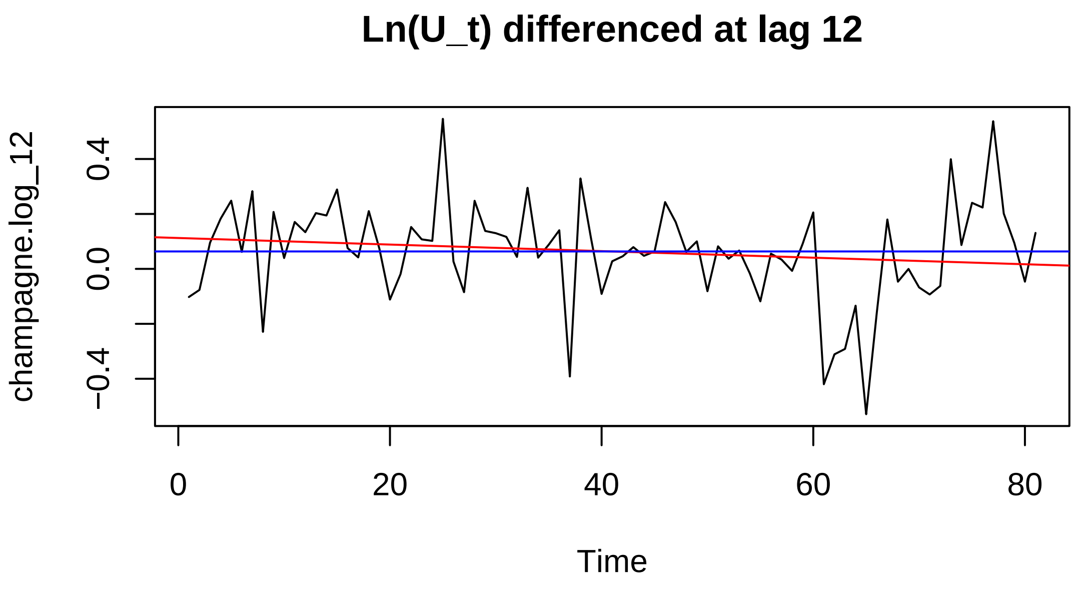
```r
champagne.log_12 <- diff(champagne.log, lag=12)
plot.ts(champagne.log_12, main="Ln(U_t) differenced at lag 12")
var(champagne.log_12)
```

```
## [1] 0.03360596
```

```r
fit <- lm(champagne.log_12 ~ as.numeric(1:length(champagne.log_12))); abline(fit, col="red")
mean(champagne.log_12)
```

```
## [1] 0.0634575
```

```r
abline(h=mean(champagne.log_12), col="blue")
```



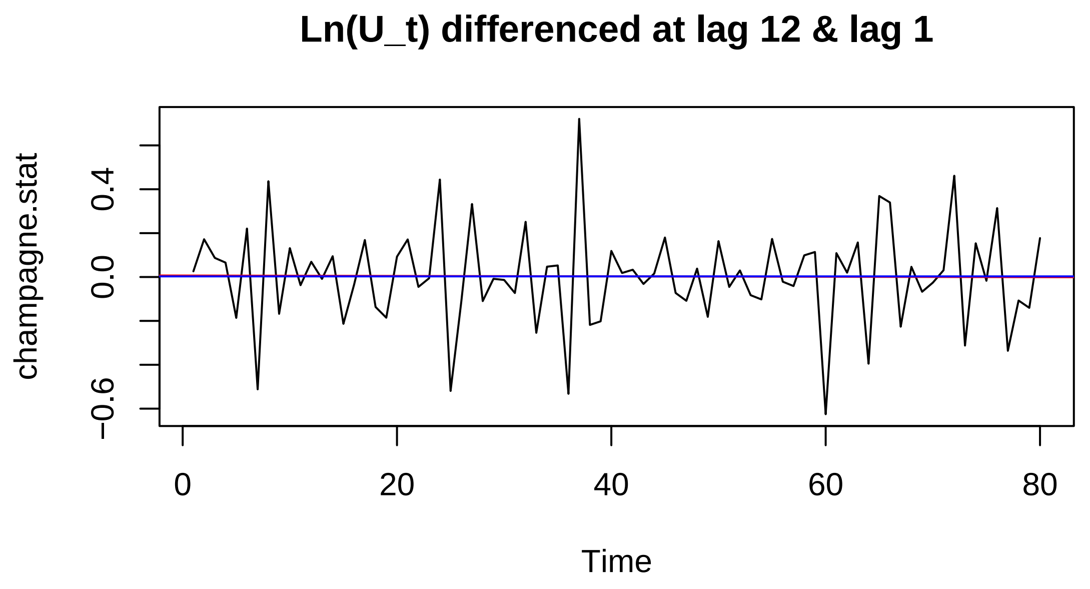
The variance decreases to 0.03360596 after differencing, and seasonality is no longer apparent. However, a trend is still present.

The second difference is taken at a lag of 1 to account for the trend.

```
champagne.stat <- diff(champagne.log_12, lag=1)
plot.ts(champagne.stat, main="Ln(U_t) differenced at lag 12 & lag 1")
fit <- lm(champagne.stat ~ as.numeric(1:length(champagne.stat))); abline(fit, col="red")
mean(champagne.stat)
```

```
## [1] 0.002915936
```

```
abline(h=mean(champagne.stat), col="blue")
```



```r
var(champagne.stat)
```

```
## [1] 0.05247805
```

The variance decreases to 0.05247805 after differencing, seasonality remains not apparent, and now, the trend is not apparent.

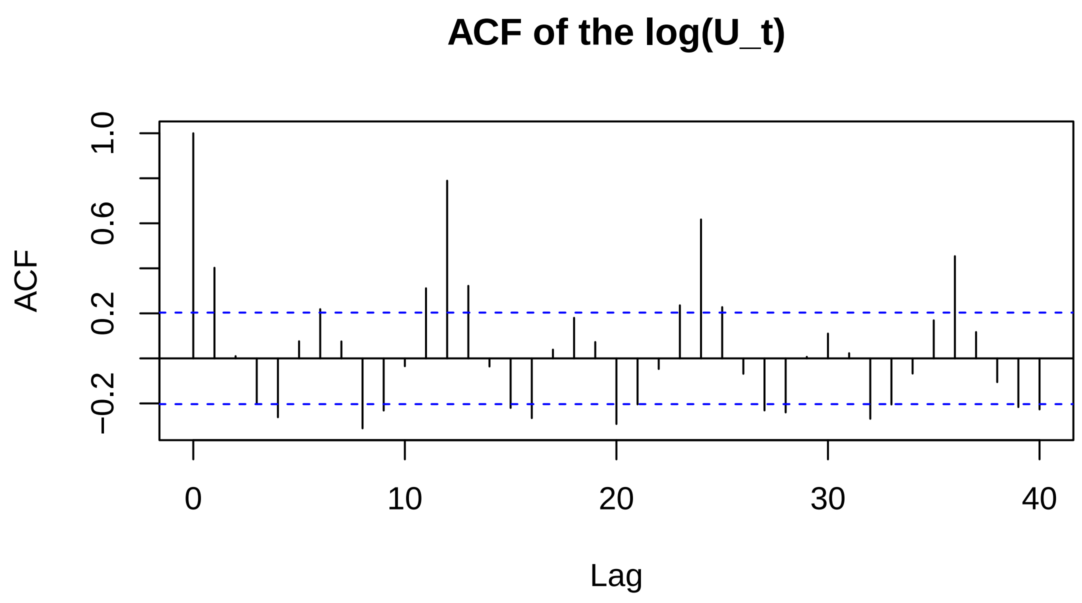
Although the data looks good, we are going to check ACF.

Plot ACF and PACF

To further assess the stationarity and identify any lingering autocorrelation in the transformed and differenced time series, we examine the autocorrelation function (ACF). The ACF provides insights into the correlation between each observation and its lagged values.

The first plot illustrates the ACF of the log-transformed series (champagne.log). This analysis helps us understand the autocorrelation structure of the original log-transformed data.

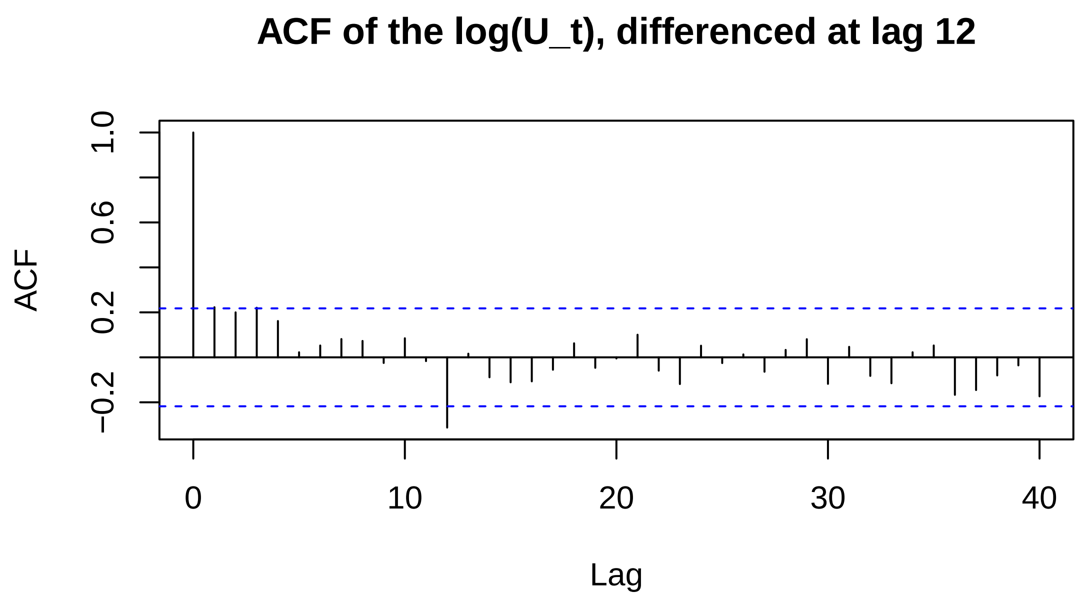
```r
acf(champagne.log, lag.max=40, main="ACF of the log(U_t)")
```



In ACF of the log(U_t), seasonality is apparent.

In the second plot, we examine the ACF of the first-differenced log-transformed series at a lag of 12 (champagne.log_12). This step aims to assess the autocorrelation after addressing seasonality through differencing at a lag of 12.

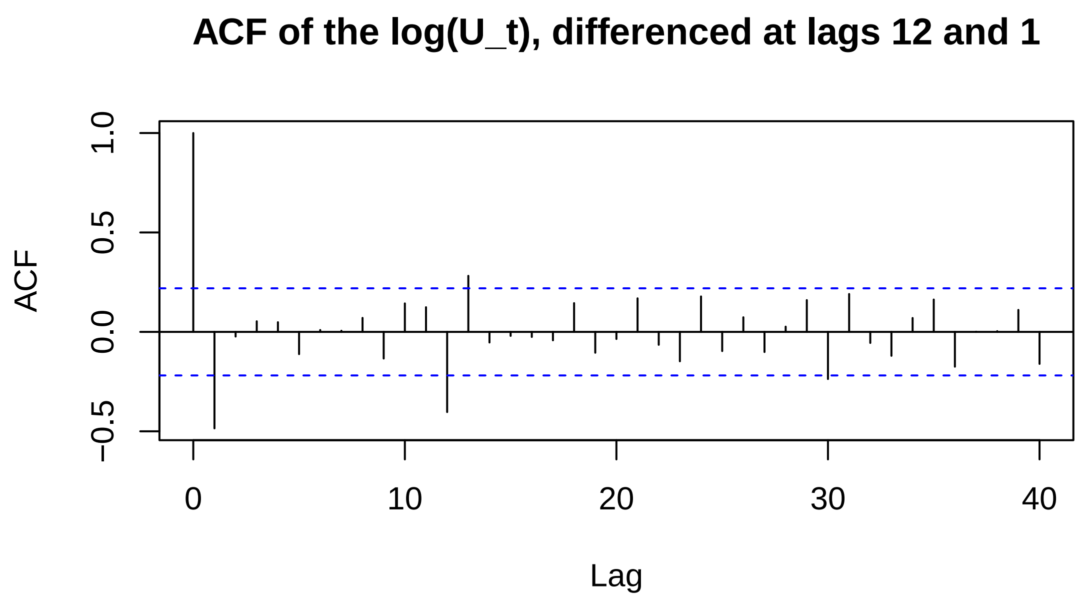
```
acf(champagne.log_12, lag.max=40, main="ACF of the log(U_t), differenced at lag 12")
```



In ACF of the log(U_t) differenced at lag 12, seasonality is no longer apparent and the ACF decay corresponds to a stationary process.

The final plot explores the ACF of the second-differenced log-transformed series at lags 12 and 1 (champagne.stat). This additional differencing at lag 1 aims to capture any residual autocorrelation and ensure the time series is adequately stationary.

```r
acf(champagne.stat, lag.max=40, main="ACF of the log(U_t), differenced at lags 12 and 1")
```



In ACF of the log(U_t) differenced at lag 12 and 1, we confirm that seasonality is not apparent and the ACF decay corresponds to a stationary process.

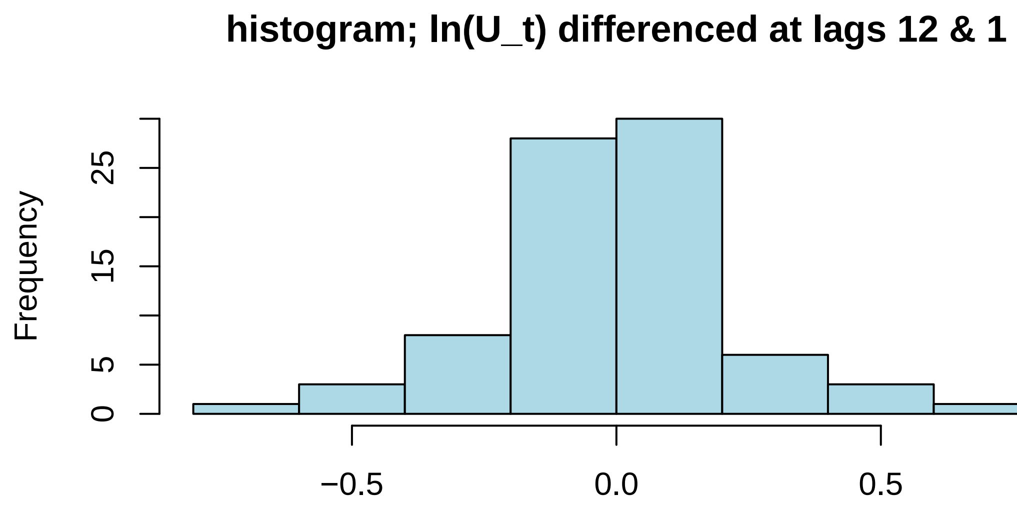
We can come to the conclusion that we can work with data $\nabla_1 \nabla_{12} ln(U_t)$ with U_t representing the first 93 observations of the original data.

- Compare Histograms of ln(U_t)

Let's also take a look at the histograms.

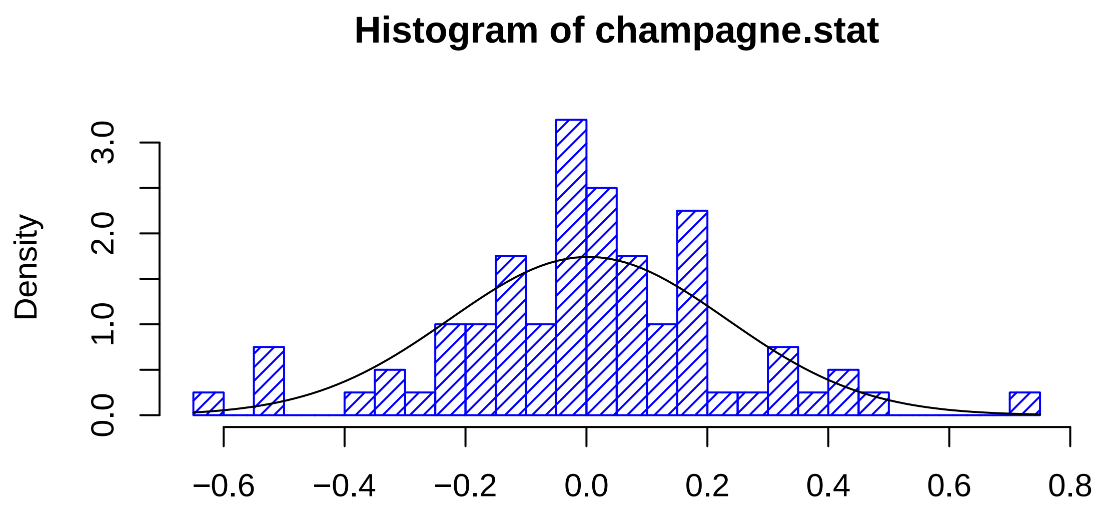
The first histogram is a histogram of $\nabla_1 \nabla_{12} ln(U_t)$. It provides a visual representation of the distribution of the stationary time series.

```r
hist(champagne.stat, col="light blue", xlab="", main="histogram; ln(U_t) differenced at lags 12 & 1")
```



To assess the normality of the stationary series, we overlay a normal distribution curve on the histogram. The blue bars represent the observed data, and the curve represents a theoretical normal distribution based on the calculated mean (m) and standard deviation (std).

```
hist(champagne.stat, density=20, breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(champagne.stat)
std <- sqrt(var(champagne.stat))
curve( dnorm(x,m,std), add=TRUE )
```



This histogram showcases a non-Gaussian pattern. The overlaid curve highlights deviations from the typical symmetric and bell-shaped Gaussian curve. Nevertheless, we move forward.

- ACF and PACF of champagne.stat=$\nabla_1\nabla_{12}ln(U_t)$

Analyzing both ACF and PACF offers a comprehensive understanding of autocorrelation and partial autocorrelation patterns. This insight is vital for parameter selection in forecasting models, like Seasonal AutoRegressive Integrated Moving Average (SARIMA) models. Given the differencing at both lag 12 and lag 1 in our preprocessing, we opt for a SARIMA model.

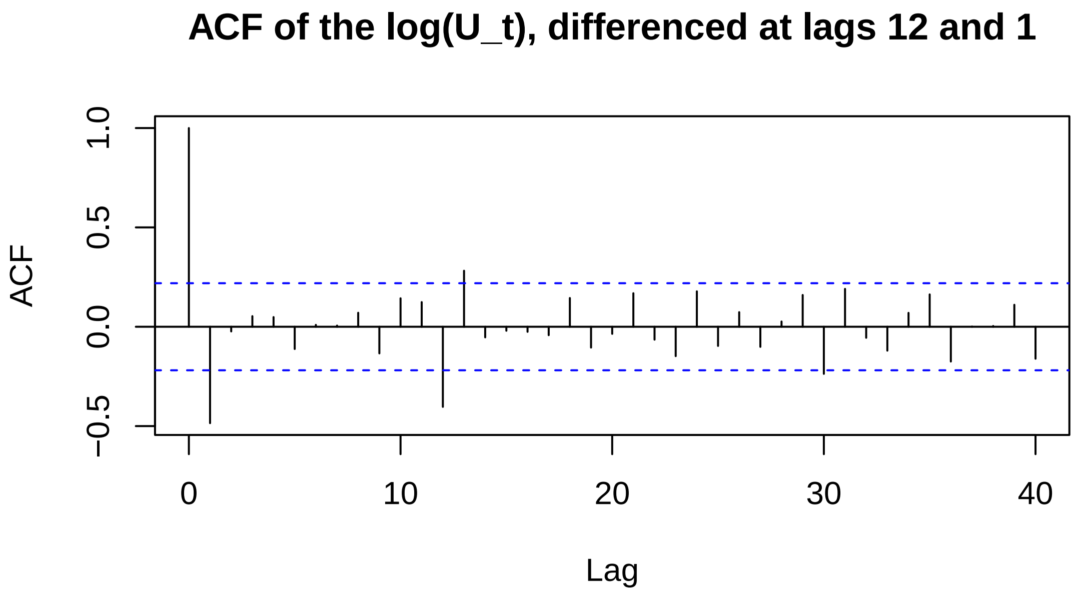
We have our model SARIMA$(p, d, q)(P, D, Q)_s$.

We can assume:

s=12 (the period), D=1 (differenced 1 time at lag 12), d=1 (differenced 1 time at lag 1)

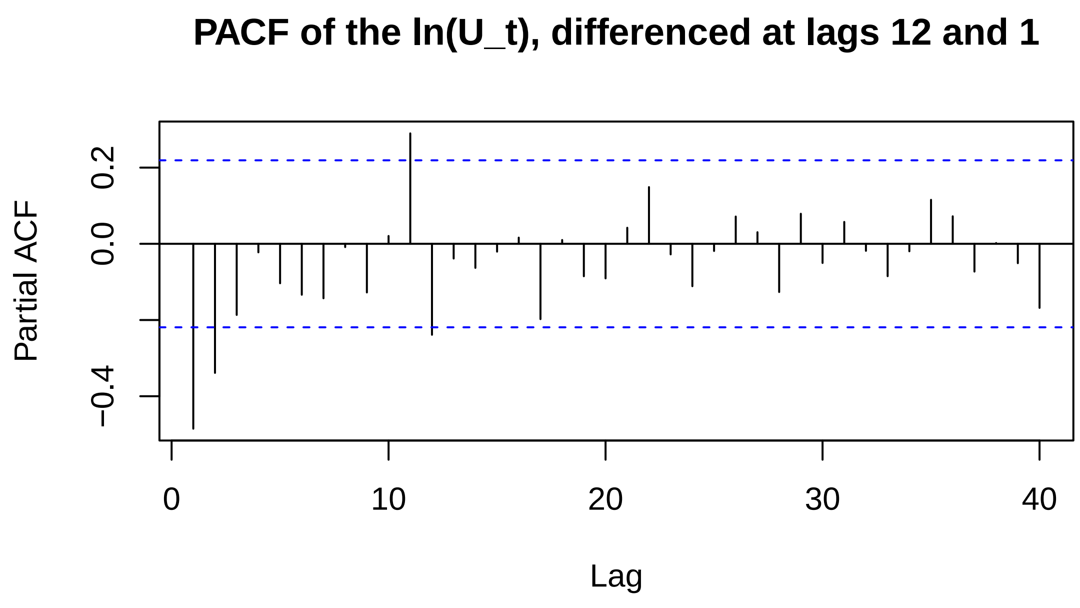
To find q, Q, p, and P, we look at our ACF and PACF.

```r
acf(champagne.stat, lag.max=40, main="ACF of the log(U_t), differenced at lags 12 and 1")
```



From the above, we can identify ACF outside the confidence interval at lag 1, 3, 12, and 30. We can assume Q can be 1 or 2 as ACF lies outside the confidence interval at lag 12, which corresponds to 1, and lag 30 (close to lag 24, a multiple of 12), which corresponds to 2. Because ACF also lies outside the confidence interval at lag 1 and 3, we will consider those for q.

```r
pacf(champagne.stat, lag.max=40, main="PACF of the ln(U_t), differenced at lags 12 and 1")
```



From the above, we can identify PACF outside the confidence interval at lag 1, 2, 11, and 12. We can assume P can be 1 as lag 12 lies outside the confidence interval. Because PACF also lies outside the confidence interval at lag 1 and 2, we will consider those for p.

Therefore, our list of candidate models to try are:

SARIMA for $\ln(U_t)$

s=12, D=1, d=1, Q=1 or 2, q=1 or 3, P=1, p=1 or 2

Fit models

We believe the best model will be SARIMA$(0,1,1)(0,1,1)_{12}$, but let's compare other models and perform diagnostic checking to make sure.

```
#install.packages("qpcR")
library(qpcR)
```

```
## Loading required package: minpack.lm

## Loading required package: rgl

## Loading required package: robustbase

## Loading required package: Matrix
```

```
arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML")
```

```
##
## Call:
## arima(x = champagne.log, order = c(0, 1, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ma1      ma2     sma1
##       -0.768  -0.0464  -0.4962
## s.e.   0.109   0.1134   0.1166
##
## sigma^2 estimated as 0.02561:  log likelihood = 30.81,  aic = -53.62
```

```
AICc(arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML"))
```

```
## [1] -53.34819
```

```
arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML")
```

```
##
## Call:
## arima(x = champagne.log, order = c(0, 1, 1), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ma1     sma1
##       -0.7973  -0.4953
## s.e.   0.0860   0.1176
##
## sigma^2 estimated as 0.02569:  log likelihood = 30.72,  aic = -55.45
```

```
AICc(arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML"))
```

```
## [1] -55.31645
```

In model selection, we rely on the Akaike Information Criterion corrected (AICc) for small sample sizes, favoring the model with the lowest AICc. Comparing SARIMA$(0,1,2)(0,1,1)_{12}$ and SARIMA$(0,1,1)(0,1,1)_{12}$, the latter has a lower AICc of -55.31645 compared to -53.34819 for the former. We will further compare these two models to determine the "best" one.

Now, let's write what the models are:

Model (A):

$$\nabla_1 \nabla_{12} ln(U_t) = (1 - 0.768_{(0.109)} B - 0.0464_{(0.1134)} B^2)(1 - 0.4962_{(0.1166)} B^{12}) Z_t$$

$$\hat{\sigma}_Z^2 = 0.02561$$

Model (B):

$$\nabla_1 \nabla_{12} ln(U_t) = (1 - 0.7973_{(0.0860)} B)(1 - 0.4953_{(0.1176)} B^{12}) Z_t$$

$$\hat{\sigma}_Z^2 = 0.02569$$

- Checking Stationarity and Invertibility (Model A and Model B)

To ensure the suitability of the selected models, we need to examine their stationarity and invertibility.

Stationarity:

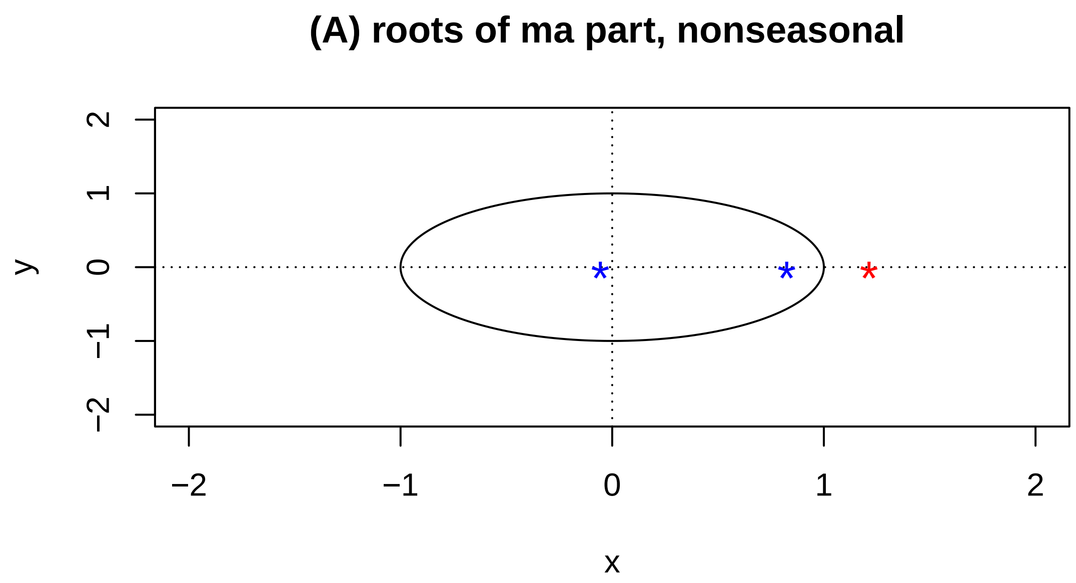
Model (A) is automatically stationary because it is a pure Moving Average model.

Model (B) is automatically stationary because it is a pure Moving Average model.

Invertibility:

Model (A) is invertible because all roots lie outside the unit circle.

```r
source("plot.roots.R")
plot.roots(NULL,polyroot(c(1, -0.768, -0.0464)), main="(A) roots of ma part, nonseasonal ")
```



Model (B) is invertible because $|\theta_1| < 1; |\Theta_1| < 1$

- Diagnostic Checking

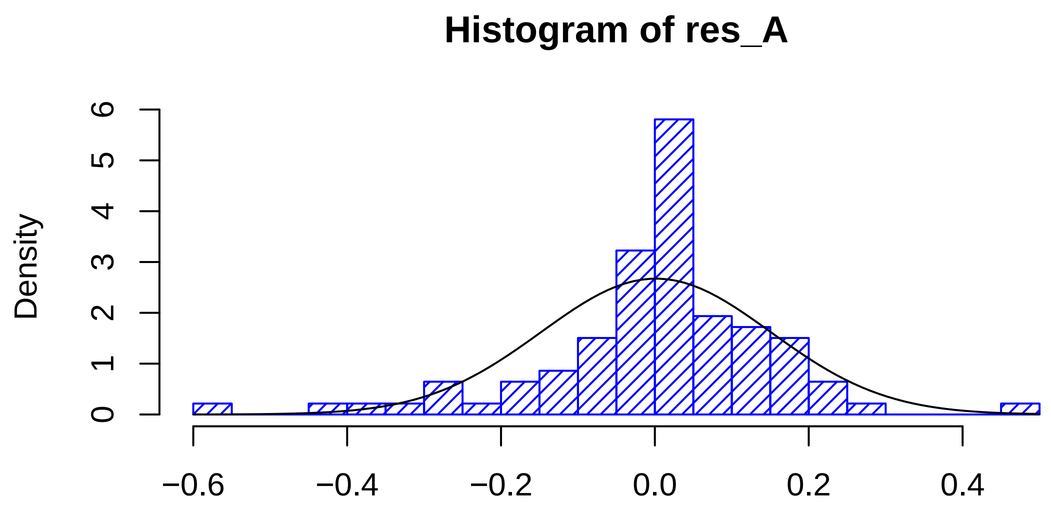
We conduct thorough diagnostic checks to validate model appropriateness. This process ensures adherence to model assumptions and evaluates deviations or patterns in residuals. Diagnostic checks encompass visual inspections via histograms and Q-Q plots, examination of autocorrelation and partial autocorrelation functions, and application of statistical tests.

Model (A):

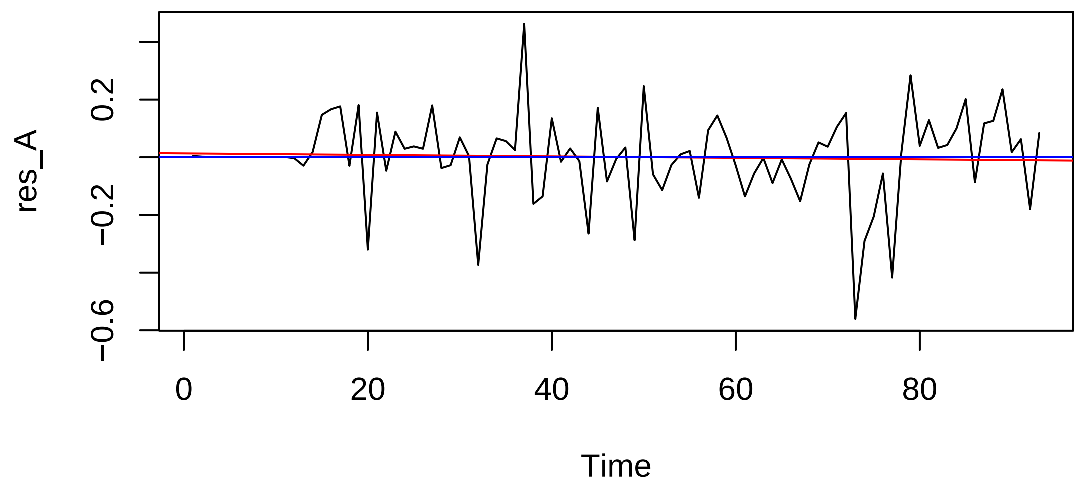
```r
fit_A <- arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="M
res_A <- residuals(fit_A)
hist(res_A,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m_A <- mean(res_A)
m_A
```

```
## [1] 0.001363134
```

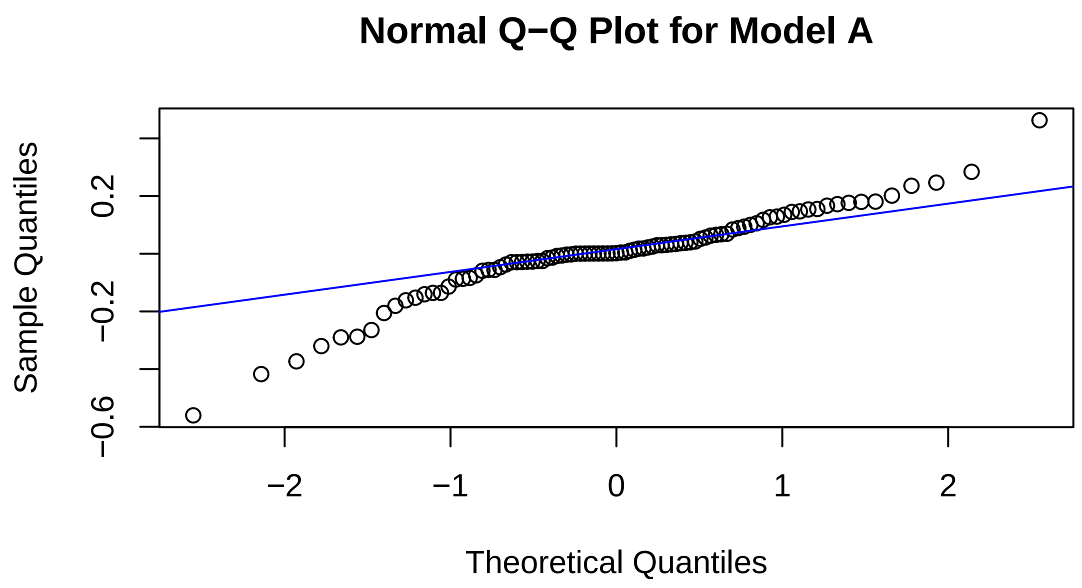
```
std_A <- sqrt(var(res_A))
curve( dnorm(x,m_A,std_A), add=TRUE )
```



```
plot.ts(res_A)
fit_1 <- lm(res_A ~ as.numeric(1:length(res_A))); abline(fit_1, col="red")
abline(h=mean(res_A), col="blue")
```



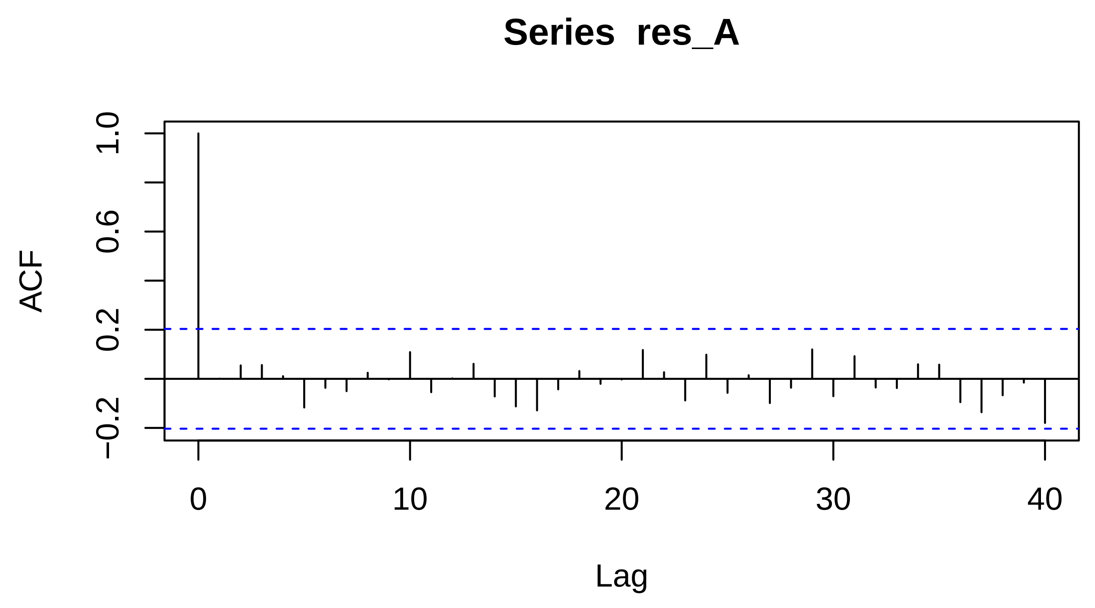
```r
qqnorm(res_A,main= "Normal Q-Q Plot for Model A")
qqline(res_A,col="blue")
```



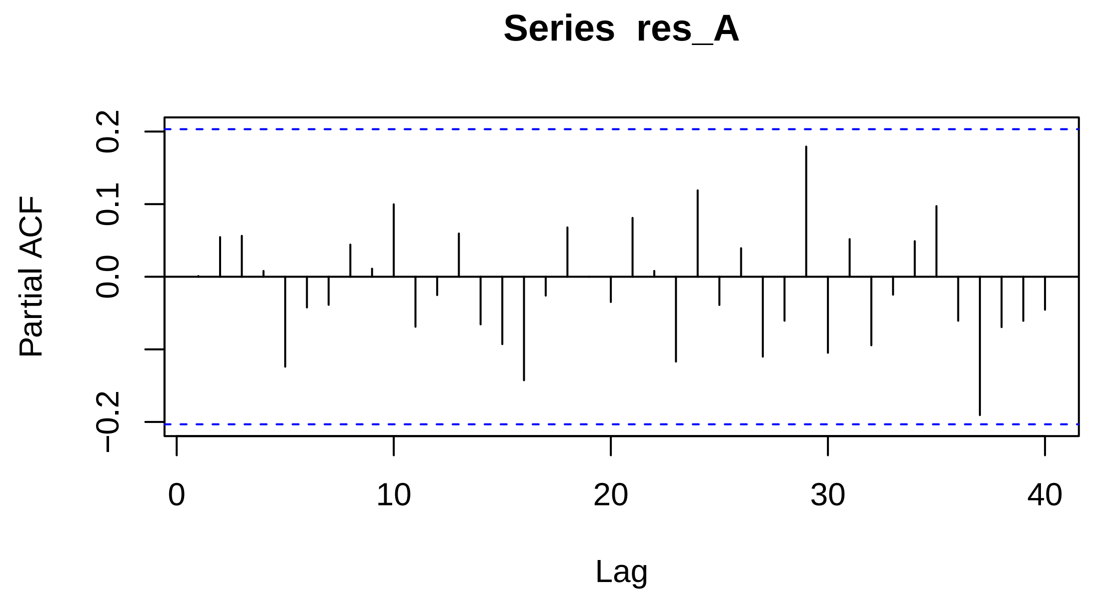
The plots reveal no trend, change of variance, or seasonality. The sample mean is close to zero at 0.001363134. While the histogram and Q-Q plot seem acceptable, they indicate a non-Gaussian distribution, as previously identified.

To ensure that there are no significant patterns in the residual, autocorrelation and partial autocorrelation functions are examined.

```r
acf(res_A, lag.max=40)
```



```r
pacf(res_A, lag.max=40)
```



All ACF and PACF of residuals fall within the confidence intervals.

To quantitatively assess the normality of the residuals, we employed the Shapiro-Wilk normality test.

```r
shapiro.test(res_A)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  res_A
## W = 0.92901, p-value = 8.041e-05
```

Results indicate a statistically significant departure from normality ($p\text{-value} < 0.05$) with a value of 8.041e-05. This confirms our earlier suspicions from the histogram and Q-Q plot that the residuals do not follow a Gaussian distribution.

Subsequently, we performed the Box-Pierce test and Box-Ljung test to evaluate autocorrelation in residuals up to lag 10. The lag value corresponds to approximately the square root of our sample size. The fitdf parameter, indicating the number of coefficients estimated in the model, was set to 3. Additionally, the McLeod-Li test had fitdf set to 0.

```r
Box.test(res_A, lag = 10, type = c("Box-Pierce"), fitdf = 3)
```

```
##
##  Box-Pierce test
##
## data:  res_A
## X-squared = 3.3766, df = 7, p-value = 0.8481
```

```r
Box.test(res_A, lag = 10, type = c("Ljung-Box"), fitdf = 3)
```

```
##
##  Box-Ljung test
##
## data:  res_A
## X-squared = 3.7093, df = 7, p-value = 0.8126
```

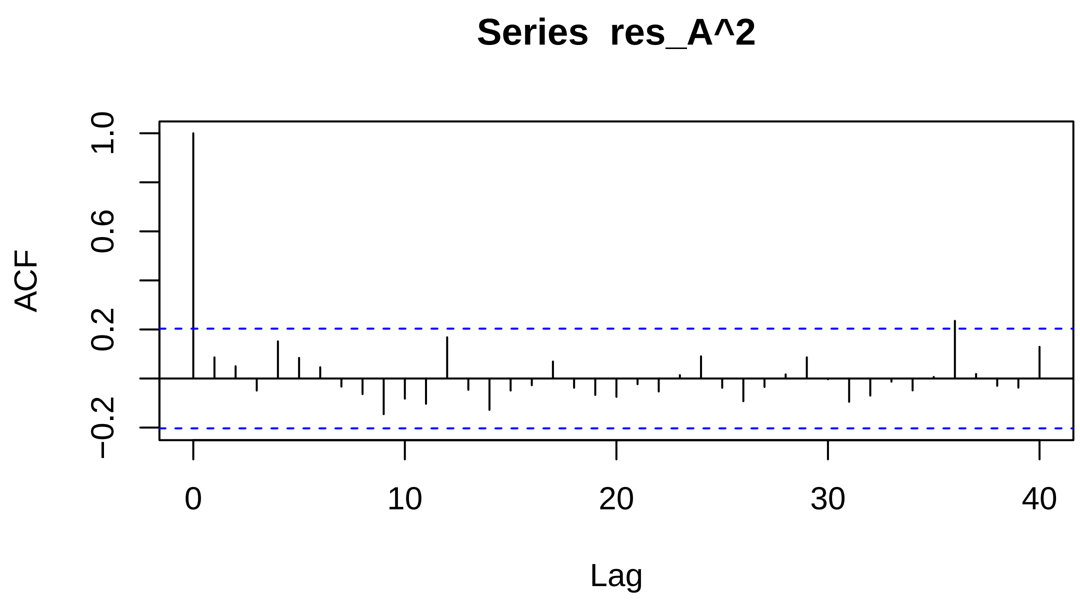
```r
Box.test(res_A^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res_A^2
## X-squared = 7.8867, df = 10, p-value = 0.6399
```

All diagnostic test p-values exceed the 0.05 significance threshold, indicating no significant evidence of autocorrelation in residuals or conditional heteroscedasticity up to lag 10. These findings validate our model assumptions, reinforcing confidence in our modeling approach.

In addition to the previous diagnostic tests, we perform a final check using the Yule-Walker method on the residuals from Model A.

```r
acf(res_A^2, lag.max=40)
```



```
ar(res_A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res_A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.02227
```

The Yule-Walker method automatically selected an autoregressive order of 0 for our residuals. This further strengthens our confidence in the diagnostic checking process for Model A. With all diagnostic tests yielding satisfactory results, Model A passes diagnostic checking and can be used for forecasting purposes.

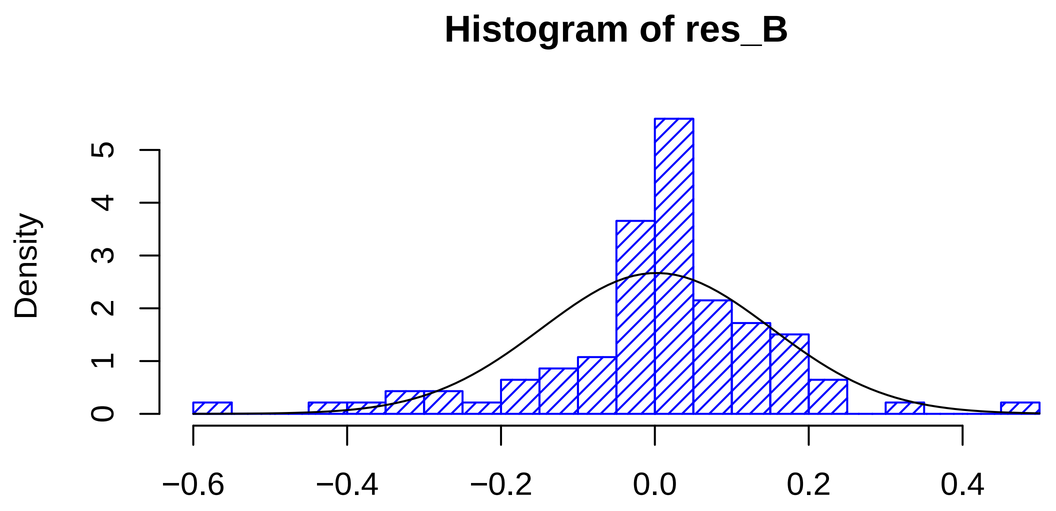
Now, turning our attention to Model B, we initiate a similar diagnostic checking process to ensure its appropriateness for forecasting.

Model (B):

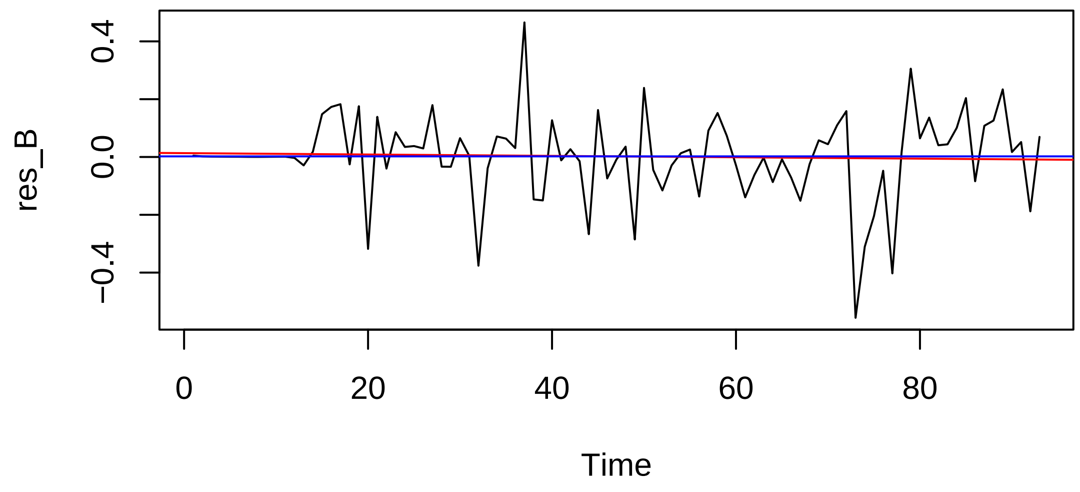
```
fit_B <- arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="MI
res_B <- residuals(fit_B)
hist(res_B,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m_B <- mean(res_B)
m_B
```

```
## [1] 0.002127006
```

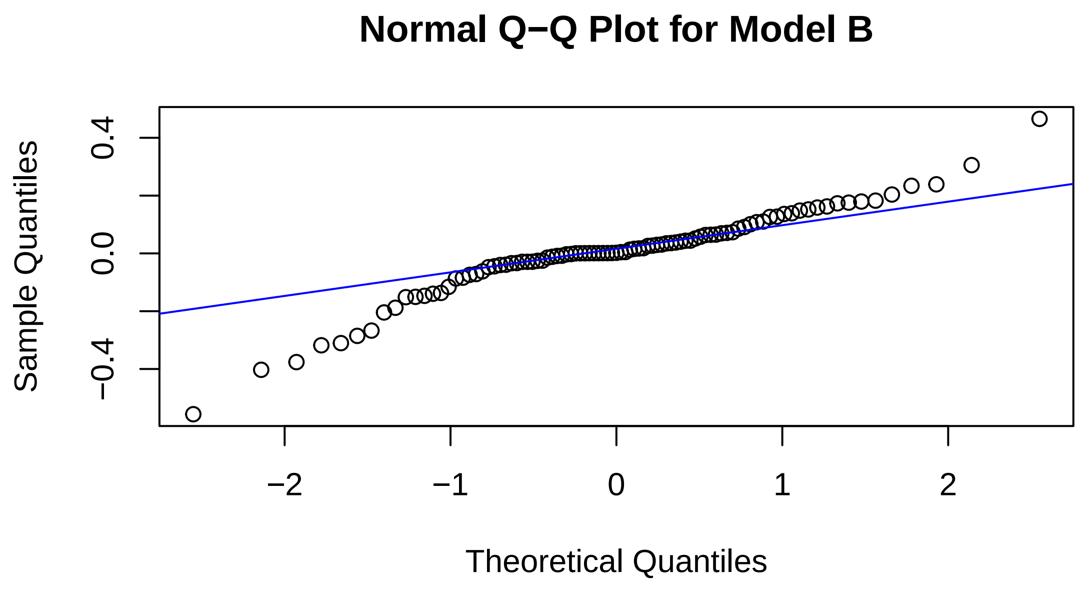
```
std_B <- sqrt(var(res_B))
curve( dnorm(x,m_B,std_B), add=TRUE )
```



```
plot.ts(res_B)
fit_2 <- lm(res_B ~ as.numeric(1:length(res_B))); abline(fit_2, col="red")
abline(h=mean(res_B), col="blue")
```



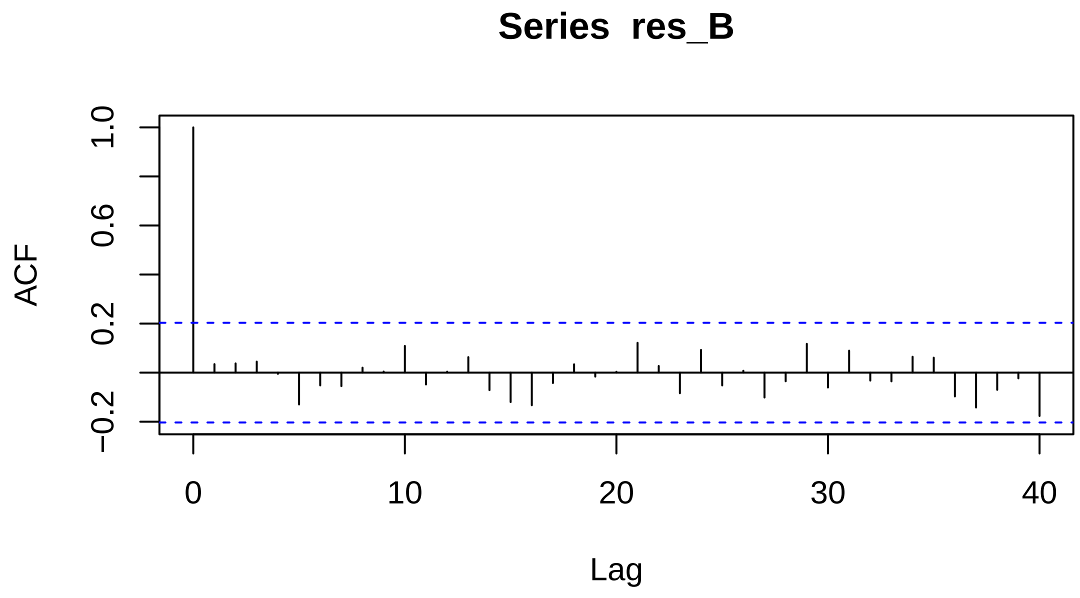
```
qqnorm(res_B,main= "Normal Q-Q Plot for Model B")
qqline(res_B,col="blue")
```



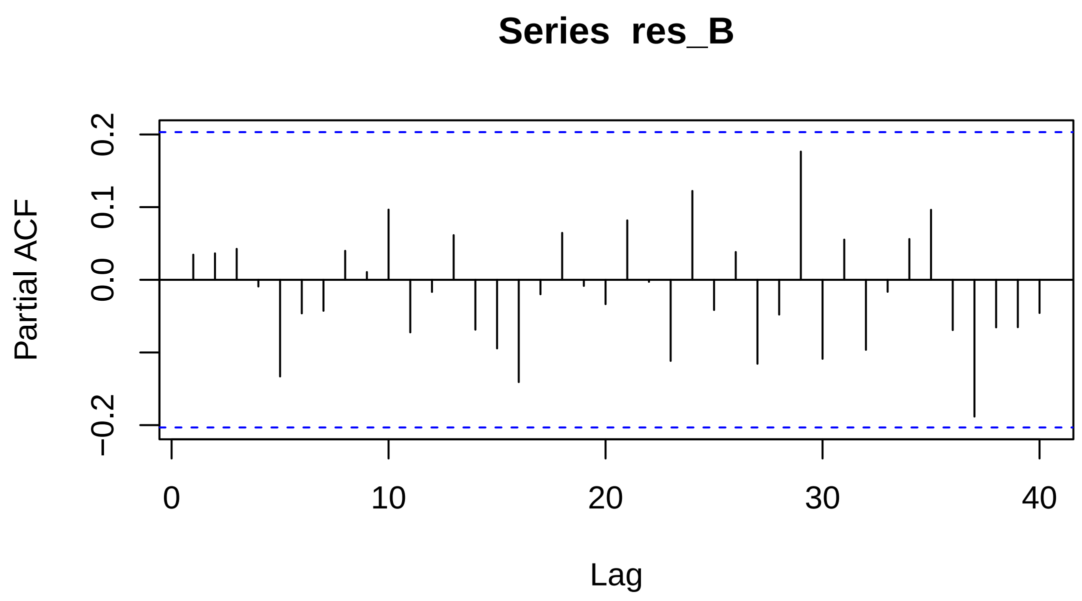
The plots reveal no trend, variance change, or seasonality. The sample mean is close to zero at 0.002127006. While the histogram and Q-Q plot seem acceptable, they do indicate a non-Gaussian distribution, as previously identified.

To ensure that there are no significant patterns in the residual, autocorrelation and partial autocorrelation functions are examined.

```r
acf(res_B, lag.max=40)
```



```r
pacf(res_B, lag.max=40)
```



All ACF and PACF of residuals fall within the confidence intervals.

To quantitatively assess the normality of the residuals, we employed the Shapiro-Wilk normality test.

```r
shapiro.test(res_B)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_B
## W = 0.92917, p-value = 8.2e-05
```

The test results showed a statistically significant departure from normality (p-value $<$ 0.05) with a value of 8.2e-05. This suggests that the residuals do not adhere to a Gaussian distribution, confirming earlier suspicions from the histogram and Q-Q plot.

Following, we conducted the Box-Pierce and Box-Ljung tests to evaluate autocorrelation in residuals up to lag 10. The fitdf parameter, representing the number of coefficients estimated in the model, was set to 2. The fitdf of the McLeod-Li test remains set to 0.

```r
Box.test(res_B, lag = 10, type = c("Box-Pierce"), fitdf = 2)
```

```
##
##  Box-Pierce test
##
## data:  res_B
## X-squared = 3.6798, df = 8, p-value = 0.8848
```

```r
Box.test(res_B, lag = 10, type = c("Ljung-Box"), fitdf = 2)
```

```
##
##  Box-Ljung test
##
## data:  res_B
## X-squared = 4.0417, df = 8, p-value = 0.8533
```

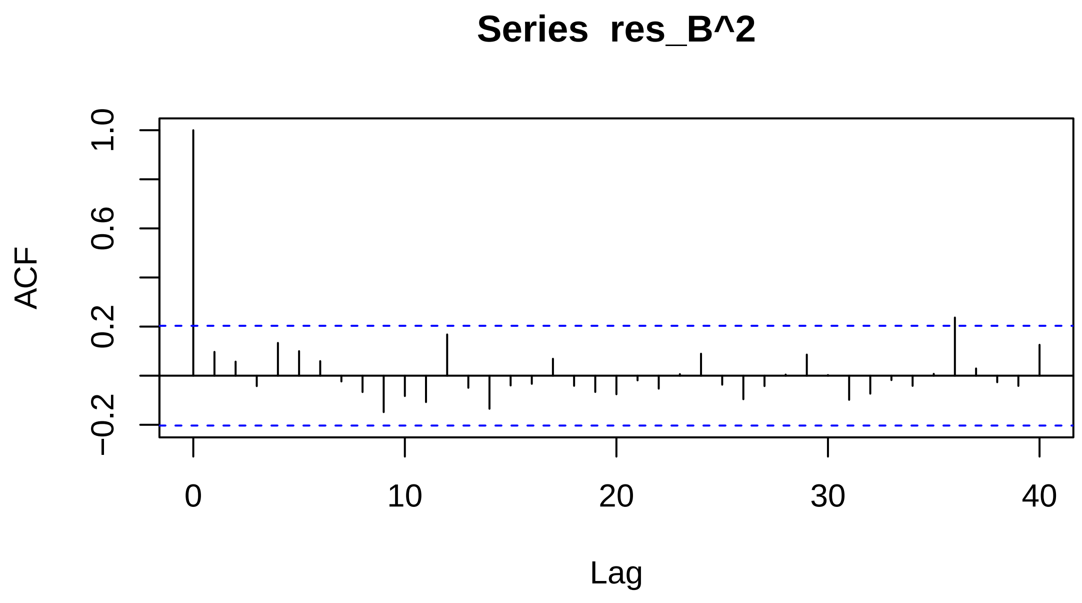
```r
Box.test(res_B^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  res_B^2
## X-squared = 8.0877, df = 10, p-value = 0.6203
```

All diagnostic test p-values surpassed the 0.05 threshold, suggesting no significant evidence of autocorrelation in residuals or conditional heteroscedasticity up to lag 10. These findings validate our model assumptions, instilling confidence in the reliability of our modeling approach.

In addition to the previous diagnostic tests, we perform a final check using the Yule-Walker method on the residuals from Model B.

```r
acf(res_B^2, lag.max=40)
```



```r
ar(res_B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
```

```
## ar(x = res_B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.02234
```

The Yule-Walker method automatically selected an autoregressive order of 0 for our residuals. This further strengthens our confidence in the diagnostic checking process for Model B. With all diagnostic tests yielding satisfactory results, Model B passes diagnostic checking and can be used for forecasting purposes.

Despite the non-Gaussian distribution of residuals, the absence of significant autocorrelation and other patterns ensures the reliability of both models for forecasting purposes.

Due to the comparable performance of Model (A) and Model (B) in diagnostic testing, the selection of the "best" model was based on the AICc. With Model B having the lowest AICc value, we selected Model (B) as the "best" model as for its suitability for forecasting.

Forecasting

Having successfully transformed and modeled the time series data, we proceed with forecasting future values while incorporating confidence intervals.

We fit a Model (B) to the log-transformed data and use it to generate forecasts for the next 12 time points:

```
#install.packages("forecast")
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

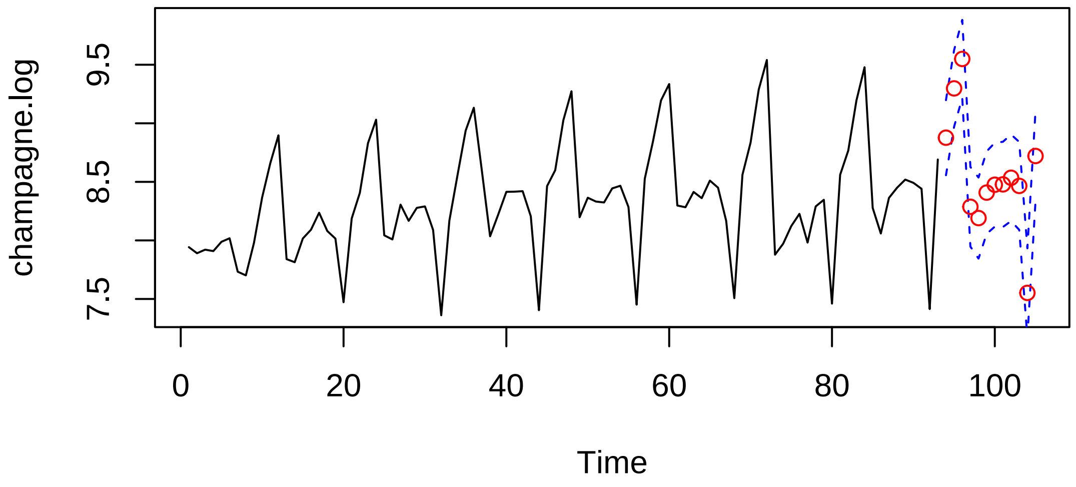
```
## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify
```

```
fit.B <- arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="M
forecast(fit.B)
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
##  94       8.877039 8.671628 9.082449 8.562890  9.191187
##  95       9.298006 9.088419 9.507593 8.977470  9.618541
##  96       9.549466 9.335784 9.763148 9.222668  9.876264
##  97       8.287256 8.069558 8.504954 7.954315  8.620196
##  98        8.190895 7.969252 8.412538 7.851921  8.529868
##  99        8.408384 8.182865 8.633903 8.063483  8.753285
## 100        8.475635 8.246306 8.704964 8.124907  8.826363
## 101        8.478553 8.245476 8.711630 8.122092  8.835013
## 102        8.534957 8.298191 8.771723 8.172855  8.897059
## 103        8.465418 8.225020 8.705815 8.097761  8.833074
## 104        7.551998 7.308022 7.795974 7.178869  7.925127
## 105        8.720660 8.473158 8.968162 8.342138  9.099182
## 106        8.933806 8.646804 9.220807 8.494875  9.372737
## 107        9.354773 9.061015 9.648531 8.905509  9.804037
## 108        9.606233 9.305870 9.906596 9.146868 10.065599
## 109        8.344023 8.037197 8.650848 7.874774  8.813272
## 110        8.247662 7.934507 8.560816 7.768733  8.726590
## 111        8.465151 8.145793 8.784509 7.976735  8.953568
## 112        8.532402 8.206959 8.857846 8.034679  9.030125
## 113        8.535320 8.203903 8.866737 8.028461  9.042179
## 114        8.591724 8.254439 8.929010 8.075891  9.107557
## 115        8.522185 8.179132 8.865237 7.997531  9.046839
## 116        7.608766 7.260041 7.957490 7.075437  8.142094
## 117        8.777427 8.423121 9.131733 8.235562  9.319291
```

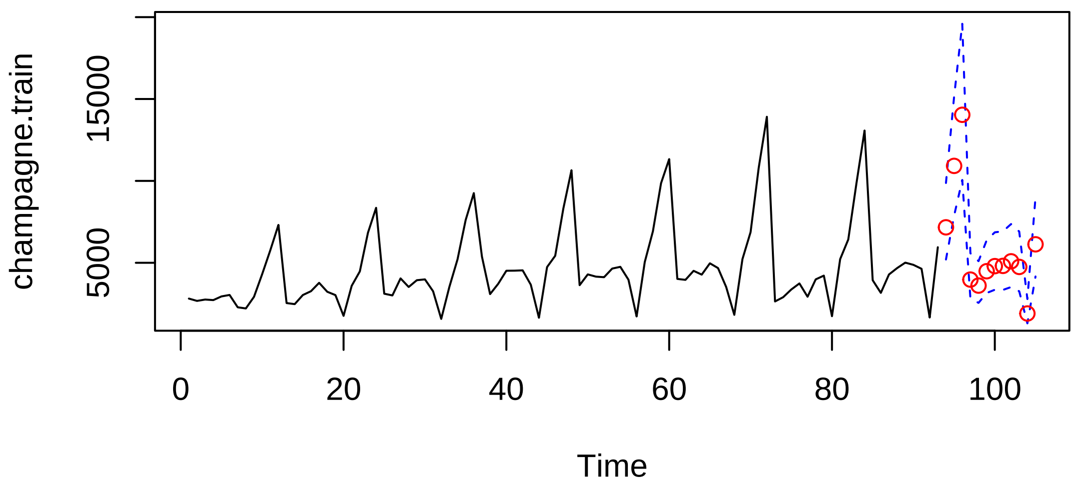
To produce graph with 12 forecasts on transformed data:

```r
pred.tr <- predict(fit.B, n.ahead = 12)
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se
ts.plot(champagne.log, xlim=c(1,length(champagne.log)+12), ylim = c(min(champagne.log),max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(champagne.log)+1):(length(champagne.log)+12), pred.tr$pred, col="red")
```



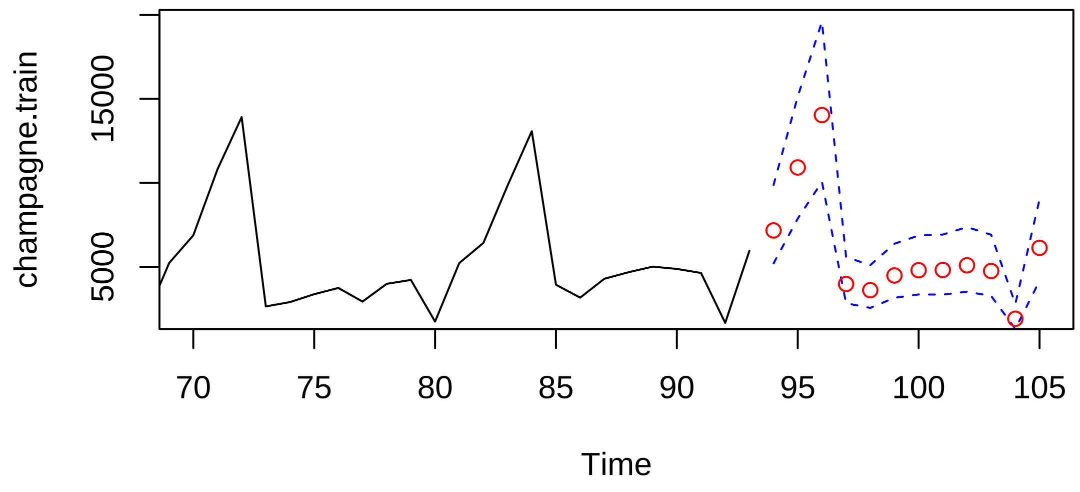
To produce graph with forecasts on original data:

```
pred.orig <- exp(pred.tr$pred)
U= exp(U.tr)
L= exp(L.tr)
ts.plot(champagne.train, xlim=c(1,length(champagne.train)+12), ylim = c(min(champagne.train),max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="red")
```



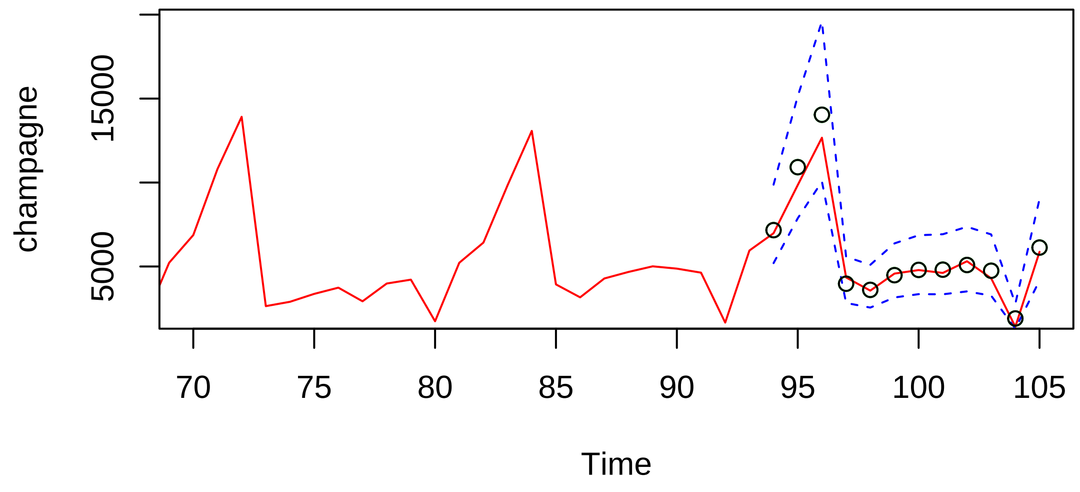
To zoom the graph, starting from entry 70:

```
ts.plot(champagne.train, xlim = c(70,length(champagne.train)+12), ylim = c(2000,max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="red")
```



To plot zoomed forecasts and true values:

```
ts.plot(champagne, xlim = c(70,length(champagne.train)+12), ylim = c(2000,max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="green")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="black")
```



The red line represents the values from the test set. They all fall within the confidence interval and closely align with the forecasted values, which is denoted by the black circles.

Conclusion

In conclusion, this project aimed to analyze and forecast champagne sales using time series analysis. We successfully achieved our goals by transforming the data, selecting a suitable model, and conducting thorough diagnostic checks.

We selected Model (B): SARIMA$(0,1,1)(0,1,1)_{12}$,

$$\nabla_1\nabla_{12}ln(U_t) = (1 - 0.7973_{(0.0860)}B)(1 - 0.4953_{(0.1176)}B^{12})Z_t$$

Model (B) demonstrated its effectiveness in forecasting future values with the incorporation of confidence intervals. The diagnostic checks confirmed the model's stationarity, invertibility, and adherence to key assumptions. Despite the non-Gaussian distribution of residuals, the absence of significant autocorrelation and other patterns validated the reliability of our forecasting model. Additionally, I would like to express my gratitude to Dr. Raya Feldman who provided valuable insights and assistance throughout the project. Overall, the forecasting model provides a robust foundation for understanding and predicting champagne sales patterns, with potential applications in business.

References

Agnihotri, P. (2020). French Champagne Dataset. Retrieved December 2023 from https://www.kaggle.com/datasets/piyushagni5/monthly-sales-of-french-champagne/data.

Appendix

```r
# Read champagne sales data from a CSV file
champagne.csv <- read.table("champagne_sales.csv", sep = ",", header=F, skip=1)
head(champagne.csv)

# Plot raw time series data
champagne = champagne.csv[,2] # Extract the sales column from the dataset
fig.dim = c(4, 2) # Set the dimensions
ts_champagne = ts(champagne, start = c(1964,1), frequency = 12)
ts.plot(ts_champagne, main = "Raw Data")

# Plot time series data
plot.ts(champagne)

# Fit a linear regression line
nt = length(champagne)
fit <- lm(champagne ~ as.numeric(1:nt)); abline(fit, col="red")
mean(champagne)
abline(h=mean(champagne), col="blue")

# Split data into training and test sets
champagne.train = champagne[c(1: 93)]
champagne.test = champagne[c(94:105)]

# Plot training set
```

```r
plot.ts(champagne.train)
fit <- lm(champagne.train ~ as.numeric(1:length(champagne.train))); abline(fit, col="red")
abline(h=mean(champagne), col="blue")

# Plot histogram of training set
hist(champagne.train, col="light blue", xlab="", main="histogram; champagne sales data")

# Plot ACF of training set
acf(champagne.train,lag.max=40, main="ACF of the Champagne Sales Data")

# Perform Box-Cox transformation
#install.packages("MASS")
library(MASS)
bcTransform <- boxcox(champagne.train~ as.numeric(1:length(champagne.train)))
bcTransform$x[which(bcTransform$y == max(bcTransform$y))]

# Extract lambda parameter
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
champagne.bc = (1/lambda)*(champagne.train^lambda-1)
champagne.log <- log(champagne.train)

# Plot Box-Cox and log-transformed data
plot.ts(champagne.bc)
plot.ts(champagne.log)

# Plot histograms
hist(champagne.log, col="light blue", xlab="", main="histogram; ln(U_t)")
hist(champagne.bc, col="light blue", xlab="", main="histogram; bc(U_t)")

# Decompose log-transformed time series
library(ggplot2)
library(ggfortify)
#install.packages("ggplot2")
#install.packages("ggfortify")
y <- ts(as.ts(champagne.log), frequency = 12)
decomp <- decompose(y)
plot(decomp)

# Calculate and plot the variance
plot.ts(champagne.log)
var(champagne.log)

# Differencing at lag 12
champagne.log_12 <- diff(champagne.log, lag=12)
plot.ts(champagne.log_12, main="Ln(U_t) differenced at lag 12")
var(champagne.log_12)
fit <- lm(champagne.log_12 ~ as.numeric(1:length(champagne.log_12))); abline(fit, col="red")
mean(champagne.log_12)
abline(h=mean(champagne.log_12), col="blue")

# Differencing at lag 12 and 1
champagne.stat <- diff(champagne.log_12, lag=1)
plot.ts(champagne.stat, main="Ln(U_t) differenced at lag 12 & lag 1")
```

```
fit <- lm(champagne.stat ~ as.numeric(1:length(champagne.stat))); abline(fit, col="red")
mean(champagne.stat)
abline(h=mean(champagne.stat), col="blue")
var(champagne.stat)

# ACF of log-transformed data
acf(champagne.log, lag.max=40, main="ACF of the log(U_t)")

# ACF of differenced at lag 12
acf(champagne.log_12, lag.max=40, main="ACF of the log(U_t), differenced at lag 12")

# ACF of differenced at lags 12 and 1
acf(champagne.stat, lag.max=40, main="ACF of the log(U_t), differenced at lags 12 and 1")

# Histograms of differenced
hist(champagne.stat, col="light blue", xlab="", main="histogram; ln(U_t) differenced at lags 12 & 1")
hist(champagne.stat, density=20, breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(champagne.stat)
std <- sqrt(var(champagne.stat))
curve( dnorm(x,m,std), add=TRUE )

# ACF of differenced at lags 12 and 1
acf(champagne.stat, lag.max=40, main="ACF of the log(U_t), differenced at lags 12 and 1")

# PACF of differenced at lags 12 and 1
pacf(champagne.stat, lag.max=40, main="PACF of the ln(U_t), differenced at lags 12 and 1")

# Trying models
#install.packages("qpcR")
library(qpcR)
arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML")
AICc(arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="ML"))

arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML")
AICc(arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="ML"))

# To check invertibility of MA part of Model (A)
source("plot.roots.R")
plot.roots(NULL,polyroot(c(1, -0.768, -0.0464)), main="(A) roots of ma part, nonseasonal ")

# Diagnostic checking of Model (A)
fit_A <- arima(champagne.log, order=c(0,1,2), seasonal = list(order = c(0,1,1), period = 12), method="MI
res_A <- residuals(fit_A)
hist(res_A,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m_A <- mean(res_A)
m_A
std_A <- sqrt(var(res_A))
curve( dnorm(x,m_A,std_A), add=TRUE )
plot.ts(res_A)
fit_1 <- lm(res_A ~ as.numeric(1:length(res_A))); abline(fit_1, col="red")
abline(h=mean(res_A), col="blue")
qqnorm(res_A,main= "Normal Q-Q Plot for Model A")
qqline(res_A,col="blue")
```

```
acf(res_A, lag.max=40)
pacf(res_A, lag.max=40)

shapiro.test(res_A)

Box.test(res_A, lag = 10, type = c("Box-Pierce"), fitdf = 3)

Box.test(res_A, lag = 10, type = c("Ljung-Box"), fitdf = 3)

Box.test(res_A^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)

acf(res_A^2, lag.max=40)
ar(res_A, aic = TRUE, order.max = NULL, method = c("yule-walker"))

# Diagnostic checking of Model (B)
fit_B <- arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="M
res_B <- residuals(fit_B)
hist(res_B,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m_B <- mean(res_B)
m_B
std_B <- sqrt(var(res_B))
curve( dnorm(x,m_B,std_B), add=TRUE )
plot.ts(res_B)
fit_2 <- lm(res_B ~ as.numeric(1:length(res_B))); abline(fit_2, col="red")
abline(h=mean(res_B), col="blue")
qqnorm(res_B,main= "Normal Q-Q Plot for Model B")
qqline(res_B,col="blue")

acf(res_B, lag.max=40)
pacf(res_B, lag.max=40)

shapiro.test(res_B)

Box.test(res_B, lag = 10, type = c("Box-Pierce"), fitdf = 2)

Box.test(res_B, lag = 10, type = c("Ljung-Box"), fitdf = 2)

Box.test(res_B^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)

acf(res_B^2, lag.max=40)
ar(res_B, aic = TRUE, order.max = NULL, method = c("yule-walker"))

# Forecasting using Model (B)
#install.packages("forecast")
library(forecast)
fit.B <- arima(champagne.log, order=c(0,1,1), seasonal = list(order = c(0,1,1), period = 12), method="M
forecast(fit.B)

# Produce graph with 12 forecasts on transformed data
pred.tr <- predict(fit.B, n.ahead = 12)
U.tr= pred.tr$pred + 2*pred.tr$se
L.tr= pred.tr$pred - 2*pred.tr$se
ts.plot(champagne.log, xlim=c(1,length(champagne.log)+12), ylim = c(min(champagne.log),max(U.tr)))
```

```r
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(champagne.log)+1):(length(champagne.log)+12), pred.tr$pred, col="red")

# Produce graph with forecasts on original data
pred.orig <- exp(pred.tr$pred)
U= exp(U.tr)
L= exp(L.tr)
ts.plot(champagne.train, xlim=c(1,length(champagne.train)+12), ylim = c(min(champagne.train),max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="red")

# Zoom in on graph, starting from entry 70
ts.plot(champagne.train, xlim = c(70,length(champagne.train)+12), ylim = c(2000,max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="red")

# Plot zoomed forecasts and true values
ts.plot(champagne, xlim = c(70,length(champagne.train)+12), ylim = c(2000,max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="green")
points((length(champagne.train)+1):(length(champagne.train)+12), pred.orig, col="black")
```