SGOL Forecasting using ARIMA Model and Natural Spline

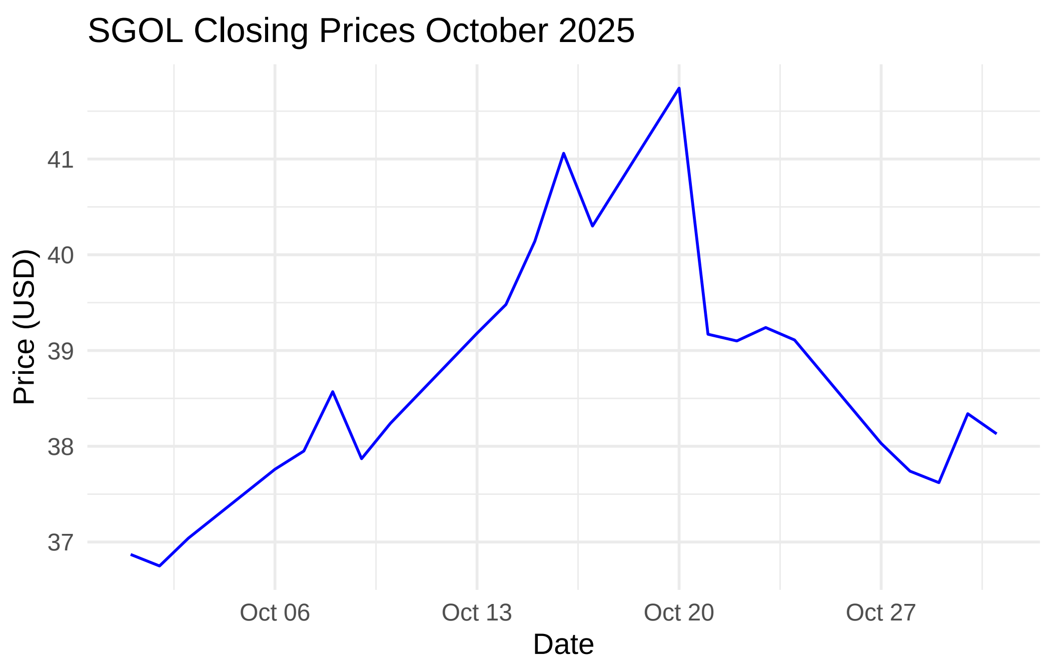
Eric L. Dias

SGOL(abrdn Physical Gold Shares ETF) Modeling

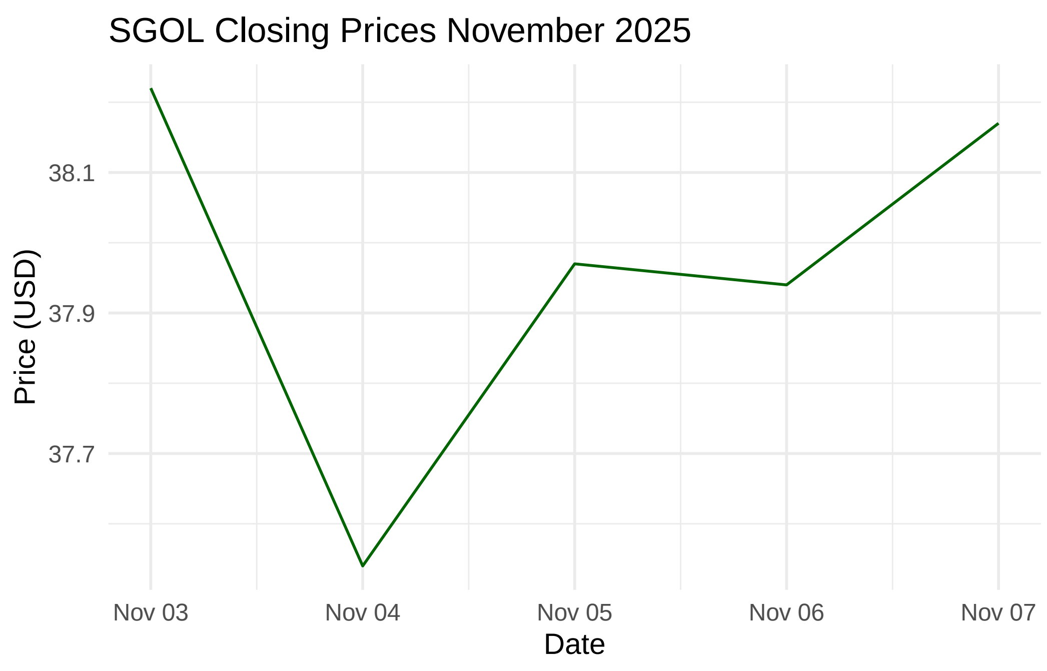
```r
library(alphavantager)
```

```r
options(readr.show_col_types = FALSE)
sgol_oct <- av_get(symbol = "SGOL",
                    av_fun = "TIME_SERIES_DAILY",
                    outputsize = "full")

options(readr.show_col_types = FALSE)
sgol_nov <- av_get(symbol = "SGOL",
                   av_fun = "TIME_SERIES_DAILY",
                   outputsize = "full")

sgol_oct <- subset(sgol_oct, timestamp >= as.Date("2025-10-01") &
                     timestamp <= as.Date("2025-10-31"))

sgol_nov <- subset(sgol_nov, timestamp >= as.Date("2025-11-01") &
                     timestamp <= as.Date("2025-11-07"))
```

```r
library(ggplot2)

ggplot(sgol_oct, aes(x = timestamp, y = close)) +
  geom_line(color = "blue") +
  labs(title = "SGOL Closing Prices October 2025",
       x = "Date", y = "Price (USD)") +
  scale_x_date(date_labels = "%b %d") +  # format x-axis dates nicely
  theme_minimal()
```



```r
ggplot(sgol_nov, aes(x = timestamp, y = close)) +
  geom_line(color = "darkgreen") +
  labs(title = "SGOL Closing Prices November 2025",
       x = "Date", y = "Price (USD)") +
  scale_x_date(date_labels = "%b %d") +  # format x-axis dates nicely
  theme_minimal()
```



Augmented Dickey-Fuller Test for Stationarity

```r
library(tseries)
```

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

```r
oct_ts <- ts(sgol_oct$close)
adf.test(oct_ts)
```

```

	Augmented Dickey-Fuller Test

data:  oct_ts
Dickey-Fuller = -1.4936, Lag order = 2, p-value = 0.7653
alternative hypothesis: stationary
```

Series Not Stationary

```r
oct_diff_ts <- diff(oct_ts)
adf_test = adf.test(oct_diff_ts)
adf_test
```

```

	Augmented Dickey-Fuller Test

data:  oct_diff_ts
Dickey-Fuller = -2.357, Lag order = 2, p-value = 0.4364
alternative hypothesis: stationary
```

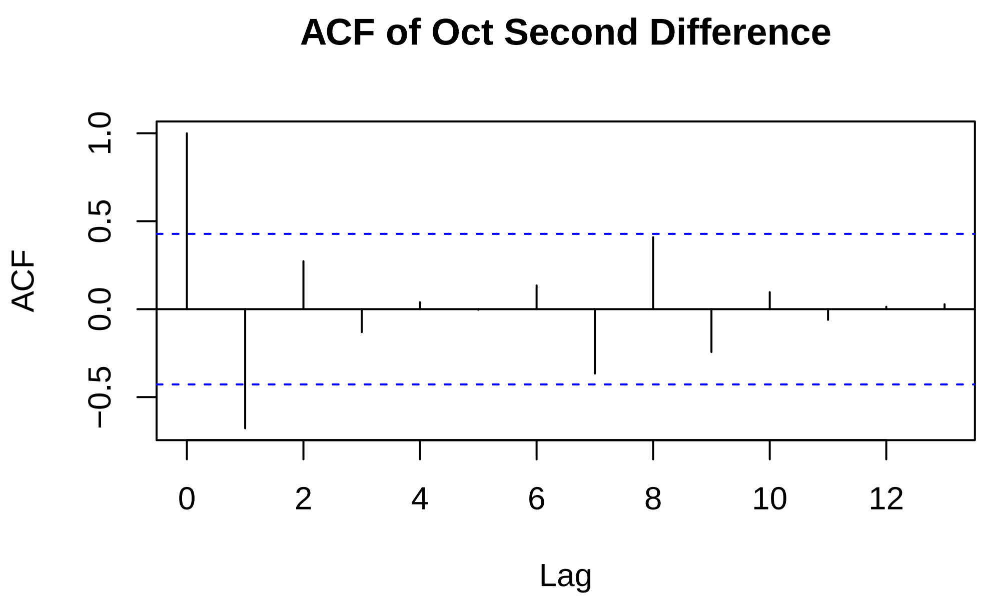
First Difference Not Stationary

```r
oct_secdiff_ts <- diff(oct_diff_ts)
adf_test = adf.test(oct_secdiff_ts)
adf_test
```

```

	Augmented Dickey-Fuller Test

data:  oct_secdiff_ts
Dickey-Fuller = -3.7904, Lag order = 2, p-value = 0.0364
alternative hypothesis: stationary
```

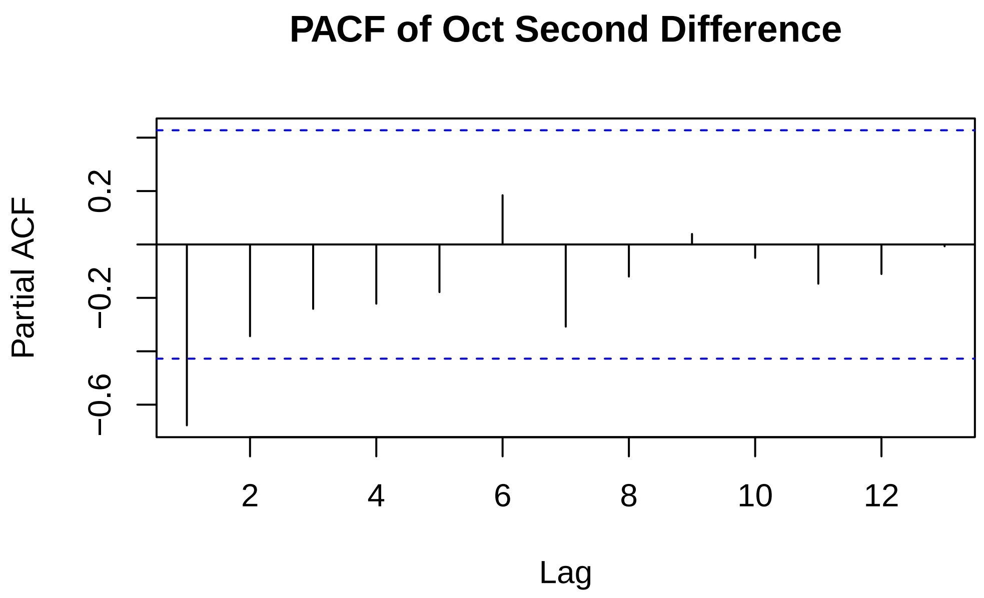
Second Difference is Stationary

```r
acf(oct_secdiff_ts, main = "ACF of Oct Second Difference")
```



MA Order is 1

```r
pacf(oct_secdiff_ts, main = "PACF of Oct Second Difference")
```



AR order is 0

Fit ARIMA(0, 2, 1) on October Data and Test on November Data

```r
library(forecast)
train_series <- c(sgol_oct$close)
model <- Arima(train_series, order = c(0, 2, 1))
predicted <- forecast(model, h=5)$mean
actual <- sgol_nov$close
```

```r
cat("Predicted: ", predicted, "\n")
```

```
Predicted:  38.15092 38.17184 38.19277 38.21369 38.23461
```

```r
cat("Actual: ", actual)
```

```
Actual:  38.22 37.54 37.97 37.94 38.17
```

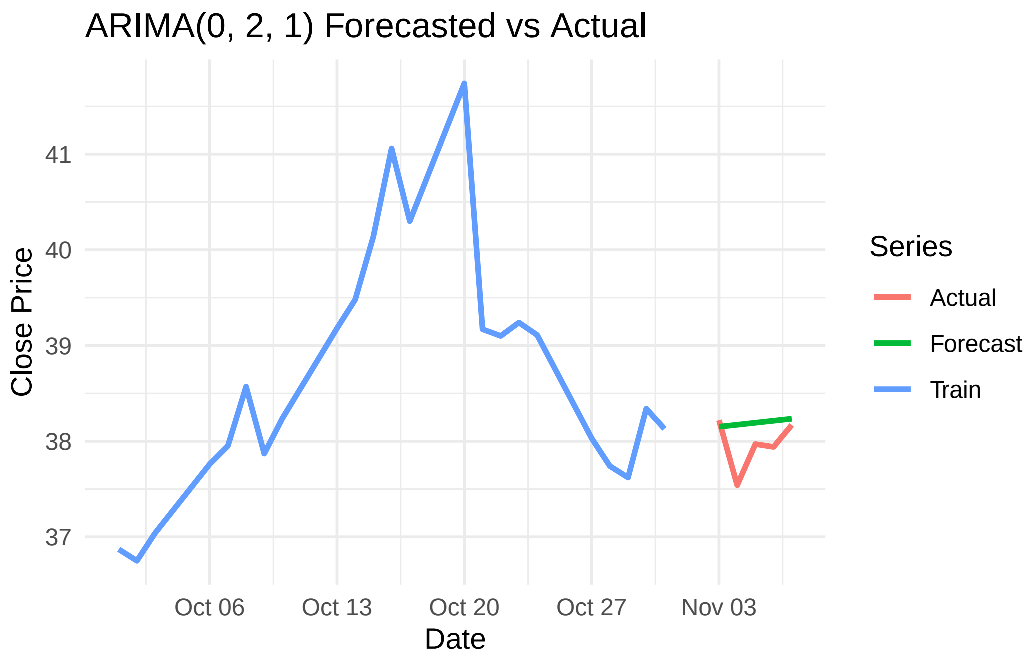
Plot Predicted and Actual

```r
library(ggplot2)
library(tibble)
library(dplyr)
```

```

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
train_dates <- sgol_oct$timestamp
test_dates <- sgol_nov$timestamp

plot_data <- tibble(
  date = c(train_dates, test_dates, test_dates),
  value = c(train_series, predicted, actual),
  type = c(
    rep("Train", length(train_series)),
    rep("Forecast", length(predicted)),
    rep("Actual", length(actual))
  )
)

ggplot(plot_data, aes(x = date, y = value, color = type)) +
  geom_line(size = 1) +
  labs(title = "ARIMA(0, 2, 1) Forecasted vs Actual",
       x = "Date",
       y = "Close Price",
       color = "Series") +
  theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Evaluation of Fit

```r
mse <- mean(predicted - actual)^2
cat("MSE: ", mse)
```

```
MSE:  0.05051951
```

```r
rss <- sum((predicted - actual)^2)
tss <- sum((actual - mean(actual))^2)
r_squared <- 1 - rss/tss
cat("R Squared: ", r_squared)
```

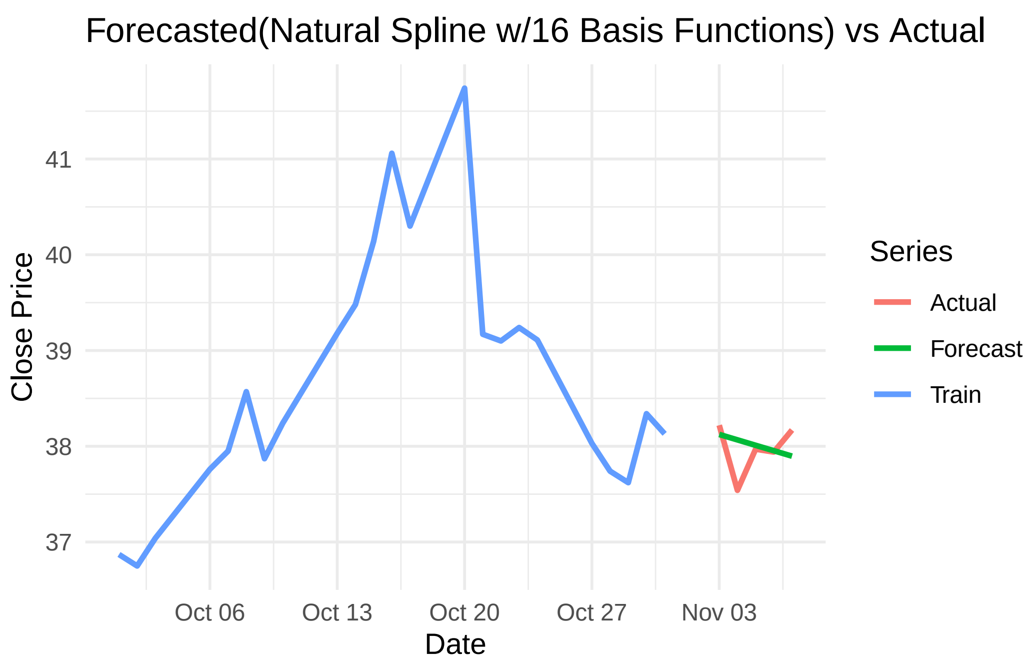
```
R Squared:  -0.8478622
```

Can we use a Natural Spline to get a Better Fit?

I am choosing 16 basis functions based on a visual inspection of the training data.

```r
library(splines)
x <- 1:length(train_series)
fit <- lm(train_series ~ ns(x, df=16))

x_test <- data.frame(x = (length(train_series) + 1):(length(train_series) + 5))
y_hat <- predict(fit, newdata=x_test, type="response")
```

```r
plot_data <- tibble(
  date = c(train_dates, test_dates, test_dates),
  value = c(train_series, y_hat, actual),
  type = c(
    rep("Train", length(train_series)),
    rep("Forecast", length(y_hat)),
    rep("Actual", length(actual))
  )
)

ggplot(plot_data, aes(x = date, y = value, color = type)) +
  geom_line(size = 1) +
  labs(title = "Forecasted(Natural Spline w/16 Basis Functions) vs Actual",
       x = "Date",
       y = "Close Price",
       color = "Series") +
  theme_minimal()
```



Evaluation of Fit

```
mse <- mean(y_hat - actual)^2
cat("MSE: ", mse)
```

```
MSE:  0.001777148
```

```
rss <- sum((y_hat - actual)^2)
tss <- sum((actual - mean(actual))^2)
r_squared <- 1 - rss/tss
cat("R Squared: ", r_squared)
```

```
R Squared:  -0.2597354
```

Conclusion

Using a Natural Spline with 16 degrees of freedom resulted in a much lower Mean Squared Error than using an ARIMA(0, 2, 1) model, meaning it is a better fit. However since both methods resulted in negative R Squared, we are better off simply predicting the mean.