Futures

2023-02-27

Contents

Exploring The Relationship Between Cryptocurrency Futures and Inflation, Stock Prices and The Market Indices **2**

Abstract . . . 2

Literature Review . . . 2

Datasets . . . 3

Importing Data . . . 3

Global Inflation Analysis . . . 4

Stock Prices Analysis . . . 6

Merging DataFrames . . . 18

Forecasting . . . 21

References . . . 26

Exploring The Relationship Between Cryptocurrency Futures and Inflation, Stock Prices and The Market Indices

Libraries

```
library(TSstudio)
library(tidyverse)
library(quantmod)
library(zoo)
library(janitor)
library(lubridate)
library(readr)
library(timetk)
library(fpp2)          # The forecasting OG
library(fpp3)          # The tidy version of fpp2
library(modeltime)     # The tidy forecasting newcomer
library(timetk)        # Companion to modeltime
library(parsnip)       # Common interface for specifying models
library(rsample)       # Training / Test data splitter
library(cowplot)
```

Abstract

The value of a particular cryptocurrency at a given time is represented through cryptocurrency futures contracts. These deals include traders agreeing to buy or sell a specific item at a defined price on a given future date. The holder of a position in a conventional futures contract is obligated to buy or sell the underlying asset at the contract price when the contract expires.

Traders could profit or lose based on their positions, such as long or short, and the price of the futures. When you believe the asset's price will rise, you can go long; conversely, if you believe it will fall, you can go short. For instance, if the price of bitcoin is $10,000 right now, you might buy (go long) or sell (go short) a futures contract in expectation of a price rise or fall. Platforms which allow for cryptocurrency futures trading a few, with the most recent being Binance.

Objectives

I was therefore curious in the following areas regarding cryptocurrency futures: 1. Does a correlation between the current global economy that is in inflation and the price of Crypto Futures ? 1. How are closely are the current cryptocurrency prices affecting the futures ? 2. What is the trend of Market Index and if there is a correlation between the crypocurrency price and its corresponding futures.

Literature Review

A previous study done on Cryptocurrency and the stock market index by (Gilana et al., 2020) had the following findings:

Using fractional integration techniques, the researchers examined the stochastic characteristics of six significant cryptocurrencies and their bilateral relationships with six stock market indices. According to the results of the univariate analysis, the unit root null hypothesis cannot be ruled out for Bitcoin and Ethereum, while the order of integration is found to be significantly higher than 1 for Litecoin, Ripple, and Stellar. For Tether, however, we find evidence in favor of mean reversion. With the exception of VIX, where mean reversion is observed, the results for the stock market indexes are more uniform and the unit root cannot be rejected in any of the series.

We show evidence that there is no cointegration between the six cryptocurrencies when considering bivariate data within the coins and testing for cointegration. Similar to this, when looking for evidence of cointegration between cryptocurrencies and stock market indexes, we discover evidence of none, suggesting that cryptocurrencies are independent of traditional financial and economic assets. The results of this study demonstrate the importance of cryptocurrencies in investor portfolios since they give investors a way to diversify their holdings, supporting the notion that cryptocurrencies are a new asset class for investments.

The study by Ahmed(2021), had the following findings: According to the results, the return on the US indices and crude oil (WTI) both illustrate the continuation of a negative and large leverage effect, whilst the cryptocurrency markets exhibit a positive asymmetric volatility effect. Additionally, all the other pairs evaluated both before and after the launch of Bitcoin Futures show indications of relatively weak dependency. Based on the Hedging Ratio and Mean-Variance Approach, this article suggests that, with the exception of WTI-Bitcoin, WTI-Dash, and WTI-Ethereum pairs, where the values of their hedge ratios are rather significant with regard to OLS regression, the investor should hold more conventional financial assets than digital assets in order to reduce risk while maintaining the same expected returns of the digital-conventional financial asset portfolio.

Datasets

The following datasets were obtained for sole the purpose of this study. The Global Infation Dataset from 1978-2022 is an open source data source that is provided freely on the Kaggle website, in the datasets sections. Because of the complexity of the global economy, it is essential to comprehend its trends and patterns in order to make wise choices regarding investments, policies, and other matters. Inflation, or the rate of price growth over time, is a significant issue that has an impact on the economy. The World Energy, Food, Consumer, and Producer Price Inflation dataset offers a thorough collection of inflation rates for 206 nations from 1970 to 2022, encompassing four significant economic sectors. The different iflation rates are averaged in the dataset and used. It is important to note that only the Rates from United States were used, since the trends in the US economy usually have a big impact on the Stock market.

The stock prices for Bitcoins, Bitcoin Futures, The SP500 index and the Shanghai Delayed Index were obtained for yahoo finance. The following steps were used to download the data from yahoo finance:

1. Look for the relevant Ticker for each of the stocks you are looking for instance; BTC for Bitcoin, and BTC = F for Bitcoin Futures
   2 .Search the ticker on yahoo finance
   3 . After getting the page for the ticker, navigate to the historical prices tab
   4 . Specify the range of dates you want to obtain data from
   5 . Once you see the option to download on the upper right corner of historical prices table, download the data.

The Adjusted closing prices were used for this analysis from each of the tickers. The distribution of the price of BTC futures is first explored and a univariate model is trained. The univariate model only considers the closing price of the BTC futures. The multivariate model considers the closing prices of the other stock indices and adds the average inflation for each year since 2017.

Importing Data

```
BTC_futures_data <- read_csv("data/BTC futures data.csv")
BTC_historical_data <- read_csv("data/BTC historical_data.csv")
Global_Inflation <- read_csv("data/Global Dataset of Inflation.csv")
Shanghai_delayed_Index <- read_csv("data/Shanghai delayed Index.csv")
SP500_Perfomance <- read_csv("data/SP500 Perfomance.csv")
```

```
btc = BTC_historical_data
btc = btc %>%
    rename("timestamp" = "...1")


btcf = BTC_futures_data
btcf = btcf %>%
    rename("timestamp" = "...1")
btcf <- slice(btcf, 1:(n() - 1))

btcf_df = btcf


sp = SP500_Perfomance
sp = sp %>%
    rename("timestamp" = "...1")
sp <- slice(sp, 1:(n() - 1))
sp_df = sp


sh = Shanghai_delayed_Index
sh <- slice(sh, 1:(n() - 1))
sh = sh %>%
    rename("timestamp" = "...1")
sh <- slice(sh, 1:(n() - 1))
sh_df = sh
```

Global Inflation Analysis

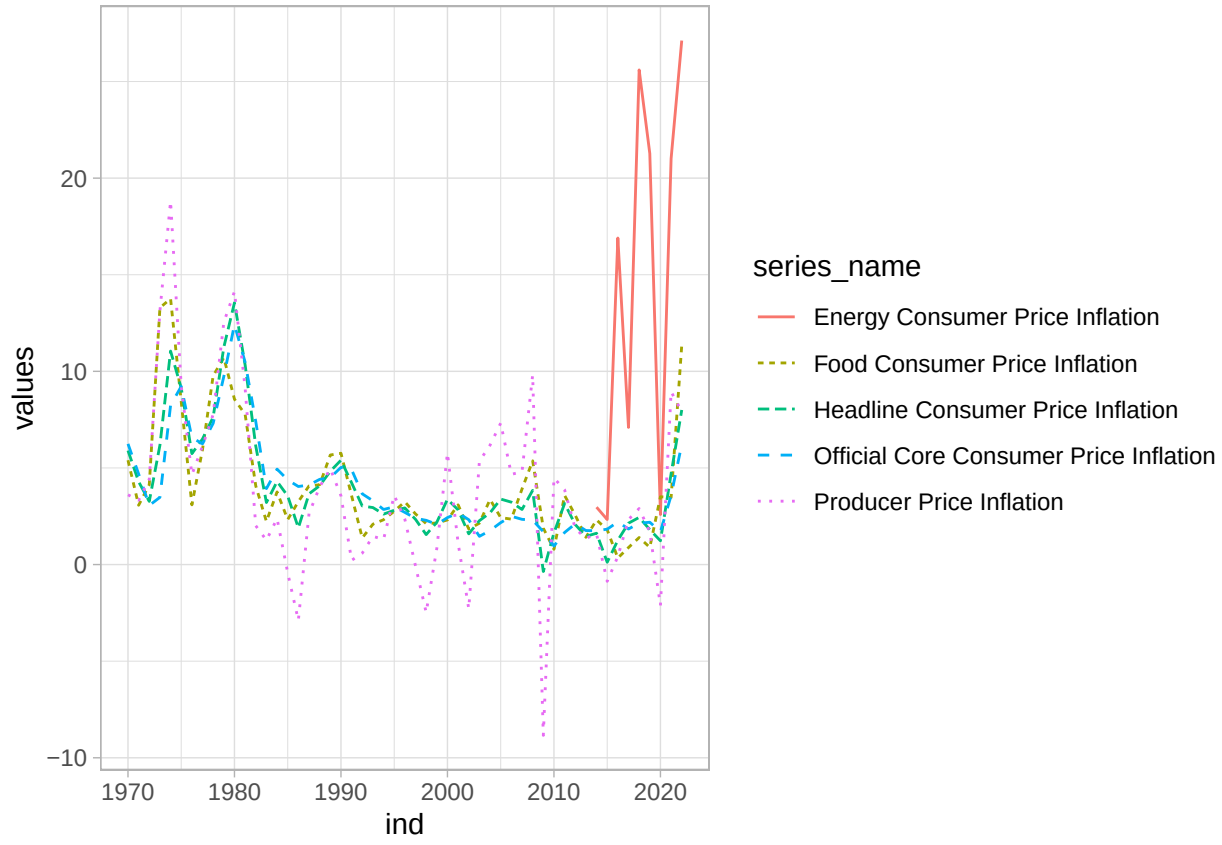
```
#INIT df
df = Global_Inflation
#change names to lower case and remove special characters
df = df %>% clean_names()
#only USA rates from
df = filter(df, df$country_code == "USA")
#remove redundant columns
df = df %>% select(-(country_code:indicator_type))
#change df from wide form to long form f
df_s  = cbind(df[1], stack(df[2:60]))
#change year indicator to numeric
df_s$ind<-gsub("x","",as.character(df_s$ind))
df_s =  df_s %>% mutate_at(c('ind', 'values'), as.numeric)
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
#remove null records
df_s = na.omit(df_s)
#convert todf
df_s = as_data_frame(df_s)
```

```
## Warning: `as_data_frame()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
```

```
# Visualization
ggplot(df_s, aes(x = ind, y = values)) +
  geom_line(aes(color = series_name, linetype = series_name)) + theme_light()
```



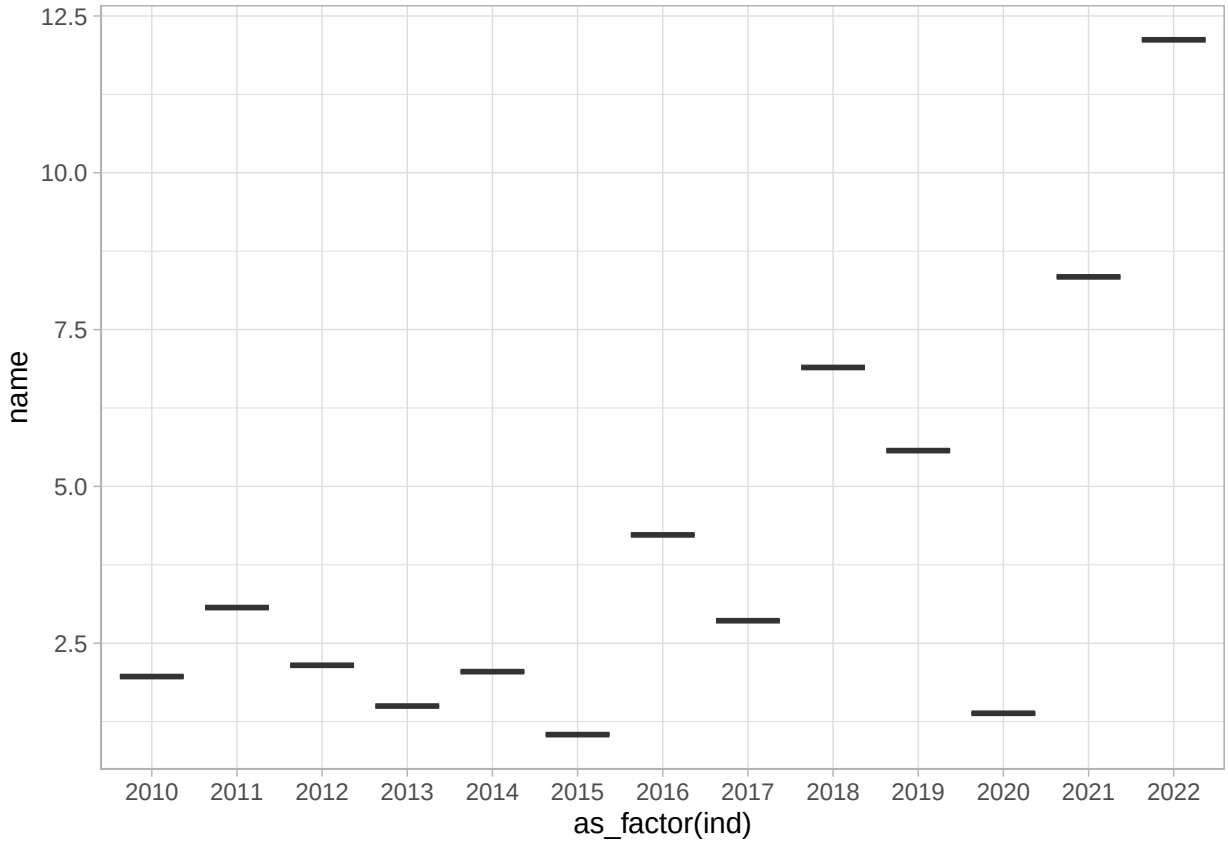
Note the inflation trend was going down since the 1978 except for the spike in 2008 and the sharp increase around 2020. Which is probably an impact of Covid-19 pandemic.

```
df_s = filter(df_s, ind >= 2010)
df_s = df_s %>% select(-c(series_name))
df_s = df_s %>%
  group_by(ind) %>%
  summarise_at(vars(values), list(name = mean))
df_s
```

```
## # A tibble: 13 x 2
##      ind  name
##    <dbl> <dbl>
##  1  2010  1.97
##  2  2011  3.07
##  3  2012  2.15
##  4  2013  1.50
```

```
##  5  2014  2.05
##  6  2015  1.04
##  7  2016  4.23
##  8  2017  2.86
##  9  2018  6.90
## 10  2019  5.57
## 11  2020  1.38
## 12  2021  8.34
## 13  2022 12.1
```

```
# grouped boxplot
ggplot(df_s, aes(x=as_factor(ind), y=name)) +
    geom_boxplot() + theme_light()
```



Discussion:

Stock Prices Analysis

Univariate Analysis

```
# Convert ts to tibble
btc_df <- btc |>
```

```
    tk_tbl(rename_index = "timestamp") %>%
    mutate(timestamp = as_date(timestamp))
```

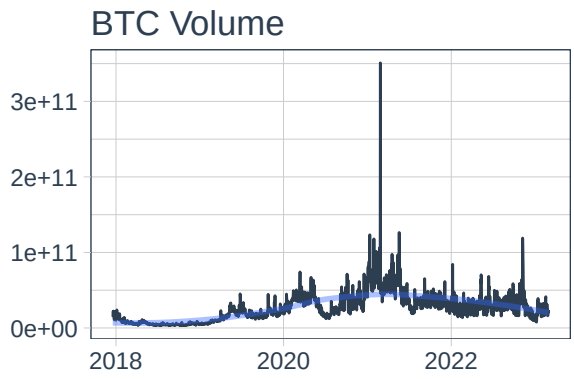
Bitcoin Stock Exploratory Data Analysis

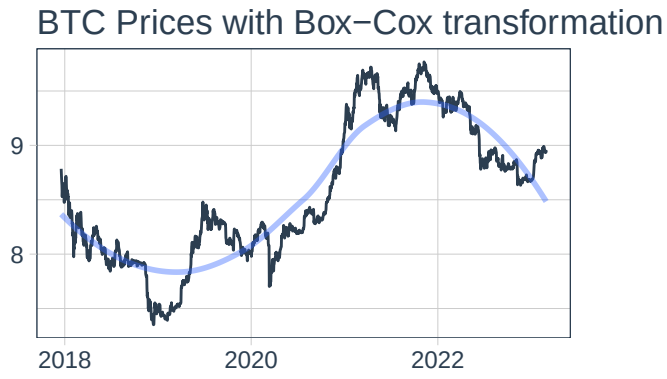
```
## Warning in tk_tbl.data.frame(btc, rename_index = "timestamp"): Warning: No index
## to preserve. Object otherwise converted to tibble successfully.
```

```
# Plot  dataset
e0 <- btc_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = volume,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Volume"
    )
#closing prices for BTC
e1 <- btc_df %>%
  plot_time_series(
      .date_var = timestamp,
      .value    = adjclose,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
      .title = "BTC Prices"
  )
#box cox transformation
# Get best lambda value to make seasonal fluctuations about the same
lambda <- btc_df %>%
    as_tsibble() %>%   # Use fabletools as a workaround
    features(adjclose, features=guerrero) %>%
    pull(lambda_guerrero)
```

```
## Using ‘timestamp‘ as index variable.
```

```
# Apply Box-Cox transformation to data using chosen lambda
btc_bc_tbl <- btc_df %>%
  mutate(box_cox = box_cox(adjclose, lambda))
# Plot data with Box-Cox transformation applied
e2 <- btc_bc_tbl %>%
    plot_time_series(
        .date_var = timestamp,
        .value = box_cox,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Prices with Box-Cox transformation"
    )
# Arrange plots
plot_grid(e0, e1, e2,nrow=2)
```





Note the naturally bellshaped trend that remains on the prices Box-cox Transformation

```
# Convert ts to tibble
sp_df <- sp |>
    tk_tbl(rename_index = "timestamp") %>%
    mutate(timestamp = as_date(timestamp))
```

SP500 Stock Exploratory Data Analysis

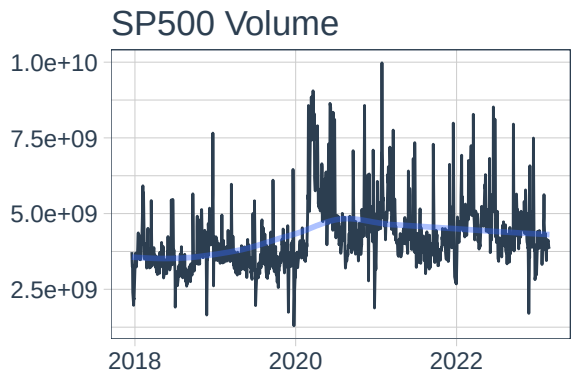
```
## Warning in tk_tbl.data.frame(sp, rename_index = "timestamp"): Warning: No index
## to preserve. Object otherwise converted to tibble successfully.
```

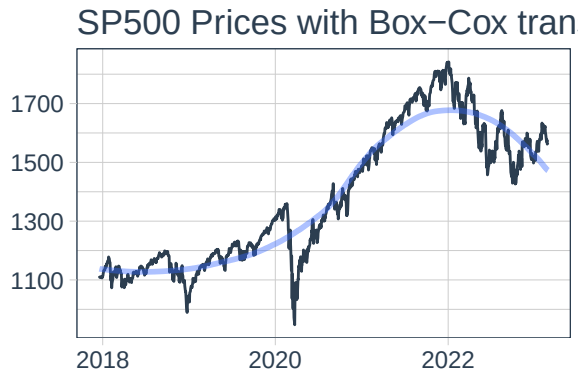
```
# Plot original data
e0 <- sp_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = volume,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "SP500 Volume"
    )

e1 <- sp_df %>%
```

```
  plot_time_series(
      .date_var = timestamp,
      .value    = adjclose,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
      .title = "SP500 Closing Prices"
  )
# Get best lambda value to make seasonal fluctuations about the same
lambda <- sp_df %>%
    as_tsibble() %>%   # Use fabletools as a workaround
    features(adjclose, features=guerrero) %>%
    pull(lambda_guerrero)
```

```
## Using ‘timestamp‘ as index variable.
```

```
# Apply Box-Cox transformation to data using chosen lambda
sp_bc_tbl <- sp_df %>%
  mutate(box_cox = box_cox(adjclose, lambda))
# Plot data with Box-Cox transformation applied
e2 <- sp_bc_tbl %>%
    plot_time_series(
        .date_var = timestamp,
        .value = box_cox,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "SP500 Prices with Box-Cox transformation"
    )
# Arrange plots
plot_grid(e0, e1, e2,nrow=2)
```






```
# Convert ts to tibble
sh_df <- sh |>
    tk_tbl(rename_index = "timestamp") %>%
    mutate(timestamp = as_date(timestamp))
```

Shanghai Delayed Index Stock Exploratory Data Analysis

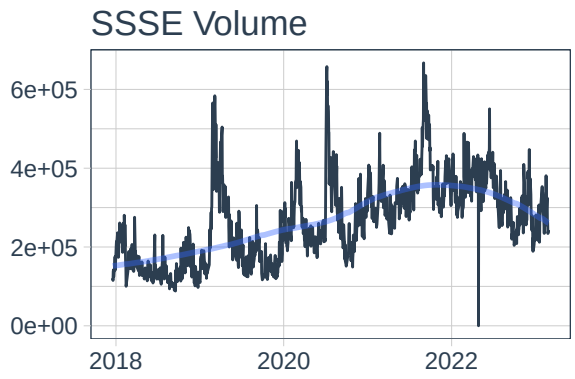
```
## Warning in tk_tbl.data.frame(sh, rename_index = "timestamp"): Warning: No index
## to preserve. Object otherwise converted to tibble successfully.
```

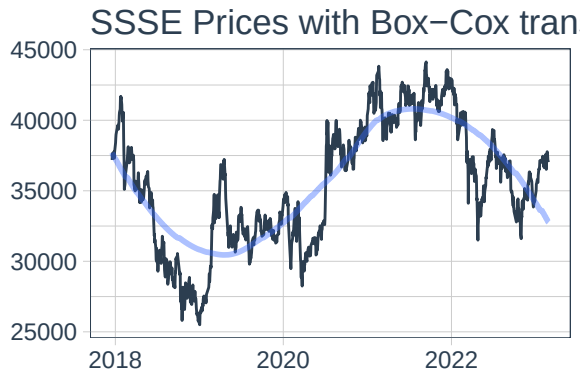
```
# Plot original data
e0 <- sh_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = volume,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "SSSE Volume"
    )

e1 <- sh_df %>%
  plot_time_series(
```

```
      .date_var = timestamp,
      .value    = adjclose,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
      .title = "SSSE Closing Prices"
  )
# Get best lambda value to make seasonal fluctuations about the same
lambda <- sh_df %>%
    as_tsibble() %>%   # Use fabletools as a workaround
    features(adjclose, features=guerrero) %>%
    pull(lambda_guerrero)
```

```
## Using ‘timestamp‘ as index variable.
```

```
# Apply Box-Cox transformation to data using chosen lambda
sh_bc_tbl <- sh_df %>%
  mutate(box_cox = box_cox(adjclose, lambda))
# Plot data with Box-Cox transformation applied
e2 <- sh_bc_tbl %>%
    plot_time_series(
        .date_var = timestamp,
        .value = box_cox,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "SSSE Prices with Box-Cox transformation"
    )
# Arrange plots
plot_grid(e0, e1, e2,nrow=2)
```





BTC Futures Exploratory Data Analysis

```
# Convert ts to tibble
f_df <- btcf_df |>
    tk_tbl(rename_index = "timestamp") %>%
    mutate(timestamp = as_date(timestamp))
```

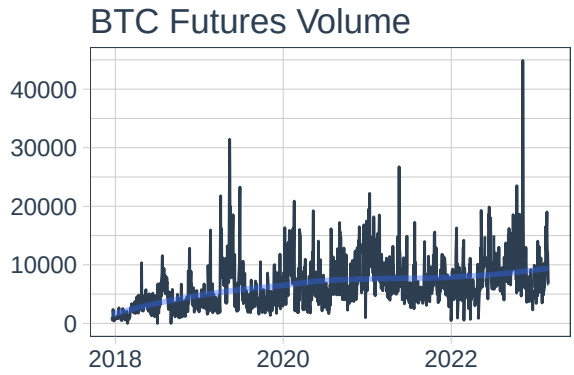
```
## Warning in tk_tbl.data.frame(btcf_df, rename_index = "timestamp"): Warning: No
## index to preserve. Object otherwise converted to tibble successfully.
```

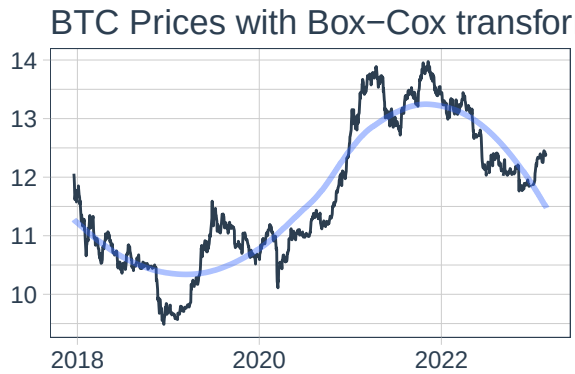
```
# Plot original data
e0 <- f_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = volume,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Futures Volume"
    )

e1 <- f_df %>%
  plot_time_series(
      .date_var = timestamp,
      .value    = adjclose,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
```

```
      .title = "BTC Prices"
  )
# Get best lambda value to make seasonal fluctuations about the same
lambda <- f_df %>%
    as_tsibble() %>%   # Use fabletools as a workaround
    features(adjclose, features=guerrero) %>%
    pull(lambda_guerrero)
```

```
## Using ‘timestamp‘ as index variable.
```

```
# Apply Box-Cox transformation to data using chosen lambda
f_bc_tbl <- f_df %>%
  mutate(box_cox = box_cox(adjclose, lambda))
# Plot data with Box-Cox transformation applied
e2 <- f_bc_tbl %>%
    plot_time_series(
        .date_var = timestamp,
        .value = box_cox,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Prices with Box-Cox transformation"
    )
# Arrange plots
plot_grid(e0, e1, e2,nrow=2)
```



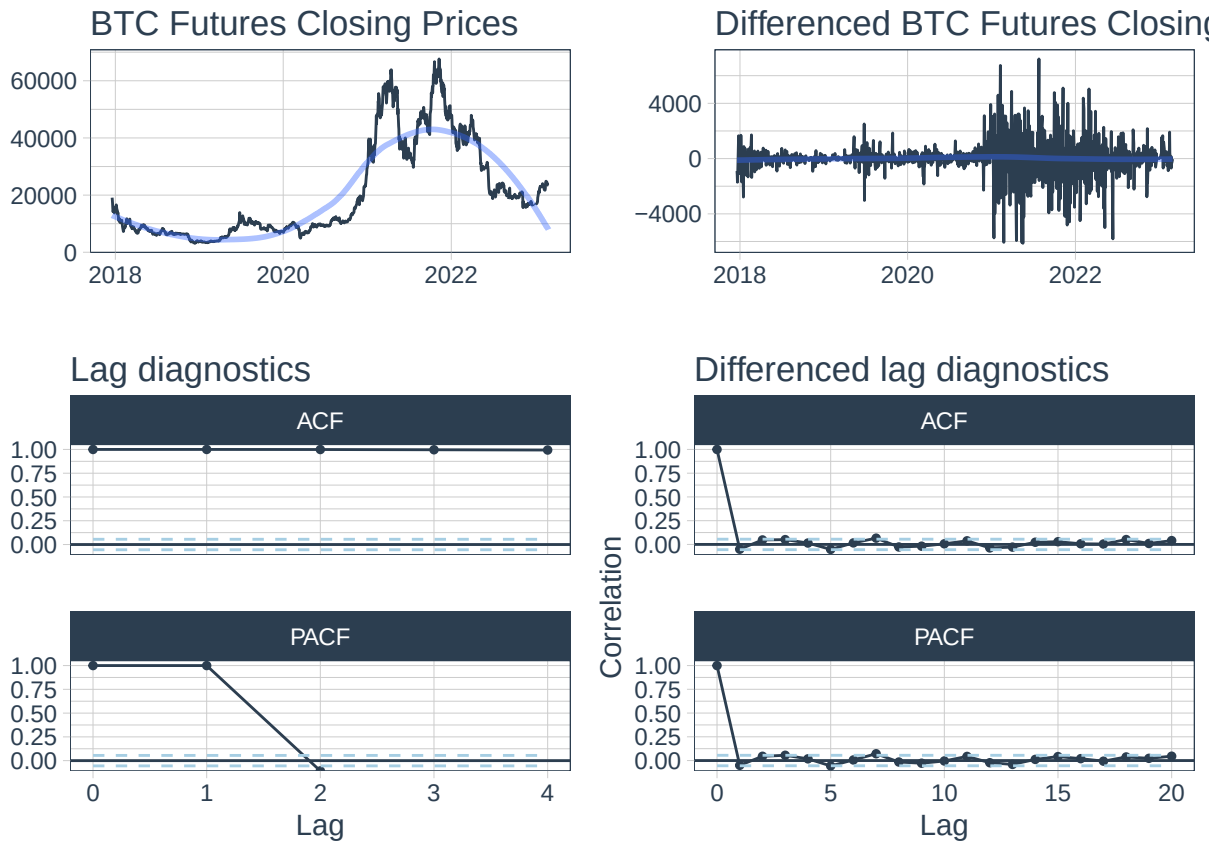


```r
# Convert ts to tibble
f_df <- f_df |>
    mutate(
        diff_value = difference(adjclose)
        )
# Plot Futures Closing Prices
m1 <- f_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = adjclose,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Futures Closing Prices"
    )
# Plot BTC Futures
m2 <- f_df %>%
    plot_time_series(
        .date_var = timestamp,
        .value = diff_value,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "Differenced BTC Futures Closing Prices"
    )
# Plot ACF/PACF
m3 <- f_df %>%
    plot_acf_diagnostics(
        .date_var = timestamp,
        .value = adjclose,
        .lags = 4,
        .show_white_noise_bars = TRUE,
        .interactive = FALSE,
        .title="Lag diagnostics"
    )
# Plot ACF
m4 <- f_df %>%
    plot_acf_diagnostics(
        .date_var = timestamp,
        .value = diff_value,
        .lags = 20,
        .show_white_noise_bars = TRUE,
        .interactive = FALSE,
        .title="Differenced lag diagnostics"
    )
# Arrange plots
plot_grid(m1, m2, m3, m4, ncol=2, rel_heights = c(1, 1.5))
```

```
## Warning: Removed 1 row containing missing values (`geom_line()`).
## Removed 1 row containing missing values (`geom_line()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```




```
## Modelling

# 1. Plot the data and identify any unusual observations.
f_df %>%
  plot_time_series(
    .date_var = timestamp,
    .value = adjclose,
    .smooth_alpha = 0.6,
    .interactive = FALSE,
    .title = "BTC Futures Closing Prices"
  )
```


```
# 3. Automatically fit multiple models and find the lowest AICc.
# Split the data 80/20
splits <- rsample::initial_time_split(f_df, prop = 0.8)
# Model Spec
model_spec_arima <- arima_reg() %>%
  set_engine("auto_arima")
# Fit Spec
model_fit_arima <- model_spec_arima %>%
  fit(adjclose ~ timestamp, data = rsample::training(splits))
```

```
## frequency = 5 observations per 1 week
```

```
# Check model report
model_fit_arima
```

```
## parsnip model object
##
## Series: outcome
## ARIMA(2,1,2)(1,0,0)[5]
##
## Coefficients:
##           ar1      ar2     ma1     ma2    sar1
##       -0.1229  -0.9164  0.0511  0.9474  0.0046
## s.e.   0.0291   0.0250  0.0248  0.0189  0.0334
##
```

```
## sigma^2 = 1411072:  log likelihood = -8930.11
## AIC=17872.23   AICc=17872.31   BIC=17901.97
```

```
# 4. Plot ACF of the residuals and do a portmanteau test to make sure they look like white noise. If no
# Create table of model to use
auto_arima_model_tbl <- modeltime_table(model_fit_arima)
# Calibrate the model to produce confidence intervals
arima_calibration_tbl <- auto_arima_model_tbl %>%
  modeltime_calibrate(new_data = rsample::testing(splits))
# Create residuals table
arima_residuals_tbl <- arima_calibration_tbl %>% modeltime_residuals()
# Plot the residuals
u1 <- arima_residuals_tbl %>%
  plot_modeltime_residuals(
    .type = "timeplot",
    .interactive = FALSE)
u2 <- arima_residuals_tbl %>%
  plot_modeltime_residuals(
    .type = "acf",
    .interactive = FALSE,
    .title = "ACF and PACF Plots")
```

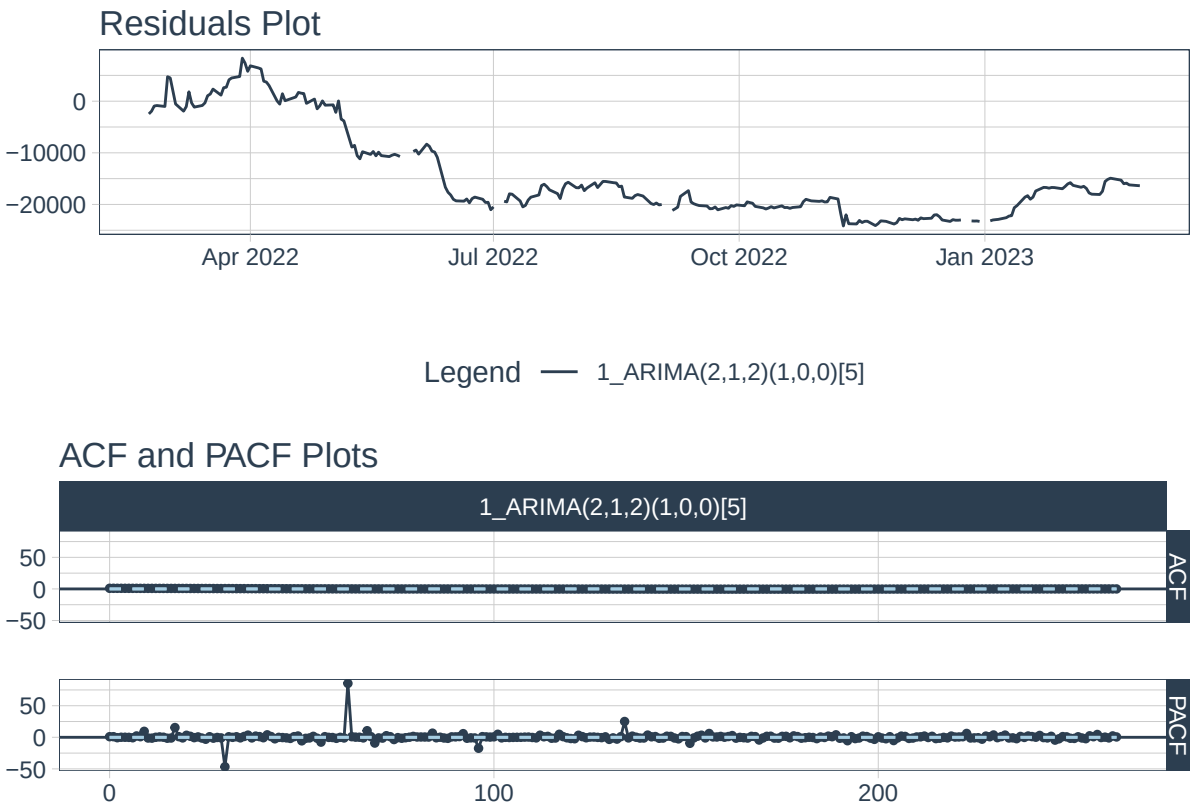
```
## Max lag exceeds data available. Using max lag: 262
```

```
# Check for white noise with a Ljung-Box test
arima_residuals_tbl %>%
  select(.residuals, .index) %>%
  as_tsibble() %>%  # fabletools workaround
  features(.residuals, ljung_box, lag=1, dof=0)
```

```
## Using ‘.index‘ as index variable.
```

```
## # A tibble: 1 x 2
##   lb_stat lb_pvalue
##     <dbl>     <dbl>
## 1    258.         0
```

```
# Arrange plots
plot_grid(u1, u2, ncol=1)
```



Ljung-Box Test The absence of autocorrelation in residuals is tested using the Ljung-Box and Box-Pierce tests. The values are autocorrelated if the p-value is low and falls below the specified significance level. If the p-value is more than 0.05, this indicates that the data residuals are independent, which is positive. The p-value in this instance is 0 0.05, which is a poor outcome.

```
# 5. Once the residuals look like white noise, calculate forecasts.
arima_calibration_tbl %>%
  modeltime_forecast(
      new_data = rsample::testing(splits),
      actual_data = f_df,
      h = "1 years"
  ) %>%
  plot_modeltime_forecast(
      .legend_max_width = 25,
      .interactive = TRUE,
      .title = "Daily Change In Closing BTC Futures Price 1-year forecast",
      .x_lab = "Period",
      .y_lab = "BTC Prices"
  )
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

Merging DataFrames

```r
sp_df = sp_df %>%
  select(-c(open, high, low, close, volume, ticker,close,high)) %>%
  rename("close_sp" = "adjclose")
sh_df = sh_df %>%
  select(-c(open, high, low, close, volume, ticker,close,high)) %>%
  rename("close_sh" = "adjclose")
btc_df = btc_df %>% select(-c(ticker, volume))
btcf_df  = btcf_df %>%
  select(-c(open,high, low, close, ticker, volume))%>%
    rename("close_f" = "adjclose")
```

```r
df = merge(btc_df,btcf_df,by="timestamp")
df = merge(df,sp_df,by="timestamp")
df = merge(df,sh_df,by="timestamp")
df$timestamp = as.POSIXct.Date(df$timestamp)
df = select(df, -c(open, close, high, low))

df$month = month(df$timestamp, label = TRUE)
df$year = year(df$timestamp)


df$infl = df$year
df$infl[df$year == 2017] = 2.858
df$infl[df$year == 2018] = 6.896
df$infl[df$year == 2019] = 5.570
df$infl[df$year == 2020] = 1.382
df$infl[df$year == 2021] = 8.340
df$infl[df$year == 2022] = 12.120
df$infl[df$year == 2023] = NA
```

```r
## Drop the BTC price to avoid multicollinearity
df |> GGally::ggpairs(columns = c(2:5, 8))
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removing 1 row that contained a missing value
```

```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 33 rows containing missing values
```

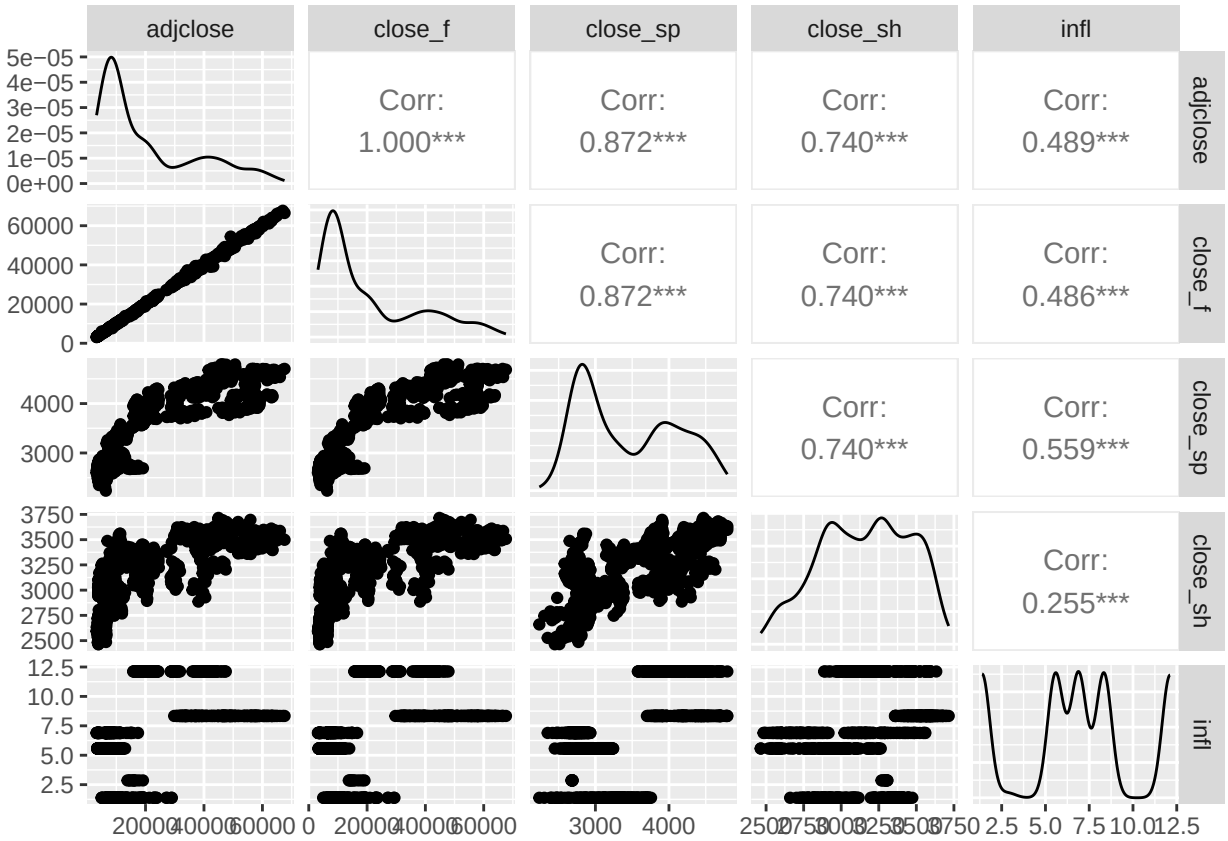
```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 1 rows containing non-finite values (`stat_density()`).
```

```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removing 1 row that contained a missing value
```

```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removing 1 row that contained a missing value
```

```
## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 34 rows containing missing values

## Warning: Removed 1 rows containing missing values (‘geom_point()‘).

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 33 rows containing missing values

## Warning: Removed 1 rows containing missing values (‘geom_point()‘).

## Warning in ggally_statistic(data = data, mapping = mapping, na.rm = na.rm, :
## Removed 33 rows containing missing values

## Warning: Removed 33 rows containing missing values (‘geom_point()‘).

## Warning: Removed 34 rows containing missing values (‘geom_point()‘).

## Warning: Removed 33 rows containing missing values (‘geom_point()‘).
## Removed 33 rows containing missing values (‘geom_point()‘).

## Warning: Removed 33 rows containing non-finite values (‘stat_density()‘).
```



Discussion

There is a strong Positve correlation that exists between the perfomance of Market indes and that of a cryptocurrency future. Note that SP500 index has a correlation of .872, the shanghai index a positive correlation of .74 and inflation of .486. This suggests that an increase in either inflation, or the stock prices of the market indices would suggest an increase in BTC Futures prices. There is a perfect correlation between Futures and Bitcoin prices.

Forecasting

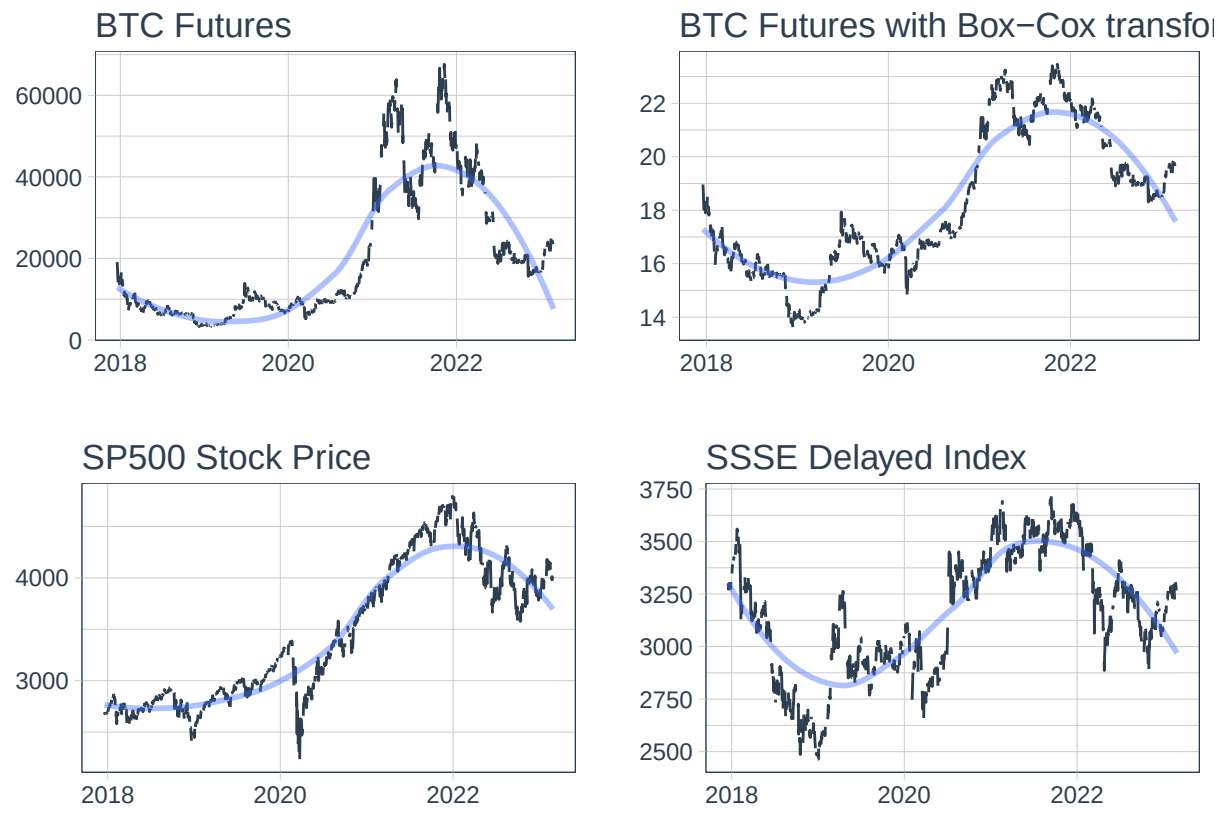
The objective of this study was not to forecast BTC futures but to explore the relationships with the predictors.

The closing price of BTC are removed to get rid of multicollinearity that might occur between the currency and its Futures.

```
df = select(df, -c(adjclose))
```

```
df |>
  mutate(timestamp = as_date(timestamp)) |>
  as_tsibble(index = timestamp) |> group_by_key() %>% fill_gaps()-> df_ts


p1 <- df_ts %>%
  plot_time_series(
      .date_var = timestamp,
      .value    = close_f,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
      .title = "BTC Futures"
  )
# Get best lambda value to make seasonal fluctuations about the same
lambda <- df_ts %>%
    as_tsibble() %>%   # Use fabletools as a workaround
    features(close_f, features=guerrero) %>%
    pull(lambda_guerrero)
# Apply Box-Cox transformation to data using chosen lambda
df_ts_tbl <- df_ts %>%
  mutate(box_cox = box_cox(close_f, lambda))
# Plot data with Box-Cox transformation applied
p2 <- df_ts_tbl %>%
    plot_time_series(
        .date_var = timestamp,
        .value = box_cox,
        .interactive = FALSE,
        .smooth_alpha = 0.4,
        .title = "BTC Futures with Box-Cox transformation"
    )
p3 <- df_ts %>%
  plot_time_series(
      .date_var = timestamp,
      .value    = close_sp,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
      .title = "SP500 Stock Price"
  )
p4 <- df_ts %>%
  plot_time_series(
      .date_var = timestamp,
      .value    = close_sh,
      .interactive = FALSE,
      .smooth_alpha = 0.4,
```

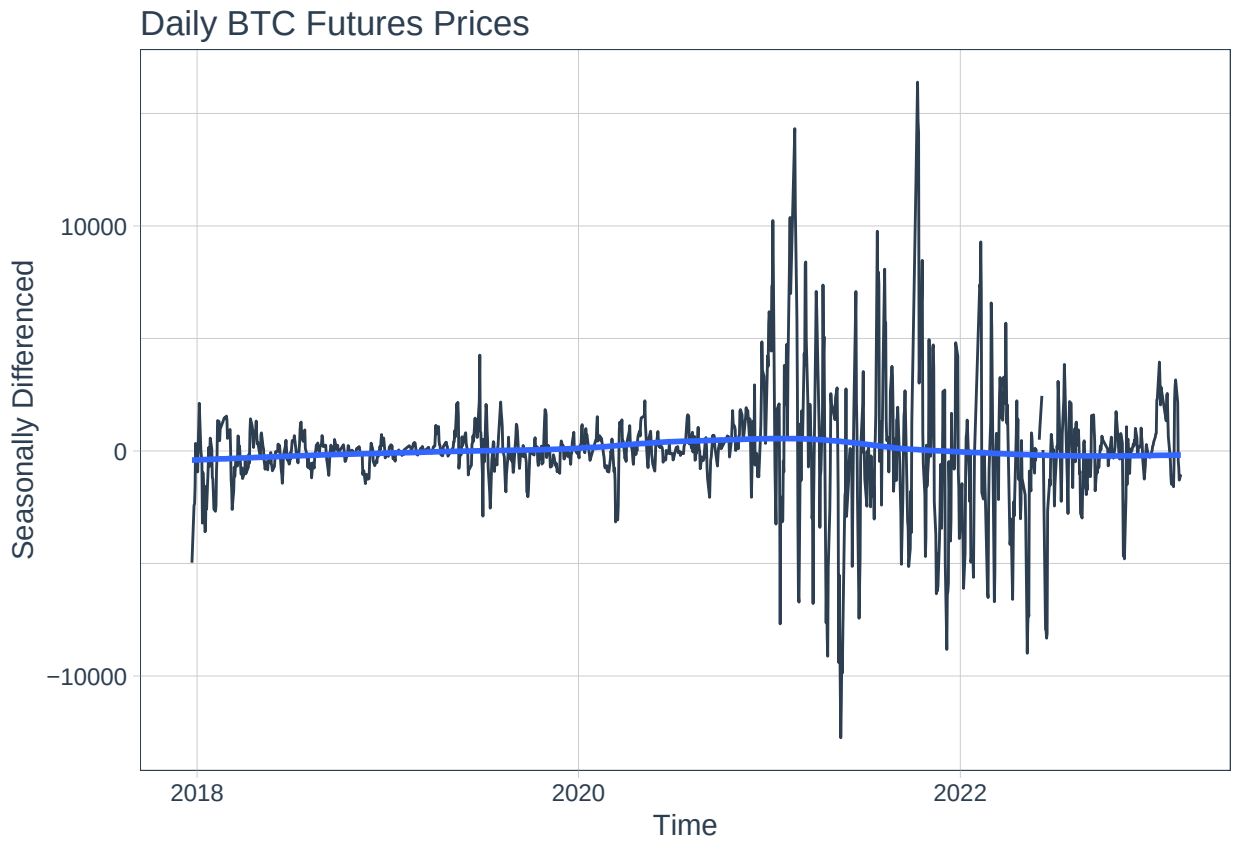
```
      .title = "SSSE Delayed Index"
  )
# Arrange plots
plot_grid(p1, p2, p3,p4, nrow=2)
```




```
# Add a column of seasonally differenced data
df_sd_tbl <- df |> tk_tbl(preserve_index = FALSE) %>%
  tk_augment_differences(
    .value = close_f,
    .lags = 4,
    .differences = 1
  )

# Plot seasonally-differenced data
df_sd_tbl %>%
  plot_time_series(
    .date_var = timestamp,
    .value = close_f_lag4_diff1, # Seasonally differenced data
    .interactive = FALSE,
    .title = "Daily BTC Futures Prices",
    .x_lab = "Time",
    .y_lab = "Seasonally Differenced"
  )
```

```
## Warning: Removed 4 rows containing missing values (‘geom_line()‘).
```

```
## Removed 4 rows containing missing values (`geom_line()`).
```

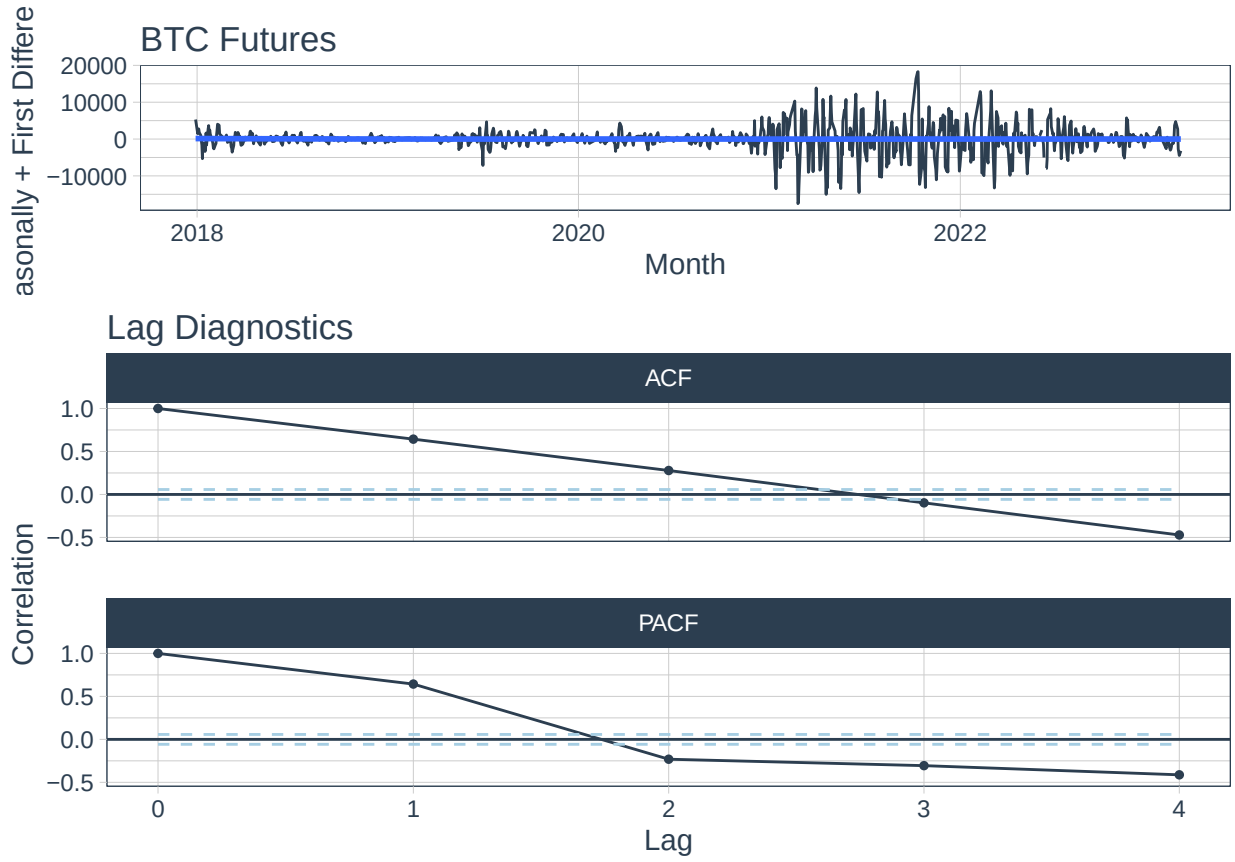



```
# Apply a first difference
df_sd_diff_tbl <- df_sd_tbl %>%
  tk_augment_differences(
    .value = close_f_lag4_diff1,
    .lags = 4,
    .differences = 1
  )
# Check that differencing was applied
df_sd_diff_tbl %>% tail()
```

```
## # A tibble: 6 x 9
##   timestamp           close_f close_sp close~1 month  year  infl close~2 close~3
##   <dttm>                <dbl>    <dbl>   <dbl> <ord> <dbl> <dbl>   <dbl>   <dbl>
## 1 2023-02-17 03:00:00   24870    4079.   3224. Feb    2023    NA    3160    4750
## 2 2023-02-21 03:00:00   24490    3997.   3307. Feb    2023    NA    2155    2705
## 3 2023-02-22 03:00:00   23800    3991.   3291. Feb    2023    NA    -450   -2690
## 4 2023-02-23 03:00:00   23920    4012.   3287. Feb    2023    NA    -720   -3585
## 5 2023-02-24 03:00:00   23576    3970.   3267. Feb    2023    NA   -1294   -4454
## 6 2023-02-27 03:00:00   23445    3982.   3258. Feb    2023    NA   -1045   -3200
## # ... with abbreviated variable names 1: close_sh, 2: close_f_lag4_diff1,
## #   3: close_f_lag4_diff1_lag4_diff1
```

```
# Plot seasonally + first differenced data
df1 <- df_sd_diff_tbl %>%
  plot_time_series(
    .date_var = timestamp,
    .value = close_f_lag4_diff1_lag4_diff1, # Seasonal + first differenced data
    .interactive = FALSE,
    .title = "BTC Futures",
    .x_lab = "Month",
    .y_lab = "Seasonally + First Differenced"
  )
# Plot lag diagnostics of seasonally + first differenced data
df2 <- df_sd_diff_tbl %>%
  plot_acf_diagnostics(
    .date_var = timestamp,
    .value =close_f_lag4_diff1_lag4_diff1,
    .lags = 4,
    .interactive = FALSE,
    .show_white_noise_bars = TRUE
  )
# Arrange plots
plot_grid(df1,df2, ncol = 1, rel_heights = c(1, 2))
```

```
## Warning: Removed 8 rows containing missing values (‘geom_line()‘).
## Removed 8 rows containing missing values (‘geom_line()‘).
```



```r
# 3. Automatically fit multiple models and find the lowest AICc.
# Split the data 80/20 into a training and test set
splits <- rsample::initial_time_split(df_ts_tbl, prop = 0.8)
# Manually fit an ARIMA(0,1,2)(0,1,1)[12] model
model_fit_arima_012_011 <- arima_reg(
  non_seasonal_ar          = 0,  # p
  non_seasonal_differences = 1,  # d
  non_seasonal_ma          = 2,  # q
  seasonal_ar              = 0,  # P
  seasonal_differences     = 1,  # D
  seasonal_ma              = 1,  # Q
  seasonal_period          = 4  # Daily data
  ) %>%
  set_engine("arima") %>%
  fit(close_f ~ ., data = training(splits))
# Manually fit an ARIMA(2,1,0)(0,1,1)[12] model
model_fit_arima_210_011 <- arima_reg(
  non_seasonal_ar          = 2,  # p
  non_seasonal_differences = 1,  # d
  non_seasonal_ma          = 0,  # q
  seasonal_ar              = 0,  # P
  seasonal_differences     = 1,  # D
  seasonal_ma              = 1,  # Q
  seasonal_period          = 4  # Daily data
  ) %>%
  set_engine("arima")%>%
  fit(close_f ~ ., data = training(splits))
# Automatically fit an ARIMA model
model_fit_auto_arima <- arima_reg() %>%
  set_engine("auto_arima")%>%
  fit(close_f ~ ., data = training(splits))
```

```
## frequency = 5 observations per 1 week
```

```r
# Add fitted models to a modeltime table
arima_models_tbl <- modeltime_table(
  model_fit_arima_012_011,
  model_fit_arima_210_011,
  model_fit_auto_arima)
# Check modeltime table
arima_models_tbl
```

```
## # Modeltime Table
## # A tibble: 3 x 3
##   .model_id .model   .model_desc
##       <int> <list>   <chr>
## 1         1 <fit[+]> REGRESSION WITH ARIMA(0,1,2)(0,1,1)[4] ERRORS
## 2         2 <fit[+]> REGRESSION WITH ARIMA(2,1,0)(0,1,1)[4] ERRORS
## 3         3 <fit[+]> REGRESSION WITH ARIMA(0,0,1)(0,0,2)[5] ERRORS
```

References

Gil-Alana, L. A., Abakah, E. J. A., & Rojo, M. F. R. (2020). Cryptocurrencies and stock market indices. Are they related?. Research in International Business and Finance, 51, 101063.
Jeribi, A., & Fakhfekh, M. (2021). Portfolio management and dependence structure between cryptocurrencies and traditional assets: evidence from FIEGARCH-EVT-Copula. Journal of Asset Management, 22(3), 224-239.
https://www.kaggle.com/datasets/belayethossainds/global-inflation-dataset-212-country-19702022