CA4

Pavel Gulin, Erling Mysen, Kristian Røhne

2025-11-23

```
#1 a)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.2
## ✔ ggplot2   3.5.2     ✔ tibble    3.3.0
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.1.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# load both datasets one is the first five keywords (Python, Java, Typescript, Interest Rates, Regulation) worldw
ide
#Second one is (Volatility, NHL, Ice Skates, Sports Training) #here ice hockey is seasonal
df1 <- read_csv("multiTimeline.csv", skip = 2) #first two rows are just text from google trends and empty column.
```

```
## Rows: 262 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl  (5): Python: (Worldwide), Java: (Worldwide), TypeScript: (Worldwide), I...
## date (1): Week
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

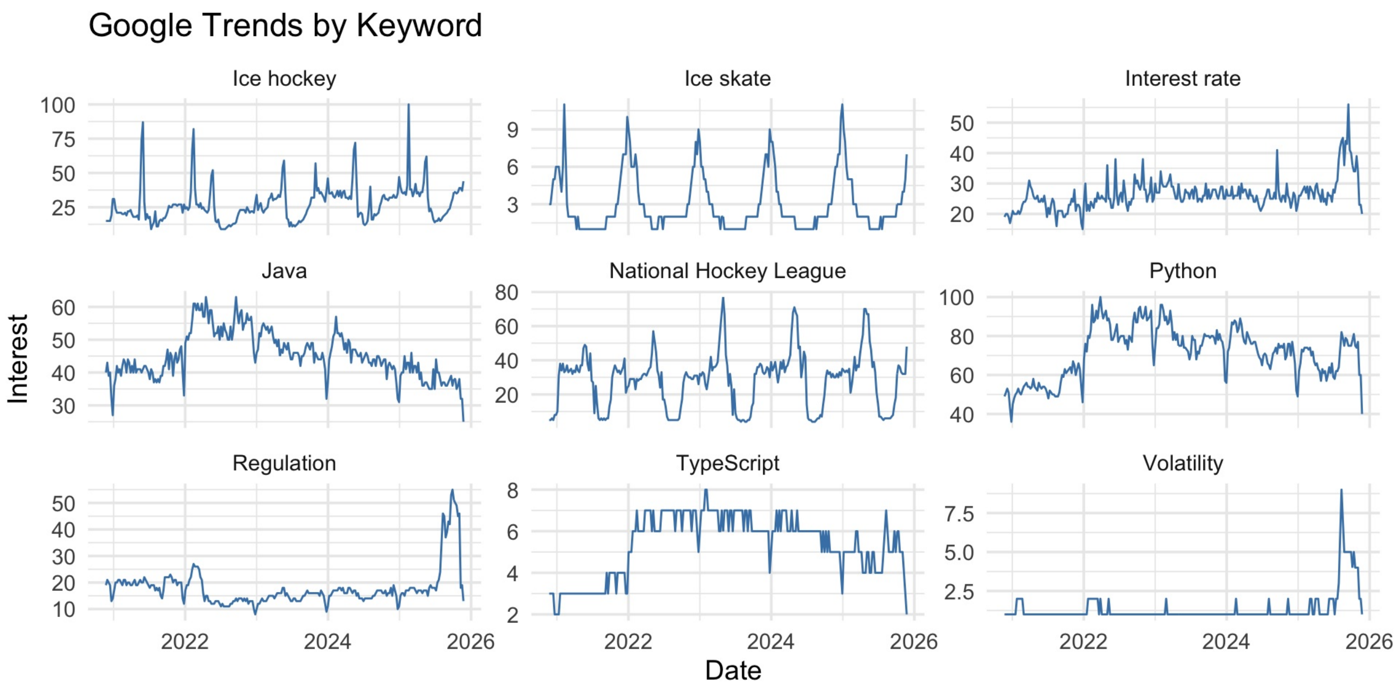
```
df2 <- read_csv("multiTimeline2.csv", skip = 2)
```

```
## Rows: 262 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl  (4): Volatility: (Worldwide), National Hockey League: (Worldwide), Ice ...
## date (1): Week
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df1 <- df1 %>% rename(date = 1)
df2 <- df2 %>% rename(date = 1)

# convert to date
df1$date <- as.Date(df1$date)
df2$date <- as.Date(df2$date)

# full join to combine all dates
df <- full_join(df1, df2, by = "date") %>%
  arrange(date)
```

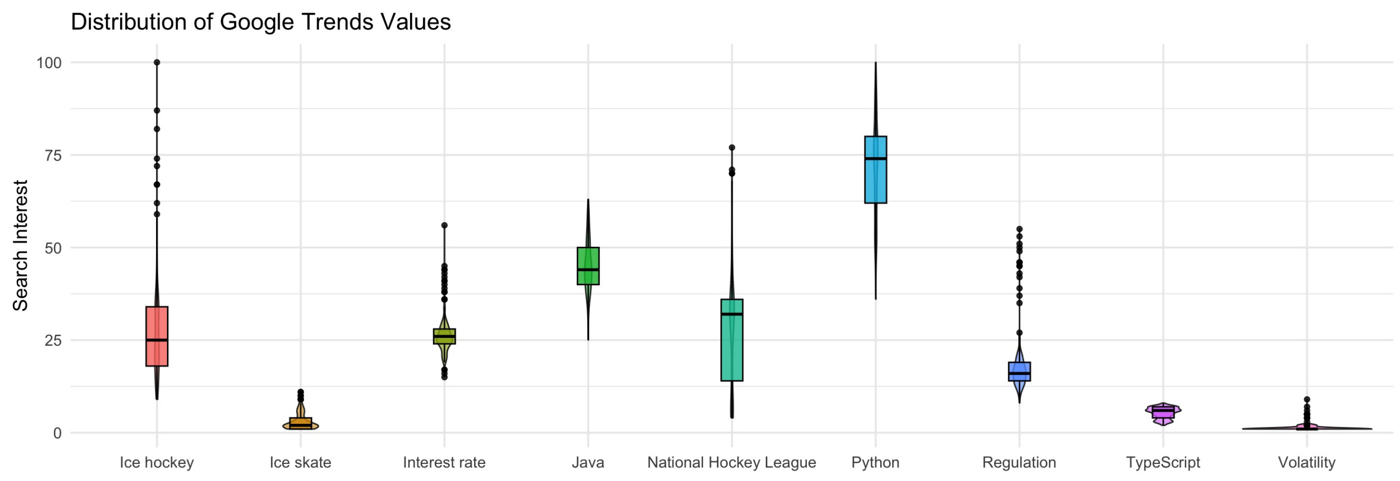
```
df_clean <- df %>% #a lot of wired values in the google dataset
  mutate(across(
    .cols = -date,
    .fns = ~ {
      x <- str_trim(.x)
      x <- str_replace_all(x, "<1", "0")
      x <- str_replace_all(x, "--", "0")
      x <- as.numeric(x)
      x[is.na(x)] <- 0   # remove NA
      x
    }
  ))

names(df_clean) <- names(df_clean) |> #remove the "(Worldwide)" after each keyword
  str_replace(": \\(Worldwide\\)", "") |>
  str_trim()
```

```
df_long <- df_clean %>%
  pivot_longer(
    cols = -date,
    names_to = "keyword",
    values_to = "value"
  )

ggplot(df_long, aes(date, value)) +
  geom_line(color = "steelblue") +
  facet_wrap(~ keyword, scales = "free_y") +
  theme_minimal(base_size = 14) +
  labs(
    title = "Google Trends by Keyword",
    x = "Date",
    y = "Interest"
  )
```



Here we see that the keywords in the same groups somewhat correlate between each other. Especially the Ice Skate and NHL keywords. Which is expected as Ice hockey is a seasonal sport. We can also observe that there is a spike in interest for Regulation, Interest Rate and Volatility in 2025, which also makes sense as all three keywords are related to finance. And the spike might have something to do with the Trumps tariffs and it's impacts on the market. Programming languages also correlate somewhat, but in a less degree than other keyword. However we can see that there is a clear downward sloping trend for all three with a sharp fall in the end of 2025.

```
ggplot(df_long, aes(keyword, value, fill = keyword)) +
  geom_violin(alpha = 0.6) +
  geom_boxplot(width = 0.15, color = "black", alpha = 0.8) +
  theme_minimal(base_size = 14) +
  labs(
    title = "Distribution of Google Trends Values",
    x = "",
    y = "Search Interest"
  ) +
  theme(legend.position = "none")
```



The distributions reveal clear differences in search behaviour across the selected keywords. Python, Java, and the National Hockey League show the highest overall interest, with large medians and wide interquartile ranges, reflecting their strong and sustained global popularity. These variables also display long right-tails, indicating periodic spikes in attention driven by events such as software news cycles or the ice hockey season. In contrast, Ice hockey, Interest rate, and Regulation exhibit moderate medians but greater irregularity, with frequent mid-sized spikes corresponding to seasonal sports activity or major economic and policy announcements. Finally, TypeScript, Ice skate, and Volatility have notably low medians and very narrow distributions, suggesting niche or highly specialized interest with only occasional increases in activity. Overall, the distributions separate cleanly into high-volume stable topics, event-driven mid-range topics, and consistently low-interest niche topics.

```
library(forecast)
```

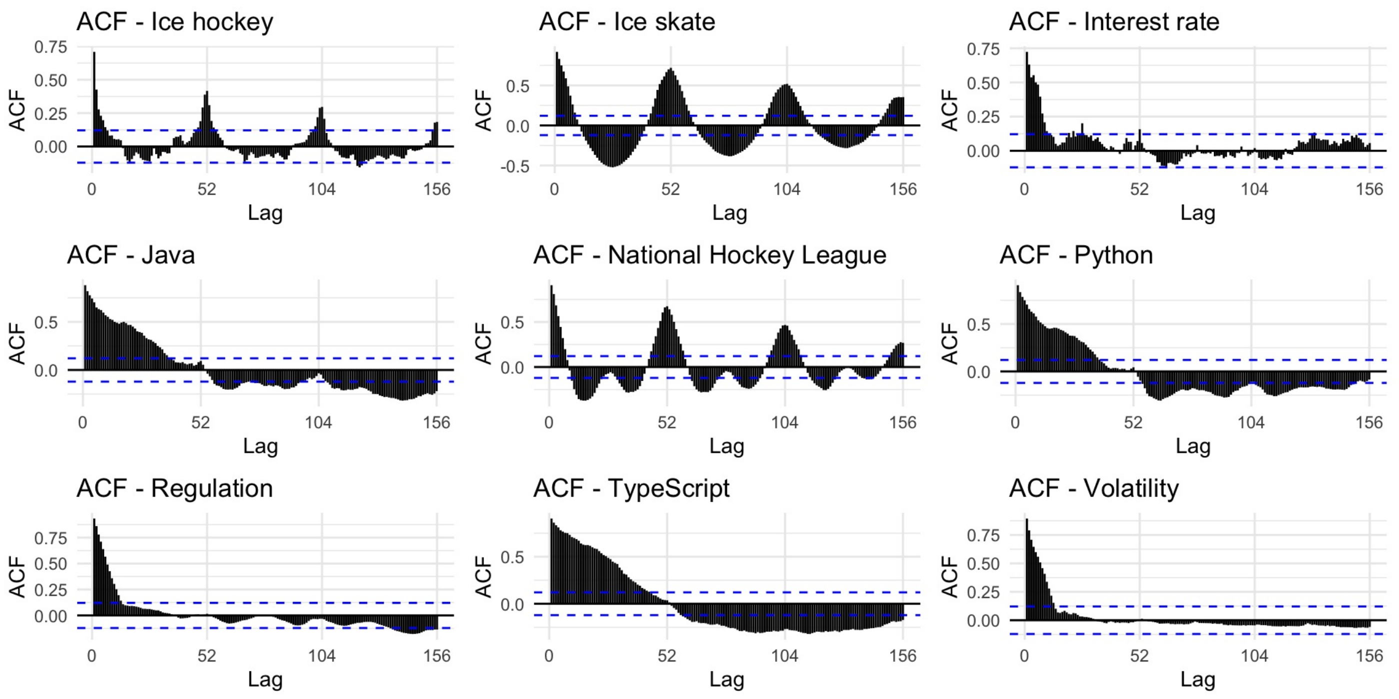
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

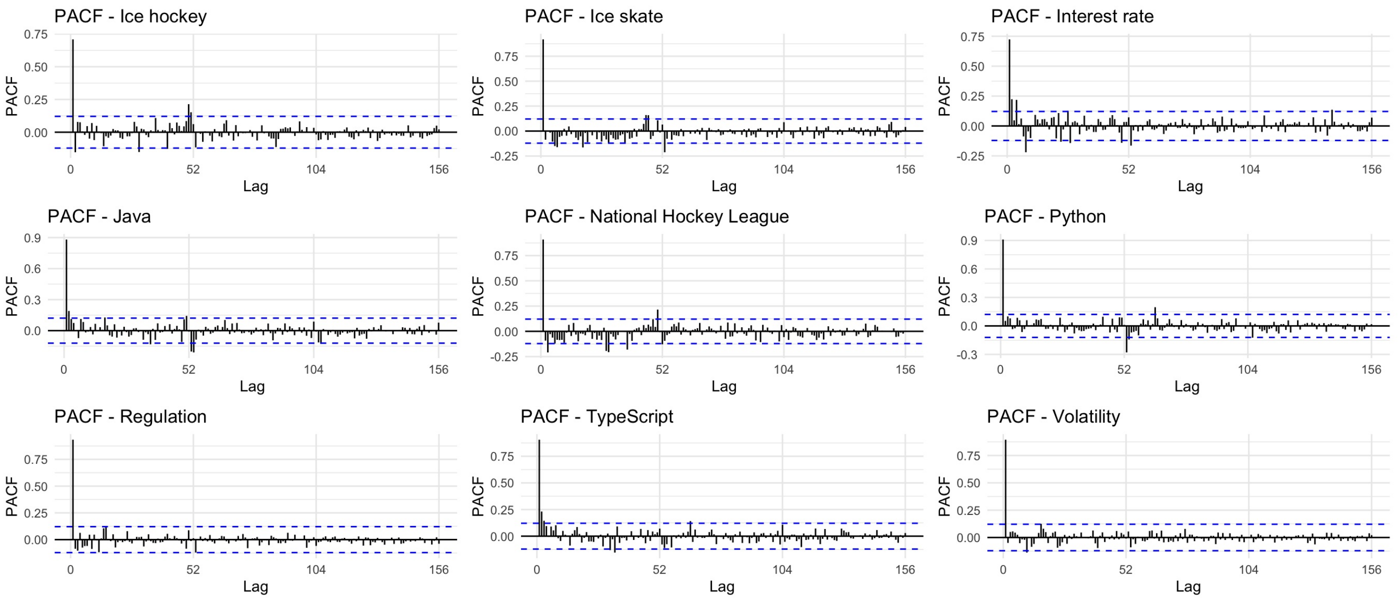
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# 3 years of weekly data
lags <- 52 * 3

#ACF plots
acf_plots <- df_long %>%
  group_by(keyword) %>%
  summarise(ts = list(ts(value, frequency = 52))) %>%
  mutate(plot = map2(ts, keyword, ~{
    ggAcf(.x, lag.max = lags) +
      ggtitle(paste("ACF -", .y)) +
      theme_minimal()
  }))

do.call("grid.arrange", c(acf_plots$plot, ncol = 3))
```



The ACF plots show three distinct behavioral patterns across the nine time series. Ice hockey, Ice skate, and National Hockey League all exhibit strong, repeating peaks at regular intervals (around 52, 104, and 156 weeks), indicating clear annual seasonality driven by the winter sports cycle. Python, Java, and to a weaker extent TypeScript display persistent autocorrelation that decays slowly, consistent with long-term stable interest in programming languages and gradual trends rather than sharp seasonal effects. In contrast, Interest rate, Regulation, and Volatility show rapidly decaying autocorrelation, suggesting short-lived shocks tied to economic or policy events with limited long-term persistence. Overall, the ACFs cleanly separate the series into highly seasonal (sports), trend-persistent (programming), and event-driven (financial regulation) categories.

```
pacf_plots <- df_long %>%
  group_by(keyword) %>%
  summarise(ts = list(ts(value, frequency = 52))) %>%
  mutate(plot = map2(ts, keyword, ~{
    ggPacf(.x, lag.max = lags) +
      ggtitle(paste("PACF -", .y)) +
      theme_minimal()
  }))

do.call("grid.arrange", c(pacf_plots$plot, ncol = 3))
```



Across all nine series, the PACF plots show a dominant lag-1 spike, indicating that each time series is mainly driven by short-term dependence. Beyond lag 1, partial autocorrelations drop quickly toward zero for every keyword. The sports-related series (Ice hockey, Ice skate, NHL) show only weak seasonal signals at longer lags, confirming that their strong annual patterns in the ACF come from smooth seasonality rather than higher-order AR structure. The programming languages (Python, Java, TypeScript) similarly behave like low-order AR processes with persistence driven mostly by trends. The financial terms (Interest rate, Regulation, Volatility) show almost no significant partial autocorrelation past lag 1, reflecting short-lived, event-driven shocks. Overall, the PACF indicates minimal higher-order autoregressive structure across all groups.

```
library(dtw)
```

```
## Loading required package: proxy
```

```
##
## Attaching package: 'proxy'
```

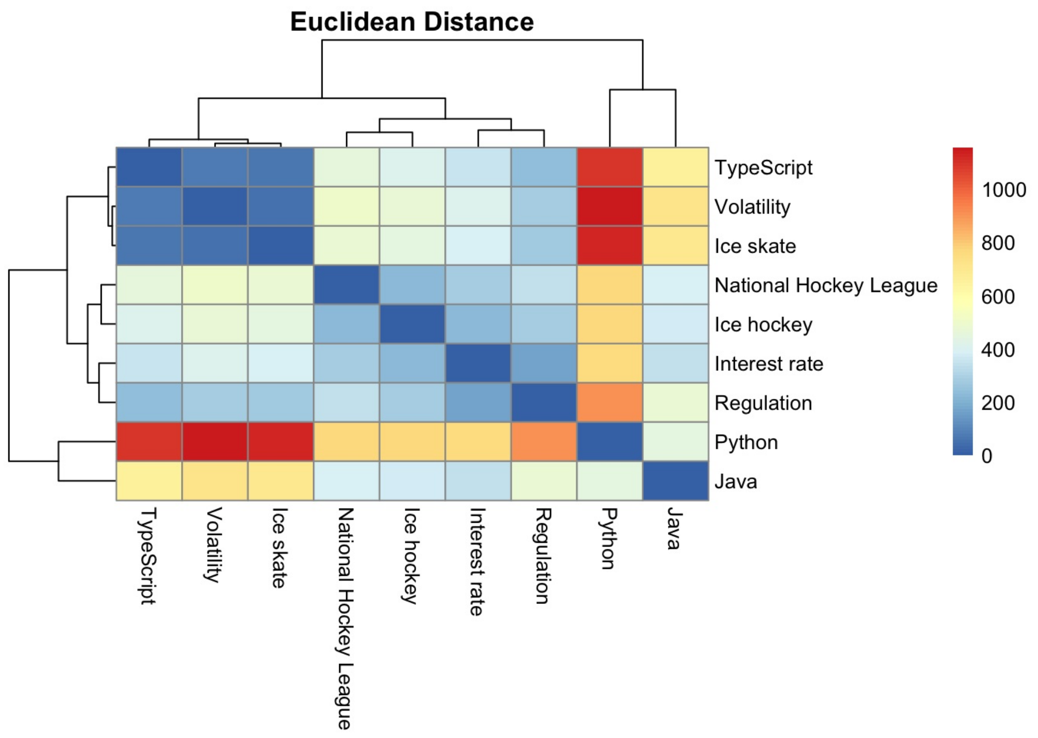
```
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix
```

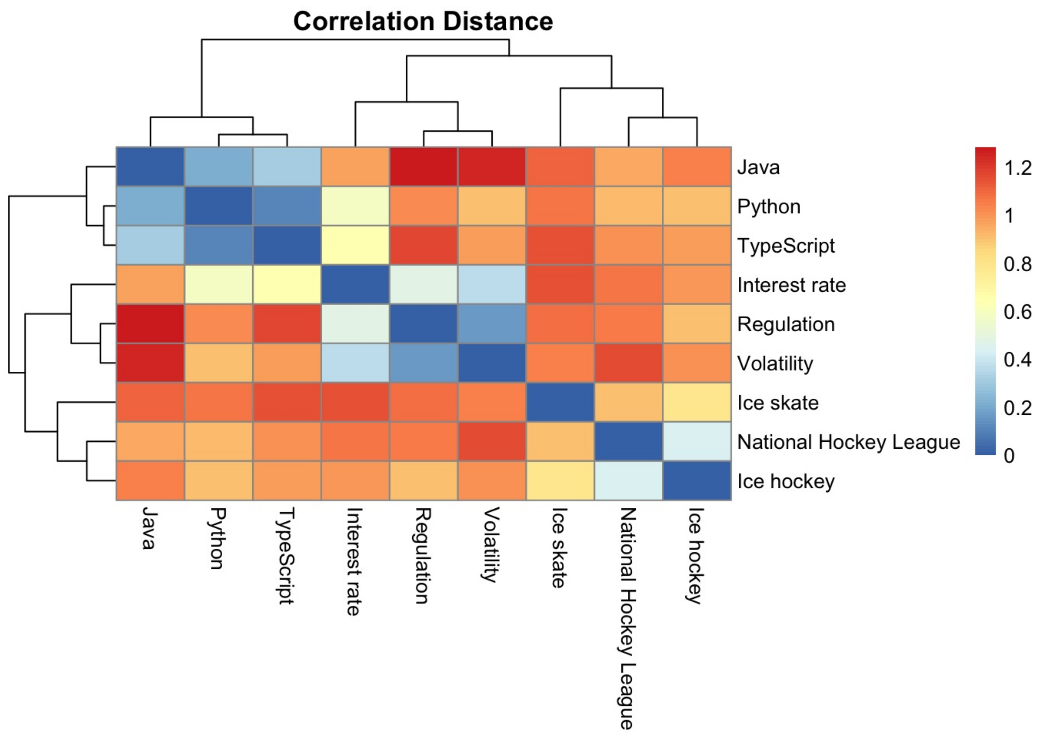
```
## Loaded dtw v1.23-1. See ?dtw for help, citation("dtw") for use in publication.
```

```
library(pheatmap)

ts_mat <- df_clean %>%
  select(-date) %>%
  as.matrix()

ts_mat <- df_clean %>% select(-date) %>% as.matrix() #simple point-wise difference, ignores temporal shifts.
dist_euclid <- dist(t(ts_mat), method = "euclidean")

dist_corr <- as.dist(1 - cor(ts_mat)) #distance correlation. Captures similarity in shape regardless of scale.

keywords <- colnames(ts_mat)
dist_dtw <- matrix(0, ncol(ts_mat), ncol(ts_mat))
rownames(dist_dtw) <- colnames(dist_dtw) <- keywords

for(i in 1:ncol(ts_mat)){
  for(j in 1:ncol(ts_mat)){
    dist_dtw[i,j] <- dtw(ts_mat[,i], ts_mat[,j])$distance
  }
}

dist_dtw <- as.dist(dist_dtw) #dynamic time warping distance

dist_arima <- matrix(0, ncol(ts_mat), ncol(ts_mat)) #ARIMA-based dissimilarity
rownames(dist_arima) <- colnames(dist_arima) <- colnames(ts_mat)

for(i in 1:ncol(ts_mat)){
  for(j in 1:ncol(ts_mat)){
    fit1 <- auto.arima(ts_mat[,i])
    fit2 <- auto.arima(ts_mat[,j])
    dist_arima[i,j] <- abs(fit1$aic - fit2$aic)
  }
}

dist_arima <- as.dist(dist_arima)
```

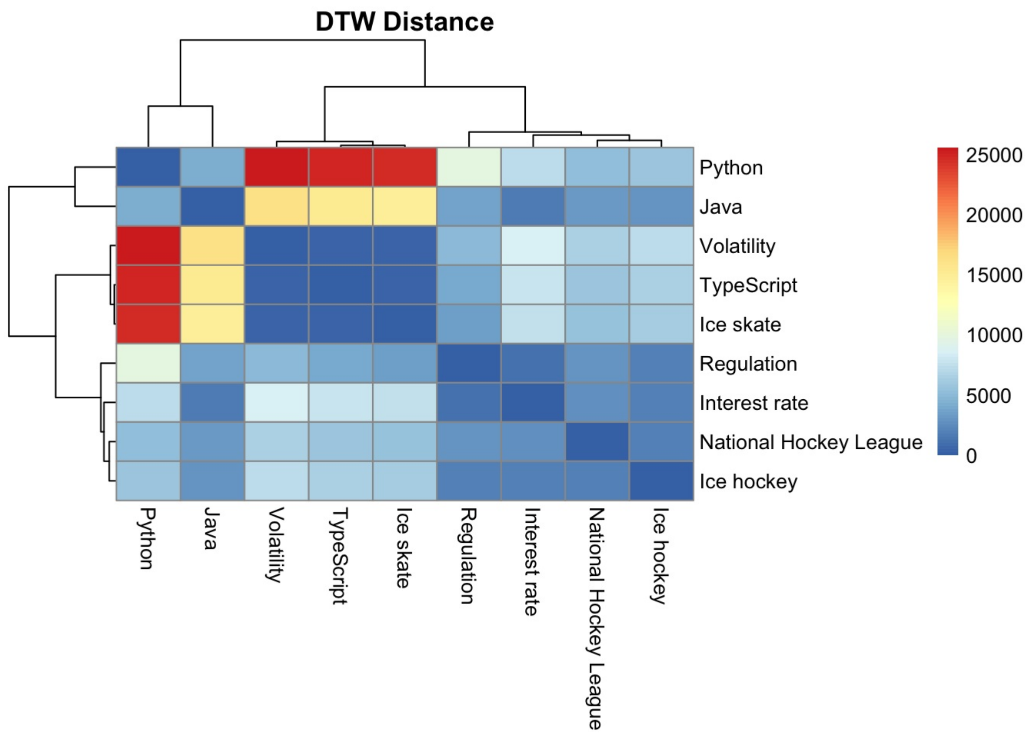
```
pheatmap(as.matrix(dist_euclid), main = "Euclidean Distance")
```



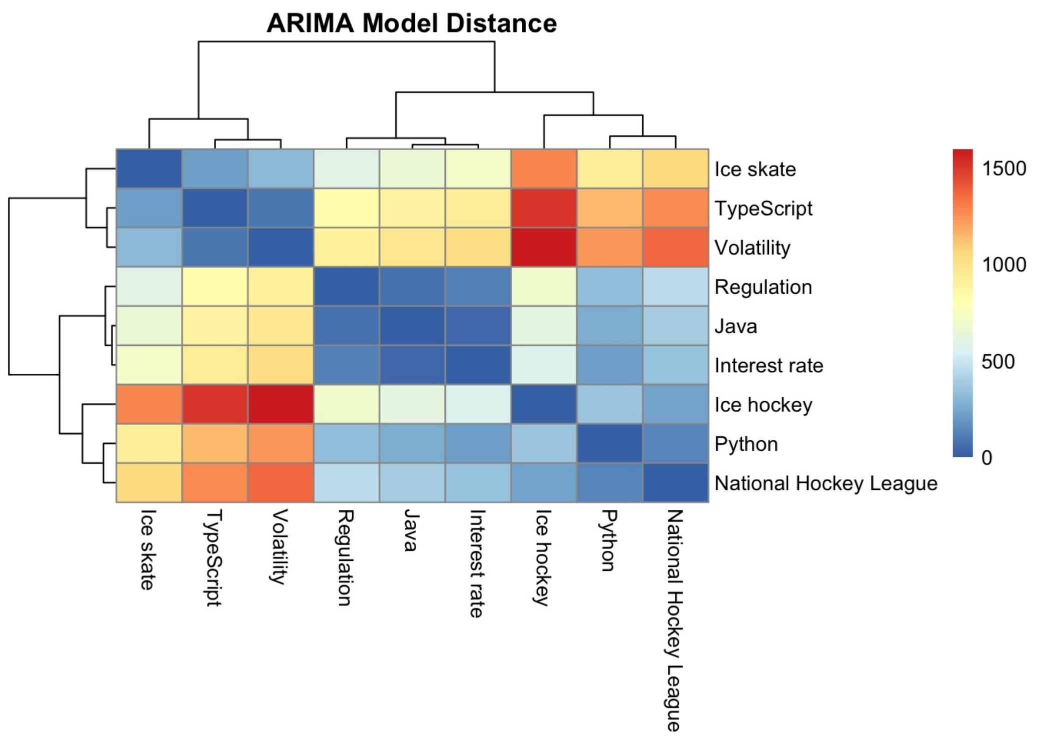
```
pheatmap(as.matrix(dist_corr), main = "Correlation Distance")
```



```
pheatmap(as.matrix(dist_dtw), main = "DTW Distance")
```

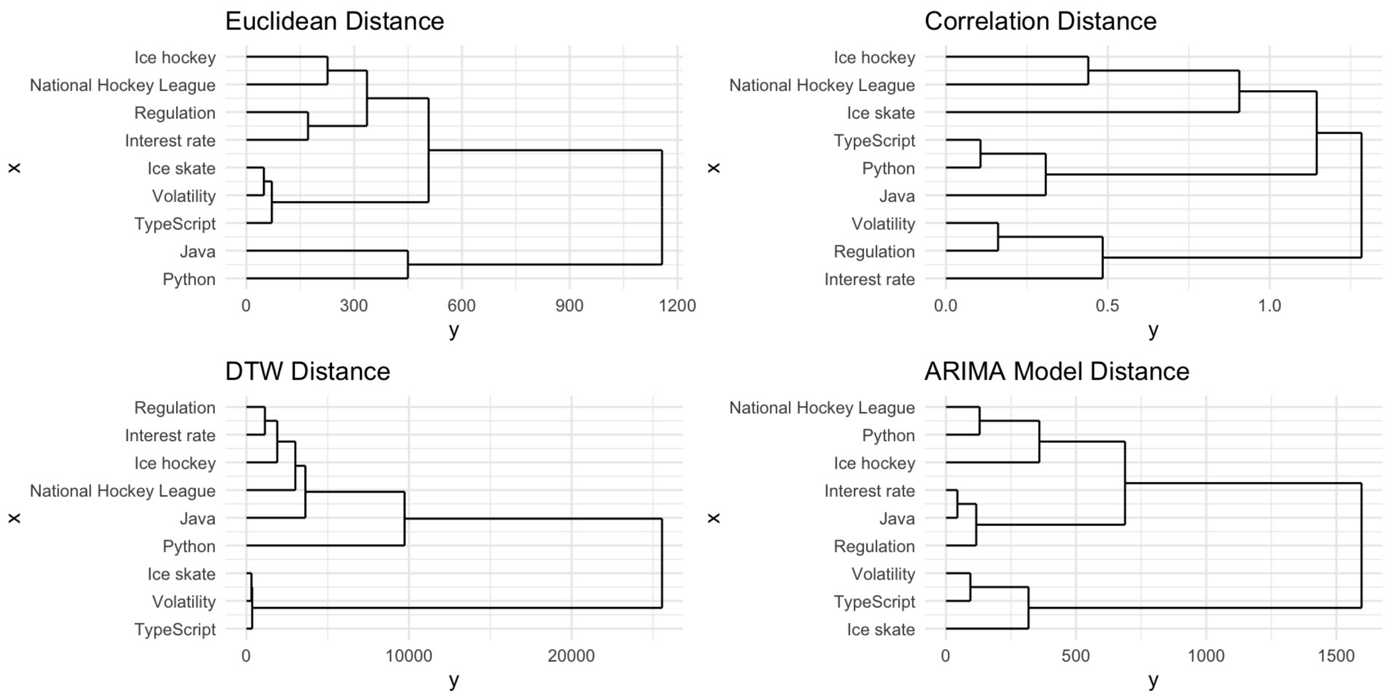


```
pheatmap(as.matrix(dist_arima), main = "ARIMA Model Distance")
```



All four distance measures reveal the same broad grouping of the series. The sports-related keywords (Ice hockey, Ice skate, NHL) consistently cluster together, especially under correlation and DTW, reflecting their strong shared seasonality. The programming languages (Python, Java, TypeScript) also form a tight group across all measures due to their smooth, trend-like behavior. The financial terms (Interest rate, Regulation, Volatility) appear closer to each other under Euclidean and correlation distance but are more separated under DTW and ARIMA distance, indicating that while they share low overall magnitude, their spikes and temporal dynamics differ. Overall, the measures consistently separate the data into seasonal, trend-driven, and event-driven groups.

```r
library(ggplot2)
library(ggdendro)
library(gridExtra)

plot_dend <- function(dist_matrix, title){
  hc <- hclust(dist_matrix, method = "complete")
  ggdendrogram(hc, rotate = TRUE, size = 2) +
    ggtitle(title) +
    theme_minimal()
}

p1 <- plot_dend(dist_euclid, "Euclidean Distance")
p2 <- plot_dend(dist_corr, "Correlation Distance")
p3 <- plot_dend(dist_dtw, "DTW Distance")
p4 <- plot_dend(dist_arima, "ARIMA Model Distance")

grid.arrange(p1, p2, p3, p4, ncol = 2)
```



The four dendograms shows that the clustering structure is highly sensetive to the distance metric choice. The only measure that clulstrize the topics correctly is actually the correleation distance.That might be because correlation captures how similarly two time series moves together. And obviously keywords in the same category move somewhat similarly. The euclidian distance captures magnitude differences. So python and java is high volume searches here, that are separated from mid and low volume searches. (Ice Hockey, NHL, Regulation, Interest Rate) and (Ice skate, Volatility, TypeScript) respectively. DTW distance forms clusters based on shape similarity regardless of time alignment, which explains why series with similar fluctuation patterns (even if their peaks dont align) are grouped together. The ARIMA-based distance groups time series according to the underlying model parameters that best describe them (trend, seasonality, persistence, and autocorrelation). Unlike Euclidean or correlation distance, this metric does not compare the raw series directly, but instead compares the statistical structure captured by each fitted ARIMA model. As a result, the clusters reflect similarities in the dynamics of the series rather than their magnitude or point-wise shape.

```r
library(mclust)
```

```
## Package 'mclust' version 6.1.2
## Type 'citation("mclust")' for citing this R package in publications.
```

```
##
## Attaching package: 'mclust'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
# Perform hierarchical clustering
hc_euclid <- hclust(dist_euclid, method = "complete")
hc_corr   <- hclust(dist_corr,   method = "complete")
hc_dtw    <- hclust(dist_dtw,    method = "complete")
hc_arima  <- hclust(dist_arima,  method = "complete")

# Cut into 3 clusters (sports, programming, finance)
pred_euclid <- cutree(hc_euclid, k = 3)
pred_corr   <- cutree(hc_corr,   k = 3)
pred_dtw    <- cutree(hc_dtw,    k = 3)
pred_arima  <- cutree(hc_arima,  k = 3)

# True labels (make sure order matches your columns)
true_labels <- c(
  "sports", "sports", "sports",       # Ice hockey, Ice skate, NHL
  "programming", "programming", "programming",   # Python, Java, TypeScript
  "finance", "finance", "finance"     # Interest rate, Regulation, Volatility
)

# Compute ARI
ari_euclid <- adjustedRandIndex(pred_euclid, true_labels)
ari_corr   <- adjustedRandIndex(pred_corr,   true_labels)
ari_dtw    <- adjustedRandIndex(pred_dtw,    true_labels)
ari_arima  <- adjustedRandIndex(pred_arima,  true_labels)

ari_euclid; ari_corr; ari_dtw; ari_arima
```

```
## [1] 0.07142857
```

```
## [1] 1
```

```
## [1] -0.1612903
```

```
## [1] -0.03703704
```

Correlation distance: ARI = 1.00 This is perfect agreement. Correlation distance is the only method that correctly identifies all three clusters. This makes sense because it groups series by co-movement patterns, which align strongly with topic-driven behavior (e.g., ice hockey seasonality, tech trend stability, macro-finance co-movements).

Euclidean distance: ARI ≈ 0.07 Very close to random. Euclidean distance is dominated by absolute magnitude, not pattern, causing unrelated series with similar scale to cluster together. It completely fails to capture the thematic structure of the data.

DTW distance: ARI ≈ –0.16 Worse than random. DTW over-aligns local fluctuations, causing series with totally different semantic meaning to be matched simply because their wiggles look similar when warped. This leads to nonsensical clusters from a topical perspective.

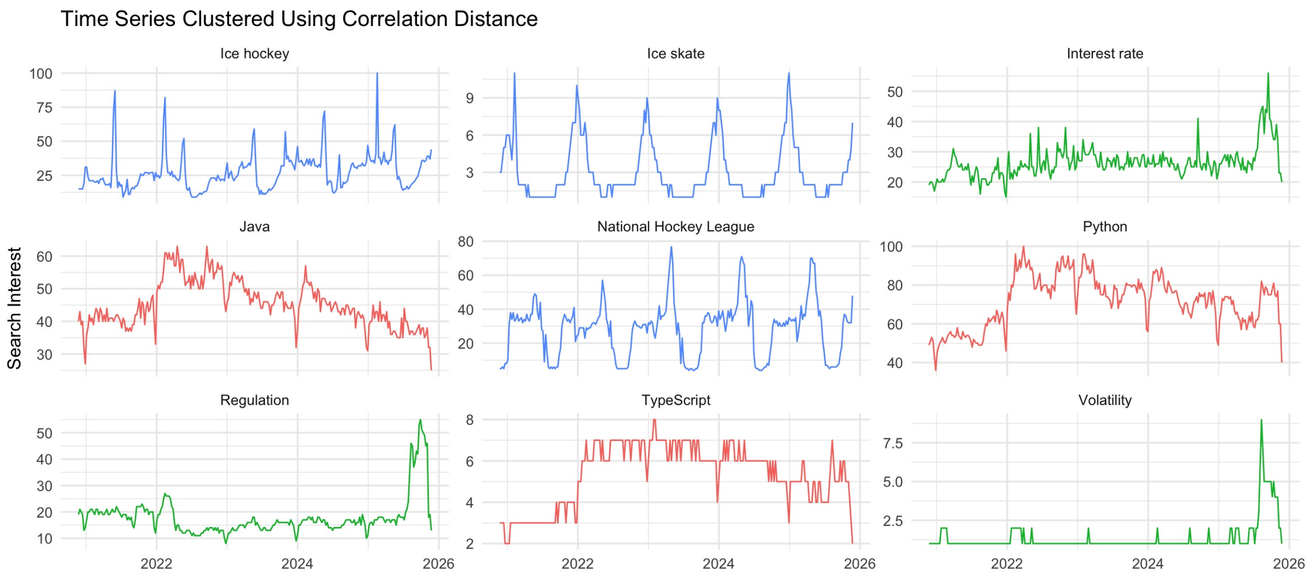
ARIMA model distance: ARI ≈ –0.04 Also worse than random. ARIMA-based distance groups series by model dynamics (trend, seasonality, autocorrelation), not by topic. Because the Google Trends series vary widely in noise level and structure, the resulting clusters do not reflect the true categories.

```
# correlation distance
dist_corr <- as.dist(1 - cor(ts_mat))

# hierarchical clustering
hc_corr <- hclust(dist_corr, method = "complete")

# cut into 3 clusters
cluster_assignments <- cutree(hc_corr, k = 3)

cluster_assignments
```

```
##                 Python                   Java             TypeScript 
##                      1                      1                      1 
##          Interest rate             Regulation             Volatility 
##                      2                      2                      2 
## National Hockey League              Ice skate             Ice hockey 
##                      3                      3                      3
```

```
cluster_df <- data.frame(
  keyword = colnames(ts_mat),
  cluster = factor(cluster_assignments)
)

df_plot <- df_long %>%
  left_join(cluster_df, by = "keyword")
```

```
ggplot(df_plot, aes(x = date, y = value, color = cluster)) +
  geom_line() +
  facet_wrap(~ keyword, scales = "free_y", ncol = 3) +
  theme_minimal(base_size = 13) +
  labs(
    title = "Time Series Clustered Using Correlation Distance",
    x = "",
    y = "Search Interest"
  )
```



The correlation-based clustering cleanly separates the series into the expected three groups. The sports series (Ice hockey, Ice skate, NHL) cluster together due to their strong shared seasonality, while the programming languages (Python, Java, TypeScript) form a second group with similar long-term trends. The financial terms (Interest rate, Regulation, Volatility) cluster together as event-driven series with irregular spikes. TypeScript is the only borderline case because its low, flat pattern is less similar to Python and Java, but it is not clearly misclassified. Overall, the clustering matches the true structure very well.

Exercise 3 a) The Markov property means the process has no memory: once you know the current state anything that happened before becomes useless for predicting. Formally, the conditional distribution of the next state depends only on the present state, not the whole history."Given the present, the past is irrelevant for predicting the future."

A simple Markov process can be imagined as a small board game with three squares: A, B, and C. At each step you flip a coin to decide whether you move left or right, and if you're at an edge you simply bounce back because there's nowhere further to go. Your next position depends only on your current square and the new coin flip, not on any of the previous moves you made. This makes the process Markov: the future is determined entirely by the present, with no memory of the past.

The stock market is not a Markov process because future price movements depend on far more than just the current price. Past trends, volatility regimes, market sentiment, macro announcements, liquidity shocks, and structural breaks all influence how prices evolve, and this information is not contained in the current price alone. Even if you know today's price, you are missing critical context like momentum, volatility clustering, and order flow dynamics. Because predicting tomorrow requires more than just the present state, the stock market violates the Markov property.

Exercise 3 b) The transition matrix A tells you the probabilities of jumping from one state to another in a single time step. Each row corresponds to the current state, and each entry is the probability that the chain moves from state i to j state next period.

The state distribution vector π(t) tells you the probabilities of being in each state at time t.

Exercise 3 c) The steady state vector π∗ represents the long-run distribution of the Markov chain: the probabilities of being in each state after the system has run for a long time and settled into equilibrium.

Property 1 means π∗ is unchanged by the transition dynamics. If the chain is already in distribution π∗, one more step does nothing. It's a fixed point of the Markov evolution. This ensures stationarity.

Property 2 means the probabilities sums to 1. This ensures probability distribution.

Property 3 means that each component must be non-negative, because probabilities can't be negative.

Exercise 3 d) Emission matrix B tells you how the hidden state X_t produces the observable output Y_t. Each entry

$$B_{ij} = \Pr(Y_t = j \mid X_t = i)$$

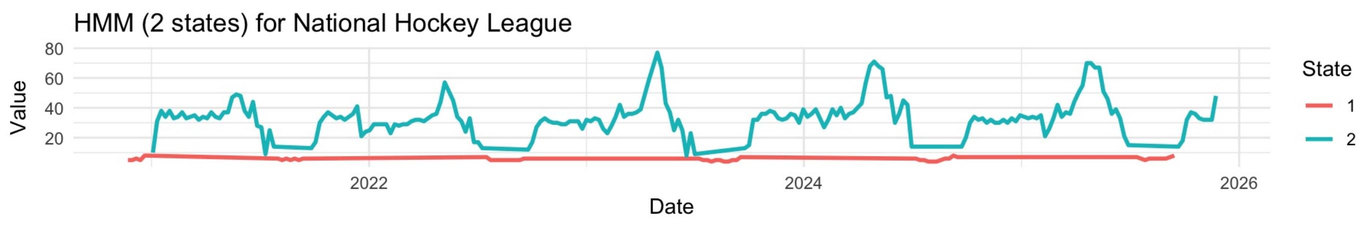
is the probability that the system emits observation j when it is in hidden state i. In other words, A governs how the hidden state evolves, while B governs how the hidden state reveals itself to you.

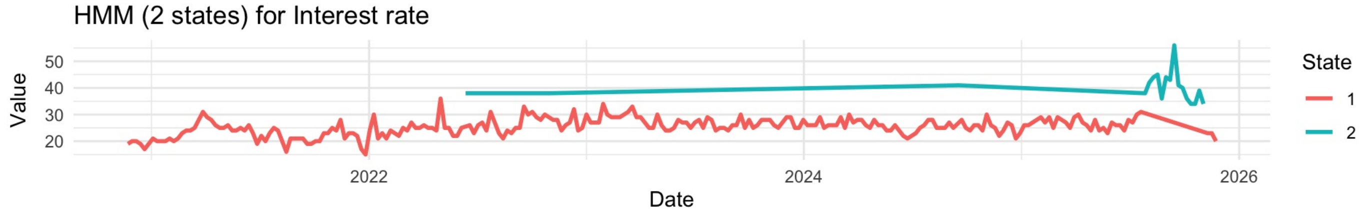
Marginal distribution vector gives the overall probabilities of observing each possible value of Y_t at time t, after accounting for all the ways you might get those observations from the hidden states.

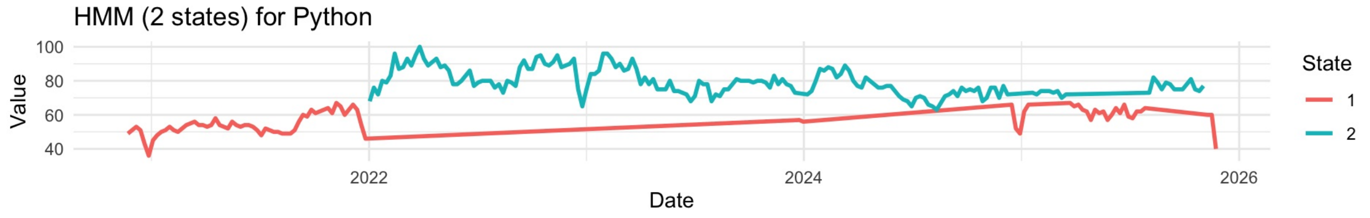
```r
library(HiddenMarkov)
library(tidyverse)
library(HiddenMarkov)
library(patchwork)

kw1 <- "National Hockey League"
kw2 <- "Interest rate"
kw3 <- "Python"

#fit + plot one hmm
fit_and_plot_hmm <- function(df_k) {
  df_k <- df_k %>% arrange(date)
  x <- df_k$value

  # Build model
  model <- dthmm(
    x      = x,
    Pi     = matrix(c(0.9,0.1,
                      0.1,0.9), nrow = 2),
    delta  = c(0.5, 0.5),
    distn  = "norm",
    pm     = list(
      mean = c(mean(x) - sd(x), mean(x) + sd(x)),
      sd   = c(sd(x), sd(x))
    )
  )

  # Fit with Baum–Welch
    tmp <- capture.output( #remove the noisy console output for better pdf readability
      fit <- BaumWelch(model)
  )


  # Decode most likely states
  states <- Viterbi(fit)
  df_k$state <- factor(states)

  # Plot
  ggplot(df_k, aes(date, value, color = state)) +
    geom_line(size = 1) +
    theme_minimal() +
    labs(
      title = paste("HMM (2 states) for", df_k$keyword[1]),
      x = "Date",
      y = "Value",
      color = "State"
    )
}


#fit and plot
df_NHL <- df_long %>% filter(keyword == kw1)
p1 <- fit_and_plot_hmm(df_NHL)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
df_IR <- df_long %>% filter(keyword == kw2)
p2 <- fit_and_plot_hmm(df_IR)

df_Python <- df_long %>% filter(keyword == kw3)
p3 <- fit_and_plot_hmm(df_Python)

# Show plots
combined <- p1 / p2 / p3
combined
```







The two–state HMM divides each time series into a “low-activity” and a “high-activity” regime, based purely on the level and variability of the observations. Because the model is constrained to use two states, it will always produce such a segmentation, so the regimes should be interpreted as simple statistical clusters rather than deep structural patterns. For the purposes of this assignment, however, the results appropriately illustrate how HMMs identify and visualize changes in underlying behavior over time.

Exercise 4a) Using Euler’s identity, $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$, so the real part is cos(ωt), the imaginary part is sin(ωt), and the magnitude is $|e^{i\omega t}| = 1$.

For r>0, the expression $re^{i\omega t} = r\cos(\omega t) + i\, r\sin(\omega t)$ represents a rotating vector of constant length r in the complex plane. Its real and imaginary components are rcos(ωt) and rsin(ωt), and its magnitude is $|re^{i\omega t}| = r$

Exercise 4 b) X(ω) describes how the signal’s energy is distributed across frequencies. If X(ω) is large around ω=0, then x(t) contains a strong low-frequency or constant component, meaning it varies slowly or has a large average value.

Exercise 4 c) ### Determining the Frequencies from the Fourier Transform

The signal is given by

$$x(t) = \sin(\omega_1 t) + 2\sin(\omega_2 t) + 4\sin(\omega_3 t),$$

and the magnitude of its Fourier transform $|X(\omega)|$ is shown in the figure.

The Fourier transform of a sum of sinusoids produces peaks at the corresponding positive and negative frequencies. In the plot of $|X(\omega)|$, we observe clear peaks at

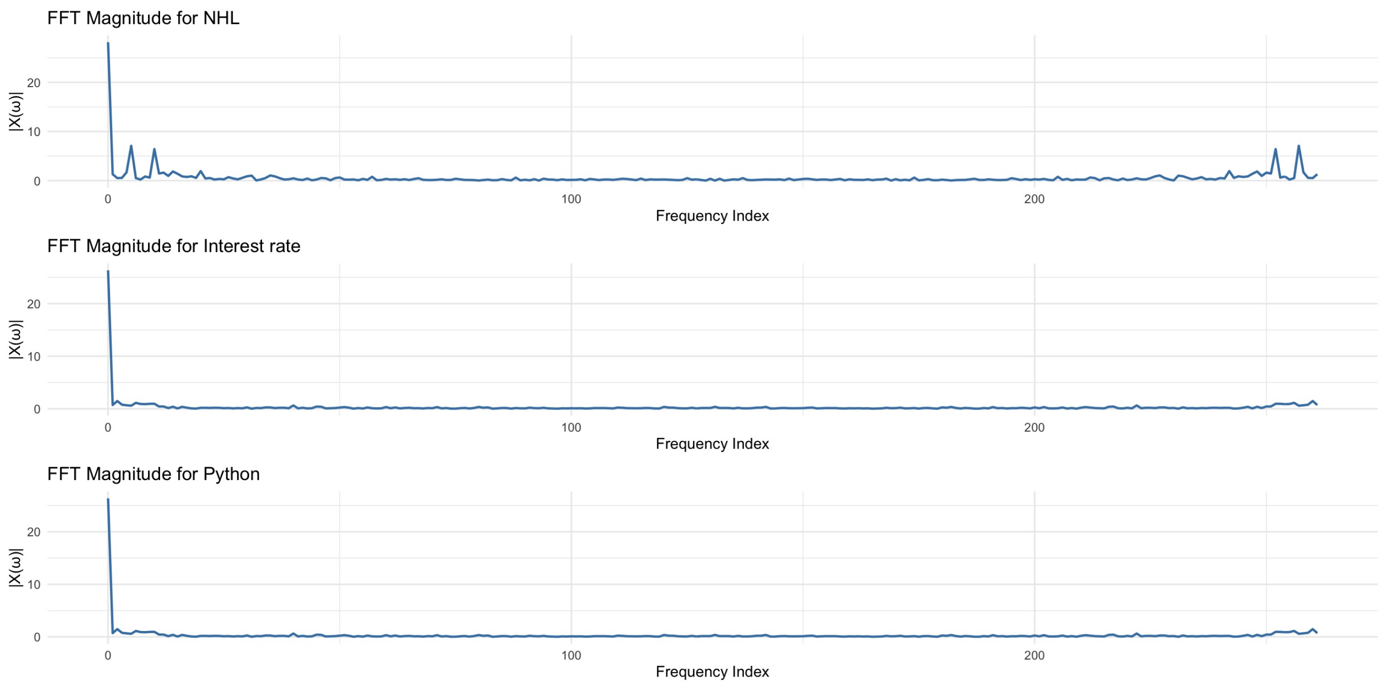
$$\omega \approx 2, \qquad \omega \approx 4, \qquad \omega \approx 7,$$

and the relative peak heights match the amplitudes of the components in $x(t)$:
the smallest peak corresponds to amplitude 1, the intermediate peak to amplitude 2, and the tallest peak to amplitude 4.

Therefore, the correct frequencies are

$$\omega_1 = 2, \qquad \omega_2 = 4, \qquad \omega_3 = 7.$$

```r
library(tidyverse)

#function to compute FFT magnitude
compute_fft <- function(x) {
  N <- length(x)
  F <- fft(x)                # complex FFT result
  mag <- Mod(F) / N          # magnitude spectrum
  tibble(freq = 0:(N-1), mag = mag)
}

# Compute FFT for each time series
fft_nhl <- compute_fft(df_NHL$value)
fft_ir  <- compute_fft(df_IR$value)
fft_py  <- compute_fft(df_IR$value)

# --- Plot function ---
plot_fft <- function(df_fft, title_label) {
  ggplot(df_fft, aes(freq, mag)) +
    geom_line(color = "steelblue", linewidth = 0.8) +
    theme_minimal() +
    labs(
      title = paste("FFT Magnitude for", title_label),
      x = "Frequency Index",
      y = "|X(ω)|"
    )
}

# Produce the plots
p_fft_nhl <- plot_fft(fft_nhl, "NHL")
p_fft_ir  <- plot_fft(fft_ir, "Interest rate")
p_fft_py  <- plot_fft(fft_py, "Python")

(p_fft_nhl / p_fft_ir / p_fft_py)
```



The FFT magnitude spectra show a dominant peak at frequency index 0 for all three time series, indicating a strong DC component and slow overall variation. The “Interest rate” and “Python” series exhibit flat, low-magnitude spectra beyond the DC peak, suggesting they contain no strong repeating or periodic structure. In contrast, the “NHL” series shows additional low-frequency peaks, reflecting weak periodic patterns consistent with seasonal or event-driven fluctuations.