Predictive Analytics of French CPI (ARIMA)

Import Package

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(lubridate)
library(tsibble)
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
##
## Attaching package: 'tsibble'
##
## The following object is masked from 'package:lubridate':
##
##     interval
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```r
library(forecast)
```

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

```r
library(tseries)
library(rstudioapi)
library(ggplot2)
library(slider)
library(dplyr)
library(feasts)
```

```
## Loading required package: fabletools
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(urca)
library(fable)
# Load the data
fcpi <- read.csv("fcpi.csv",sep = ';')
head(fcpi)
```

```
##         time      cpi
## 1 1960-01-01 9.797457
## 2 1960-02-01 9.820050
## 3 1960-03-01 9.820050
## 4 1960-04-01 9.835111
## 5 1960-05-01 9.812519
## 6 1960-06-01 9.804988
```

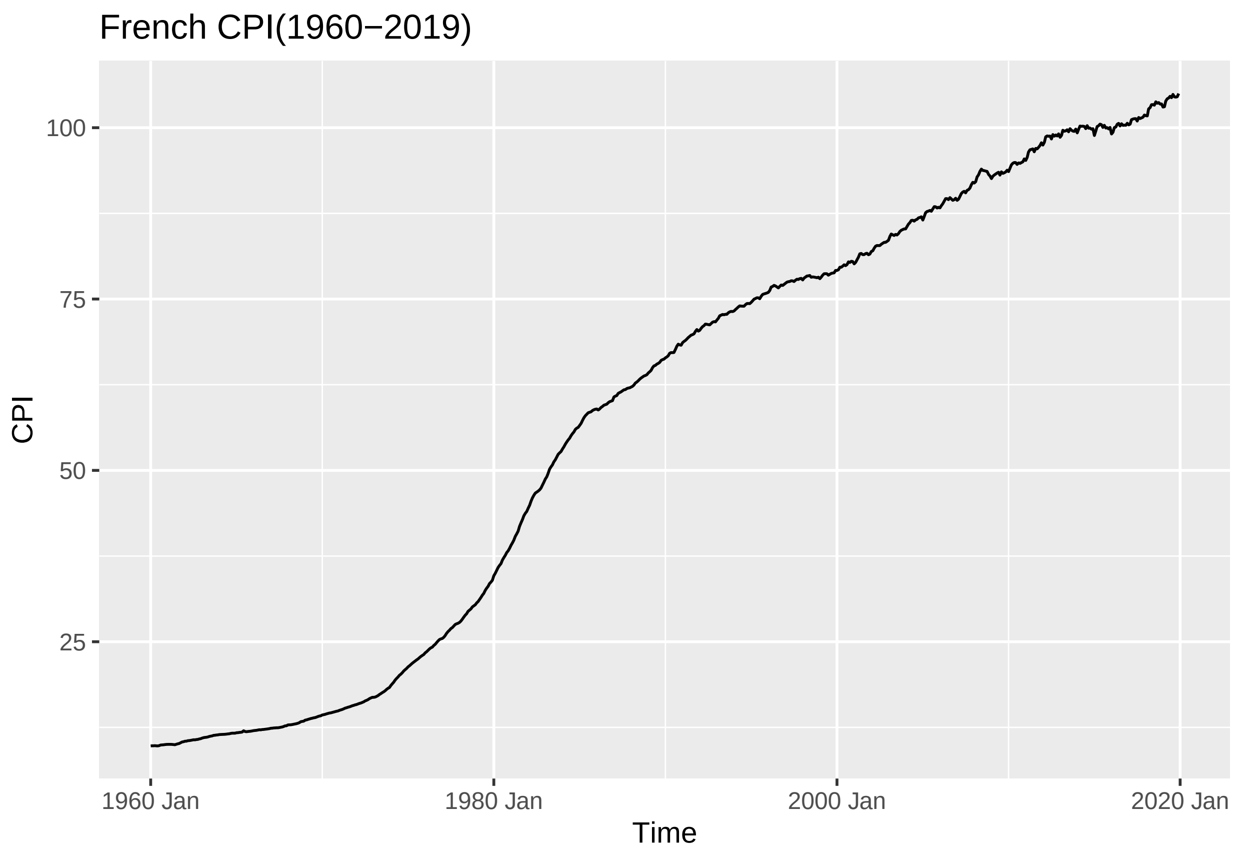
```r
# Convert date format
fcpi$time <- as.Date(fcpi$time, format="%Y-%m-%d")
# Convert to yearmonth for tsibble format
fcpi_tsibble <- fcpi %>%
  mutate(time = yearmonth(time)) %>%
  as_tsibble(index = time)
glimpse(fcpi_tsibble)
```

```
## Rows: 720
## Columns: 2
## $ time <mth> 1960 Jan, 1960 Feb, 1960 Mar, 1960 Apr, 1960 May, 1960 Jun, 1960 ~
## $ cpi  <dbl> 9.797457, 9.820050, 9.820050, 9.835111, 9.812519, 9.804988, 9.842~
```

1. Plot the data and the autocorrelation function.

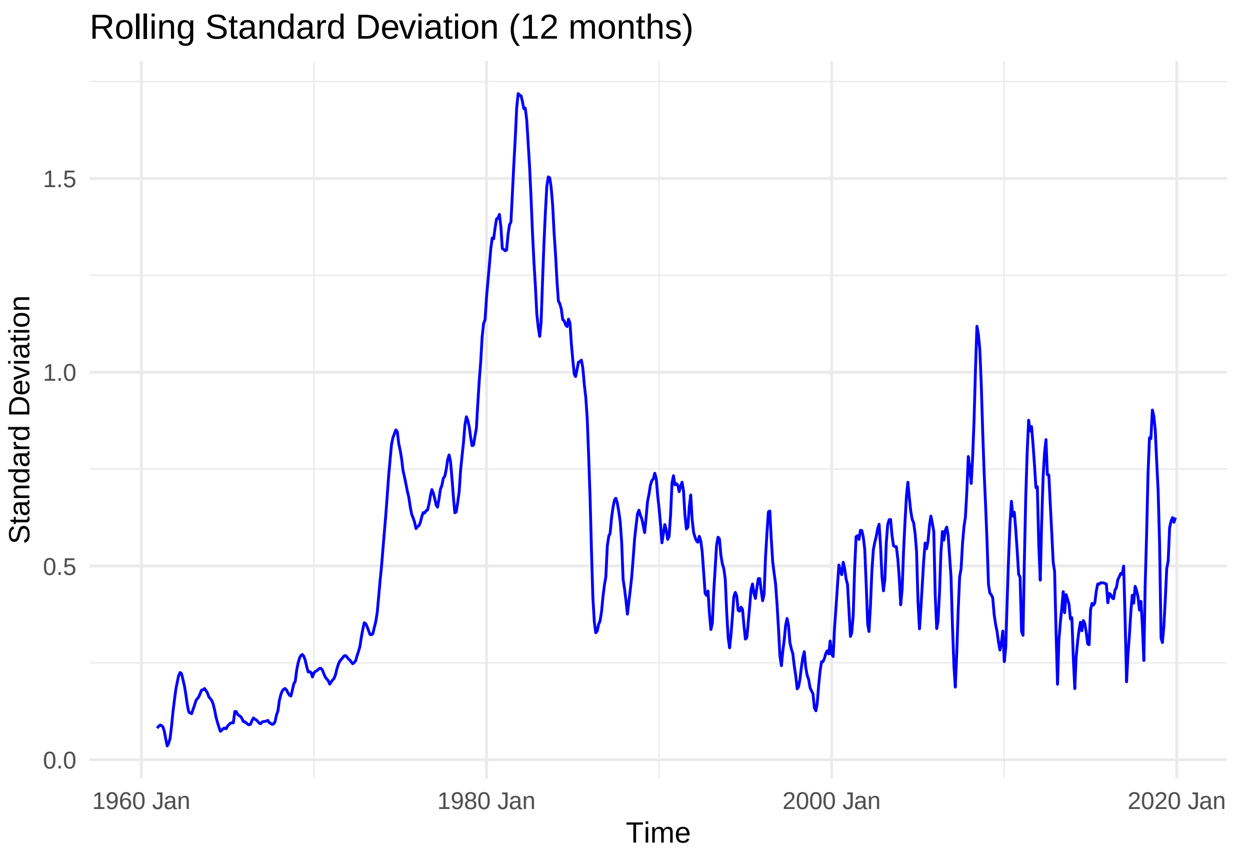
```r
# Plot the time series
autoplot(fcpi_tsibble)+
  ggtitle('French CPI(1960-2019)')+
  xlab("Time") + ylab("CPI")
```

```
## Plot variable not specified, automatically selected `.vars = cpi`
```



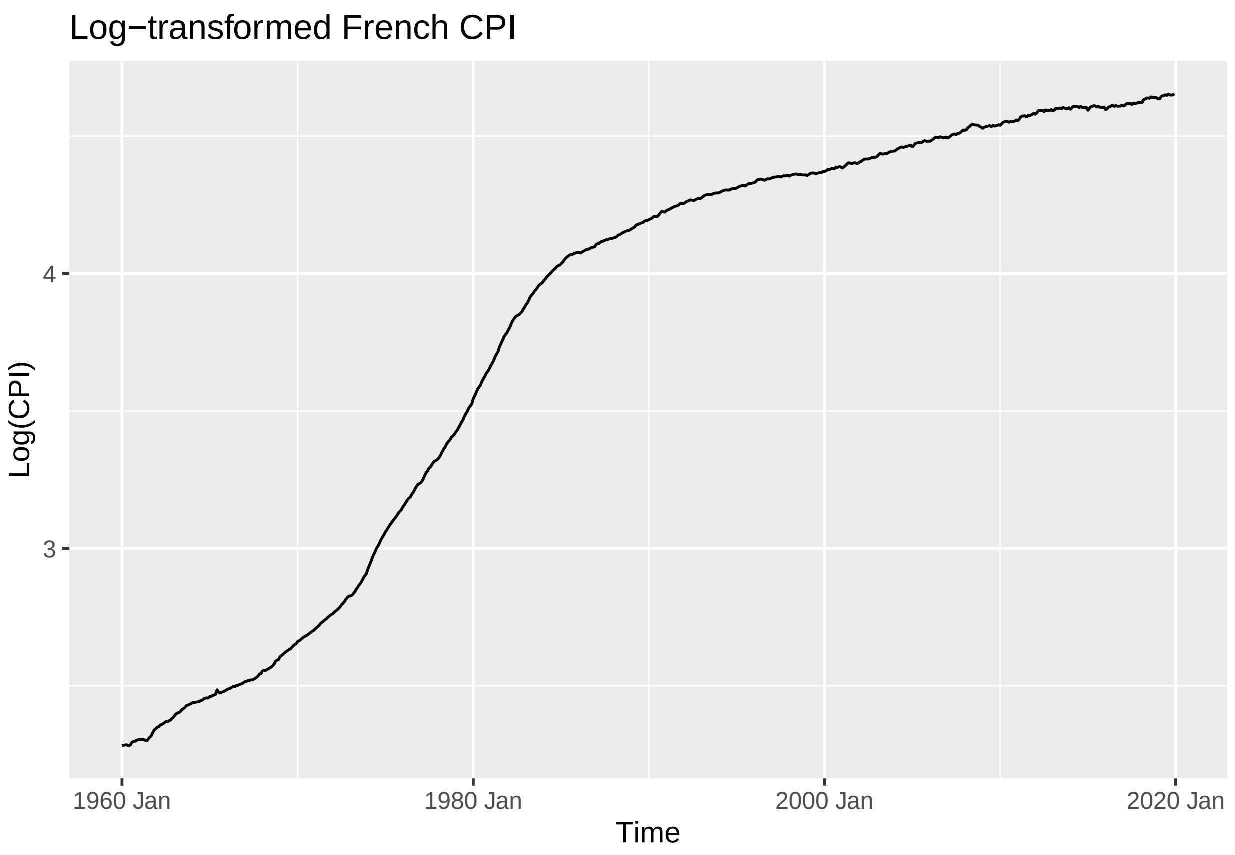
```r
# Plot rolling standard deviation
fcpi_tsibble <- fcpi_tsibble %>%
  mutate(rolling_sd = slide_dbl(cpi, sd, .before = 11, .complete = TRUE))

ggplot(fcpi_tsibble, aes(x = time, y = rolling_sd)) +
  geom_line(color = "blue") +
  labs(title = "Rolling Standard Deviation (12 months)", x = "Time", y = "Standard Deviation") +
  theme_minimal()
```

```
## Warning: Removed 11 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



# 2. Based on the above plot, the variance of the series appears to be heteroscedastic, suggesting that a log transformation is necessary to stabilize the variance.

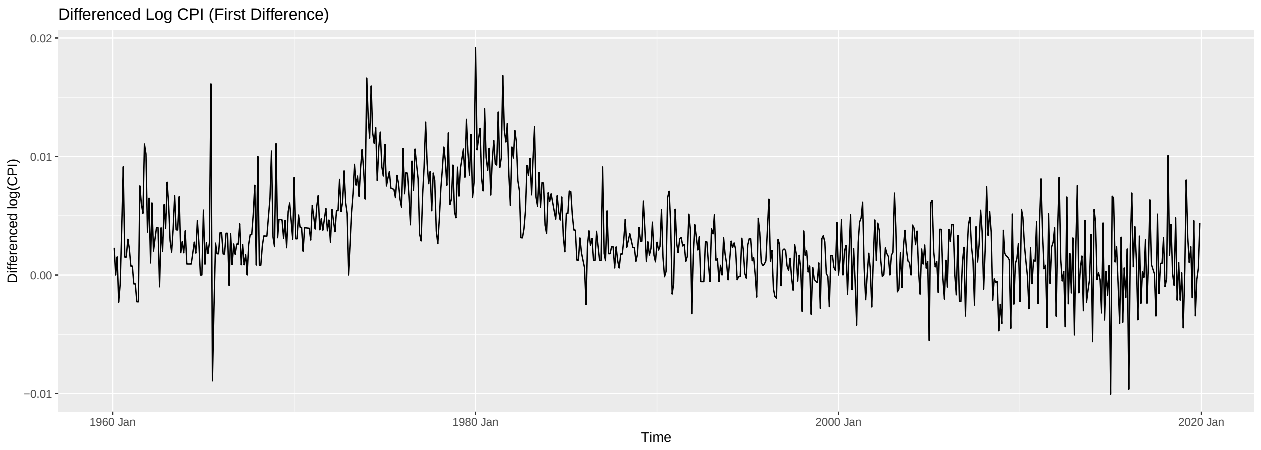
```
# Logarithm
fcpi_tsibble <- fcpi_tsibble %>%
  mutate(fcpi_log=log(cpi))
# Plot the log time series
autoplot(fcpi_tsibble,fcpi_log)+
  ggtitle('Log-transformed French CPI')+
  xlab('Time')+ylab('Log(CPI)')
```



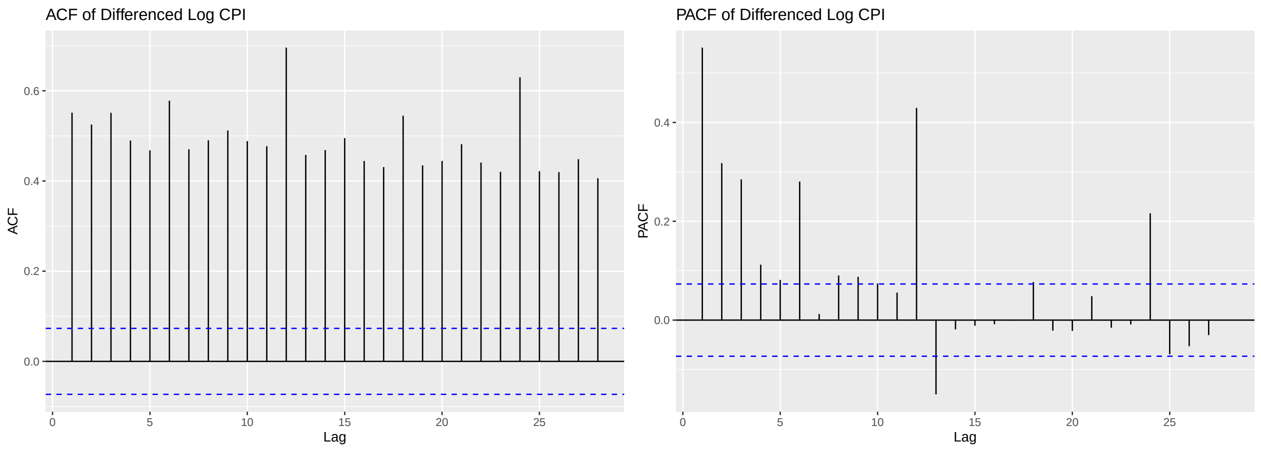
# 3. This series exibits a clear upward trend and is non-stationary, indicating the need for differencing (first differencing).

```
# First-Differencing
fcpi_tsibble <- fcpi_tsibble %>%
  mutate(fcpi_log_diff = difference(fcpi_log, differences = 1))

# Plot differenced time series
autoplot(fcpi_tsibble, fcpi_log_diff) +
  ggtitle("Differenced Log CPI (First Difference)") +
  xlab("Time") + ylab("Differenced log(CPI)")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



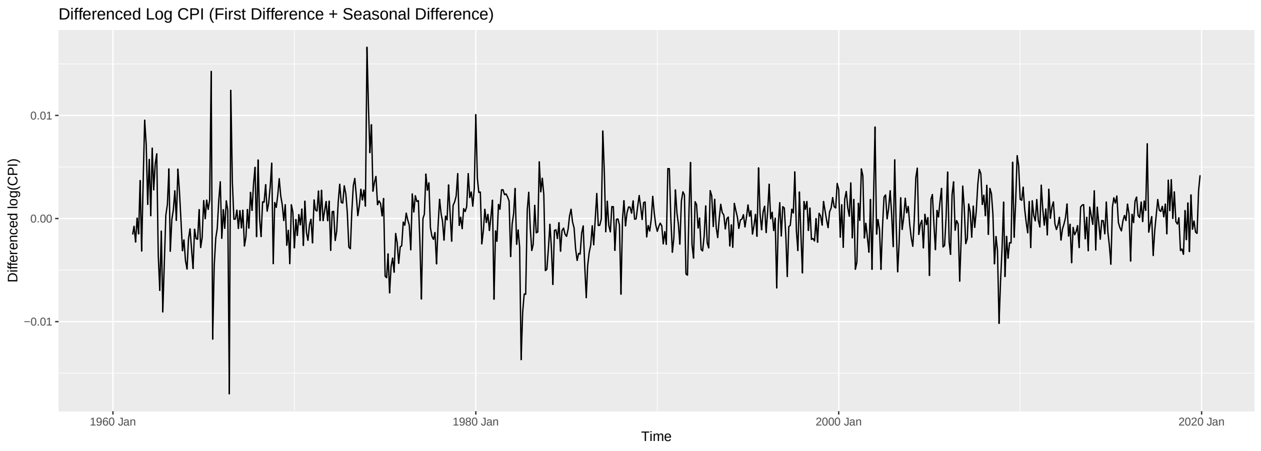
```r
# Plot ACF and PACF
acf_diff_plot <- ggAcf(fcpi_tsibble$fcpi_log_diff) +
  ggtitle("ACF of Differenced Log CPI")

pacf_diff_plot <- ggPacf(fcpi_tsibble$fcpi_log_diff) +
  ggtitle("PACF of Differenced Log CPI")

grid.arrange(acf_diff_plot, pacf_diff_plot, ncol = 2)
```



# 4. As observed in the ACF plot, the series exhibits strong seasonality, as evidenced by significant correlations at the 12-lag and 24-lag intervals. Therefore, seasonal differencing will be applied to address this seasonality.

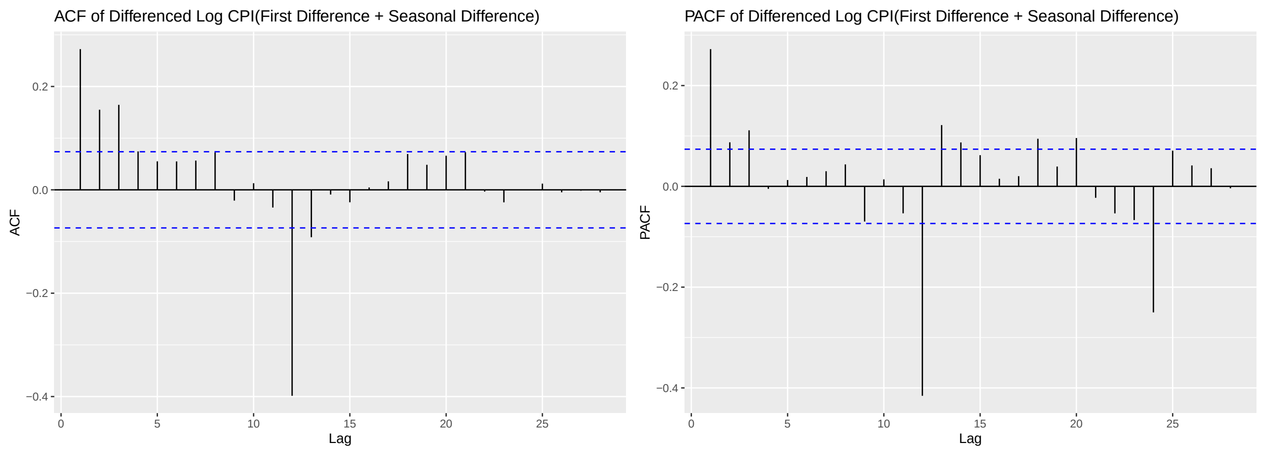
```r
# Seasonal differencing
fcpi_tsibble <- fcpi_tsibble %>%
  mutate(fcpi_log_seasonal_diff = c(rep(NA,12),diff(fcpi_log_diff, lag = 12)))
# Plot the seasonally differenced time series
autoplot(fcpi_tsibble, fcpi_log_seasonal_diff) +
  ggtitle("Differenced Log CPI (First Difference + Seasonal Difference)") +
  xlab("Time") + ylab("Differenced log(CPI)")
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



```r
# Plot ACF and PACF
acf_seasonal_diff_plot <- ggAcf(fcpi_tsibble$fcpi_log_seasonal_diff) +
  ggtitle("ACF of Differenced Log CPI(First Difference + Seasonal Difference)")
```

```
pacf_seasonal_diff_plot <- ggPacf(fcpi_tsibble$fcpi_log_seasonal_diff) +
  ggtitle("PACF of Differenced Log CPI(First Difference + Seasonal Difference)")

grid.arrange(acf_seasonal_diff_plot, pacf_seasonal_diff_plot, ncol = 2)
```



# 5. Now we apply ADF test and KPSS test to assess the stationary of the processed time series.

```
adf_test <- ur.df(na.omit(fcpi_tsibble$fcpi_log_seasonal_diff), type = "drift", selectlags = "AIC")
summary(adf_test)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0173565 -0.0014800  0.0001232  0.0015647  0.0166061
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.750e-06  1.087e-04  -0.025   0.9798
## z.lag.1     -6.629e-01  4.544e-02 -14.589   <2e-16 ***
## z.diff.lag  -8.802e-02  3.765e-02  -2.338   0.0197 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002886 on 702 degrees of freedom
## Multiple R-squared:  0.3678, Adjusted R-squared:  0.366
## F-statistic: 204.2 on 2 and 702 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -14.5889 106.422
##
```

```
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

ADF:

The ADF test checks if a time series is stationary. The null hypothesis is that the series has a unit root (non-stationary), and the alternative is that it is stationary.

Key Results: Test Statistic: -14.5889, which is much smaller than the critical value at the 1% level (-3.43), so we reject the null hypothesis.

p-value: Very small (well below 0.05), confirming the rejection of the null hypothesis.

Conclusion: Since both the test statistic and p-value suggest rejecting the null hypothesis, we conclude that the series is stationary.

```r
kpss_test <- kpss.test(na.omit(fcpi_tsibble$fcpi_log_seasonal_diff), null = "Level")
```

```
## Warning in kpss.test(na.omit(fcpi_tsibble$fcpi_log_seasonal_diff), null =
## "Level"): p-value greater than printed p-value
```

```r
kpss_test
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  na.omit(fcpi_tsibble$fcpi_log_seasonal_diff)
## KPSS Level = 0.12678, Truncation lag parameter = 6, p-value = 0.1
```

KPSS:

The KPSS test checks for stationarity around a level. The null hypothesis is that the series is stationary, and the alternative is that it is non-stationary.

Test Results: KPSS statistic: 0.12678

p-value: 0.1

At the 5% significance level, we fail to reject the null hypothesis.

Conclusion: The series is stationary.

6. Next, fit SARIMA models. The PACF plot shows a clear cutoff after lag 3 in the non-seasonal component, suggesting a potential value of p = 3. In the seasonal component, significant spikes are observed at lags 12 and 24, indicating that P = 2 may be appropriate. The ACF plot exhibits a gradual decay in the non-seasonal part, and thus is not used for identifying q. However, a prominent spike at lag 12 in the seasonal ACF suggests that Q = 1 could be a suitable choice. Consequently, this initial analysis suggests that a possible model for these data is an SARIMA(3,1,0)(2,1,1)[12]. I fit this model, along with some variations on it, and compute the AIC, BIC and AICc values.

```
# Fit several SARIMA models

model_result <- fcpi_tsibble %>%
  model(
    sarima_310_211 = ARIMA(fcpi_log ~ pdq(3,1,0) + PDQ(2,1,1,period=12)+0),
    sarima_311_211 = ARIMA(fcpi_log ~ pdq(3,1,1) + PDQ(2,1,1,period=12)+0),
    sarima_312_211 = ARIMA(fcpi_log ~ pdq(3,1,2) + PDQ(2,1,1,period=12)+0),
    sarima_310_111 = ARIMA(fcpi_log ~ pdq(3,1,0) + PDQ(1,1,1,period=12)+0),
    sarima_311_111 = ARIMA(fcpi_log ~ pdq(3,1,1) + PDQ(1,1,1,period=12)+0),
    sarima_312_111 = ARIMA(fcpi_log ~ pdq(3,1,2) + PDQ(1,1,1,period=12)+0),
    sarima_310_011 = ARIMA(fcpi_log ~ pdq(3,1,0) + PDQ(0,1,1,period=12)+0),
    sarima_311_011 = ARIMA(fcpi_log ~ pdq(3,1,1) + PDQ(0,1,1,period=12)+0),
    sarima_312_011 = ARIMA(fcpi_log ~ pdq(3,1,2) + PDQ(0,1,1,period=12)+0),
    sarima_210_211 = ARIMA(fcpi_log ~ pdq(2,1,0) + PDQ(2,1,1,period=12)+0),
    sarima_211_211 = ARIMA(fcpi_log ~ pdq(2,1,1) + PDQ(2,1,1,period=12)+0),
    sarima_212_211 = ARIMA(fcpi_log ~ pdq(2,1,2) + PDQ(2,1,1,period=12)+0),
    sarima_210_111 = ARIMA(fcpi_log ~ pdq(2,1,0) + PDQ(1,1,1,period=12)+0),
    sarima_211_111 = ARIMA(fcpi_log ~ pdq(2,1,1) + PDQ(1,1,1,period=12)+0),
    sarima_212_111 = ARIMA(fcpi_log ~ pdq(2,1,2) + PDQ(1,1,1,period=12)+0),
    sarima_210_011 = ARIMA(fcpi_log ~ pdq(2,1,0) + PDQ(0,1,1,period=12)+0),
    sarima_211_011 = ARIMA(fcpi_log ~ pdq(2,1,1) + PDQ(0,1,1,period=12)+0),
    sarima_212_011 = ARIMA(fcpi_log ~ pdq(2,1,2) + PDQ(0,1,1,period=12)+0),
  )
```

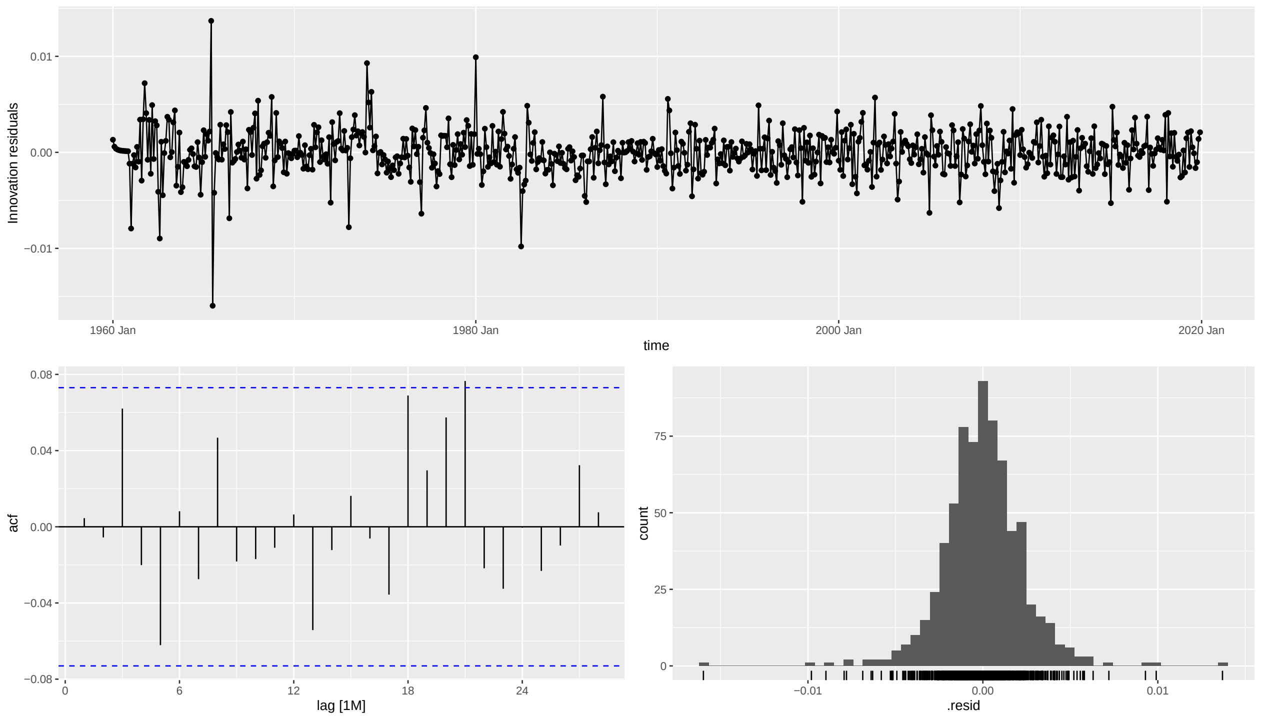
```
## Warning in sqrt(diag(best$var.coef)): NaNs produced

## Warning: 1 error encountered for sarima_312_211
## [1] non-finite finite-difference value [3]

## Warning: 1 error encountered for sarima_211_211
## [1] initial value in 'vmmin' is not finite
```

```
# Compare the models
glance(model_result)
```

```
## # A tibble: 16 x 8
##    .model              sigma2 log_lik    AIC   AICc    BIC ar_roots    ma_roots
##    <chr>                <dbl>   <dbl>  <dbl>  <dbl>  <dbl> <list>      <list>
##  1 sarima_310_211 0.00000572   3268. -6523. -6523. -6491. <cpl [27]> <cpl [12]>
##  2 sarima_311_211 0.00000552   3281. -6547. -6546. -6510. <cpl [27]> <cpl [13]>
##  3 sarima_310_111 0.00000571   3268. -6525. -6524. -6497. <cpl [15]> <cpl [12]>
##  4 sarima_311_111 0.00000551   3281. -6548. -6548. -6516. <cpl [15]> <cpl [13]>
##  5 sarima_312_111 0.00000552   3281. -6547. -6546. -6510. <cpl [15]> <cpl [14]>
##  6 sarima_310_011 0.00000571   3268. -6526. -6526. -6503. <cpl [3]>  <cpl [12]>
##  7 sarima_311_011 0.00000551   3281. -6550. -6550. -6523. <cpl [3]>  <cpl [13]>
##  8 sarima_312_011 0.00000551   3281. -6548. -6548. -6517. <cpl [3]>  <cpl [14]>
##  9 sarima_210_211 0.00000596   3254. -6495. -6495. -6468. <cpl [26]> <cpl [12]>
## 10 sarima_212_211 0.00000551   3282. -6547. -6547. -6511. <cpl [26]> <cpl [14]>
## 11 sarima_210_111 0.00000595   3254. -6497. -6497. -6475. <cpl [14]> <cpl [12]>
## 12 sarima_211_111 0.00000551   3281. -6550. -6550. -6523. <cpl [14]> <cpl [13]>
## 13 sarima_212_111 0.00000551   3281. -6549. -6549. -6517. <cpl [14]> <cpl [14]>
## 14 sarima_210_011 0.00000595   3254. -6499. -6499. -6481. <cpl [2]>  <cpl [12]>
## 15 sarima_211_011 0.00000550   3281. -6552. -6552. -6529. <cpl [2]>  <cpl [13]>
## 16 sarima_212_011 0.00000550   3281. -6551. -6551. -6523. <cpl [2]>  <cpl [14]>
```

7. Among all the candidate models, the SARIMA(2,1,1)(0,1,1)[12] model exhibited the lowest values for AIC, BIC, and AICc, indicating superior model performance. Consequently, it was chosen as the optimal model. The following is the model diagnostics for SARIMA(2,1,1)(0,1,1)[12].

```r
# Select the best model from the list
best_sarima_model <- model_result %>% select(sarima_211_011)

# Generate residual diagnostic plots
gg_tsresiduals(best_sarima_model)
```



Based on residual diagnostic:

- The residuals fluctuate randomly around zero, this means there is no trend or pattern left in the residuals.
- The ACF plot shows all autocorrelations are inside the confidence bands, this suggests there is no significant autocorrelation in the residuals.
- The histogram of the residuals looks like a normal distribution, this indicates the residuals are approximately normally distributed.
- There is no sign of seasonality or structure in the residual plot, this means the model has captured the seasonal and trend components well.

Conclusion: The residuals behave like white noise. This confirms that SARIMA(2,1,1)(0,1,1)[12] is a good model and fits the CPI data well.

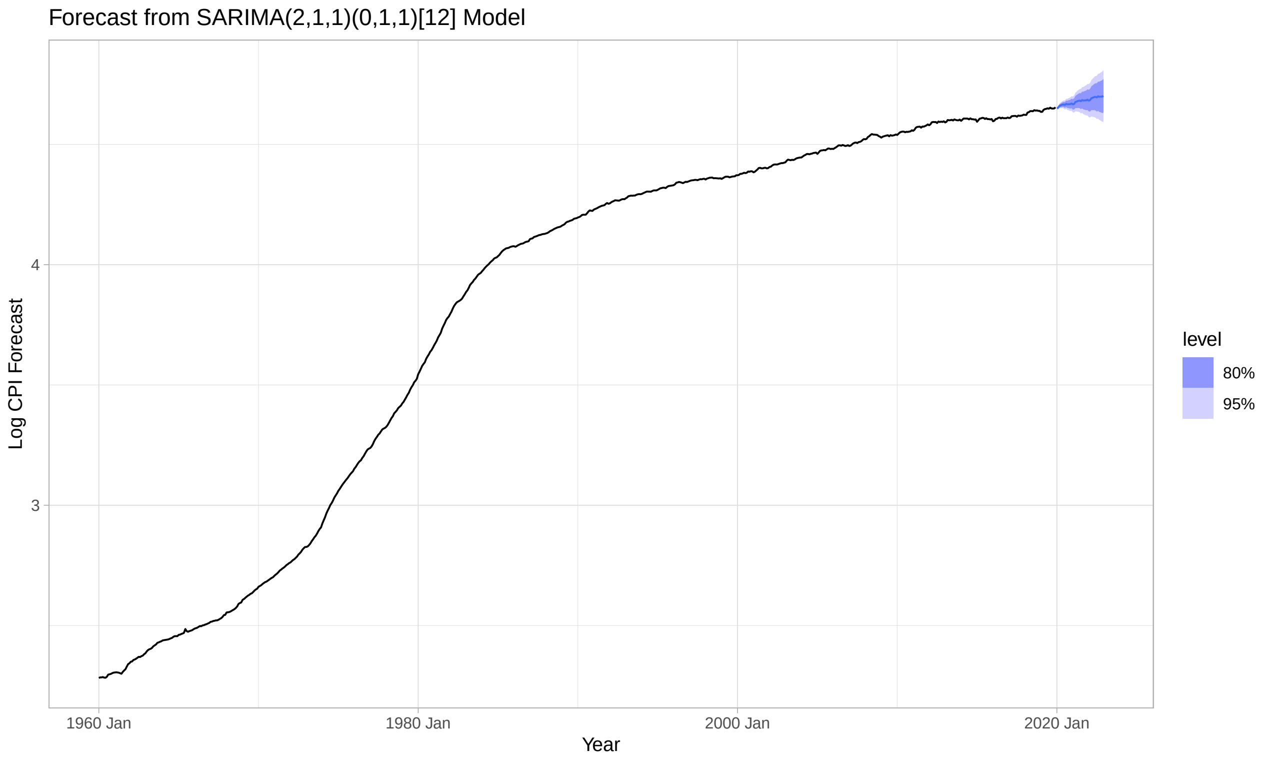
8. Ljung Box Test.

```r
augment(best_sarima_model) %>%
  features(.innov, ljung_box)
```

```
## # A tibble: 1 x 3
##   .model          lb_stat lb_pvalue
##   <chr>             <dbl>     <dbl>
## 1 sarima_211_011  0.0149     0.903
```

9. Since the p-value is greater than 0.05, we fail to reject the null hypothesis. This means that the residuals from the SARIMA(2,1,1)(0,1,1)[12] model do not exhibit significant autocorrelation. In the end, the following plots show the 36-month forecast generated by the selected SARIMA model, with one plot representing the logged data and the other showing the original data.

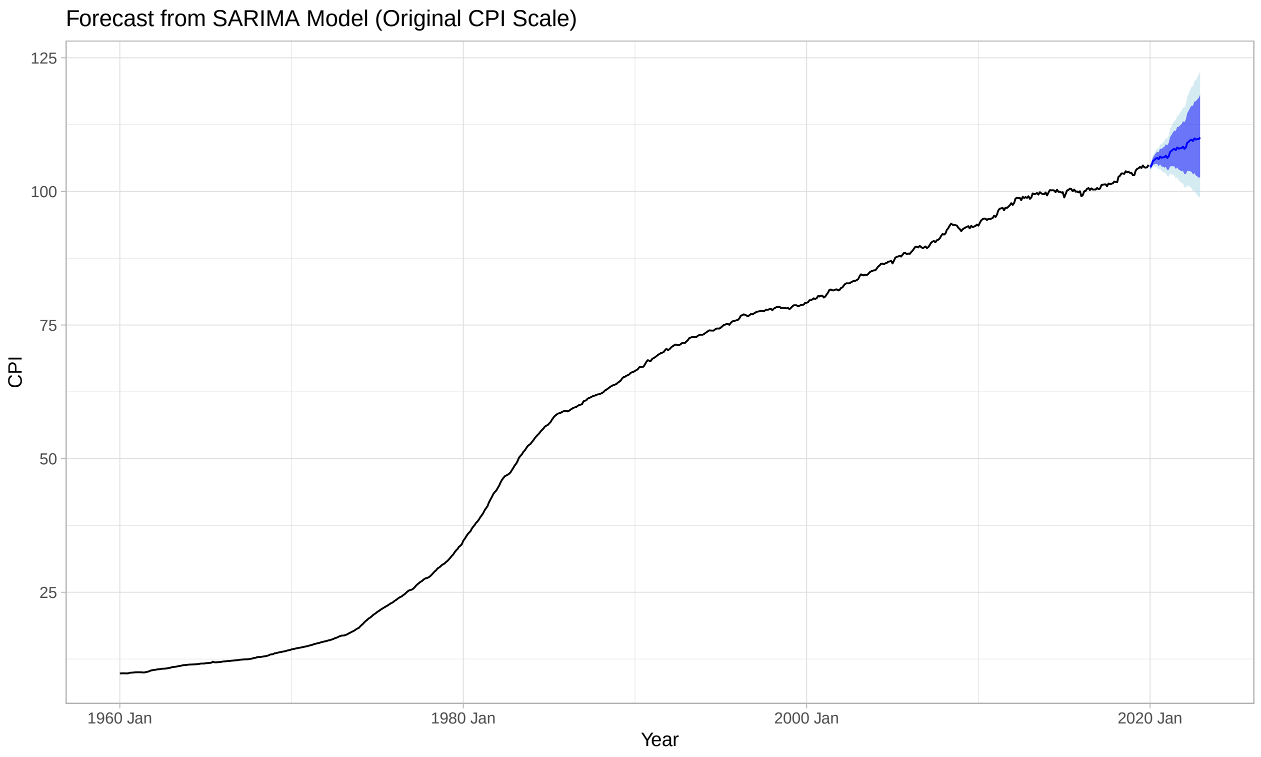
```r
# Generate 36-month forecast
sarima_forecast <- best_sarima_model %>%
  forecast(h = "36 months")

# Convert forecast to tibble
forecast_tbl <- sarima_forecast %>% as_tibble()

# Extract forecast interval (80% & 95%)
forecast_interval <- sarima_forecast %>%
  hilo(level = c(80, 95)) %>%
  unpack_hilo(`80%`, names_sep = "_") %>%
  unpack_hilo(`95%`, names_sep = "_")

# Plot forecast with uncertainty intervals
autoplot(sarima_forecast, fcpi_tsibble) +
  ggtitle("Forecast from SARIMA(2,1,1)(0,1,1)[12] Model") +
  xlab("Year") + ylab("Log CPI Forecast") +
  theme_light()
```



```r
# Convert log CPI forecast to original CPI scale
forecast_full_original <- forecast_interval %>%
  mutate(
    mean_original = exp(.mean),
    lower_80_original = exp(`80%_lower`),
    upper_80_original = exp(`80%_upper`),
    lower_95_original = exp(`95%_lower`),
    upper_95_original = exp(`95%_upper`)
  )

forecast_full_original
```

```
## # A tsibble: 36 x 13 [1M]
## # Key:       .model [1]
##    .model              time
##    <chr>              <mth>
##  1 sarima_211_011 2020 Jan
##  2 sarima_211_011 2020 Feb
##  3 sarima_211_011 2020 Mar
##  4 sarima_211_011 2020 Apr
##  5 sarima_211_011 2020 May
##  6 sarima_211_011 2020 Jun
##  7 sarima_211_011 2020 Jul
##  8 sarima_211_011 2020 Aug
##  9 sarima_211_011 2020 Sep
## 10 sarima_211_011 2020 Oct
## # i 26 more rows
## # i 11 more variables: fcpi_log <dist>, .mean <dbl>, `80%_lower` <dbl>,
## #   `80%_upper` <dbl>, `95%_lower` <dbl>, `95%_upper` <dbl>,
## #   mean_original <dbl>, lower_80_original <dbl>, upper_80_original <dbl>,
## #   lower_95_original <dbl>, upper_95_original <dbl>
```

```
# Plot forecast in Original Data
ggplot() +
  # 95% prediction interval
  geom_ribbon(data = forecast_full_original,
              aes(x = time, ymin = lower_95_original, ymax = upper_95_original),
              fill = "lightblue", alpha = 0.5) +
  # 80% prediction interval
  geom_ribbon(data = forecast_full_original,
              aes(x = time, ymin = lower_80_original, ymax = upper_80_original),
              fill = "blue", alpha = 0.5) +
  # Forecast mean (original CPI)
  geom_line(data = forecast_full_original,
            aes(x = time, y = mean_original), color = "blue") +
  # Historical data (original CPI)
  geom_line(data = fcpi_tsibble,
            aes(x = time, y = cpi), color = "black") +
  labs(
    title = "Forecast from SARIMA Model (Original CPI Scale)",
    x = "Year",
    y = "CPI"
  ) +
  theme_light()
```


Forecast from SARIMA Model (Original CPI Scale)
CPI
1960 Jan
1980 Jan
2000 Jan
2020 Jan
Year