Cocoa Price Prediction Model for Ghana*

Forecasting Cocoa Price Flutuation Using Time Series

Shanjie Jiao Edward Hong Lilian Sun Haoya Wang

April 1, 2025

Table of contents

1 Model

This study aims to develop a predictive model to capture future fluctuations in cocoa prices. To enhance the model's forecasting accuracy, a range of exogenous variables are considered, spanning both agricultural asepcts and macroeconomic dimensions. Climatic factors such as precipitation and temperature are considered due to their indirect influence on market prices through their effects on cocoa yield, which is considered as the most important factor pushing cocoa price. In addition, agricultural indicators—including labor input, cultivated area, yield per hectare—as well as productivity-related metrics such as total factor productivity (TFP),

*Code and data are available at: https://github.com/Jie-jiao05/Cocoa_price_preditcion.

are integrated into the framework to comprehensively evaluate their potential impact on price. By incorporating these variables, we hope to explore how these environmental and economic variables explain their impact on cocoa prices.

To investigate the potential impact of external variables on cocoa prices, the Generalized Additive Model (GAM), Autoregressive Integrated Moving Average (ARIMA), and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models are considered as candidate approaches.

1.1 Model Set-up

1.1.1 Generalized Additive Model (GAM)

Price, as the response variable in this study, is continuous, strictly positive, and reflects actual measured values rather than frequencies or binary outcomes for decision-making purposes. Thus, the Gamma distribution is selected. The use of a log link function ensures that predicted prices remain positive and allows the model to capture nonlinear and multiplicative relationships between the response and explanatory variables. This makes the Gamma distribution a theoretically appropriate and practically robust choice for modeling the influence of external factors on cocoa prices.

Since the dataset is organized by month (from January 2015 to December 2023) and includes only the Ghana region, there is no hierarchical or nested structure in the data. Furthermore, the temporal dimension is explicitly available through the monthly time variable. Therefore, random effects are not included in the model; instead, we focus on fixed effects, along with a smooth function of time. The smooth term is incorporated to capture nonlinear trends in the response over time. Additionally, since the outcome variable is cocoa price, a continuous quantity rather than a rate or count so offset term will not be considered in the model.

The model is defined as follows:

$$
\begin{aligned}
Y_t \mid U &\sim \mathrm{Gamma}(\mu_t, \theta), \quad g(\mu_t) = X_t\beta + U(t) \\
g(\mu_t) = \log(\mu_t) &= \beta_0 + s_1(\mathrm{Month\_Index}_t) + s_2(\mathrm{Temp}_t) + s_3(\mathrm{Fert}_t) + s_4(\mathrm{TFP\_Index}_t) \\
&\quad + s_5(\mathrm{Capital\_Index}_t) + s_6(\mathrm{Land\_Q}_t) + s_7(\mathrm{Labor\_Q}_t) + s_8(\mathrm{Cropland\_Q}_t) \\
&\quad + s_9(\mathrm{prep}_t) + \beta_{10} \cdot \mathrm{Production\_tonnes}_t + \beta_{11} \cdot \mathrm{Yield\_tonnes\_per\_hectare}_t \\
&\quad + U(t) \\
U(t) &\sim \mathrm{IWP}_2(\sigma) \quad \text{(Smooth Trend)}
\end{aligned}
$$

```
Family: Gamma
Link function: log
```

```
Formula:
Price ~ s(Month_Index) + s(Temp) + s(Fert) + s(TFP_Index) + s(Capital_Index) +
    s(Land_Q) + s(Labor_Q) + s(Cropland_Q) + s(prep) + Production_tonnes +
    Yield_tonnes_per_hectare

Parametric coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               2.048e+00  3.081e+00   0.665   0.5093
Production_tonnes        -1.753e-07  1.811e-07  -0.968   0.3378
Yield_tonnes_per_hectare  1.087e+01  5.768e+00   1.885   0.0651 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                   edf Ref.df      F  p-value
s(Month_Index)   8.105  8.746  8.737 4.53e-07 ***
s(Temp)          1.484  1.818  0.439  0.70213
s(Fert)          4.391  5.345  3.495  0.00646 **
s(TFP_Index)     1.000  1.000 21.047 3.00e-05 ***
s(Capital_Index) 1.000  1.000  0.015  0.90356
s(Land_Q)        1.000  1.000 10.654  0.00196 **
s(Labor_Q)       1.945  2.208  2.466  0.08152 .
s(Cropland_Q)    1.000  1.000 10.582  0.00203 **
s(prep)          1.000  1.000  2.409  0.12686
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.939   Deviance explained = 95.2%
GCV = 0.0030818  Scale est. = 0.0021012  n = 75

[1] 953.5679
```

1.1.2 Autoregressive Integrated Moving Average (ARIMA)

The second model we select is ARIMA, as our dataset provides accurate monthly records from January 2015 to December 2023. ARIMA effectively models how past values influence future outcomes, making it ideal for capturing temporal dependencies and trends. It also handles non-stationarity through differencing, which stabilizes the data and facilitates more reliable model construction.

From the initial plot of the cocoa price data, there is no clear evidence of a seasonal trend. The series appears to fluctuate irregularly over time. However, the ACF and PACF plots of the original (undifferenced) series reveal signs of non-stationarity, as the autocorrelations decay slowly. To address this, we apply first-order differencing, which yields a series that appears stationary. The ACF and PACF plots of the differenced series indicate an autoregressive structure of order 2. Based on these diagnostics, we propose an ARIMA(2,1,0) model for the cocoa price series.

The model is defined as follows:

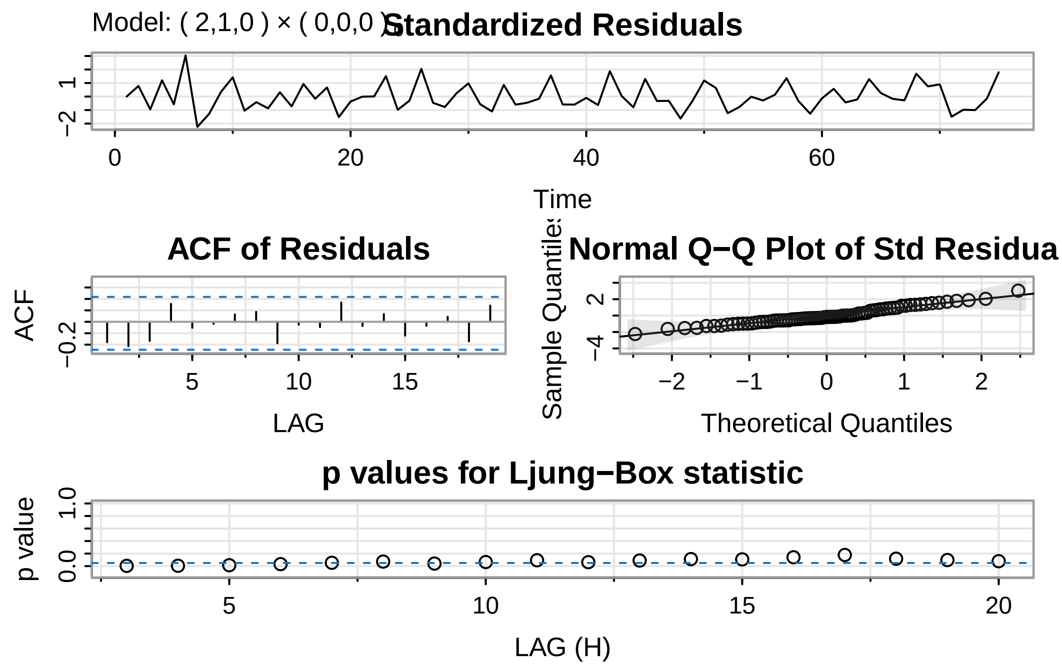
$$\Delta y_t = \phi_1 \Delta y_{t-1} + \phi_2 \Delta y_{t-2} + \varepsilon_t$$

```
initial  value 6.586000
iter   2 value 6.391957
iter   3 value 6.273983
iter   4 value 6.264059
iter   5 value 6.258127
iter   6 value 6.246465
iter   7 value 6.246399
iter   8 value 6.246397
iter   8 value 6.246397
final  value 6.246397
converged
initial  value 6.249680
iter   2 value 6.249670
iter   3 value 6.249664
iter   4 value 6.249657
iter   4 value 6.249657
iter   4 value 6.249657
final  value 6.249657
converged
<><><><><><><><><><><><><><>

Coefficients:
         Estimate      SE t.value p.value
ar1       -0.8714  0.1024 -8.5110  0.0000
ar2       -0.4932  0.1021 -4.8322  0.0000
constant   0.7308 25.5678  0.0286  0.9773

sigma^2 estimated as 264647.1 on 71 degrees of freedom

AIC = 15.4453  AICc = 15.44993  BIC = 15.56984
```



1.1.3 Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

While ARIMA and GAM models primarily focus on modeling the conditional mean of a time series, they typically assume homoskedasticity — that is, constant variance of the error terms over time. However, in financial and commodity markets such as cocoa prices, time series often exhibit heteroskedasticity, particularly in the form of volatility clustering, where periods of high volatility tend to cluster together, as do periods of low volatility. To address this, the third model introduced in this study is the GARCH model, which is specifically designed to capture such dynamic behavior in volatility.

In the context of this research, modeling the volatility of cocoa prices is crucial for understanding the risks and uncertainties associated with price movements over time. The GARCH framework allows the conditional variance to evolve dynamically, providing a more realistic and robust approach to capturing the stylized facts of the cocoa price series. By accommodating time-varying volatility, the GARCH model serves as an essential complement to mean-based models and enhances the overall forecasting framework. A generalized GARCH(1,1) model is combined with an ARMA(1,0) structure in the mean equation to account for potential autocorrelation in the return series without overcomplicating the model.

$$
\begin{aligned}
r_t &= \sigma_t \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0,1) \\
\sigma_t^2 &= \alpha_0 + \alpha_1 r_{t-1}^2 + \beta_1 \sigma_{t-1}^2
\end{aligned}
$$

```
NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
attached to the search() path when 'fGarch' is attached.

If needed attach them yourself in your R script by e.g.,
        require("timeSeries")


Series Initialization:
 ARMA Model:                arma
 Formula Mean:              ~ arma(1, 0)
 GARCH Model:               garch
 Formula Variance:          ~ garch(1, 1)
 ARMA Order:                1 0
 Max ARMA Order:            1
 GARCH Order:               1 1
 Max GARCH Order:           1
 Maximum Order:             1
 Conditional Dist:          norm
 h.start:                   2
 llh.start:                 1
 Length of Series:          74
 Recursion Init:            mci
 Series Scale:              0.2652516

Parameter Initialization:
 Initial Parameters:          $params
 Limits of Transformations:   $U, $V
 Which Parameters are Fixed?  $includes
 Parameter Matrix:
                     U          V       params includes
    mu     -0.13515703   0.135157  0.003181607     TRUE
    ar1    -0.99999999   1.000000 -0.571748121     TRUE
    omega   0.00000100 100.000000  0.100000000     TRUE
    alpha1  0.00000001   1.000000  0.100000000     TRUE
    gamma1 -0.99999999   1.000000  0.100000000    FALSE
    beta1   0.00000001   1.000000  0.800000000     TRUE
    delta   0.00000000   2.000000  2.000000000    FALSE
    skew    0.10000000  10.000000  1.000000000    FALSE
    shape   1.00000000  10.000000  4.000000000    FALSE
 Index List of Parameters to be Optimized:
    mu    ar1  omega alpha1  beta1
     1      2      3      4      6
```

```
 Persistence:                  0.9


--- START OF TRACE ---
Selected Algorithm: nlminb

R coded nlminb Solver:

  0:     91.064114: 0.00318161 -0.571748 0.100000 0.100000 0.800000
  1:     90.455959: 0.00318170 -0.571842 0.0897750 0.0905958 0.790040
  2:     90.316810: 0.00318221 -0.571797 0.0898436 0.0745243 0.784218
  3:     89.985143: 0.00318332 -0.571631 0.115168 0.0516874 0.781787
  4:     89.915065: 0.00318359 -0.572289 0.115928 0.0326938 0.777451
  5:     89.753768: 0.00318315 -0.574041 0.129223 0.0209313 0.785349
  6:     89.687118: 0.00318177 -0.574220 0.129099 0.00322959 0.793544
  7:     89.670747: 0.00318143 -0.574212 0.132190 0.00251832 0.797994
  8:     89.666646: 0.00318103 -0.574383 0.129110 1.00000e-08 0.798616
  9:     89.661812: 0.00317441 -0.581730 0.132342 1.00000e-08 0.798062
 10:     89.659576: 0.00317382 -0.573691 0.131753 1.00000e-08 0.798063
 11:     89.659568: 0.00317378 -0.573778 0.132404 1.00000e-08 0.798530
 12:     89.659340: 0.00317376 -0.573824 0.132066 1.00000e-08 0.798316
 13:     89.659278: 0.00317341 -0.574597 0.131915 1.00000e-08 0.798486
 14:     89.658942: 0.00312354 -0.574964 0.127006 1.00000e-08 0.805855
 15:     89.641626: 0.00239224 -0.576949 0.0468896 1.00000e-08 0.925190
 16:     89.574067: 0.00177248 -0.575470 1.00000e-06 1.00000e-08 0.999035
 17:     89.566713: 0.00171828 -0.572768 1.00000e-06 1.00000e-08 0.998914
 18:     89.546522: 0.00150324 -0.564921 1.00000e-06 1.00000e-08 0.998230
 19:     89.544925: 0.00151573 -0.566926 1.00000e-06 1.00000e-08 0.998118
 20:     89.544540: 0.00155521 -0.569248 1.00000e-06 1.00000e-08 0.998081
 21:     89.544538: 0.00157059 -0.569388 1.00000e-06 1.00000e-08 0.998085
 22:     89.544536: 0.00159147 -0.569432 1.00000e-06 1.00000e-08 0.998087
 23:     89.544532: 0.00169140 -0.569526 1.00000e-06 1.00000e-08 0.998090
 24:     89.544526: 0.00186373 -0.569577 1.00000e-06 1.00000e-08 0.998092
 25:     89.544521: 0.00208418 -0.569534 1.00000e-06 1.00000e-08 0.998091
 26:     89.544518: 0.00218035 -0.569433 1.00000e-06 1.00000e-08 0.998087
 27:     89.544518: 0.00217572 -0.569382 1.00000e-06 1.00000e-08 0.998085
 28:     89.544518: 0.00216577 -0.569375 1.00000e-06 1.00000e-08 0.998084
 29:     89.544518: 0.00216433 -0.569376 1.00000e-06 1.00000e-08 0.998084

Final Estimate of the Negative LLH:
 LLH:  -8.65913    norm LLH:  -0.1170153
           mu            ar1          omega         alpha1          beta1
 5.740910e-04 -5.693755e-01  7.035843e-08  1.000000e-08  9.980845e-01
```

```
R-optimhess Difference Approximated Hessian Matrix:
                 mu          ar1         omega        alpha1         beta1
mu     -1582.706434     2.131551     -9003.343     -249.3021     -379.4792
ar1        2.131551  -106.648072     -7894.958     -354.9384     -342.2954
omega  -9003.343304 -7894.957628 -34491841.703 -1646364.2282 -1504977.7310
alpha1  -249.302089  -354.938383  -1646364.228   -78923.2855   -72375.7127
beta1   -379.479209  -342.295391  -1504977.731   -72375.7127   -66631.6164
attr(,"time")
Time difference of 0.002393007 secs

--- END OF TRACE ---


Time to Estimate Parameters:
 Time difference of 0.01388192 secs
```

```r
library(Metrics)
```

```
Attaching package: 'Metrics'


The following object is masked from 'package:forecast':

    accuracy
```

```r
test$Date <- as.Date(test$Date)  # Convert Date column
test$Month_Index <- as.numeric(as.factor(test$Date))  # Numeric index for smooth time trend
gam_pred <- predict(gam_model, newdata = test, type = "response")


arima_pred <- forecast(arima_model, h = nrow(test))$mean

garch_forecast <- predict(garch_model, n.ahead = nrow(test))
garch_mean <- garch_forecast$meanForecast

# For GAM and ARIMA (level)
actual <- test$Price
test_returns  <- diff(log(test$Price))

rmse <- function(pred, actual) {
```

```r
  sqrt(mean((pred - actual)^2))
}

rmse_gam   <- rmse(gam_pred, actual)
rmse_arima <- rmse(arima_pred, actual)

# For GARCH - only meaningful if you're comparing returns
rmse_garch <- rmse(garch_mean, test_returns)  # optional
```

```
Warning in pred - actual: longer object length is not a multiple of shorter
object length
```

```r
test$Date <- as.Date(test$Date)

# Add Month_Index if used in GAM
test$Month_Index <- as.numeric(as.factor(test$Date))

# ---- Predict from previously trained models ----
# Replace gam_model, arima_model, garch_model with your trained model names

# 1. GAM
gam_pred <- predict(gam_model, newdata = test, type = "response")

# 2. ARIMA
arima_pred <- forecast(arima_model, h = nrow(test))$mean

# 3. GARCH
garch_forecast <- predict(garch_model, n.ahead = nrow(test))
garch_mean_return <- garch_forecast$meanForecast
last_price <- tail(train$Price, 1)  # use last value from training set
garch_pred <- last_price * exp(cumsum(garch_mean_return))

# ---- Calculate RMSE ----
actual <- test$Price

rmse <- function(pred, actual) sqrt(mean((pred - actual)^2))

rmse_gam   <- rmse(gam_pred, actual)
rmse_arima <- rmse(arima_pred, actual)
rmse_garch <- rmse(garch_pred, actual)
```

```r
# ---- Output ----
cat("RMSE (GAM):   ", round(rmse_gam, 4), "\n")
```

```
RMSE (GAM):    682.3146
```

```r
cat("RMSE (ARIMA): ", round(rmse_arima, 4), "\n")
```

```
RMSE (ARIMA):  514.4683
```

```r
cat("RMSE (GARCH): ", round(rmse_garch, 4), "\n")
```

```
RMSE (GARCH):  626.6236
```

1.2 Final Model

1.3 Model Diagonistic

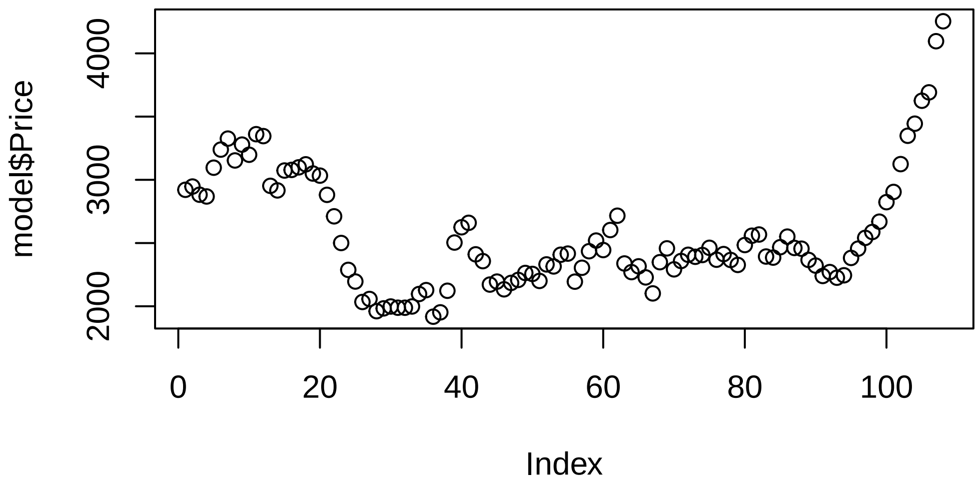
2 Results

2.1 Model Performance Validatioin

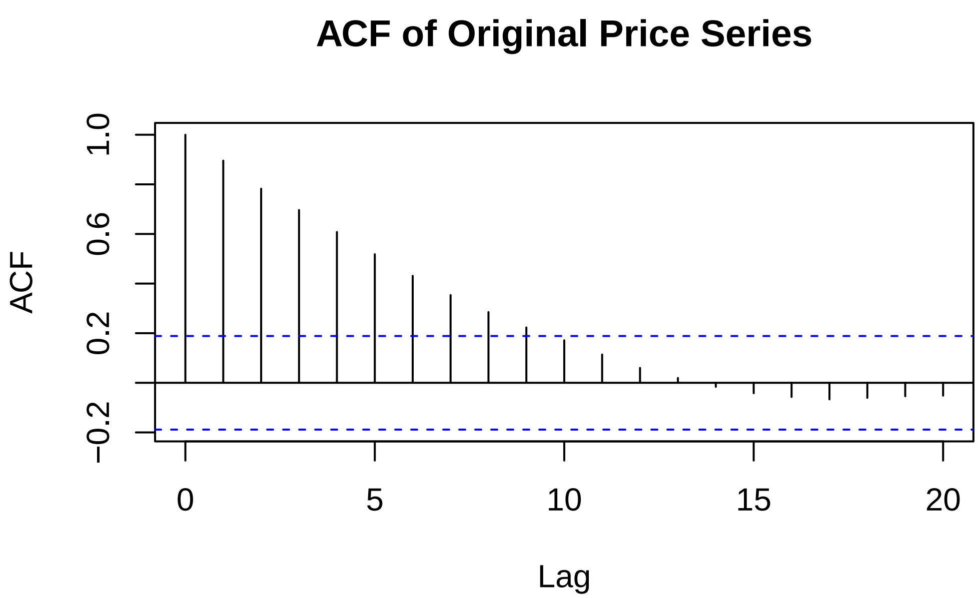
2.2 Forecasting

Appendix

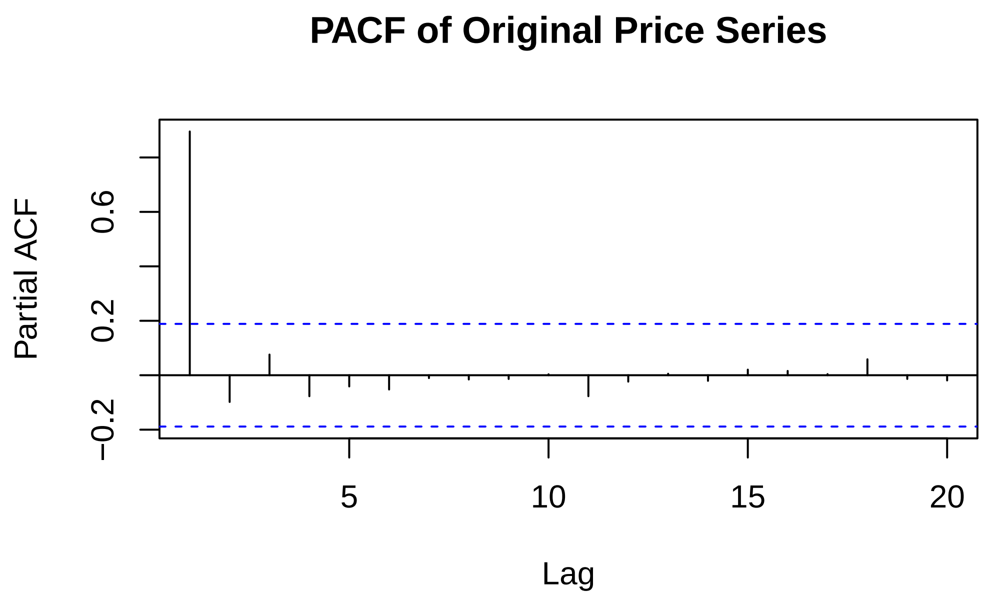
```r
plot(model$Price)
```



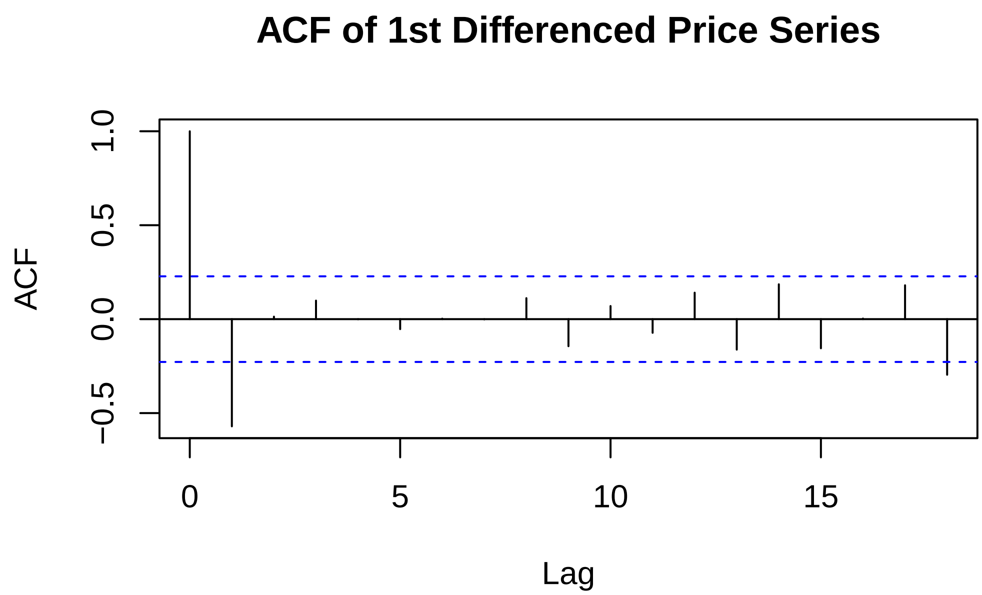

```
# Augmented Dickey-Fuller test
acf(model$Price, main = "ACF of Original Price Series")
```



```r
pacf(model$Price, main = "PACF of Original Price Series")
```




```r
price_diff <- diff(train$Price)  # first-order difference
acf(price_diff, main = "ACF of 1st Differenced Price Series")
```




```
pacf(model$Price, main = "PACF of 1st Differenced Price Series")
```