PCA based construction of cryptocurrency index
Process Report

Zeyan Huang

April 30, 2025

Abstract

1 Introduction

After reviewing the paper "Principal Component Analysis-Based Construction and Evaluation of a Cryptocurrency Index"[**1**], I decided to replicate its results and explore potential improvements for analyzing the cryptocurrency market.

2 Data

To construct the dataset for this study, I collected daily market capitalization data for the top 1,200 cryptocurrencies from CoinGecko.com, covering the period from March 29, 2024, to March 28, 2025. The selection of these cryptocurrencies was based on their rankings as of March 28, 2025, ensuring that the dataset reflects the market structure as observed at the end of the analysis period.

One notable issue with the dataset is the presence of missing data points. Specifically, some cryptocurrencies had not yet launched at the start of the data period, resulting in incomplete time series that could complicate further analysis.

To address this issue, I adopted different approaches for handling missing data when calculating the static index and the dynamic index.

3 Static Index

The methodology presented in the paper uses the first principal component (PC1) as a set of weights, which are multiplied by the market capitalizations of the corresponding cryptocurrencies on a given day. The weighted values are then summed to compute the index value for each specific day.

In this replication, I applied principal component analysis to the full dataset—consisting of 1,200 cryptocurrencies over 365 days—and used PC1 as the weighting scheme to construct the daily index. However, the variance explained by PC1 was only 36%, which is relatively low and suggests that the resulting index may not be as reliable or informative as implied in the original paper. Here I used the sum of the 1200 cryptocurrencies' market capitalization as the total market capitalization.

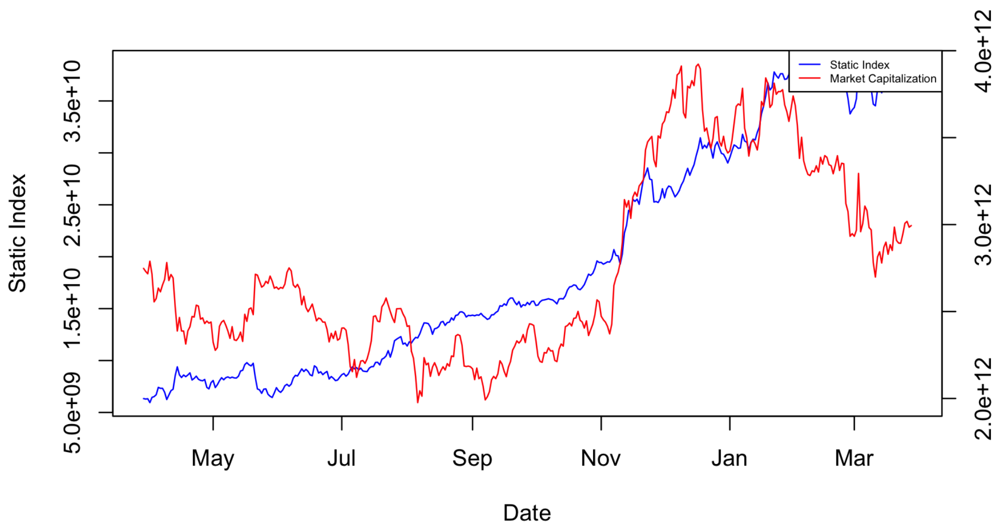

Figure 1: Static Index using PC1

As shown in the plot above, the calculated static index does not fully capture the overall movement of the cryptocurrency market. However, there are several periods where the index exhibits similar short-term trends to the total market capitalization, suggesting that the method may still partially reflect market dynamics.

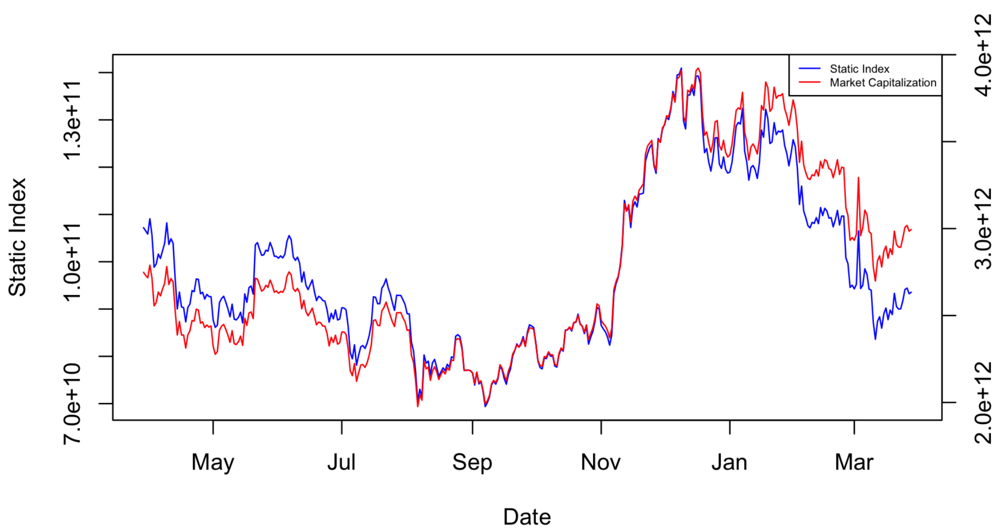

Figure 2: Static Index using absolute value of PC1 and PC2

Interestingly, when using the absolute value of the sum of PC1 and PC2 loadings as weights, the resulting static index appears to effectively capture the overall movement of the cryptocurrency market, even though the combined variance explained is only 61.5%.

The static index without using absolute value provided below.

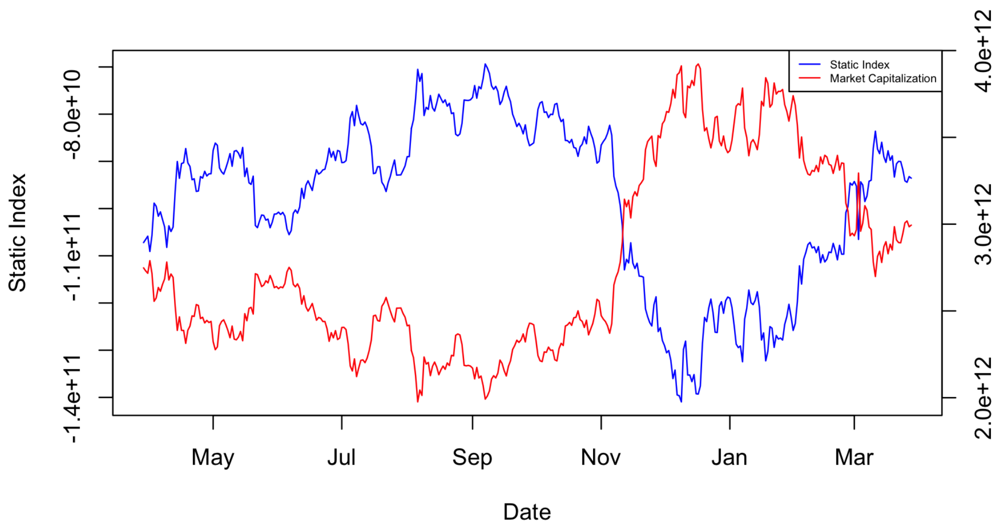

Figure 3: Static Index using PC2

Question!!! Why the absolute method here work well?

4 Dynamic Index Construction Methodology

Based on the PCA-based dynamic index methodology presented in the original paper, the calculation of the dynamic index consists of two main steps.

In the **first step**, a rolling window of T_n days (90 days in our implementation) is used to conduct PCA and correlation analysis to determine the number of constituent cryptocurrencies to include in the index. We define an update interval of T_{n_gap} days (30 days in our test). Every 30 days, we extract the preceding 90-day window of data, apply PCA, and identify the subset of cryptocurrencies whose individual indices (constructed using PC1) have a correlation higher than 0.99 with the index derived from the full set of cryptocurrencies. This subset of cryptocurrencies is saved for use in the next step, and the number of selected assets is denoted as nc.

In the **second step**, the index is computed using a forward-looking window. Specifically, for each update period, we take the previous T_w days (also 30 in our implementation) of data for the n_c selected cryptocurrencies and apply PCA. The first principal component (PC1) is used as the weighting vector. The index value a_t for the subsequent T_w days is calculated as:

$$a_t = \sum_{i=1}^{n_c} PC_{1,i} \cdot M_{t,i}$$

where $PC_{1,i}$ is the loading of the i-th asset on the first principal component, and $M_{t,i}$ is the market capitalization of asset i on day t.

Once the daily values of a_t are obtained, the dynamic PCA-based index I_{PCA} is computed as:

$$I_{PCA} = \frac{a_t}{a_{base}} \times m$$

Here, m is a multiplier (set to 1000 for this test), and a_{base} is the value of a_t on the base day t_0, which is the first day of each new T_w-day sub-period. This ensures that the index resets to m at the beginning of each sub-period, maintaining consistency with the methodology described in the paper.

The plot below compares the dynamic PCA-based index with the total market capitalization, using the first principal component (PC1) as the weighting vector. While the dynamic index generally tracks the overall movement of the market quite well, a noticeable divergence appears after December, where the index underperforms relative to the market capitalization. I believed this gap is because our m value is set as 1000 and not updated follows the market, so the next step should focus on the calculation methodology of choosing m.

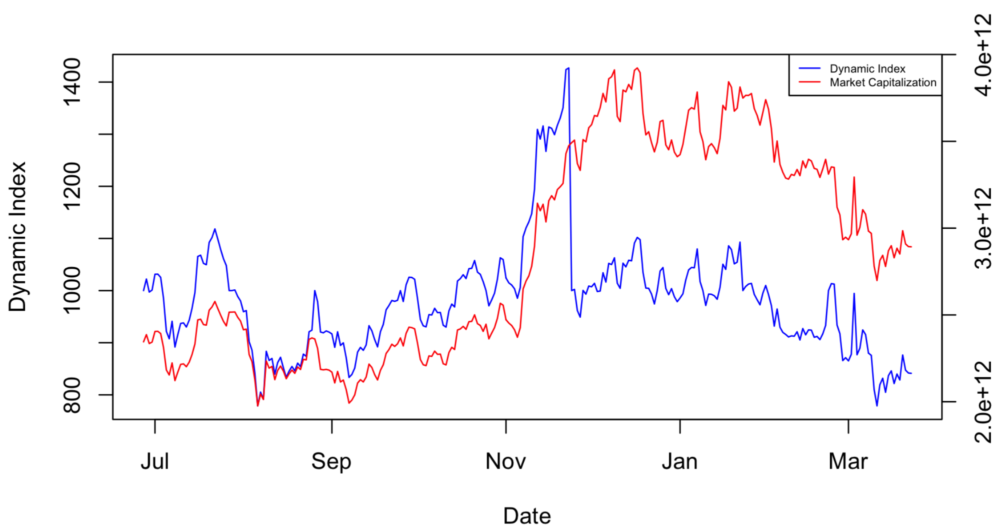

Figure 4: Dynamic index using PC1

References

[1] Principal component analysis based construction and evaluation of cryptocurrency index. Retrieved from `https://doi.org/10.1016/j.eswa.2020.113796`

Appendix

R Code for Data Preprocessing

Listing 1: Data Preprocessing

```
# Data
# Data preparation
# data from Coingecko
folder_path <- "market_cap_data/"
all_files <- list.files(folder_path)


dataset <- lapply(all_files, function(x){
  data <- read.csv(paste0(folder_path, x), header = TRUE)
  data <- data[-nrow(data), ]
  data$Date <- as.Date(data$date)
  data <- data[order(data$Date),]
  data <- data[!duplicated(data$Date), ]


  price <- as.numeric(data$market_cap)
  names(price) <- data$Date
  return(price)
}
  )

# Name the data list
file_names <- gsub("\\.csv", "", all_files)
names(dataset) <- file_names

# Step 1: Convert each price series to a dataframe with Date + market cap
dataset_aligned <- lapply(names(dataset), function(name) {
  df <- data.frame(Date = as.Date(names(dataset[[name]])),
                   Cap = dataset[[name]])
  colnames(df)[2] <- name
  return(df)
})

# Step 2: Merge all dataframes by Date, using full outer join
data <- Reduce(function(x, y) merge(x, y, by = "Date", all = TRUE),
                      dataset_aligned)

# Step 3: set all NA value as 0
# data[is.na(data)] <- 0

# Set Date as row index
rownames(data) <- data$Date
data$Date <- NULL
```

R Code for Static Index

Listing 2: SI

```
# Static Index
# Data prep
data_static_index <- data

data_static_index[is.na(data_static_index)] <- 0

(pca_static_index$sdev[1:7])^2
sum((pca_static_index$sdev)^2)
sum((pca_static_index$sdev[1:2])^2)/sum((pca_static_index$sdev)^2)*100

static_index <- as.matrix(data_static_index) %*% as.vector(pca_static_index$rotation[,1])

static_index <- as.data.frame(static_index)

static_index$Date <- as.Date(rownames(static_index))

plot(static_index$Date, static_index$V1, type = "l", xlab = "Date", ylab = "PCA Static Index")

# Get the daily market capitalization
data_mc <- data.frame(RowSum = rowSums(data_static_index))

rownames(data_mc) <- rownames(data_static_index)

# Combine the data_mc with static_index
data_mc_si <- merge(static_index, data_mc, by = "row.names")
rownames(data_mc_si) <- data_mc_si$Row.names
data_mc_si$Row.names <- NULL

(data_mc_si$Date, data_mc_si$V1, type = "l", col = "blue", xlab = "Date", ylab = "Static Index")

par(new = TRUE)
plot(data_mc_si$Date, data_mc_si$RowSum, type = "l", col = "red", axes = FALSE, xlab = "", ylab = "")
axis(side = 4)  # Add right-side axis
mtext("Market Capitalization", side = 4, line = 3)  # Label for right Y-axis

# Add legend
legend("topright", legend = c("Static Index", "Market Capitalization"),
       col = c("blue", "red"), lty = 1, cex = 0.5)

static_index_pc2 <- abs(as.matrix(data_static_index) \%*\% as.vector(rowSums(pca_static_index$rotation[,1:2])))

static_index_pc2  <- as.data.frame(static_index_pc2 )

static_index_pc2$Date <- as.Date(rownames(static_index_pc2))

# Combine the data_mc with static_index
data_mc_si_pc2 <- merge(static_index_pc2, data_mc, by = "row.names")
rownames(data_mc_si_pc8) <- data_mc_si_pc2$Row.names
```

```
data_mc_si_pc2$Row.names <- NULL

plot(data_mc_si_pc2$Date, data_mc_si_pc2$V1, type = "l", col = "blue", xlab = "Date", ylab = "Static Index")

par(new = TRUE)
plot(data_mc_si_pc2$Date, data_mc_si_pc2$RowSum, type = "l", col = "red", axes = FALSE, xlab = "", ylab = "")
axis(side = 4)  # Add right-side axis
mtext("Market Capitalization", side = 4, line = 3)  # Label for right Y-axis

# Add legend
legend("topright", legend = c("Static Index", "Market Capitalization"),
       col = c("blue", "red"), lty = 1, cex = 0.5)
```

R Code for Nc constituents Update function

Listing 3: NC

```
Nc_cal <- function(dataframe, threshold = 0.99){
  # Step 1: Calculate PC1 for all columns
  pca_full_temp <- prcomp(dataframe, scale. = TRUE)

  # Initialize a list to store PC1 for each subset of columns
  pc1_list_temp <- list()
  index_list_temp <- list()
  variance_explained_list <- list()

  # Loop through subsets of columns (from 1:2, 1:3, ..., 1:ncol)
  for (i in 2:ncol(dataframe)) {
    subset_data <- dataframe[, 1:i]
    pca_subset <- prcomp(subset_data, scale. = TRUE)
    pc1_subset <- pca_subset$rotation[, 1]

    pc_1_v <- (pca_subset$sdev[1])^2
    pc_all_v <- sum((pca_subset$sdev)^2)
    pc_1_explained_v_pct <- pc_1_v/pc_all_v
    variance_explained_list[[i]] <- list(pc_1_v, pc_all_v, pc_1_explained_v_pct)

    temp_index <- as.matrix(subset_data) %*% as.vector(pc1_subset)
    pc1_list_temp[[i]] <- pc1_subset  # Store in list (index starts at 1)
    index_list_temp[[i]] <- temp_index
  }

  # Initialize a vector to store the correlations
  correlations_temp <- numeric(length(pc1_list_temp) - 1)

  # Index with all crypto
  index_all_temp <- index_list_temp[[length(index_list_temp)]]

  # Calculate the Correlation
  for (i in 2: (length(index_list_temp))){
```

```r
    correlations_temp[i] <- cor(index_list_temp[[i]], index_all_temp)
  }

  # Find the first index where correlation >= threshold
  first_reach_index_temp <- which(correlations_temp >= threshold)[1]
  chosen_crypto_list <- c(colnames(dataframe)[1:first_reach_index_temp])

  output_list <- list()
  output_list[['Index']] <- index_list_temp
  output_list[['Correlation']] <- correlations_temp
  output_list[['Nc']] <- first_reach_index_temp
  output_list[['Choosen Crypto']] <- chosen_crypto_list
  output_list[['Variance']] <- variance_explained_list
  return(output_list)
}
```

R Code for Basic dynamic index calculation function for single period

Listing 4: DI$_{b}ase$

```r
dynamic_index_base <- function(dataframe, Tw = 30){
  a_list <- c()
  a_base <- as.numeric(dataframe[Tw+1,])%*%as.vector(prcomp(dataframe[1:Tw,], scale.= TRUE)$rotation[,1])
  a_list <- append(a_list, 1)
  # for (i in 2:Tw){
  #   a_t <- (as.numeric(dataframe[Tw+i,])%*%as.vector(prcomp(dataframe[i:(Tw+i-1),], scale.= TRUE)$rotation[,1]))/a_base
  #   a_list <- c(a_list, a_t)
  #
  # }
  # a_t_df <- data.frame(Value = a_list, row.names = rownames(dataframe[(Tw+1):(Tw+Tw),]))
  a_t <- as.matrix(dataframe[(Tw+2):(Tw+Tw),])%*%as.vector(prcomp(dataframe[1:Tw,], scale.= TRUE)$rotation[,1])

  a_list <- c(a_list, as.vector(a_t)/c(a_base))
  a_t_df <- data.frame(Value = a_list, row.names = rownames(dataframe[(Tw+1):(Tw+Tw),]))

  return(a_t_df)
}
```

R Code for Dynamic Index function

Listing 5: DI$_{base}$

```r
dynamic_index_cal <- function(dataframe, Tn = 90, Tn_gap = 30, Tw = 30, threshold = 0.99){

  # Nc Choosing process
  nc_result <- list()
```

```
  n_iter <- floor((nrow(dataframe) - Tn) / Tn_gap)
  for (i in 0:n_iter){
    data_90 <- dataframe %>%
    slice((1+i*30):(90+i*30)) %>%        # Select the first 90 rows (time window)
    select(where(~ all(!is.na(.)))) %>% # Drop columns that contain any NA values
    select(where(~ all(. != 0))) %>%     # Drop columns that contain any 0 values
    select(where(~ {                     # Drop columns with 0 or NA standard deviation
      sd_x <- sd(., na.rm = TRUE)        # Calculate standard deviation ignoring NA
      is.finite(sd_x) && sd_x > 0        # Keep only columns with valid, non-zero SD
    })) %>%
    { .[, order(-as.numeric(.[1, ]))] } # Sort the remaining columns by the first row, descending

    nc_result[[i+1]] <- Nc_cal(data_90, threshold = threshold)
  }

  # Dynamic Index Calculation
  dynamic_index_results <- list()
  for (i in 1:length(nc_result)){
    data_60 <- dataframe %>%
    slice((31+i*30):(90+i*30)) %>%        # Select the required 60 rows (time window)
    select(where(~ all(!is.na(.)))) %>% # Drop columns that contain any NA values
    select(where(~ all(. != 0))) %>%
    select(all_of(nc_result[[i]][["Choosen Crypto"]]))# Filter by selected crypto
    if (nrow(data_60) < 60){
      break
    }
    dynamic_index_results[[i]] <- dynamic_index_base(data_60)
  }

  final_results <- list(nc_result, dynamic_index_results)
  return(final_results)
}
```