Can the "Sell in May and Go Away" Strategy Produce Excess Returns in the TSX?

Isaiah Sinclair and Mahir Mehta

Professor Y. Sun

FIN*4100

5 April 2024

Abstract

“Sell in May and Go Away” (also known as the “Halloween Effect”) is a calendar effect which posits that the returns in the stock market from the 6-month period of May-October are outperformed by the returns from November-April. This research investigates the strength of this effect in the TSX with samples from over 20 years of data across 4 oldest and largest ETFs by assets under Management. However, we performed most of our research using the XIU ETF, which is well-diversified in the 60 top companies in the TSX. We perform tests on the models for the difference in return between the two 6-month periods. After demonstrating some statistically significant evidence of a Halloween Effect, we construct a Halloween Effect trading strategy, where we buy risk-free assets during May-October, and invest in the stock market during the November-April period. We compare the results of this model with the results of a standard buy-and-hold strategy in the XIU ETF, evaluating if there is a statistically significant alpha value that can be derived using the Capital Asset Pricing Model (CAPM). We also analyze the parameter stability of these models.

Introduction and Background

“Sell in May and Go Away” is an old saying that states that investors are better off pulling their money out of the market in May until the end of October. But as an old saying in England goes, “remember to buy back by St. Legers’ Day”. Similarly, O'Higgins (1990) investigated a similar strategy and timing of it. This effect was first investigated in a seminal study by Bouman and Jacobsen (2002). They find statistically significant differences between the May–October and the November–April stock returns in 36 of 37 countries, with data covering the January 1970 to August 1998 period. 6-month returns in the May–October period average 11.1% lower than the November-April returns.

Jacobsen and Zhang (2012) argue that the Halloween Effect strengthened rather than weakened in recent years and that it does not suffer from Murphy’s Law, which suggests that the return from trying to profit from a market anomaly will usually be negative (Dimson and Marsh, 1999). Several researchers suggest that these seasonal stock market returns could be tied to factors such as seasonal affective disorder (SAD), the weather, or daylight savings time (Garrett et al., 2005). Other profitable strategies may coincide with or even resemble the Halloween Effect. Afik and Lahav (2015) demonstrated that although the Halloween Effect is statistically significant in the US stock market, it can be easily outperformed by holding a market index from March to November each year. Their study revealed that this strategy remained consistent over 43 years on the S&P 500 index. However, they acknowledge the challenge of distinguishing the January effect from the Halloween Effect. Schabek and Castro (2017) similarly found that strategies initiated in October, November, and December also yielded abnormal returns.

Pastun et al. (2019) conducted research regarding the Halloween Effect for the Canadian stock market for the period of 1915-2014 to perform a sub-period analysis of the returns noting the returns in recent history are more prominent then in earlier years, consistent with findings in other research reports.

We retested the significance of the difference in the returns for the two 6-month periods and performed tests for the Canadian stock market in general, testing for the ETFs tracking the largest/widest pool of stocks (by market capitalization) on the Toronto Stock Exchange. We performed the tests to see if there were any violation of the Classical Linear Regression Model (CLRM) for the models we constructed on different ETFs. We also created a Halloween Effect strategy model to determine if we could find statistically significant excess returns. These returns are denoted by the alpha constant value in the CAPM regression equation that is run.

In recent years, Lo (2004) introduced the adaptive market hypothesis (AMH), which emphasizes the dynamic nature of market efficiency when multiple "species" (groups of market participants) compete for limited resources (profit opportunities), the market tends to be highly efficient. Consequently, some investors may incur losses and exit the market, reducing competition and increasing profit opportunities over time. This attracts new investors, leading to a renewed increase in market efficiency. Thus, the AMH proposes that market efficiency fluctuates cyclically, rather than following a consistent trend of increasing efficiency as suggested by the efficient market hypothesis (EMH). Consequently, trading strategies based on predictable patterns in price behavior (e.g., calendar anomalies) are also expected to vary in performance over time.

In recent decades, several studies have investigated calendar anomalies within the framework of the AMH. For instance, Urquhart and McGroarty (2014) utilized subsample and rolling window techniques to examine the evolution of four calendar anomalies (January, Halloween, Monday, and turn-of-the-month effects) in the Dow Jones Industrial Average from 1900 to 2013.

Plastun et al. (2020) conducted a comprehensive investigation of the evolution of various monthly anomalies (the January, December, and Mark Twain effects) in the US stock market for its entire history. The results show that the January Effect was the only prevalent anomaly, providing exploitable profit opportunities and that the behavior of the market is consistent with the AMH. Finally, Alekneviciene et al. (2022) tested the AMH regarding calendar anomalies in the Baltic stock markets using GARCH regressions, Kruskal–Wallis statistics, and rolling windows. The authors found three significant anomalies (the monthly, turn-of-the-month, and Friday effects), concluding that the behavior of the market is consistent with the AMH.

Furthermore, Lucey & Zhao (2008) found no evidence of the Halloween Effect in the US stock market. Lucey and Zhao (2008) instead found that the January Effect and the Halloween effect were confounded. The January Effect largely made up most of what is denoted as the Halloween Effect. Therefore, Lucey & Zhao (2008) concluded that the existence of the Halloween Effect in the US was uncertain. This is our basis for extended analysis for the January Effect relation to the Halloween Effect, hence tests to further investigate it were conducted as well.

Data

We used the following ETFs for our models and tests, primarily using XIU:

ETF Details	
Ticker	**ETF Description**
XIU	Tracks the largest 60 companies listed on the TSX and is the oldest and largest ETF in Canada.
XIC	Seeks to replicate the performance of the S&P/TSX Capped Composite Index.
XMD	Seeks to replicate the returns of the S&P/TSX Completion Index which consists of the mid-cap and small-cap companies.
XRE	Seeks to replicate the returns of the REIT Index on the S&P/TSX.

We downloaded the adjusted close prices of each of these ETFs and computed the monthly returns with these index prices. The data was downloaded from Yahoo Finance. Specifically with the XIU ETF, we used data from January 2000 to December 2023. With the other ETFs, we used data from January 2003 to December 2023. We did not conduct as many tests with the other ETFs, XIC, XMD, and XRE. These ETFs were used to see if the Halloween Effect exists in specific markets, not just the largest companies in the TSX.

The risk-free returns for the Halloween Effect strategy was derived from the 6-month Canadian treasury bill yield averaged over the weekly auction for the month calculated. This data was downloaded from the Bank of Canada and was cleaned using Microsoft Excel and the Pandas library in Python for our use.

	XIU.TO Returns (r_t)	**XIC.TO Returns** (r_t)	**XMD.TO Returns** (r_t)	**XRE.TO Returns** (r_t)	**Effect Period** (S_t)	**January Effect** (J_t)	**Effect Period*** (S_t^{adj})	**Risk free rate** (R_f)
Count	*288.0000*	*181.0000*	*181.0000*	*181.0000*	*288.0000*	*288.0000*	*288.0000*	*288.0000*
Mean	*0.6151*	*0.7879*	*0.7832*	*0.8894*	*0.5000*	*0.0833*	*0.4167*	*0.1712*
S.D.	*4.0837*	*3.8135*	*4.3261*	*4.4887*	*0.5008*	*0.2768*	*0.4939*	*0.1328*
Min	*-16.4558*	*-18.5698*	*25.4284*	*-27.2430*	*0.0000*	*0.0000*	*0.0000*	*0.0079*
25%	*-1.4663*	*-1.2738*	*-1.0840*	*-1.2958*	*0.0000*	*0.0000*	*0.0000*	*0.0699*
50%	*1.0755*	*0.8833*	*0.8838*	*1.1081*	*0.5000*	*0.0000*	*0.0000*	*0.1368*
75%	*3.1374*	*3.0761*	*3.0846*	*3.2747*	*1.0000*	*0.0000*	*1.0000*	*0.2454*
Max	*12.5702*	*12.4548*	*19.6697*	*17.2612*	*1.0000*	*1.0000*	*1.0000*	*0.4946*

*Table 1: descriptive Statistics for the returns of the ETFs, * effect period excludes January.*

Methodology

Classical Linear Regression Model Assumptions

For all the models that we constructed below, we considered the following Classical Linear Regression Model (CLRM) assumptions, to ensure that our results were consistent and Best Linear Unbiased Estimators(BLUE).

Assumption #1: Errors Have Zero Mean

$$E(u_t) = 0$$

We obtained the mean of the residuals of the model. If the mean was very close to zero, then this assumption is not violated.

Assumption #2: Homoskedasticity

$$var(u_t) = \sigma^2 < \infty$$

This assumes that the variance of the error term is constant across time. If it is violated, the errors are said to be 'heteroskedastic'. We performed the Breusch-Pagan (Breusch and Pagan, 1979) test for it. It is a χ^2 test with k degrees of freedom and if the test statistic exceeds the critical test statistic value, then the null hypothesis is violated and hence, we reject the assumption of homoskedasticity. Generally, we addressed the violations of the homoskedasticity assumption by using robust standard errors.

Assumption #3: No Serial Correlation

$$cov(u_i, u_j\ ) = 0\ for\ i \neq j$$

This states that there should be no covariance between the error terms from different time periods. The Breusch-Godfrey test (Breusch and Godfrey 1981) uses the LM test statistic which, if greater than Critical value of the test statistic will violate the assumption. We did not find any evidence of serial correlation in any of our models, so we did not need to adjust our models to address any violations of this assumption.

Assumption #4: Explanatory Variables are Non-Stochastic and Suffer No Multicollinearity

The dummy variables we created are non-stochastic and do not have collinear relationships with each other.

Assumption #5: The Error Term is Normally Distributed

$$u_t \sim N(0, \sigma^2)$$

To test this assumption, we performed the Bera-Jarque test to use the property of the normally distributed random variable characterized by the first two moments (Jarque, 1980). The null hypothesis of normal assumption is rejected if the BJ test statistic exceeds the critical value and hence assumption is violated. This assumption was violated in many of our models, however, given that our sample size was quite large (over 200 monthly returns), we expect that this had a very minimal effect on our results. This is because with large sample sizes, the central limit theorem ensures that the test statistics will asymptotically follow the appropriate distribution.

Models

Standard Dummy Variable Approach for the Halloween Returns

We test the significance of the Halloween Effect on monthly returns by constructing the following regression model:

$$r_t = \mu + \alpha_1 S_t + u_t$$

where r_t is the monthly return, the dummy variable S_t is 1 if the month t is between November-April (inclusive) and 0 if the month t is between May-October (inclusive). μ represents the intercept term and u_t denotes the error term.

We tested this model for parameter instability using CUSUM and CUSUMQ tests. This was to ensure that the constant and coefficient did not have a statistically significant change over time. We tested any suspected structural breaks with the Chow test (Chow, 1960).

The January Effect Model

Given that excess returns during the November-April period was statistically significant in the XIU ETF, we tested for whether there was a confounding relationship between the Halloween Effect and the January Effect as Lucey & Zhao (2008) had concluded that the existence of the Halloween effect was uncertain and that mainly the excess returns reflected the January Effect. We added a dummy variable to the above regression model representing the January Effect:

$$r_t = \mu + \alpha_1 S_t^{adj} + \alpha_2 J_t + u_t$$

where r_t captures the monthly return, the dummy variable $S_t^{adj}$is 1 if the month t falls between November and April, excluding January, and 0 for if the month t is between May and October. J_t is 1 if the if the month t is January and 0 otherwise. μ represents the intercept term and u_t denotes the error term in the month t.

Halloween Strategy Effect Model

Finally, a Halloween Effect strategy was constructed to see if the Halloween Effect exhibited within the XIU ETF could be used to produce excess returns relative to the underlying index. This strategy consists of using the 6-month Canadian treasury bill returns for the return during the May to October period. During the November-April period, our strategy switches to investing in the XIU ETF, earning returns from the stock market. We conducted our analysis by comparing this to a standard buy and hold strategy where the XIU ETF is bought and not sold. We then constructed a CAPM model to determine if there are statistically significant excess returns that are produced from the Halloween Strategy over buying the XIU ETF index and holding it. To run the CAPM model as a regression with an alpha value (representing the excess returns), we re-arranged the CAPM model:

$$r_a = r_f + \beta(r_m - r_f) + \alpha$$
$$\leftrightarrow\ r_a - r_f = \alpha + \beta(r_m - r_f)$$

Using this model, we can interpret the constant as the alpha value, which represents the Halloween Effect strategy's excess returns over the buy-and-hold strategy over the overall market index. Two new variables were constructed in Python to run this regression, specifically the difference between the actual return and the risk-free rate, $r_a - r_f$ and the market risk premium, $r_m - r_f$.

Analysis and Results

Standard Dummy Variable Approach for the Halloween Returns

The results for the model were found that none of the Breusch-Pagan and Breusch-Godfrey Test had test statistic that could reject the null hypothesis at the 10% significance level. Thus, the model errors were homoscedastic and no evidence for serial correlation was found among all the ETF models. Jera-Barque test was violated for all the models but according to the central limit theorem, the sample sizes were sufficiently large hence the effect of this assumption violation are insignificant.

	XIU.TO	XIC.TO	XMD.TO	XRE.TO
Breusch-Pagan Test (p-value)	*0.5535 (0.4568)*	*0.1222 (0.7264)*	*0.9844 (0.7537)*	*1.7953 (0.1802)*
Breusch-Godfrey Test (p-value)	*6.7084 (0.3486)*	*8.1224 (0.2291)*	*2.9417(0.8162)*	*6.6322(0.3562)*

Table 2: Breusch-Pagan Test and Breusch-Godfrey Test

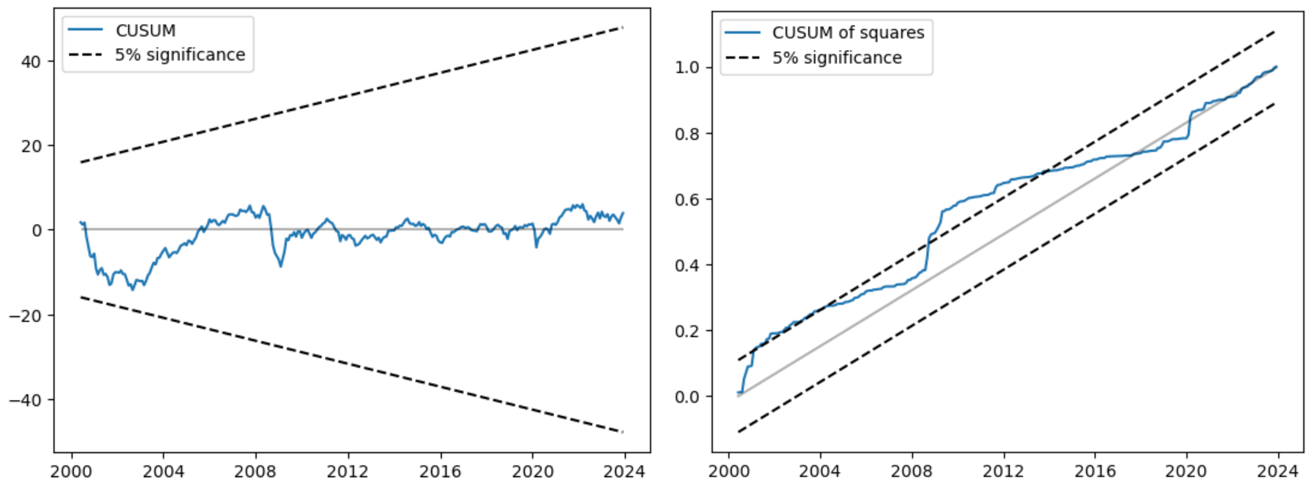


Figure 1: CUSUM and CUSUMSquared Plots for the Halloween effect returns

Performing the CUSUM and CUSUMSQ tests we suspected 2 structural breaks, which then tested by Chow test led to the p-values of 0.8549 and 0.3968 respectively, thus confirming that there were no statistically significant structural breaks at the 5% level of significance.

Performing the analysis for the standard dummy variable approach, we get that the constant term (μ) was not statistically significant for any of the ETF returns. XIU.TO had a statistically significant difference in returns through the period effect being significant at the 10% level of confidence (p-value=0.097<0.100) from the results reported under Table 3. XIC, XMD and XRE did not have any statistically significant results as to there being a difference in the returns between the two 6-month periods tested for at the 10% level.

		Coefficient (Standard Error)	**t-value**	**p-value (P>|t|)**
XIU.TO	*Constant*(μ)	*0.2153(0.339)*	*0.635*	*0.526*
	Period Effect (α_1)	*0.7996(0.480)**	*1.667*	*0.097**
XIC.TO	*Constant* (μ)	*0.4299 (0.349)*	*1.232*	*0.219*
	Period Effect (α_1)	*0.6463 (0.492)*	*1.313*	*0.191*
XMD.TO	*Constant* (μ)	*0.2846 (0.391)*	*0.728*	*0.467*
	Period Effect (α_1)	*0.8438 (0.552)*	*1.53*	*0.127*
XRE.TO	*Constant* (μ)	*0.3727 (0.408)*	*0.914*	*0.361*
	Period Effect (α_1)	*0.7369 (0.575)*	*1.281*	*0.201*

Table 3: Results of the Halloween returns model

The January Effect Model

The January Effect model was tested for the XIU ETF and not the other ones as the difference in returns for the Halloween period was only statistically significant for XIU, and not the other ETFs. Based on the model constructed for the January effect, we found the results in the Table 4. We also found no violations of the CLRM assumptions for it. Based on the test statistic and the associated p-values for the intercept and both effect terms, we conclude that there is no statistically significant evidence that the returns produced by Halloween period are correlated to the January Effect returns for the XIU ETF.

		Coefficient (standard error)	**t-value**	**p-value (P>|t|)**
XIU.TO	*Constant*(μ)	*0.2153 (0.340)*	*0.634*	*0.527*
	Period Effect (α_1)	*0.7274 (0.504)*	*1.444*	*0.150*
	January Effect(α_2)	*1.1604 (0.899)*	*1.291*	*0.198*

Table 4: Results of the Model investigating both Halloween and January effect simultaneously

Halloween Effect Strategy Model

CAPM Evaluation

We constructed the CAPM model for the excess return through the Halloween effect and the regressor being the market risk premium for the strategy. It violated the homoskedasticity assumption (Breusch-Pagan Test with the test-statistic 39.03 and p-value<0.01). Thus, as a correction, we performed the model with robust standard errors to rectify for the violation of the assumption. Note that once we used heteroskedastic-robust standard errors, the test statistics are compared against the normal distribution instead of the t-distribution, as shown in the table below. There were a variety of possible break quarters based on the CUSUM and CUSUMQ tests we conducted. When we conducted Chow tests, we found 1 statistically significant break around 2010 (the F-test statistic was 10.075 for this specific Chow test). As a result, we also provided 2 additional separate results for the same CAPM model equation but with 2000-2010 data and 2011-2023 data.

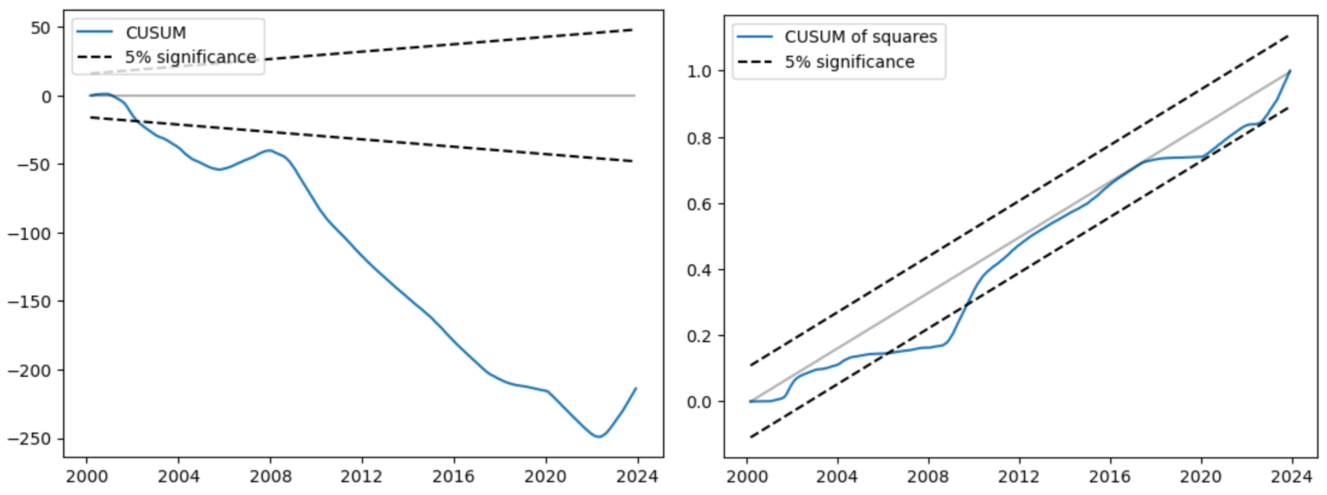


Figure 2: CUSUM and CUSUM Squared tests for the CAPM model

Hence, we analysed, the full CAPM model as well as the CAPM models with the data separated based on this structural break. The risk premium coefficient was significant at the 1% level of significance for the full model as well as the two models with the separated datasets. Breaking the model in two time periods, we were not able to find any statistically significant coefficient for the intercept (alpha) in any of the either full or reduced models. Note that this alpha value denotes the excess returns for the Halloween Effect trading strategy (see Methodology section for more details). Thus, we found no statistically significant evidence of excess returns for the Halloween Effect strategy for the XIU ETF.

		Coefficient (standard error)	z-value	p-value (P>|z|)
2001-2023	*Excess Returns* (α)	*0.2131 (0.131)*	*1.632*	*0.103*
	Halloween Risk Premium (β)	*0.4743 (0.065)****	*7.345*	*0.000*
2000-2010	*Excess Returns* (α)	*0.2065 (0.214)*	*0.214*	*0.334*
	Halloween Risk Premium (β)	*0.4173 (0.086)****	*4.874*	*0.000*
2011-2023	*Excess Returns* (α)	*0.1906 (0.153)*	*1.242*	*0.214*
	Halloween Risk Premium t (β)	*0.5628 (0.084)****	*6.705*	*0.000*

Table 5: Results of the CAPM model

Conclusion

Using the standard dummy variable approach to stocks within the TSX, we found that there is some statistically significant evidence of the Halloween Effect. It appears that during the months from May-October, monthly returns tend to be lower than returns from November-April. We did not find statistically significant evidence of the Halloween Effect at the 10% significance level within other ETFs that invest in more specific markets, such as the XIC, XMD and XRE indexes.

With this consideration, we investigated whether there is a confounding relationship between the January Effect, which says that January tends to have higher returns than other months of the year. While the inclusion of the dummy variable associated with this effect reduced the p-value of the adjusted Halloween Effect dummy variable coefficient, we still found some evidence of the Halloween Effect, just not at a 10% level of significance. Thus, it appears that the January Effect may drive some of the Halloween Effect, but not entirely as the coefficient remained similar, and the p-value only increased marginally.

Given the statistically significant higher returns within the XIU index from the months of November-April, we decided to construct a trading strategy that attempted to take advantage of this effect for profit opportunities. This strategy invested in Canadian Treasury bills during the worse performing months (May-October) and invested in the XIU index during the other six months of the year. After using over 20 years of data to determine the monthly returns for this strategy, we created a CAPM model to evaluate whether our trading strategy exhibited excess returns relative to its risk level. After considering additional trading costs, taxation, and the added benefit of dividends from the stock index, it is further shown that this calendar effect does not appear to exhibit profit opportunities in comparison to a buy-and-hold strategy in the XIU index.

References

Afik, Z., & Lahav, Y. (2015, January). A better ‘autopilot’ than Sell-in-May? 40 years in the US market. *Journal of Asset Management*, 16(1), 41–51. https://doi.org/10.1057/jam.2014.41

Alekneviciene, V., Klasauskaitė, V., & Aleknevičiūtė, E. (2022). Behavior of calendar anomalies and the adaptive market hypothesis: Evidence from the Baltic stock markets. *Journal of Baltic Studies*, 53, 187–210. https://doi.org/10.1080/01629778.2021.1990094

Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica,* 47(5), 1287–1294. https://doi.org/10.2307/1911963

Breusch, T. S., & Godfrey, L. G. (1981). A review of recent work on testing for autocorrelation in dynamic simultaneous models. In D. Currie, R. Nobay, & D. Peel (Eds.), Macroeconomic Analysis: Essays in Macroeconomics and Econometrics (pp. 63–105). Croom Helm.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regression. Econometrica, 26(3), 591–605.

Dichtl, H., & Drobetz, W. (2014). Are stock markets really so inefficient? The case of the “Halloween Indicator.” *Finance Research Letters*, 11(2), 112–121. https://doi.org/10.1016/j.frl.2013.10.001

Dimson, E., & Marsh, P. (1999). Murphy’s law and market anomalies. *Journal of Portfolio Management*, 25, 53–69.

Garrett, I., Kamstra, M. J., & Kramer, L. A. (2005). Winter blues and time variation in the price of risk. *Journal of Empirical Finance*, 12(2), 291–316.

Haggard, K. S., & Witte, H. D. (2010). The Halloween Effect: Trick or treat? *International Review of Financial Analysis*, 19(5), 379–387.

Jacobsen, B., & Zhang, C. Y. (2012). The Halloween indicator: Everywhere and all the time. Working Paper. Massey University.

Jarque, C., & Bera, A. (1980). Efficient tests for normality homoscedasticity and serial independence of regression residuals. *Econometric Letters*, 6, 255–259.

Lo, A. W. (2004). The Adaptive Markets Hypothesis. *The Journal of Portfolio Management*, 30, 15–29.

Lobão, J., & Costa, A. C. (2023). The Adaptive Dynamics of the Halloween Effect: Evidence from a 120-Year Sample from a Small European Market. *International Journal of Financial Studies*, 11(1), 13. https://doi.org/10.3390/ijfs11010013

Lucey, B. M., & Zhao, S. (2008). Halloween or January? Yet another puzzle. *International Review of Financial Analysis*, 17(5), 1055–1069.

O'Higgins, M., & Downs, J. (1990). Beating the Dow, A High-Return-Low-Risk method investing in Industrial Stocks with as little as $5000. Harper Collins.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1), 100–115.

Plastun, A., Sibande, X., Gupta, R., & Wohar, M. E. (2020). Halloween Effect in developed stock markets: A historical perspective. *International Economics*, 161, 130–138. https://doi.org/10.1016/j.inteco.2019.11.009

Schabek, T., & Castro, H. (2017). Sell not only in May. Seasonal effect on emerging and developed stock markets. *Dynamic Econometric Models*, 17(1), 5–18.

Urquhart, A., & McGroarty, F. (2014). Calendar effects, market conditions and the Adaptive Market Hypothesis: Evidence from long-run U.S. data. *International Review of Financial Analysis*, 35, 154–166.

FIN4100_Term_Project_Code_Output

April 5, 2024

```
[ ]: # Importing Libraries
     import matplotlib.pyplot as plt
     import yfinance as yf
     import pandas as pd
     import statsmodels.api as sm
     import numpy as np
     import statsmodels.regression.recursive_ls as rls
     import datetime
     import statsmodels.stats.diagnostic as dg
```

```
[ ]: # Getting Data, note date format is in 'YYYY-MM-DD'
     prices = yf.download("XIU.TO", start="1999-12-01", end="2024-01-31",␣
      ↪interval="1mo")

     # Creating a Normal Index Column and a DataFrame
     prices.reset_index(inplace=True)

     # Dates say the first day of the month, but that should be ignored -- it is␣
      ↪reflecting the end of the month when it is calculating the price change
     # This is just an issue with downloading monthly data from Yahoo finance
     prices = pd.DataFrame({'date': prices['Date'], 'monthlyPrice': prices['Adj␣
      ↪Close']})

     # Obtaining percentage changes, dropping top row since it does not have a␣
      ↪percentage change
     prices['monthlyReturn'] = prices['monthlyPrice'].pct_change() * 100
     prices = prices.drop(0)

     # Dropping monthly price since it is not needed
     prices.drop(columns=['monthlyPrice'], inplace=True)

     # Should be 12 * 24 = 288 observations (monthly)
     len(prices)

     # Construct Halloween Effect Factor Variable
     #  along with it also constructing a January dummy variable for another test
     halloweenEffectFactor = []
```

```
for i in range(len(prices)):
    if prices.iloc[i]['date'].month > 4 and prices.iloc[i]['date'].month < 11:
        halloweenEffectFactor.append(0)
    else:
        halloweenEffectFactor.append(1)

prices['effectPeriod'] = halloweenEffectFactor

# Constructing a January dummy variable for another test
januaryEffectFactor = []

for i in range(len(prices)):
    if prices.iloc[i]['date'].month == 1:
        januaryEffectFactor.append(1)
    else:
        januaryEffectFactor.append(0)

prices['januaryEffect'] = januaryEffectFactor

#Halloween effect without January effect
halloweenEffectPostFactor = []


for i in range(len(prices)):
    if prices.iloc[i]['date'].month > 4 and prices.iloc[i]['date'].month < 11 :
        halloweenEffectPostFactor.append(0)
    elif prices.iloc[i]['date'].month == 1:
        halloweenEffectPostFactor.append(0)
    else:
        halloweenEffectPostFactor.append(1)

prices['effectPeriodAdjusted'] = halloweenEffectPostFactor


# Set index back to the date variable for better display in graphs
prices.set_index('date', inplace=True)

# Getting Risk Free Rate (Bank of Canada 1 Month Treasury Bill Yield, Monthly␣
 ↪Rate)

# Create an adjusted returns data frame that includes the risk free rate for␣
 ↪Halloween effect periods

# Using Bank of Canada T-Bill Data
tbillReturns = pd.read_csv("tbill_all.csv")
```

```
# Cleaning Data to Obtain 1-Month T-Bill Data
tbillReturns = tbillReturns[["date", "V80691345"]]
tbillReturns.rename(columns={"date": "date", "V80691345": "sixMoTBillYield"},␣
 ↪inplace=True)
tbillReturns.dropna(inplace=True)
tbillReturns.reset_index(inplace=True, drop=True)

# Changing Date Days to First of Month to Make it Easier to Average Values

tbillReturns['date'] = pd.to_datetime(tbillReturns['date'])

newDates = []

for i in range(len(tbillReturns)):
    newDates.append(datetime.date(tbillReturns.iloc[i]['date'].year,␣
 ↪tbillReturns.iloc[i]['date'].month, 1))

tbillReturns['date'] = newDates

# Averaging All Yields from the Same Month to Create One Value
tbillReturns = tbillReturns.groupby(['date']).mean()

# Dropping Final Few Rows that are passed the time period of the XIU ETF
for i in range(3):
    tbillReturns.drop(tbillReturns.index[len(tbillReturns) - 1], inplace=True)

# De-annualize Risk Free Rate
tbillReturns['sixMoTBillYield'] /= 12

# Add Risk Free Rate to prices DataFrame for Comparison
prices['monthlyRiskFreeRate'] = tbillReturns['sixMoTBillYield']

prices.dropna(inplace=True)

# Viewing dataframe with risk free rate data
prices.head(8)
```

```
[*********************100%%***********************]  1 of 1 completed
```

[ ]:
```
            monthlyReturn  effectPeriod  januaryEffect  effectPeriodAdjusted
date
2000-01-01       1.208480             1              1                     0  \
2000-02-01       4.278614             1              0                     1
2000-03-01       9.064881             1              0                     1
2000-04-01      -1.603264             1              0                     1
2000-05-01       0.715572             0              0                     0
```

```
2000-06-01      10.834780            0             0                     0
2000-07-01       3.094705            0             0                     0
2000-08-01       7.092701            0             0                     0

            monthlyRiskFreeRate
date
2000-01-01             0.445833
2000-02-01             0.445833
2000-03-01             0.452500
2000-04-01             0.467500
2000-05-01             0.494667
2000-06-01             0.484375
2000-07-01             0.479583
2000-08-01             0.479000
```

[ ]:
```
# Constructing another DataFrame of different ETF prices (non XIU.TO), so they␣
 ↪can be tested for the Halloween Effect

otherETFs = ["XIC.TO", "XMD.TO", "XRE.TO"]

# Getting Data, note date format is in 'YYYY-MM-DD'
# Note the smaller available sample size of these ETFs
otherPrices = yf.download(otherETFs, start="2002-12-01", end="2024-01-31",␣
 ↪interval="1mo")

# Creating a Normal Index Column and a DataFrame
otherPrices.reset_index(inplace=True)

tempOtherPrices = pd.DataFrame(data={"date": otherPrices['Date']})

# Constructing a DataFrame with only the adjusted close prices
for i in otherETFs:
    tempOtherPrices[i] = otherPrices['Adj Close'][i]


otherPrices = tempOtherPrices

# Replacing ETF Closing Prices with their Return
# Obtaining percentage changes, dropping top row since it does not have a␣
 ↪percentage change
for i in otherETFs:
    otherPrices[i] = otherPrices[i].pct_change() * 100

otherPrices = otherPrices.drop(0)

# Construct Halloween Effect Factor Variable
#  along with it also constructing a January dummy variable for another test
```

```
halloweenEffectFactor = []

for i in range(len(otherPrices)):
    if otherPrices.iloc[i]['date'].month > 4 and otherPrices.iloc[i]['date'].
 ↪month < 11:
        halloweenEffectFactor.append(0)
    else:
        halloweenEffectFactor.append(1)


otherPrices['effectPeriod'] = halloweenEffectFactor

otherPrices.set_index('date', inplace=True)

# Viewing dataframe
otherPrices.head(8)
```

```
[*********************100%%**********************]  3 of 3 completed
```

[ ]:

```
              XIC.TO    XMD.TO    XRE.TO  effectPeriod
date
2003-01-01 -0.721692 -0.329308  3.484330             1
2003-02-01  1.216508 -2.105230  1.025667             1
2003-03-01 -3.485559 -4.659551 -3.553269             1
2003-04-01  3.888699  5.871434  3.611301             1
2003-05-01  4.337402  5.581965  3.092738             0
2003-06-01  1.270133  3.037142  0.500113             0
2003-07-01  3.582642  2.739031  6.165782             0
2003-08-01  4.336235  3.415179  1.913948             0
```

[ ]:

```
# Construct Regression to Test if Halloween Effect Exists

exogSimple = sm.add_constant(data=prices[['effectPeriod']])
simpleModel = sm.OLS(prices['monthlyReturn'], exogSimple).fit()

simpleModel.summary()

# Statistically significant evidence against null hypothesis with Jera-Barque␣
 ↪test, but sample size is quite large so this shouldn't matter too much for␣
 ↪our results
```

[ ]:

Dep. Variable:	monthlyReturn	**R-squared:**	0.010
Model:	OLS	**Adj. R-squared:**	0.006
Method:	Least Squares	**F-statistic:**	2.777
Date:	Fri, 05 Apr 2024	**Prob (F-statistic):**	0.0967
Time:	22:27:50	**Log-Likelihood:**	-811.98
No. Observations:	288	**AIC:**	1628.
Df Residuals:	286	**BIC:**	1635.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.2153	0.339	0.635	0.526	-0.452	0.883
effectPeriod	0.7996	0.480	1.667	0.097	-0.145	1.744

Omnibus:	39.614	**Durbin-Watson:**	1.809
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	75.544
Skew:	-0.741	**Prob(JB):**	3.94e-17
Kurtosis:	5.024	**Cond. No.**	2.62

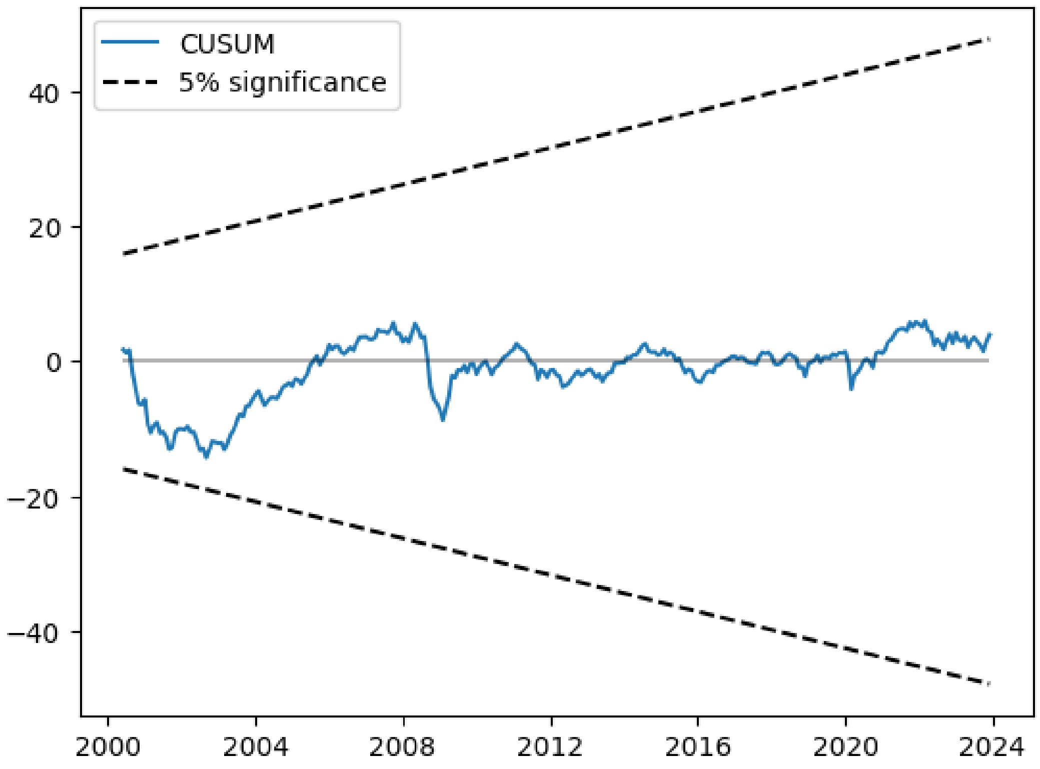
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

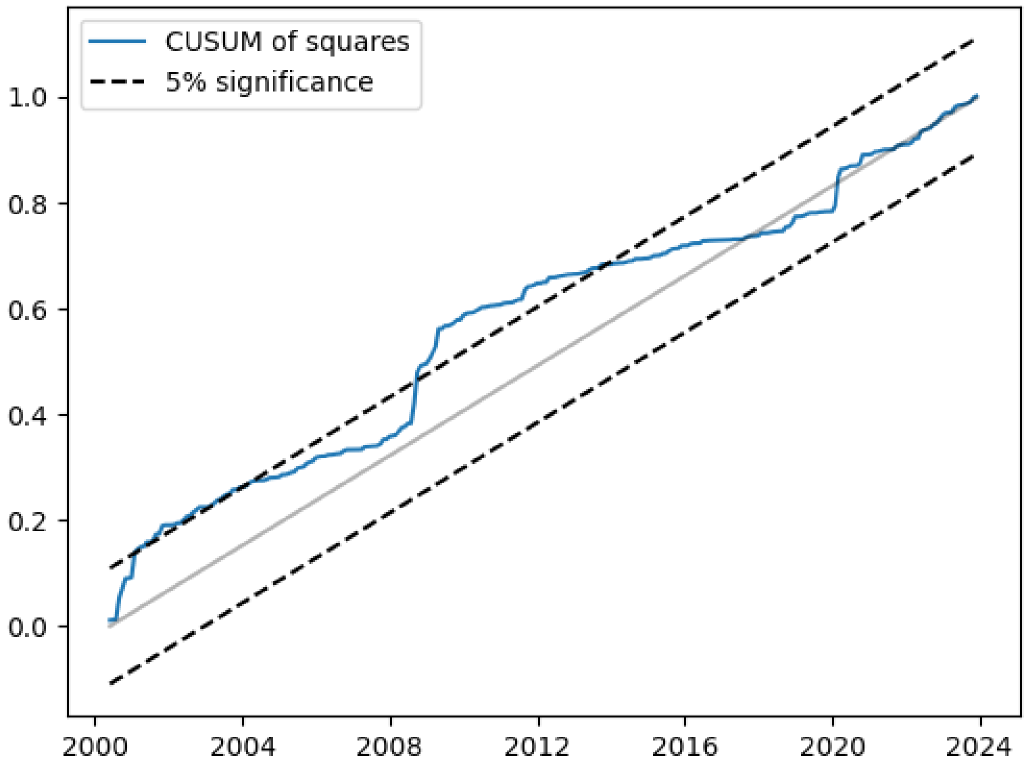
```
[ ]: # Test for CLRM model assumptions

# Test to See that residuals add up to zero
print(simpleModel.resid.mean())

# Test for to see if homoskedasticity assumption is violated
# Set robust = False as I want to test that residuals are normally distributed␣
 ↪(original Breush-Pagan)
heteroskedasticityTest = dg.het_breuschpagan(simpleModel.resid, exogSimple,␣
 ↪robust=False)

print(heteroskedasticityTest)

# Does not violate homoskedasticity assumption

# # Test for autocorrelation (testing across effect period length of 6 months)
autocorrelationTest = dg.acorr_breusch_godfrey(simpleModel, nlags=6)

print(autocorrelationTest)
# No statistically significant evidence of autocorrelation using last 6 months␣
 ↪of data
```

```
2.220446049250313e-16
(0.553503263467519, 0.4568897851942031, 0.2734336285678295, 0.6014425658746108)
(6.708446258242244, 0.34865333467690346, 1.1129407236736633,
0.35480893534565383)
```

```
[ ]: # Conduct parameter instability tests for the model above
     simpleModelRecursive = rls.RecursiveLS(prices['monthlyReturn'], exogSimple).
      ↪fit()

     # Plotting CUSUM and CUSUM^2 graphs
     cusumResults = simpleModelRecursive.plot_cusum(alpha=0.05, legend_loc='upper␣
      ↪left')
     cusumSquaredResults = simpleModelRecursive.plot_cusum_squares(alpha=0.05,␣
      ↪legend_loc='upper left')

     # It appears that according the CUSUM^2 graph, there is some parameter␣
      ↪instability. I will test to verify this.
```

```
c:\Users\isss1\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```





[ ]:
```
# Testing to verify parameter instability
# Break quarters appear to be from 2000-2008, 2009-2014, 2015-2023

# 2000-2008
pricesGroup1 = prices.iloc[:108]
exogPricesGroup1 = sm.add_constant(data=pricesGroup1[['effectPeriod']])
pricesGroup1Model = sm.OLS(pricesGroup1['monthlyReturn'], exogPricesGroup1).
  ↪fit()



# 2009-2014
pricesGroup2 = prices.iloc[108:180]
exogPricesGroup2 = sm.add_constant(data=pricesGroup2[['effectPeriod']])
pricesGroup2Model = sm.OLS(pricesGroup2['monthlyReturn'], exogPricesGroup2).
  ↪fit()

# 2015-2023
pricesGroup3 = prices.iloc[180:]
exogPricesGroup3 = sm.add_constant(data=pricesGroup3[['effectPeriod']])
pricesGroup3Model = sm.OLS(pricesGroup3['monthlyReturn'], exogPricesGroup3).
  ↪fit()
```

```
# # Using Chow Tests to Determine if These Periods have␣
↪statistically-significant parameter instability
k = 1

# # Compare 2000-2008 data vs 2009-2014 data
chowTest1Observations = pricesGroup1.shape[0] + pricesGroup2.shape[0]
chowTest1SecondDF = chowTest1Observations - 2 * k

# Obtain SSR for combined model
testGroup1 = prices.iloc[0:180]
exogTestGroup1 = sm.add_constant(data=testGroup1[['effectPeriod']])
testGroup1Model = sm.OLS(testGroup1['monthlyReturn'], exogTestGroup1).fit()

# # Calculating Chow Test in 2 Steps
chowTest1 = (testGroup1Model.ssr - (pricesGroup1Model.ssr + pricesGroup2Model.
↪ssr)) / k
chowTest1 = chowTest1 / ((pricesGroup1Model.ssr + pricesGroup2Model.ssr)/␣
↪chowTest1Observations)

print("Result of the Chow Test (comparing 2000-2008 vs 2009-2014) is: " +␣
↪str(chowTest1))

# Compare 2009-2014 data vs 2015-2023 data
chowTest2Observations = pricesGroup2.shape[0] + pricesGroup3.shape[0]
chowTest2SecondDF = chowTest2Observations - 2 * k

# Obtain SSR for combined model
testGroup2 = prices.iloc[108:]
exogTestGroup2 = sm.add_constant(data=testGroup2[['effectPeriod']])
testGroup2Model = sm.OLS(testGroup2['monthlyReturn'], exogTestGroup2).fit()

# Calculating Chow Test in 2 Steps
chowTest2 = (testGroup2Model.ssr - (pricesGroup2Model.ssr + pricesGroup3Model.
↪ssr)) / (pricesGroup2Model.ssr + pricesGroup3Model.ssr)
chowTest2 = chowTest2 * (chowTest2SecondDF / k)

print("Result of the Chow Test (comparing 2009-2014 vs 2015-2023) is: " +␣
↪str(chowTest2))

# Did not detect structural breaks at the 5% level, despite the CUSUM^2 results
```

```
Result of the Chow Test (comparing 2000-2008 vs 2009-2014) is:
0.8549516985937874
Result of the Chow Test (comparing 2009-2014 vs 2015-2023) is:
0.39689866027323767
```

```
[ ]: # Test a some other ETFs to see if the halloween effect (with the simple␣
     ↪regression model) still occurs in specific markets of the TSX
     # Note the smaller sample size
     for i in otherETFs:
         exogSimple = sm.add_constant(data=otherPrices[['effectPeriod']])
         simpleModel = sm.OLS(otherPrices[i], exogSimple).fit()
         print(simpleModel.summary())

         # Test for CLRM model assumptions

         # Test to See that residuals add up to zero
         print(simpleModel.resid.mean())

         # Test for to see if homoskedasticity assumption is violated
         # Set robust = False as I want to test that residuals are normally␣
     ↪distributed (original Breush-Pagan)
         heteroskedasticityTest = dg.het_breuschpagan(simpleModel.resid, exogSimple,␣
     ↪robust=False)

         print(heteroskedasticityTest)

         # Does not violate homoskedasticity assumption

         # # Test for autocorrelation (testing across effect period length of 6␣
     ↪months)
         autocorrelationTest = dg.acorr_breusch_godfrey(simpleModel, nlags=6)

         print(autocorrelationTest)
         # No statistically significant evidence of autocorrelation using last 6␣
     ↪months of data
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 XIC.TO   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     1.723
Date:                Fri, 05 Apr 2024   Prob (F-statistic):              0.191
Time:                        22:27:50   Log-Likelihood:                -703.32
No. Observations:                 253   AIC:                             1411.
Df Residuals:                     251   BIC:                             1418.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.4299      0.349      1.232      0.219      -0.257       1.117
effectPeriod     0.6463      0.492      1.313      0.191      -0.323       1.616
```

```
==============================================================================
Omnibus:                       57.758   Durbin-Watson:                   1.949
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              185.103
Skew:                          -0.948   Prob(JB):                     6.39e-41
Kurtosis:                       6.737   Cond. No.                         2.62
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
3.721221837478786e-16
(0.12241168330990604, 0.7264334024846564, 0.0423426677401238,
0.8371347632260674)
(8.124439564003197, 0.22912760935320764, 1.3547613664376856,
0.23362793292731523)
                            OLS Regression Results
==============================================================================
Dep. Variable:                 XMD.TO   R-squared:                       0.009
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     2.341
Date:                Fri, 05 Apr 2024   Prob (F-statistic):              0.127
Time:                        22:27:50   Log-Likelihood:                -732.04
No. Observations:                 253   AIC:                             1468.
Df Residuals:                     251   BIC:                             1475.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.2846      0.391      0.728      0.467      -0.485       1.054
effectPeriod     0.8438      0.552      1.530      0.127      -0.242       1.930
==============================================================================
Omnibus:                       87.593   Durbin-Watson:                   1.887
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              614.200
Skew:                          -1.177   Prob(JB):                    4.25e-134
Kurtosis:                      10.261   Cond. No.                         2.62
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
3.2297397080004555e-16
(0.09844144672024413, 0.7537080362954806, 0.021092367901702543,
0.8846441644388566)
(2.9417251377672295, 0.8161246236405642, 0.4803697985666641, 0.8226806944052569)
                            OLS Regression Results
==============================================================================
Dep. Variable:                 XRE.TO   R-squared:                       0.006
```

```
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     1.641
Date:                Fri, 05 Apr 2024   Prob (F-statistic):              0.201
Time:                        22:27:50   Log-Likelihood:                -742.70
No. Observations:                 253   AIC:                             1489.
Df Residuals:                     251   BIC:                             1496.
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.3727      0.408      0.914      0.361      -0.430       1.175
effectPeriod     0.7369      0.575      1.281      0.201      -0.396       1.870
==============================================================================
Omnibus:                       90.055   Durbin-Watson:                   1.800
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              642.910
Skew:                          -1.212   Prob(JB):                    2.48e-140
Kurtosis:                      10.424   Cond. No.                         2.62
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
2.5276223801742696e-16
(1.7953232140680484, 0.1802789253767555, 0.3785817044998204, 0.5389207183525313)
(6.632335220366771, 0.3561898876155624, 1.0992528385325286, 0.36334050534791834)
```

[ ]:
```python
# Adding January Effect to the simple model
# We want to see if the January effect is being confounded with the Halloween␣
↪Effect (since it is said that stocks go up more than usual in January)
# Want to see if the January Effect really causing the Halloween Effect.

januaryEffectExog = sm.add_constant(prices[['effectPeriodAdjusted',␣
↪'januaryEffect']])
januaryEffectModel = sm.OLS(prices['monthlyReturn'], januaryEffectExog)
januaryEffectModelResults = januaryEffectModel.fit()

print(januaryEffectModelResults.summary())

# Test for CLRM Model Assumptions
# Test to See that residuals add up to zero
print(januaryEffectModelResults.resid.mean())

# Test for to see if homoskedasticity assumption is violated
# Set robust = False as I want to test that residuals are normally distributed␣
↪(original Breush-Pagan)
```

```
heteroskedasticityTest = dg.het_breuschpagan(januaryEffectModelResults.resid,␣
↪januaryEffectExog, robust=False)

print(heteroskedasticityTest)

# Does not violate homoskedasticity assumption

# # Test for autocorrelation (testing across effect period length of 6 months)
autocorrelationTest = dg.acorr_breusch_godfrey(januaryEffectModelResults,␣
↪nlags=6)

print(autocorrelationTest)
# No statistically significant evidence of autocorrelation using last 6 months␣
↪of data
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          monthlyReturn   R-squared:                       0.010
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     1.498
Date:                Fri, 05 Apr 2024   Prob (F-statistic):              0.225
Time:                        22:27:50   Log-Likelihood:                -811.87
No. Observations:                 288   AIC:                             1630.
Df Residuals:                     285   BIC:                             1641.
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
const                   0.2153      0.340      0.634      0.527      -0.453
0.884
effectPeriodAdjusted    0.7274      0.504      1.444      0.150      -0.264
1.719
januaryEffect           1.1604      0.899      1.291      0.198      -0.609
2.930
==============================================================================
Omnibus:                       39.065   Durbin-Watson:                   1.804
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.875
Skew:                          -0.735   Prob(JB):                     9.08e-17
Kurtosis:                       4.999   Cond. No.                         4.26
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

```
specified.
-6.41462192005646e-16
(1.6012910537024254, 0.449039003804509, 0.39737685553367214, 0.6724524885549918)
(6.980616225128305, 0.32264352390470025, 1.155075675237077, 0.3307015736147754)
```

[ ]:
```
# Construct Strategy Returns, one for the standard buy and hold strategy and␣
↪one for the Halloween Effect Strategy

# Buy and hold strategy already exists (monthly market returns) as a variable
# So I need to construct a Halloween Effect trading strategy variable
# This variable will hold stocks from November-April (inclusive) and hold the␣
↪risk free treasury bills from May-October(inclusive)

halloweenStrategyMonthlyReturn = []

# Creating a new data frame that comparisons the standard buy and hold strategy␣
↪with the halloween effect buying strategy
prices.reset_index(inplace=True)

for i in range(len(prices)):
    if prices.iloc[i]['date'].month > 4 and prices.iloc[i]['date'].month < 11:
        halloweenStrategyMonthlyReturn.append(prices.
↪iloc[i]['monthlyRiskFreeRate'])
    else:
        halloweenStrategyMonthlyReturn.append(prices.iloc[i]['monthlyReturn'])

# Setting index to date so we can see it on the CUSUM graphs
prices.set_index('date', inplace=True)

# Adding Halloween Strategy returns
prices['halloweenStrategyReturn'] = halloweenStrategyMonthlyReturn

# Constructing a variable that subtracts the risk free from the returns of the␣
↪portfolio
prices['halloweenStrategyDifference'] = prices['halloweenStrategyReturn'] -␣
↪prices['monthlyRiskFreeRate']

# Constructing a market risk premium variable
prices['halloweenStrategyRiskPremium'] = prices['monthlyReturn'] -␣
↪prices['monthlyRiskFreeRate']

prices.head(8)
```

[ ]:
```
            monthlyReturn  effectPeriod  januaryEffect  effectPeriodAdjusted
date
2000-01-01       1.208480             1              1                     0  \
2000-02-01       4.278614             1              0                     1
```

```
2000-03-01       9.064881          1              0                    1
2000-04-01      -1.603264          1              0                    1
2000-05-01       0.715572          0              0                    0
2000-06-01      10.834780          0              0                    0
2000-07-01       3.094705          0              0                    0
2000-08-01       7.092701          0              0                    0

            monthlyRiskFreeRate  halloweenStrategyReturn
date
2000-01-01             0.445833                 1.208480  \
2000-02-01             0.445833                 4.278614
2000-03-01             0.452500                 9.064881
2000-04-01             0.467500                -1.603264
2000-05-01             0.494667                 0.494667
2000-06-01             0.484375                 0.484375
2000-07-01             0.479583                 0.479583
2000-08-01             0.479000                 0.479000

            halloweenStrategyDifference  halloweenStrategyRiskPremium
date
2000-01-01                     0.762647                      0.762647
2000-02-01                     3.832781                      3.832781
2000-03-01                     8.612381                      8.612381
2000-04-01                    -2.070764                     -2.070764
2000-05-01                     0.000000                      0.220906
2000-06-01                     0.000000                     10.350405
2000-07-01                     0.000000                      2.615122
2000-08-01                     0.000000                      6.613701
```

[ ]:
```python
# CAPM Model Evaluation

# Using variable constructed above, I will run create a CAPM model to evaluate␣
↪the performance of the Halloween Strategy
# Want to see what the Alpha is, so I will add a constant to the CAPM model to␣
↪represent an alpha value
exogCapmModel = sm.add_constant(prices['halloweenStrategyRiskPremium'])
capmModel = sm.OLS(prices['halloweenStrategyDifference'], exogCapmModel).fit()

# The constant is 0.2131, which is the alpha value

# Test for CLRM Model Assumptions
# Test to See that residuals add up to zero
print(capmModel.resid.mean())

# Test for to see if homoskedasticity assumption is violated
# Set robust = False as I want to test that residuals are normally distributed␣
↪(original Breush-Pagan)
```

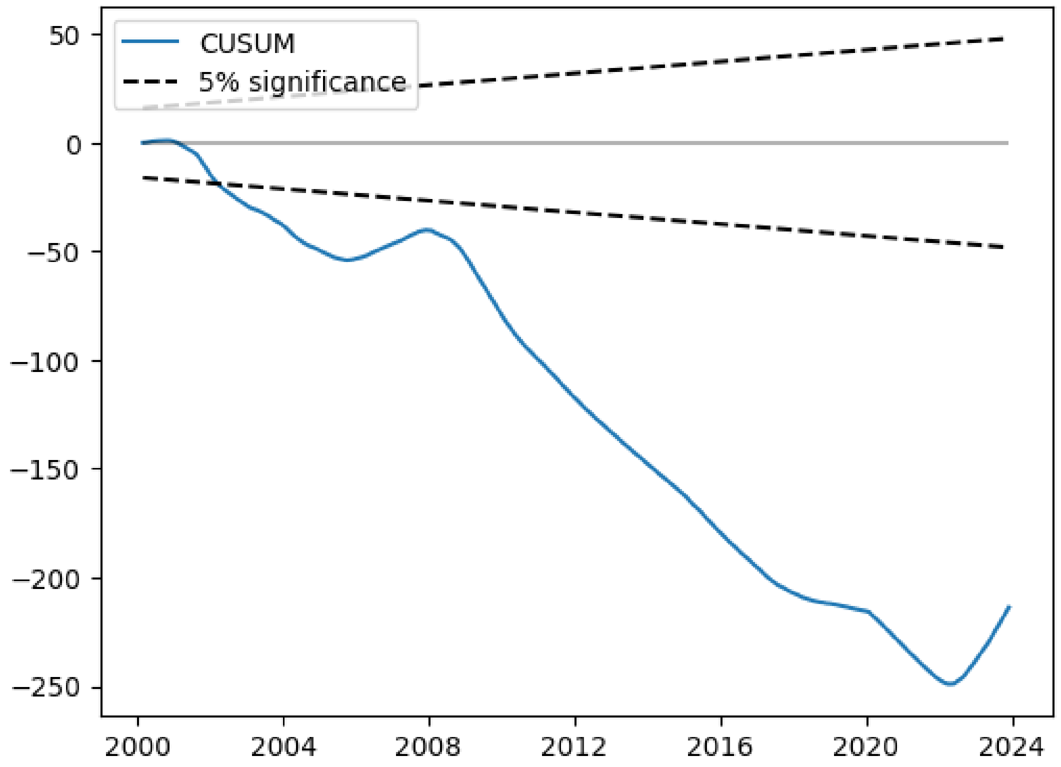
```
heteroskedasticityTest = dg.het_breuschpagan(capmModel.resid, exogCapmModel,␣
↪robust=False)

print(heteroskedasticityTest)

# Homoskedasticity assumption is violated -- must re-compute test with robust␣
↪standard errors

capmModel = sm.OLS(prices['halloweenStrategyDifference'], exogCapmModel).
↪fit(cov_type="HC1")
print(capmModel.summary())

# # Test for autocorrelation (testing across effect period length of 6 months)
autocorrelationTest = dg.acorr_breusch_godfrey(capmModel, nlags=6)

print(autocorrelationTest)
# No statistically significant evidence of autocorrelation using last 6 months␣
↪of data
```

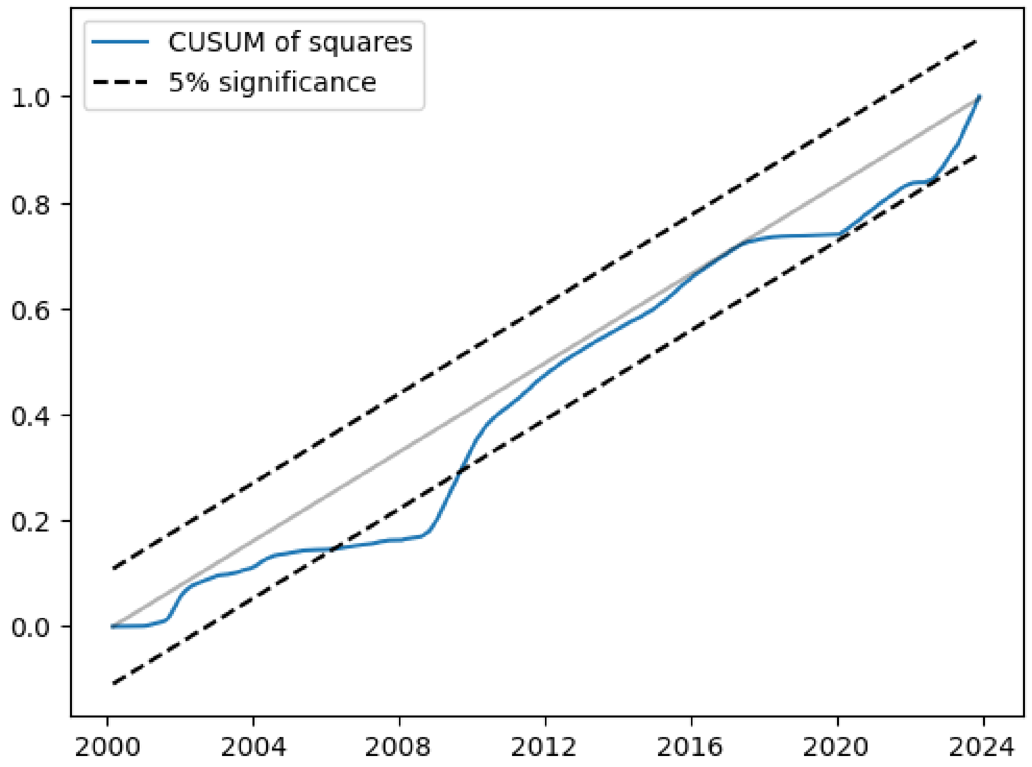
```
-3.0839528461809902e-18
(39.03724132240836, 4.157976177482545e-10, 22.585505210171352,
3.1873848596675246e-06)
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:     halloweenStrategyDifference   R-squared:
0.474
Model:                                     OLS   Adj. R-squared:
0.472
Method:                          Least Squares   F-statistic:
53.95
Date:                         Fri, 05 Apr 2024   Prob (F-statistic):
2.15e-12
Time:                                 22:27:50   Log-Likelihood:
-614.28
No. Observations:                          288   AIC:
1233.
Df Residuals:                              286   BIC:
1240.
Df Model:                                    1
Covariance Type:                           HC1
================================================================================
================
                                  coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
```

```
const                              0.2131      0.131      1.632      0.103
-0.043       0.469
halloweenStrategyRiskPremium     0.4743      0.065      7.345      0.000
0.348       0.601
==============================================================================
Omnibus:                       14.285   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               35.073
Skew:                          -0.070   Prob(JB):                     2.42e-08
Kurtosis:                       4.704   Cond. No.                         4.14
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC1)
(7.885744790639148, 0.24659404347836691, 1.3137547151540423, 0.2509100442830801)
```

[ ]:
```python
# Test CAPM Model for Parameter Instability

# Conduct parameter instability tests for the model above
capmModelRecursive = rls.RecursiveLS(prices['monthlyReturn'], exogCapmModel).
 ↪fit()

# Plotting CUSUM and CUSUM^2 graphs
cusumResults = capmModelRecursive.plot_cusum(alpha=0.05, legend_loc='upper␣
 ↪left')
cusumSquaredResults = capmModelRecursive.plot_cusum_squares(alpha=0.05,␣
 ↪legend_loc='upper left')

# It appears that according the CUSUM^2 graph, there is some parameter␣
 ↪instability.
```

```
c:\Users\isss1\AppData\Local\Programs\Python\Python311\Lib\site-
packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
```


CUSUM
5% significance
50
0
−50
−100
−150
−200
−250
2000
2004
2008
2012
2016
2020
2024


CUSUM of squares
5% significance
1.0
0.8
0.6
0.4
0.2
0.0
2000
2004
2008
2012
2016
2020
2024

```
[ ]: # Evaluate for statistically significant parameter instabilty based on the␣
      ↪CUSUM and CUSUM^2 graph

     # Define Test Statistic Calculation
     def computeChowTest(pricesGroup1Model, pricesGroup2Model, testGroupModel,␣
      ↪chowTestObservations):
         chowTest = (testGroupModel.ssr - (pricesGroup1Model.ssr + pricesGroup2Model.
      ↪ssr)) / k
         chowTest = chowTest / ((pricesGroup1Model.ssr + pricesGroup2Model.ssr)/␣
      ↪chowTestObservations)

         return chowTest


     # From both of the CUSUM and CUSUM^2 tests we've identified the following break␣
      ↪quarters
     # 2000-2002, 2002-2006, 2007-2010, 2011-2024

     # 2000-2002
     pricesGroup1 = prices.iloc[:36]

     exogCapmP1 = sm.add_constant(pricesGroup1['halloweenStrategyRiskPremium'])
     capmModelP1 = sm.OLS(pricesGroup1['halloweenStrategyDifference'], exogCapmP1).
      ↪fit()

     # 2003-2006
     pricesGroup2 = prices.iloc[36:84]
     exogCapmP2 = sm.add_constant(pricesGroup2['halloweenStrategyRiskPremium'])
     capmModelP2 = sm.OLS(pricesGroup2['halloweenStrategyDifference'], exogCapmP2).
      ↪fit()

     # 2007-2010
     pricesGroup3 = prices.iloc[84:132]
     exogCapmP3 = sm.add_constant(pricesGroup3['halloweenStrategyRiskPremium'])
     capmModelP3 = sm.OLS(pricesGroup3['halloweenStrategyDifference'], exogCapmP3).
      ↪fit()

     # 2011-2024
     pricesGroup4 = prices.iloc[132:]
     exogCapmP4 = sm.add_constant(pricesGroup4['halloweenStrategyRiskPremium'])
     capmModelP4 = sm.OLS(pricesGroup4['halloweenStrategyDifference'], exogCapmP4).
      ↪fit()

     # Compare 2000-2002 vs 2003-2006
```

```
chowTest1Observations = pricesGroup1.shape[0] + pricesGroup2.shape[0]
chowTest1SecondDF = chowTest1Observations - 2 * k

# Obtain SSR for combined model
testGroup1 = prices.iloc[0:84]
exogTestGroup1 = sm.add_constant(testGroup1['halloweenStrategyRiskPremium'])
testGroup1Model = sm.OLS(testGroup1['halloweenStrategyDifference'],␣
 ↪exogTestGroup1).fit()

# Calculating Chow Test in 2 Steps
chowTest1 = computeChowTest(capmModelP1, capmModelP2, testGroup1Model,␣
 ↪chowTest1Observations)

print("Result of the Chow Test (comparing 2000-2002 vs 2003-2006) is: " +␣
 ↪str(chowTest1))

# Compare 2003-2006 vs 2007-2010
chowTest2Observations = pricesGroup3.shape[0] + pricesGroup4.shape[0]
chowTest2SecondDF = chowTest2Observations - 2 * k

# Obtain SSR for combined model
testGroup2 = prices.iloc[84:132]
exogTestGroup2 = sm.add_constant(testGroup2['halloweenStrategyRiskPremium'])
testGroup2Model = sm.OLS(testGroup2['halloweenStrategyDifference'],␣
 ↪exogTestGroup2).fit()

# Calculating Chow Test in 2 Steps
chowTest = computeChowTest(capmModelP2, capmModelP3, testGroup2Model,␣
 ↪chowTest2Observations)

print("Result of the Chow Test (comparing 2003-2006 vs 2007-2010) is: " +␣
 ↪str(chowTest2))

# Compare 2007-2010 vs 2011-2024
chowTest3Observations = pricesGroup3.shape[0] + pricesGroup4.shape[0]
chowTest3SecondDF = chowTest3Observations - 2 * k

# Obtain SSR for combined model
testGroup3 = prices.iloc[84:]
exogTestGroup3 = sm.add_constant(testGroup3['halloweenStrategyRiskPremium'])
testGroup3Model = sm.OLS(testGroup3['halloweenStrategyDifference'],␣
 ↪exogTestGroup3).fit()

# Calculating Chow Test in 2 Steps
chowTest3 = computeChowTest(capmModelP3, capmModelP4, testGroup3Model,␣
 ↪chowTest3Observations)
```

```
print("Result of the Chow Test (comparing 2007-2010 vs 2011-2024) is: " +␣
 ↪str(chowTest3))

# Found one structural break between 2010 and 2011
# Estimating the two different models with the data
pricesCapm1 = prices[:132]
pricesCapm2 = prices[132:]

# 2000-2010
exogCapm1 = sm.add_constant(pricesCapm1['halloweenStrategyRiskPremium'])
capmModel1 = sm.OLS(pricesCapm1['halloweenStrategyDifference'], exogCapm1).
 ↪fit(cov_type="HC1")

print(capmModel1.summary())

# 2011 - 2024
exogCapm2 = sm.add_constant(pricesCapm2['halloweenStrategyRiskPremium'])
capmModel2 = sm.OLS(pricesCapm2['halloweenStrategyDifference'], exogCapm2).
 ↪fit(cov_type="HC1")

print(capmModel2.summary())
```

```
Result of the Chow Test (comparing 2000-2002 vs 2003-2006) is:
1.5010101178234885
Result of the Chow Test (comparing 2003-2006 vs 2007-2010) is:
0.39689866027323767
Result of the Chow Test (comparing 2007-2010 vs 2011-2024) is:
10.075047779934657
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:     halloweenStrategyDifference   R-squared:
0.417
Model:                                     OLS   Adj. R-squared:
0.413
Method:                          Least Squares   F-statistic:
23.76
Date:                         Fri, 05 Apr 2024   Prob (F-statistic):
3.13e-06
Time:                                 22:27:51   Log-Likelihood:
-298.52
No. Observations:                          132   AIC:
601.0
Df Residuals:                              130   BIC:
606.8
Df Model:                                    1
```

```
Covariance Type:                     HC1
================================================================================
================
                                  coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
const                           0.2065      0.214      0.966      0.334
-0.212       0.626
halloweenStrategyRiskPremium    0.4173      0.086      4.874      0.000
0.250       0.585
==============================================================================
Omnibus:                        4.929   Durbin-Watson:                   1.975
Prob(Omnibus):                  0.085   Jarque-Bera (JB):                6.390
Skew:                          -0.135   Prob(JB):                       0.0410
Kurtosis:                       4.043   Cond. No.                         4.74
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC1)
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:     halloweenStrategyDifference   R-squared:
0.561
Model:                                     OLS   Adj. R-squared:
0.559
Method:                          Least Squares   F-statistic:
44.96
Date:                         Fri, 05 Apr 2024   Prob (F-statistic):
3.60e-10
Time:                                 22:27:51   Log-Likelihood:
-306.50
No. Observations:                          156   AIC:
617.0
Df Residuals:                              154   BIC:
623.1
Df Model:                                    1
Covariance Type:                           HC1
================================================================================
================
                                  coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
----------------
const                           0.1906      0.153      1.242      0.214
-0.110       0.491
halloweenStrategyRiskPremium    0.5628      0.084      6.705      0.000
```

```
0.398       0.727
==============================================================================
Omnibus:                        4.788   Durbin-Watson:                   2.061
Prob(Omnibus):                  0.091   Jarque-Bera (JB):                6.455
Skew:                          -0.074   Prob(JB):                       0.0396
Kurtosis:                       3.985   Cond. No.                         3.55
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC1)
```